OO Methods in Teaching: Current Experience and Future Possibilities

Gillian Lovegrove^a, Phil McLaughlin^a, Jakub Chabik^a, Pippa Standen^b

^aSchool of Computing, Staffordshire University, Stafford UK

^bSchool of Information Systems, SIHE, Southampton UK

Keyword Codes: D.1.5; D.2.10

Keywords: OO teaching, OO analysis and design, OO methods

Abstract

The potential of object-oriented technology (OOT) to ease the problems of complexity, maintenance costs and reuse is now being realised for large systems. OOT is now recognised as a technique for "serious" software engineering, such that the number of commercial products built using OOT is on the increase. A significant amount of research and commercial activity in the object-oriented database field is seen as another sign of the importance of the OO paradigm. The Object Management Group (OMG) has produced an object model which is acceptable to over 100 companies world-wide.

OOT is sufficiently important for it to be included in every undergraduate curriculum. Introducing students to the object paradigm within traditional computing and information systems courses is a difficult task. There is a need to choose a suitable language that supports objects (OOPL) and/or the need to teach an OO analysis and design method appropriate for the newer paradigm. The timing for these is crucial and discussed in the paper.

However the paper focuses on the choice of method, which, given the plethora of OO methods on the market, is not easy. Criteria for assessing methods for teaching are given, together with an evaluation for two well known methods. The potential of two recent "hopeful" methods is assessed and guidelines given for using methods within the curriculum.

1. INTRODUCTION

Object orientation is now accepted as a software engineering technique, resulting in an increasing number of commercial products, driven partly by a trend towards componentware [1]. Object-oriented databases are seen as a significant addition to relational databases, though the technology for their integration appears to be taking a long time to mature. Nevertheless the focus on the standardisation of the object model for databases is likely to have an impact on OO technology in general.

2. LIFE CYCLES, METHODS AND LANGUAGES

Typically structured development proceeds down the waterfall cycle from requirements capture to analysis, design implementation and maintenance; systems development models are constructed involving entity relationship models, data flow diagrams, and entity life histories.

In contrast, OO analysis and design (OOAD) is an inherently iterative process with the distinction between the analysis and design processes being less well defined. Analysis might be said to be the identification of objects from the problem space, whereas design is the refinement of objects and class hierarchies in the solution space. This refinement of classes becomes somewhat dependent on the coding

language. All in all therefore, the notion of a waterfall cycle is less appropriate to OOAD than traditional methods.

2.1 Analysis and design

OO analysis starts by an attempt to identify objects and their relationship with each other (client, server, responsibility); this is a difficult task. Ideally the analyst would have previous experience in OO modelling which has implications with respect to teaching.

The design phase consists of the generation of the class hierarchies, subsystems and modules. In practice it is rare that any software system is so fully specified that no further changes are required during design and before implementation. Also, the design phase can be affected by characteristics of the implementation language such as the existence of single or multiple inheritance and file structures.

2.2 The different methods

The design and use of OO methods is currently an area of great debate with at least 40 different methods available to choose from [2,3,4]. Methods have been developed from different backgrounds, such as modelling and simulation, or more traditional information systems background. Some concentrate more on analysis and some mainly on design.

Each of the methods hopes to achieve a dominant position in the market. With the experience of traditional methods in mind, it is unlikely that any one method will dominate for all domains and situations, but as the technology matures a smaller number of methods will become popular. These will undoubtedly have a good CASE tool associated with them for a reasonable price, and capable of producing code.

3. OVERVIEW OF CURRENT STATE OF THE ART

There are many books which teach an OOPL first and incorporate little in the way of design [5,6,7,8,9,10]; Meyer encourages formal approaches to software specification, construction and verification [11,12]. Others concentrate on a design method but incorporate examples in languages [17,18]. Weiner and Pinson describe current practice in university courses, short courses and seminars [13]. They assert that in general four steps are followed: providing an understanding of the OOP approach, illustration of the properties and features of OOP by teaching an OOP language, presentation of a features analysis, and integration of the first three steps into a logical approach to problem solving.

Wu describes a three-stage, incremental approach in teaching OOP [13]. The first stage includes making students able to write a simple program, unlearning old methods of programming. The second stage introduces the benefits of code reuse, abstraction and encapsulation. Stage three introduces students to client/server programming, the concept of inheritance and polymorphism, and introduces an OO design method. A later article [15] gives more detail of the design stage. He asserts that OOP is ultimately learned only through constant use and experimentation.

We see that there is support for a broad approach in the USA integrating OO concepts with language and design, allowing the student time to merge the ideas arising from theory and practice. The fundamental differences between the American and the European methods for systems development may be partly due to cultural differences between America and Europe and in part by the different educational systems [16]. American methods are strongly focused on pragmatism and informality, whereas European methods are more formal, resulting in better predictability and system quality. The dilemma for the European teacher is that a pragmatic approach during teaching aids faster understanding, whilst teaching a design method more formally, runs the risk of overloading the students with the details of the method at the expense of understanding the underlying principles of object technology.

4. CRITERIA FOR METHODS FOR TEACHING

In the light of the experience above, the amount of familiarity students have with object-oriented concepts before they start concentrating on analysis and design is a factor in deciding which method would be a good one to adopt. It is assumed here that the previous experience is likely to exist but be limited. The criteria for choosing a method are given below:

- (a) students can do it: it is conceptually simple; draws on previous experience
- (b) it is well documented (in a book)
- (c) it is easy to visualise and the notation is simple
- (d) it is useful in industry, well tested and popular
- (e) there is a CASE tool for the method which is affordable
- (f) it encourages group work

Criteria (a), (b), (c), (d) and (e) are self-explanatory. The last one reflects the increased emphasis on group activities which arises from using object-oriented methods. It also reflects the need in the undergraduate curriculum to include group activities, since this experience will be a support for their careers at a later date.

5. ASSESSMENT OF SELECTED METHODS

CRC cards were designed and used especially for the teaching environment [20] before the book associated with the method [21] was published. The method appeared effective and cheap and was adopted at the University of Southampton. OMT is the method promoted by Rumbaugh, whose book was one of the first to appear [18] and offered a more detailed prescriptive view of objects; this was adopted at Staffordshire University. A number of other methods are known to have been used in other UK universities; these include Booch, Coad/Yourdon and Shlaer/Mellor, chosen largely because of the availability of the books.

5.1 CRC or Wirfs-Brock method

This method might be said to have an exploratory, then analysis and design phase. The exploratory phase deals with gathering of information, in particular the identification of classes and their associated responsibilities. This phase relies heavily on heuristics and uses CRC cards (Class, Responsibility and Collaboration) as the primary vehicle for team communication. CRC cards are index cards or PostIts on which designers write information about proposed classes.

(a) students can do it: it is conceptually simple; draws on previous experience

CRC is simple and students can pick it up easily. It does not draw on past experience.

(b) it is well documented (in a book)

The book [Wirfs-Brock 90] is a good one with well explained examples and heuristics.

(c) it is easy to visualise and the notation is simple

The notation is simpler, good for teaching though it may be lacking for commercial needs.

(d) it is useful in industry, well tested and popular

It is currently less well known in industry and less used.

(e) there is a CASE tool for the method which is affordable

The method, being based on group work and heuristics, does not lend itself to a CASE tool; if there are CASE tools, they are not well known.

(f) it encourages group work

This method is designed for group work though it can be used singly.

5.2 OMT

OMT [18] is a very popular method in the software engineering community. Having been developed from a modelling and simulation background it has been adapted for use in what would normally be considered mainstream information systems applications.

(a) students can do it: it is conceptually simple; draws on previous experience

OMT consists of three basic models. The first is the object model represented using object structure diagrams, close to the conventional entity relationship models. This allowed them to model the object structures fairly quickly. Similarly, the dynamic model shows object behaviour using state diagrams (c.f. entity life histories), and the functional model shows value mapping and functional dependencies with data flow diagrams. The three models are complementary and for simple models OMT is very clear.

(b) it is well documented (in a book)

The book [18] is clear, and Rumbaugh writes regularly in JOOP, presenting case studies.

(c) it is easy to visualise and the notation is simple

The notation is not too complex for students to grasp.

(d) it is useful in industry, well tested and popular

OMT has been applied fairly extensively. It is less of an academic tool than a practical method and has thus found acceptance with C++ and Smalltalk developers

(e) there is a CASE tool which is affordable

A number of CASE tools support OMT, including low cost tools (e.g. Select OMT).

(f) it encourages group work

OMT neither enforces or discourages group working.

6. EXPERIENCE WITH THESE TWO METHODS

Experience with students is mixed. Some students are able to grasp the modelling concepts with ease, whilst some have great difficulty; students rarely perceive the importance of analysis and design, and tend to divorce it from system development. Thus their analytical ability is below what would be expected in industry. Often they lack general business or industrial knowledge. Modelling requires some understanding of the domain, but more importantly an understanding of the concepts of object technology. Initial identification of object structures and relationships is vital; CRC is better than OMT in this respect. Having identified the classes, students find it difficult to classify them into inheritance hierarchies and aggregations due to their relative lack of experience.

A more radical change in concept for students to come to grips with, is the notion of object to object communication via message passing. In CRC this concept is exercised by groups working through scenarios, in contrast to OMT where it is modelled using state diagrams. Experience shows that stronger students gain an understanding of the basic OO concepts relatively easily, but weaker students find they are a significant hurdle to be overcome before they can concentrate on using the chosen method.

7. NEW METHODS THAT MIGHT HELP

ObjectOry and Fusion are relatively recent methods which may prove of greater use with students. Jacobson's method, ObjectOry, would arguably build on students' familiarity with the traditional SDLC. Fusion combines the best of several methods and is a mix of Rumbaugh's OMT, Wirfs-Brock's method, Booch's OOAD and formal methods.

7.1 ObjectOry

Jacobson's ObjectOry method takes a global view and centres on an analysis of the way a system is actually used; his use of three types of objects (Interface, Entity and Control) which separate interface behaviour from entity objects and keep these independent of controls, helps to construct sophisticated designs and provides a robust model. Both the processes and the method are represented as objects; in the opinion of some people, this may lead to software which is not truly object-oriented.

Central to ObjectOry is the concept of Use Cases, which identify a scenario, process, particular pattern or usage, which link the complete SDLC. Organisation analysis and design models around user interaction and actual usage scenarios. Each Use Case is identified through the actors of each scenario and the students can decide the actors' roles for each process, identifying the main tasks of each actor. The thread provided by Use Cases runs through the whole project.

(a) students can do it: it is conceptually simple; draws on previous experience

Use Cases call for a degree of role play which can appeal to students. The method is conceptually simple, with some underpinning by traditional and structured SDLC methods.

(b) it is well documented (in a book)

The book is a good one [3].

(c) it is easy to visualise and the notation is simple

ObjectOry is good here too.

(d) it is useful in industry, well tested and popular

It appears to becoming quite popular.

(e) there is a CASE tool which is affordable

ObjectOrySE, ObjectOry Support Environment
is the CASE tool.

(f) it encourages group work

Use Case role play encourages supportive and participative group work.

7.2 Fusion

(a) students can do it: it is conceptually simple; draws on previous experience

Fusion is very easy to pick up. In contrast to OMT, it supports the earliest stage - object identification - and focuses on analysis.

(b) it is well documented (in a book)

The book [23] is excellent, especially for students, in that it is short, communicative and based on examples rather than high level formality.

(c) it is easy to visualise and the notation is simple

The notation is simple and has similarities with OMT.

(d) it is useful in industry, well tested and popular

of several methods and also supports invariants, preconditions and postconditions.

(e) there is a CASE tool which is affordable

There are three CASE tools mentioned in the book.

(f) it encourages group work

Not explicitly.

8. CONCLUSIONS

Our experience supports that of Weiner and Wu [13,14,15]. In addition, it appears that the more formal approach of European education may lead to greater difficulties in students understanding and using object-oriented technology. Thus whilst many methods may be highly thought of in a commercial setting, it is useful for students inexperienced in OO to learn a method in a curriculum that strengthens these weaknesses. Existing methods, like OMT, may be used more successfully with the addition of

working through scenarios and role playing. Alternatively, a new method like Fusion, with a greater emphasis on finding objects, might be an improvement. There is some concern about ObjectOry in that it may not lead to strictly object-oriented systems, but since this whole technology is relatively young, this appealing method should not be ignored.

Undergraduate curricula need to contain many more contact hours for object technology, including experience with concepts and use of a suitable language before methods can be taught successfully. In time we believe this will happen, but meanwhile many students may find themselves baffled. For them, the power and potential of OO may be obscure.

References

1. Byte Magazine, May 1994.
2. Graham, I. *Object-Oriented Methods* (2nd edition). Addison Wesley.
3. Jacobson, I. et al. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison Wesley 1992.
4. Hutt, A. *Object Analysis and Design: Description of Methods*. John Wiley 1994.
5. Graham, N. *Learning C++*. McGraw Hill 1991.
6. Kaehler, T. & Patterson, D. *A Taste of Smalltalk*. Norton and Co. 1986.
7. Lippman, S. *C++ Primer*. Addison Wesley 1990.
8. Shafer, D. & Ritz, A.D. *Practical Smalltalk*. Springer Verlag 1991.
9. Shafer, D. *Smalltalk Programming for Windows*. Prima Publishing 1993.
10. Stroustrup, B. *The C++ Programming Language*. Addison Wesley 1987.
11. Meyer, B. *Eiffel: the language*. Prentice Hall 1992.
12. Meyer, B. *Toward an object-oriented curriculum*. JOOP May 1993 v6 n2.
13. Weiner, R. & Pinson, L. *OOP: an academic perspective*. JOOP June 1993 v6 n3.
14. Wu, C.T. *Three-stage, incremental approach in teaching OOP*. JOOP July-August 1993 v6 n4.
15. Wu, C.T. *Teaching OOP in three stages, stage 3*. JOOP September 1993 v6 n5.
16. Kristen, G. *From European Methods to Object Orientation*. *Objects in Europe*. Vol 1 No 2 Spring 1994.
17. Booch, G. *Object-oriented Design* (2nd Edition). Addison Wesley 1994.
18. Rumbaugh, J et al. *Object-oriented modelling*. Prentice Hall 1991.
19. D'Souza, D. *Working with OMT*. JOOP October 1993 v6.
20. Wirfs-Brock, R & Wilkerson, B. *Object-oriented design: a responsibility approach*. OOPSLA Proceedings, Sigplan Notices, 24. ACM Press 1989.
21. Wirfs-Brock, R et al. *Designing Object-oriented Software*. Prentice Hall 1990.
22. Coleman, D et al. *Object-oriented Development: the Fusion Method*. Prentice Hall 1994.