View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by AIS Electronic Library (AISeL)

On Teaching an Object-Oriented Database Module in Undergraduate CS/IS Curricula[1]

Billy B. L. Lim, Ph.D

KEYWORDS: *Object-Orientation, Database Management Systems, CIS Curriculum, Object-Oriented Databases, Computer Science Education.*

ABSTRACT: *As object-oriented technology becomes more and more prevalently used in the computing industry, it is important that the computing science curricula keep up with the technological trend. The literature shows numerous efforts in this direction but few have dealt with object-oriented databases. Much of the efforts are in the areas of incorporating object-orientation into introductory computing courses or integrating special topics courses on object-oriented programming into the curricula.This paper describes the experiences that the author had while integrating object-orientation into a CS/IS curriculum. In particular, it illustrates where and how object-oriented databases can be introduced in an introductory DBMS course by using some object-oriented DBMS prototypes. A sample module for introducing object-oriented databases is also presented.*

INTRODUCTION

In recent years, object-oriented (OO) technology has become one of the dominant technologies in the computing industry. In a recent survey, it was reported that over 75% of the Fortune 100 companies have adopted OO technology to some degree for their computing needs [1]. To reflect the advances in OO technology and its acceptance by the computing industry [2, 3], many computer science (CS)/information systems (IS) "sub-disciplines" have successfully integrated OO technology into their respective areas of research and development as a new approach to problem solving. This is evident from the use of the burgeoning OO technology in the areas of information system analysis and design [4, 5, 6], database management systems [7, 8], operating systems [9, 10, 11], and programming languages [12, 13], just to name a few.

Object-orientation has also been integrated into CS and IS curricula in the last few years [14, 15, 16, 17, 18, 19]. While object-orientation has been steadily incorporated into the curricula, the incorporation deals mostly with OO programming and to a lesser extent, OO analysis and design. Few have described object-orientation with respect to databases even though database courses are typical common requirements in both CS and IS curricula. This lack of interest is somewhat predictable since object-oriented database management systems (OODBMSs) have yet to make any headway in the mainstream computing industry where relational DBMSs are still predominant.

However, it is the opinion of this author that given the significant level of research and development work on OODBMS in the past few years, the topic of object-orientation deserves an adequate coverage in a database course. This position is further strengthened when one looks at how commercial OODBMSs have gained their own niche in the areas of scientific, multimedia, and geographic databases in the industry (e.g., CAD/CAM, GIS). It has been reported that OODBMS is projected to be a $425 million industry by 1996 [20]. Moreover, it is important to realize that as OO programming becomes increasingly popular, the demand for persistent objects (i.e., those that transcend over time and space) will increase and OODBMSs will be in the right place to serve the needs as they are the only repository designed to store objects. Having taken this position, this paper describes the integration of object-orientation into an undergraduate database course and documents the experience in the process.

The remainder of this paper is organized as follows. Section 2 gives background information on OODBMS and discusses some related work. The integration of object-orientation into a DBMS course is discussed in Section 3. Finally, Section 4 gives the summary and conclusions.

BACKGROUND AND RELATED WORK

OODBMS: The definition

While there is a lack of standards with regard to OODBMS concepts (with the exception of the effort by Atkinson et. al. [21] and the ongoing work by Object Management Group, a standards organization), it is commonly agreed that OO databases should have the characteristics of both object-orientation and databases. Object-orientation includes encapsulation through abstract data types (classes), complex objects, inheritance, polymorphism, and object identity. Database capabilities include the traditional features like persistence, concurrency control, recovery, querying, integrity, and security.

Atkinson et al. describe the following golden rules as the mandatory features of an OODBMS:

1) Complex objects (Thou shalt support complex objects)
2) Object identity (Thou shalt support object identity)
3) Encapsulation (Thou shalt encapsulate thine objects)

4) Types and Classes (Thou shalt support types or classes)
5) Class or Type Hierarchies (Thine classes or types shalt inherit from their ancestors)
6) Overriding, overloading, and late binding (Thou shalt not bind prematurely)
7) Computational Completeness (Thou shalt be computationally complete)
8) Extensible (Thou shalt be extensible)
9) Persistence (Thou shalt remember thy data)
10) Secondary storage management (Thou shalt manage very large databases)
11) Concurrency (Thou shalt accept concurrent users)
12) Recovery (Thou shalt recover from hardware and software failures)
13) Ad Hoc Query Facility (Thou shalt have a simple way of querying data)

The first eight of the rules are from an object model and the last five are features typically found in a DBMS.

Integrating Object-Orientation into a CS/IS Curricula

As the popularity of object-orientation continues to grow in the industry, more and more CS/IS departments are adopting the new paradigm in their curricula (albeit slowly). As mentioned above, much of the integration of object-orientation happens at the programming language level. That is, instead of using procedural languages such as Pascal, C, or COBOL, OO programming languages like C++ and Smalltalk are used to teach the introductory programming courses (e.g., [14]).

This language-level adoption also applies to upper division courses that make use of the "bells and whistles" of OO development environments and the tools that go with them (e.g., [18]). This type of course usually comes in the form of a special topics class, such as Introduction to OO Programming, newly added to the curriculum. Another form of language-level adoption appears in the traditional programming languages course where different programming paradigms like procedural, logic, functional, and more recently, object-oriented paradigms are discussed.

Another type of adoption that has surfaced recently is in the form of software development methodology. This adoption can be realized in a software engineering course where the OO paradigm is studied together with the traditional structured methodology or by itself with the structured methodology course as a prerequisite. It should, however, be noted that due to the OO methodology war [22], methodology-level adoption is not as widespread as language-level adoption.

Lastly, object-orientation can also be integrated into the curriculum in a database course. This database-level adoption, the topic of this paper, is detailed in the next section.

OO DATABASES IN CS/IS CURRICULA

OO databases are introduced in the author's department through an undergraduate introductory DBMS course called Database Processing. This course, required in the IS sequence and an elective in the CS sequence , is aimed at junior, senior, and graduate students wanting to learn the fundamentals of a DBMS as well as the new and emerging technologies like OO and client-server databases. The course has two prerequisites — Data Structures and System Development I.

"As the popularity of object-orientation continues to grow in the industry, more and more CS/IS departments are adopting the new paradigm in their curricula."

The majority of the database course is structured around the traditional set of topics such as conceptual modeling, query languages, transaction processing, recovery, integrity, etc. However, unlike a traditional undergraduate DBMS course, this one contains a knowledge unit that discusses object-orientation and how it affects the database technology. (Because OODBMS is yet to be considered mainstream, it does not yet deserve to be a course by itself in an undergraduate curriculum. It is, however, a viable advanced topic graduate course, as is commonly seen in graduate curricula.) Approximately two weeks of a 16-week DBMS course were used in incorporating the OO knowledge unit. Two different OODBMSs, Postgres and SOD, were used in the integration process. They are described next, followed by a sample outline for the 2-week module.

Postgres

Like most institutions, the department cannot afford or justify having a commercially available OODBMS for teaching purposes. For that reason, Postgres (for Post Ingres) [23], a prototype DBMS available through ftp from the University of California at Berkeley, has been used in the exploration of OODBMS characteristics. Postgres builds on the popular Ingres relational database system and extends the relational core with OODBMS features such as inheritance, complex objects, and methods. These features permit the instructor to illustrate the fundamentals of object-orientation and give the students a feel for what to expect when facing a next generation DBMS. It should be noted that Postgres does not run on top of Ingres (i.e., it is standalone) and that it has a separate graphical front-end called Picasso, downloadable through ftp as well, that runs on X Windows if GUI programming on X for Postgres is desired.

Representative examples of some of the illustrations [25] presented in the course are given below. The query language used in Postgres is called PostQUEL, an extension of QUEL, the query language of Ingres.

Example 1 (inheritance):

```
create Person (name = char(16),¬
 age = int4, location = point)
create Employee (salary = int4,¬
 manager = char(16)) inherits (person)
append Person (name = "Billy",¬
 age = 31, location = "(3.1, 6.2)")
append Employee (name = "Sharon",¬
 age = 20, location = "(1.1, 2.2)",¬
 salary = 1000, manager = "Billy")
```

[Publisher's note: Due to publication format restrictions the "¬" is used here to denote the continuation of the code onto the next line.]

This example illustrates how a type/class can inherit the attributes from another type/class. Here, the class `Employee` inherits the attributes from the class `Person` and the second append statement, which works like the INSERT statement in SQL, shows how the inherited attributes are used.

Example 2 (complex object):

```
create City (name = char(16),¬
 location = box, budget = city_budget)
append City (name = "Berkeley",¬
 location = "(0.0,0.0,10.0,10.0)",¬
 budget = "250, 300, 325, 275")
retrieve (City.name) where¬
 City.budget[1] > City.budget[2]
```

This example shows the use of complex objects to define non-primitive data types. Here, attribute `location` is of type box, a quadruple defining the two coordinates of a rectangle/box. The `budget` attribute is of type `city_budget`, an array of budgets for the four

quarters. The retrieve statement gives an example of how the values of the budget array can be used, namely it retrieves all the cities (i.e., the city names) where the first quarter budget is greater than the second quarter's.

Example 3 (method):

```
retrieve (City.name) where boxarea¬
 (City.location) > 100
retrieve (Employee.name) where¬
 overpaid(Employee)
```

This example demonstrates how methods (preloaded, compiled C functions) can be used in the querying process. Here, the method boxarea returns the area a given city occupies and the method overpaid tells if an employee is overpaid (based on some formula). Thus, the two retrieve statements retrieve the cities with area greater than 100 and employees who are overpaid, respectively.

As can be seen above, the examples together with the fundamentals of object-orientation enable the students to grasp the basics of OODBMS while still in the confine of an introductory undergraduate DBMS course. This exposure to object-orientation in Postgres running on Unix together with hands-on experience with a real world DBMS, i.e., IBM DB2 on MVS, currently used in all of the course activities, gives the students an added advantage when competing for jobs in the market place or a head start on OO technology if they are going to graduate schools.

SOD

Experience with OODBMS may also be gained with a home-grown, PC-based database system, called SOD (Simple Object Database) [26], that has been developed to aid the integration process.

SOD is an experimental OODBMS project designed and developed to serve two purposes. First, the project provides great opportunity to gain a deeper understanding of OODBMS. It allows one to know more about potential implementation difficulties of an actual OODBMS and to be more aware of the potential applications of such a system. Second, it is hoped that others can use this system as an educational tool.

Given the great expense of acquiring OODBMS technology most students are learning about OODBMS only through textbook readings. However, to teach the concepts effectively, a combination of textbook readings and experiences with an OODBMS is a must. SOD is intended to be used as a teaching tool to fill this gap. The aforementioned Postgres system certainly does the job as well but it is a fairly large system that does not permit students to bring it home to their own PCs and perform experimentation with it. (Most students do not have systems that run Unix and even if they do, they are typically not powerful enough to run a system like Postgres.) A PC-based system like SOD represents an ideal mix of platforms (DB2 on mainframe MVS, Postgres on Unix workstation, and SOD on PC DOS) where students are exposed to a variety of flavors of DBMSs that they are equally likely to encounter upon graduation.

The SOD system supports many OO features (and more) including

1) class definition
2) (single) inheritance
3) multivalue attributes
4) object identity
5) complex objects
6) behaviors
7) querying of database by user-specified conditions
8) menu-driven graphical user interface

It is developed using the FoxPro language in the FoxPro 2.0 environment and has the look and feel of the environment. A detailed description of SOD and how it can be used in a database course may be found in [26].

Sample Module

Given the vast number of conventional topics (e.g., concurrency control, SQL, query optimization, etc.) that need to be covered in a undergraduate database course, one has to make difficult decisions as to what topics to remove in order to make room for an OO module. In the database course described here, coverage of all non-relational systems are removed. That is, topics of hierarchical (e.g., IMS) and network (e.g., IDMS) databases are only discussed in a "Types of Data Models" lecture and no detailed coverage is given to any of the two. It is felt that in today's market place, graduates are most likely to encounter relational systems at work. Furthermore, while it is true that the demand for knowledge of IMS is likely to be more than that of ObjectStore or Gemstone (some of the popular OODBMSs), it is believed that the general knowledge of object-orientation coupled with exposure to OODBMS characteristics will, in the long run, outweigh any shortcomings for not covering topics like IMS.

There are a variety of activities that can be covered in a 2-week OO module. A week can, and should, be spent on introducing the fundamentals of OO paradigm and the major differences between relational and OO databases. This can be done with "standard" introductory materials found in most OO databases textbooks (e.g., [7], [8]) videotapes about objects and OODBMSs (e.g., [27], [28]), and/or materials available in the public domain designed for this purpose (e.g., [29]).

With the 1-week introduction, the second week of the module can be devoted to the coverage of more in-depth use, analysis, design, and/or implementation of OODBMSs. Depending on the background of the students, this can range from the studying of how a new generation DBMS like Postgres handles new requirements (see examples 1-3 in earlier section for sample activities) to a more ambitious one that involves studying how C++ and Smalltalk objects can be made persistent in an OODBMS. The latter can convince the students that unlike extended relational systems, objects created in an OO environment need not be "taken apart" and mapped into relational tables. Namely, the potentially complex structure of objects can be preserved when stored on disks and when retrieved, no massive amount of joins are needed to re-construct the objects.

In the course described here, neither C++ nor Smalltalk was discussed due to the background of students. However, students were encouraged, with bonus credits, to try out the systems discussed in class with simple exercises that involve creating tables with non-traditional data types and querying the database with methods. In the foreseeable future, since the curriculum in the author's department has been overhauled, i.e., students are more likely to have C++/Smalltalk as prerequisites, the possibility of introducing persistent objects for C++ or Smalltalk may be realized.

SUMMARY AND CONCLUSIONS

Object-orientation has steadily penetrated the software industry and the CS/IS curricula in recent years. While the penetration on the industry side has been fairly uniform in all areas of object-orientation (i.e., OO programming, analysis and design, databases), the level of penetration has not been the same for the curricula, especially for OO databases. This paper takes the position that while OODBMSs are still relatively immature and are not mainstream DBMS yet, they deserve coverage in the undergraduate CS/IS curricula. This coverage can be given in the traditional DBMS class with a knowledge unit in object-orientation.

The integration involves introducing object-orientation as a knowledge unit and it makes use of two prototype DBMSs that support object-orientation as the vehicles. The exposure to object-orientation together with the experience with DBMSs on different platforms put the students in a very advantageous position when competing for jobs in the market place. It should be emphasized that the OO module described here does not prepare

one to immediately participate in OO projects without further training. Since not many CS/IS curricula expose students to state-of-the-art technologies such as OO technology, for recruiting companies that are cutting edge, an OO module like the one described can go a long way in helping a student clinch a job offer.

As future work, other aspects of OODBMS and how they can be integrated into the curriculum, especially the graduate program, are being investigated. Issues like long transactions, versioning, object integrity, and OSQL (object SQL) are beyond the scope of an undergraduate DBMS course and are potential topics for a graduate advanced DBMS class.

REFERENCES

[1] Executive Summary, *Object Magazine*, July-August, Vol. 2, No. 2, 1992.

[2] Annual ACM International Conference on Object-Oriented Programming: Systems, Languages, and Applications, 1986-1993. Also as ACM SIGPLAN Notices.

[3] Cover Story: "Software Made Simple: Object-Oriented Programming," *Business Week*, September 30, 1991.

[4] Booch, G., *Object-Oriented Design with Applications*, 2nd Edition, Benjamin-Cummings, 1994.

[5] Rumbaugh, James., Blaha, Michael., Premerlani, William., Eddy, Frederick., Lorensen, William., *Object-Oriented Modeling and Design*, Prentice-Hall Inc., 1991.

[6] Jacobson, Ivar., *Object-Oriented Software Engineering*, Addison-Wesley, 1992.

[7] Maier, D., Zdonik, S., *Readings in Object-Oriented Database Systems*, Morgan-Kaufmann, 1990.

[8] Cattell, R., *Object Data Management: Object-Oriented and Extended Relational Database Systems*, Addison-Wesley, 1991.

[9] *OS/2 2.1 Operating System*, IBM Corporation, 1993.

[10] *Pink Object-Oriented Operating System*, Apple-IBM joint development, Expected 1994.

[11] *NextStep 3.1*, NeXT Computer, Inc., 1993.

[12] Ellis, Margaret., Stroustrup, Bjarne., *The Annotated C++ Reference Manual*, Addison-Wesley, 1990.

[13] Goldberg, Adele., *Smalltalk-80: The Language*, Addison-Wesley, 1989.

[14] Pugh, J. R., LaLonde, W. R., Thomas, D. A., "Introducing Object-Oriented Programming into Computer Science Curriculum," *18th SIGCSE Technical Symposium on Computer Science Education*, St. Louis, MI, February, 1987.

[15] Temte, M. C., "Let's Begin Introducing the Object-Oriented Paradigms," *22nd SIGCSE*, San Antonio, TX, March, 1992.

[16] Cain, William., "Object-Oriented Programming and the CIS Curriculum," *Journal of Information Systems Education*, Vol. 3, No. 1, 1991. p 2-7.

[17] Waguespack, Les., "Object Orientation in the IS Curriculum," *Information System Education Conference*, Phoenix, AZ, 1993.

[18] Lim, Billy B. L., "A Project-Intensive Introductory Object-Oriented Programming Course," *ISECON '93*, Phoenix, Arizona, November, 1993.

[19] Lim, Billy B. L., "Integrating Object-Orientation Into an Undergraduate CS Curriculum," *Journal of Computing for Small Colleges*, Feb, 1995.

[20] *IBM Object-Oriented Database Strategy*, IBM Santa Teresa Lab, 1992.

[21] Atkinson et. al. "Object-Oriented Database Systems Manifesto," *Proc. of the 1st International Conference on Deductive and Object-Oriented Databases*, North-Holland, 1991.

[22] Which Method Is Best? Shoot Out at the OO Corral, Panel Discussion, *International Conference on Object-Oriented Programming: Systems, Languages, and Applications*, Washington, D. C., 1993.

[23] Stonebraker, Michael., Kemnitz, Greg., "The Postgres Next Generation Database Management System," *Communication of the ACM*, Vol. 34, No. 10, 1991.

[24] Stonebraker, Michael. et al., "Third Generation Data Base Manifesto," *SIGMOD Record*, 1991.

[25] *Postgres Reference Manual*, University of California at Berkeley, 1991.

[26] Chang, C., Lim, Billy B. L., "Integrating Object-Oriented Technology into an Undergraduate Database Course Using SOD: A Simple Object Database," *ISECON '93*, Phoenix, Arizona, November, 1993.

[27] *The World of Objects*, Borland International, 1992.

[28] Loomis, Mary, Object Database Technology session, Building the Enterprise Through Object Technology, Hewlett-Packet Inc., 1994.

[29] Object-Oriented Databases section, *Object-Orientation Across Undergraduate CS/IS Curricula*, NSF Undergraduate Faculty Enhancement grant, 1994.

Billy B. L. Lim, Ph.D
Applied Computer Science Department
Illinois State University
Normal, IL 61790-5150

Dr. Billy Lim is an assistant professor at the Applied Computer Science Department of Illinois State University. His research and teaching interests include new generation database systems, object-oriented systems development, and user interface design. He has been very active in object technology training and education and has recently conducted NSF-sponsored workshops to integrate object-orientation into undergraduate CS/IS curricula.

Volume 7, Number 3

The Journal of Information Systems Education is published four times each year by EDSIG, the DPMA's special interest group for education. Papers published in the Journal are submitted and accepted through a formal, refereed process. Submissions for other sections are also welcomed. For more information about submissions, contact the appropriate editor.

Copyright © 1995, EDSIG
The Data Processing Management Association's
Special Interest Group for Education

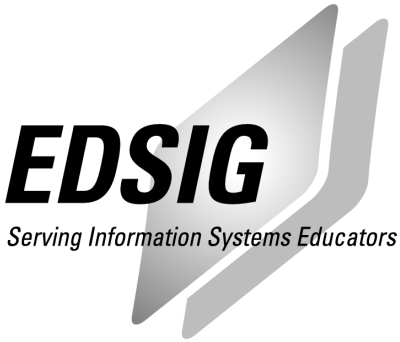
All rights reserved including those of translation. The opinions expressed by authors in the Journal of Information Systems Education are their own and do not necessarily represent those of DPMA and/or EDSIG. Letters addressed to the editor are considered submitted for publication unless the writer states otherwise. Reproduction of the Journal of Information Systems Education, in whole or part, without written permission of EDSIG, is strictly prohibited.

For information about permissions, reprints, advertising, membership, subscription and change of address, contact the publisher:

creative perspectives
310 East Main Street, Suite 220
Charlottesville, VA 22901
1-804-971-6795
creatvpers@aol.com

Reprinted from the Fall 1995 issue of the Journal of Information Systems Education.

Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

Copyright ©1995 by the Information Systems & Computing Academic Professionals, Inc. (ISCAP). Permission to make digital or hard copies of all or part of this journal for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial use. All copies must bear this notice and full citation. Permission from the Editor is required to post to servers, redistribute to lists, or utilize in a for-profit or commercial use. Permission requests should be sent to the Editor-in-Chief, Journal of Information Systems Education, editor@jise.org.

ISSN 1055-3096