4-27-1998

Trends in teaching object-oriented programming at the Community College level

Thomas W. Fargnoli
Rowan University

TRENDS IN TEACHING OBJECT-ORIENTED PROGRAMMING AT THE COMMUNITY COLLEGE LEVEL

by
Thomas W. Fargnoli

A Thesis

Submitted in partial fulfillment of the requirements of the
Master of Arts Degree in Computer Science Education
in the Graduate Division of Rowan University
1998

Approved by

Dr. John Sooy

Date Approved April 27, 1998

ABSTRACT

Thomas W. Fargnoli, Trends in Teaching Object-Oriented Programming at the Community College Level, 1998, J. Sooy, Computer Science Education

The purpose of this study was to determine the current trends in teaching object-oriented programming at the community college level. A questionnaire was developed and sent to all community colleges in the state of New Jersey to assess the extent of object-oriented material offered in computer science courses. Nine community colleges participated in the study.

All nine colleges reported offering at least one course containing some degree of object-oriented material. Of the eighty five computer science courses offered, thirty of them (35%) contained some degree of object-oriented material. The degree of the object-oriented material ranged from 5% to 100%. Fourteen of the thirty courses (48%) contained at least 50% of objected-oriented material. Eight of the thirty courses (27%) were reported to have contained 100% object-oriented material. Colleges that reported the most object-oriented content had recently revised their computer science curriculum.

The conclusions from this study indicated that the majority of computer science courses in the community college do not currently contain object-oriented material and the migration to an object-oriented paradigm is occurring slowly. The study also revealed a correlation between the computer science faculty's experience in object-oriented programming and the degree of object-oriented material offered in the curriculum.

MINI-ABSTRACT

Thomas W. Fargnoli, Trends in Teaching Object-Oriented Programming at the Community College Level, 1998, J. Sooy, Computer Science Education

A questionnaire developed to assess the extent of object-oriented material offered in computer science courses at the community college level was administered to nine community colleges in New Jersey. The study showed that the majority of computer science courses do not currently contain object-oriented material. The study also revealed a correlation between the computer science faculty's experience in object-oriented programming and the degree of object-oriented material offered in the curriculum.

ACKNOWLEDGMENTS

I would like to express my appreciation to my graduate advisor, Dr. John Sooy, for his time and assistance throughout this project.

I would also like to offer special thanks to my wife, MaryEllen, for her support and encouragement, and to my children, Tom and Janine, for their patience and understanding.

I am also grateful to the music of Andrea Bocelli. His voice soothed my nerves through a very eventful academic year.

TABLE OF CONTENTS

FIGURES

Figure

TABLES

Table

CHAPTER 1

Introduction to the Study

Introduction

Object-oriented programming offers a new and powerful model for writing computer software.[1] This approach speeds the development of new programs, and, if properly used, improves the maintenance, reusability, and modifiability of software. Learning object-oriented programming however, requires a major shift in thinking from the traditional procedural programming disciplines. The most difficult problem in teaching object-oriented programming is getting the learner to give up the global knowledge of control that is possible with procedural programs, and rely on the local knowledge of objects to accomplish their tasks.[2] This chapter presents an introduction to the study of determining the current trends in teaching object-oriented programming at the community college level.

Background

Over the last ten years there has been a major shift in programming language design from procedural languages to object-oriented languages.[3] This is in response to the increasing software maintenance costs and backlog that are prevalent in the

software industry. The recent commercialization of object-oriented software technologies has been driven by pragmatic desires to increase productivity, shorten cycle times, enhance maintainability and extensibility, and more fully satisfy user requirements.[4] Based on this trend, people trained in object-oriented design and implementation will be in demand as the industry continues the shift from a procedural to an object-oriented paradigm. Rather than try to make object design as much like procedural design as possible, the most effective way of teaching how to think with objects is to immerse the leaner in the "object-ness" of the material. To accomplish this, as much familiar material as possible had to be removed.[5]

Problem

This study is to determine the current trends in teaching object-oriented programming at the community college level.

Significance of the Problem

Object-oriented programming allows the reuse of software components across programs, thus decreasing the software maintenance costs that are prevalent in the software industry. Consequently, object-oriented technology has become one of the dominant technologies in the computing industry. Seventy five percent of the Fortune 100 companies have adopted object technology to some degree for their computing needs.[6] By the year 2000, nearly all programmers will be writing object-oriented programs and extending class libraries.[7]

Despite industry's desire to embrace object-oriented programming for overall cost reductions and increased productivity, it is faced with the difficult task of making a transition from a procedural paradigm to an object-oriented paradigm. This transition has many barriers. A recent study, detailed in the October 1997 journal *Computer*, sites obstacles in companies making this transition. The study tracked four companies that decided to transition to object-oriented technology. As a result of their study, one of the main obstacles in a company adopting object-oriented technology was related to the difficulty in *learning* object-oriented technology.[8] The reason for this is because object-oriented technology is more than just a way of programming. It is a way of thinking abstractly about a problem using real world concepts, rather than computer concepts. This may be a difficult transition for some people because older programming languages force one to think in terms of the computer and not in terms of the application.[9] It was observed that it may be better to hire fresh people, untainted by years of procedural thinking, than to re-train the existing programming staff.

Based on industry's demand and on the problems associated with the transition from a procedural paradigm to an object-oriented paradigm, interest in teaching object-oriented programming in first year computer science courses has increased substantially over the last few years.[10]

Limitations

Since this study is focused on determining the current trends in teaching object-oriented programming at the community college level, the data collected will be limited to the community college curriculums. The mission of the community college is to fulfill both career preparation and transfer functions and it often provides the first look at computer programming for a student.

This study is limited to community colleges in the state of New Jersey.

Definitions

The following definitions are taken from Object-Oriented Modeling and Design, (Rumbaugh, Blaha, Premerlani, Eddy, and Lorenson. Englewood Cliffs, New Jersey: Prentice Hall, 1991) and What is Object-Oriented Software? An Introduction, (Montlick, Terry, *Software Design Consultants*, 1997 -http:/www.softdesign.com/softinfo/objects.html)

Classification Objects with the same data structure (*attributes*) and behavior (*operations*) are grouped into a class. *Paragraph, Window, and ChessPiece* are examples of classes. Each object is said to be an instance of its class. An object is defined via its class, which determines everything about an object.

Encapsulation Providing access to an object only through its messages, while keeping the details private is referred to as encapsulation (also known as *information hiding).*

Inheritance Inheritance is the sharing of attributes and operations among classes based on a hierarchical relationship. A class can be defined broadly and then refined into

successively finer *subclasses*. Each subclass incorporates, or *inherits*, all of the properties of its *superclass* and adds its own unique properties. For example, *ScrollingWindow* and *FixedWindow* are subclasses of *Window*. Inheritance also promotes reuse. You don't have to start from scratch to write a new program.

Message Messages define the interface to the object. All communication to and from objects are done via messages. The object which a message is sent to is called the *receiver* of the message.

Object An object is a discrete, distinguishable entity. A *paragraph in a document, a window on my workstation, and the white queen in a chess game* are examples of objects. Objects can be concrete, such as a file in a file system, or conceptual, such as a scheduling policy in a multiprocessing operation system. Each object has its own inherent identity.

An object is a "black box" which receives and sends messages. A black box contains code (sequences of computer instructions) and data (information which the instructions operate on).

Object-Oriented The term "object-oriented" means that we organize software as a collection of discrete *objects* that incorporate both data structure and behavior.

OOPSLA Object-Oriented Programming Systems, Languages, and Applications. OOPSLA is an annual conference for disseminating new object-oriented ideas and application results.

Polymorphism Polymorphism means that the same operation may behave differently on different classes. The *move* operation, for example, may behave differently on the *Window* and *ChessPiece* classes. An operation is an action or transformation that an object performs or is subject to. A specific implementation of an operation by a certain class is called a method. Because an object-oriented operator is polymorphic, it may have more than one method implementing it.

Procedures

Since the purpose of this study is to determine the trends in teaching object-oriented programming at the community college level, a questionnaire will be created and distributed to community colleges in the state of New Jersey. As an introduction to this study, a brief introduction to object-oriented technology, related research, and purpose of the study will accompany the questionnaire.

The questionnaire's purpose will be twofold: First, to assess the current computer science curriculum with regard to computer programming courses offered at the school, and second, to assess the current knowledge of object-oriented technology and to what extent that technology is represented in the curriculum.

ENDNOTES

[1] Montlick, Terry, What is Object-Oriented Software? An Introduction, (*Software Design Consultants*, 1997 - http:/www.soft-design.com/softinfo/objects.html), 7

[2] Beck, Ken, Apple Computer, Inc., A Laboratory For Teaching Object-Oriented Thinking, (*OOPSLA*'89 Conference Proceedings October 1-6, 1989, New Orleans, Louisiana and the special issue of *SIGPLAN* notices Volume 24, Number 10, October 1989 - http//c2.com/doc/oopsla89/paper.html), 1

[3] Kolling, Michael and Rosenberg, John, An Object-Oriented Program Development Environment for the First Programming Course, (Proceedings of the 27th *SIGCSE* Technical Symposium on Computer Science Education, March 1996), 83-87

[4] Fishman, Robert G., and Kemere, Chris F., Object Technology and Reuse: Lessons from Early Adopters, *Computer*, October, 1997), 47

[5] Beck, Ken, Apple Computer, Inc., A Laboratory For Teaching Object-Oriented Thinking, (*SIGPLAN* notices Volume 24, Number 10, October 1989 - http//c2.com/doc/oopsla89/paper.html), 1

[6] NSF Sponsored Workshop, Illinois State University, Object-Orientation Across Undergraduate Computer Science Curricula, Program Grant Number DUE-9455119 (June 3 - June 14, 1996 - http:/www.cs.ilstu.edu/oopoverview.html), 1

[7] Wirfs-Brock, Rebecca and Wilkerson, Brian, Object-Oriented Design: A Responsibility-Driven Approach, (http//www.vpplus.com/obj1.html), 3

[8] Fishman, Robert G., and Kemere, Chris F., Object Technology and Reuse: Lessons from Early Adopters, *Computer*, October, 1997), 48

[9] Rumbaugh, Blaha, Premerlani, Eddy, Lorenson, Object-Oriented Modeling and Design. (Englewood Cliffs, New Jersey: Prentice Hall, 1991) ix

[10] Kolling, Michael, Koch, Bett, and Rosenberg, John, Requirements for a First Year Object-Oriented Teaching Language, (*SIGCSE* Bulletin, Vol. 27, No. 1, March 1995), 173-177

CHAPTER 2

Review of Related Literature and Research

Introduction

Chapter Two presents related literature and research in the area of teaching object-oriented programming. The related literature supports industry's stance on why we should embrace object-oriented technology and consequently, why we should incorporate it early in the computer science curriculum. The related research is focused on effective methods of teaching object-oriented programming, particularly with the novice in mind.

Related Literature

The recent commercialization of object-oriented software technologies has been driven by pragmatic desires to increase productivity, shorten cycle times, enhance maintainability and extensibility, and more fully satisfy user requirements.[1]

"Object-Oriented Modeling and Design" (Rumbaugh, Blaha, Premerlani, Eddy, Lorenson) focuses on object-oriented modeling and design as a new way of thinking about problems using models around real-world concepts. They cite their

experiences and pertinent studies to show evidence for usefulness of object-oriented development. In this regard, they have used object-oriented techniques for developing compilers, graphics, user interfaces, simulations, meta models, control systems, and other applications. They have used object-oriented models to document existing programs that are ill-structured and difficult to understand. They are enthusiastic supporters of object-oriented development and see no reason it should not be used on most software projects.

The annual OOPSLA conferences describe many applications that have benefited from an object-oriented approach. The studies and applications cited include developing an object-oriented operating system, a statistical analysis program, a large medical application, and signal processing applications. They report that the main benefit is not reduced development time; object-oriented development may take more time than conventional development, because it is intended to promote future reuse and reduce downstream errors and maintenance.[2]

Because object-orientation is becoming one of the primary means for problem solving, the need to teach object-orientation in undergraduate curriculum is growing.[3] This was the premise for a workshop supported by the National Science Foundation for Undergraduate Faculty Enhancement. The purpose of this project was to provide a two-week summer workshop for faculty who did not have any formal training in this area. The objectives were to introduce object-oriented concepts and to demonstrate how to deliver effective courses and units on object-orientation.

In a paper by Joseph Bergin, entitled "Teaching Object-Oriented Analysis and Design in CS 1", he points out that, typically, courses that introduce computer science focus on programming. Analysis and Design (A/D) is often ignored in the CS curriculum except for the Software Engineering course. There is some evidence that this is inadequate. (Employer dissatisfaction with recent graduates' ability in this area.) One of the advantages of the OT (Object Technology) approach is that there is a smaller divide between A/D, on the one hand, and programming on the other.[4] The tools and techniques are not as different as they are when standard structured methodologies are applied. This gives us hope for an affirmative answer to the question at hand. Most OT practitioners recommend a spiral approach to development, in which a functional subsystem is delivered to users and used as the basis for further analysis. The system grows through interaction between the users, designers, programmers, always with a growing, working system to use as a reference point. This avoids the problem that sometimes occurs with older technologies in which the system is finally delivered complete, but after a long delay, it no longer meets the needs that have evolved in the interim.

Related Research

Robert G. Fichman and Chris F. Kemerer site obstacles in companies making the transition from a procedural paradigm to an object-oriented paradigm. The study tracked four companies that decided to transition to object-oriented technology. As a result of their study, one of the main obstacles in a company adopting object-oriented technology was related to the difficulty in *learning* object-oriented technology. The reason for this is because object-oriented technology is more than just a way of programming. It is a way of thinking abstractly about a problem using real world concepts, rather than computer concepts.[5] This may be a difficult transition for some people because older programming languages force one to think in terms of the computer and not in terms of the application.

In a paper entitled "Requirements for a First Year Object-Oriented Teaching Language" , Michael Kolling, Brett Koch and John Rosenberg report that interest in teaching object-oriented programming in first year computer science courses has increased substantially over the last few years. They contend that while the theoretical advantages are clear, it is not obvious that the available object-oriented languages are suitable for this purpose.[6] The paper discusses the requirements for an object-oriented teaching language and draws attention to the deficiencies of existing languages.

This study was followed by a subsequent study introducing a new language called "Blue". Blue is a new language and integrated programming environment currently under development explicitly for object-oriented teaching to first year

students. The second paper, entitled Blue - A Language for Teaching Object-Oriented Programming, by Michael Kolling and John Rosenberg, appeared in the proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education, March 1996.

According to this study, most people agree that Pascal was a wonderful tool. The problem with Pascal today is that it is based on an outdated programming paradigm. The problem with object-oriented languages is that they are not made for teaching. Blue tries to bring these two aspects together and aims at being the objected-oriented equivalent of Pascal.

A paper entitled "A Laboratory for Teaching Object Oriented Thinking" was presented at the OOPSLA'89 Conference and appeared in the SIGPLAN journal (Volume 24, Number 10, October 1989). The study details a successful approach to teaching object-oriented design to both novice and experienced programmers.

Procedural designs can be characterized at an abstract level as having processes, data flows, and data stores, regardless of the implementation language or operating environment. In their study, they came up with a similar set of fundamental principles for object designs. They settled on three dimensions which identify the role of an object in a design: *class name, responsibilities, and collaborators*. The class name of an object creates a vocabulary for discussing a design, responsibilities identify problems to be solved, and collaborators show the relationships to other objects. CRC (for Class, Responsibility, and Collaboration) cards are used to understand objects and their behaviors.

One of the contexts in which the authors have used CRC cards is in a three-hour class entitled "Thinking with Objects," in which a data flow example is introduced (a school, with processes for teaching and administration) which is then recast in terms of objects with the responsibilities and collaborators (such as Teacher, Janitor, and Principle). The class then pairs off and spends an hour designing the objects in an automated banking machine.

The study concludes that CRC cards give the learner who has never encountered objects a physical understanding of object-ness, and prepares them to understand the vocabulary and details of particular languages. When learners pick up an object, they seem to more readily identify with it and are prepared to deal with the remainder of the design from its perspective.[7]

Figure 2.1 illustrates a CRC card for the "transaction" function (or object) of an automated banking machine. The class name is underlined (Transaction), the responsibilities of the class are listed under the class name (Validate & Perform money transfer, Keep audit info.), and the class collaborators are listed on the right side of the card (CardReader, Dispenser, RemoteDB, Action & Account).

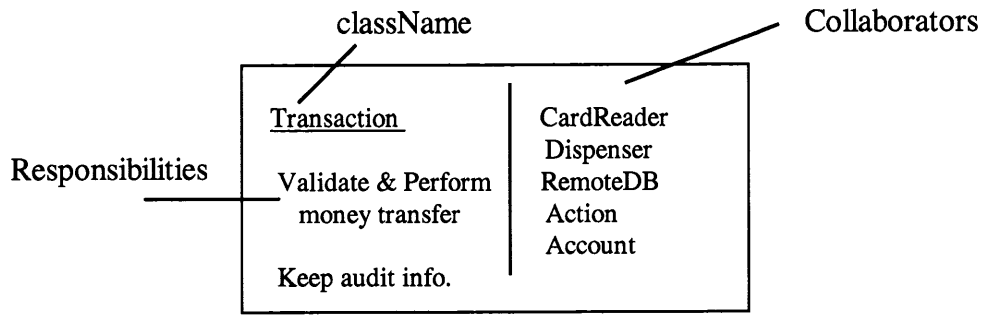

Figure 2.1

A Class-Responsibility-Collaborator (CRC) Index Card

John Traxler, School of Computing and Information Technology at the University of Wolverhampton, England, 1994, published a paper on "Teaching Programming Languages and Paradigms". For several years, final-year undergraduates in computer science and software engineering at the University of Wolverhampton have been offered a module introducing them to the issues of programming language design and encourages them to reflect on their experience of procedural programming. It also introduces them to the idea of programming paradigms and to the specific principles and practices of functional, concurrent and object-oriented programming. The paper by John Traxler draws on their responses and discusses some of the issues raised in this kind of comparative study. These issues include:

a) The extent to which students' previous experience and training skews their understanding of new programming paradigms and languages.

b) The extent to which the University's choice of a base teaching language, initial programming paradigm and programming style affects the students' subsequent progress in other languages and paradigms.

c) The relationship between the paradigms with their respective languages, on the one hand, and appropriate formal methods and analysis and design activities, on the other.

d) The need to address the industrial topicality, the relevance and credibility of programming styles and languages in an academic context.

The conclusions of this paper, based on the feedback from the students, show that students arrive at the university with a very mixed background of programming experience and subsequently, may introduce a bias into their approach to new forms of programming.[8]

CHAPTER 3

Procedures

Introduction

Chapter three presents the fundamental concept and methods used in the collection and analysis of data to assess the current trends in teaching object-oriented programming at the community college level. The procedures are focused on assessing the percentage of object-oriented programming currently being taught at the community college level. The research methodology is presented in three stages: Population, Development and Validation of the Questionnaire, and Procedures. The first phase of the research, Population, describes the group that received the survey. The next section, Development and Validation of the Questionnaire, describes the development of the questionnaire and the validation of this research instrument. The chapter concludes with a section on Procedures which describes how the questionnaire was distributed and how the responses were collected.

Population

A questionnaire was sent to all nineteen community colleges in the state of New Jersey. All community colleges in New Jersey were selected. Nine of the nineteen

colleges responded to the survey. Appendix B lists each of nine community colleges that were included in the study.

Development and Validation of the Questionnaire

The questionnaire (see Appendix A) was developed to determine the trends in teaching object-oriented programming at the community college level. In particular, the questionnaire was developed to assess the extent of object-oriented material offered in computer science courses at the community college level. The questionnaire consists of three questions. Question one is a close-form question designed to obtain the majority of the research material. It is directly concerned with the current trends in teaching object-oriented programming. It queries the respondent for the title of each computer science course offered at their college and the percentage of object-oriented material contained in that course. Questions two and three are open-form questions designed to gain insight as to how the college perceives teaching object-oriented programming at the introductory level.

The questionnaire was validated by the jury method. This validation method, conducted by a panel of three research colleagues and the author's thesis advisor, involved rating the survey in terms of how effectively it samples significant aspects of its purpose.

Procedures

A cover letter (see Appendix A) was drafted by the author providing a brief introduction to object-oriented technology, related research, and purpose of the study. The cover letter and accompanying questionnaire were mailed to all community colleges in New Jersey. The letters were addressed to the Chairperson of the Computer Science Department of each college. The cover letter, questionnaire, and stamped, addressed return envelope were mailed on January 19, 1998.

On February 23, 1998, follow-up calls were made to the colleges that did not respond to the questionnaire. The purpose of the calls was to determine if the college received the questionnaire and to either request the best person to receive a second questionnaire, or to obtain answers to the questionnaire directly over the phone. From the follow-up calls, two colleges answered the questions over the phone, and three wanted an additional survey mailed to a specific person at their college. The additional surveys were mailed on February 24, 1998.

By March 1, 1998, a total of nine of the nineteen colleges responded which comprise the sample population of the study (see Appendix B).

CHAPTER 4

Analysis of the Data

Introduction

This chapter presents the results of a study assessing the current trends in teaching object-oriented programming at the community college level. The data gathering instrument, a questionnaire (see Appendix A) was developed to determine the trends in teaching object-oriented programming at the community college level. In particular, the questionnaire was developed to assess the extent of object-oriented material offered in computer science courses at the community college level. The questionnaire was sent to all nineteen community colleges in the state of New Jersey. Nine of the nineteen (48 percent) surveys were completed and returned (see Appendix B). A comprehensive analysis of the survey data is provided.

Analysis of Survey Results

Question one of the questionnaire is a close-form question designed to obtain the majority of the research material. It is directly concerned with the current trends in teaching object-oriented programming. It queries the respondent for the title of each computer science course offered at their college and the percentage of object-oriented

material contained in that course. For the nine community colleges that the data represents, eighty five computer science courses were listed under question one. Of the eighty five courses, thirty of the courses (35%) were reported to contain some degree of object-oriented material in them. Figure 4.1 shows a pie chart depicting the ratio of courses containing some object-oriented material with courses that do not contain object-oriented material.

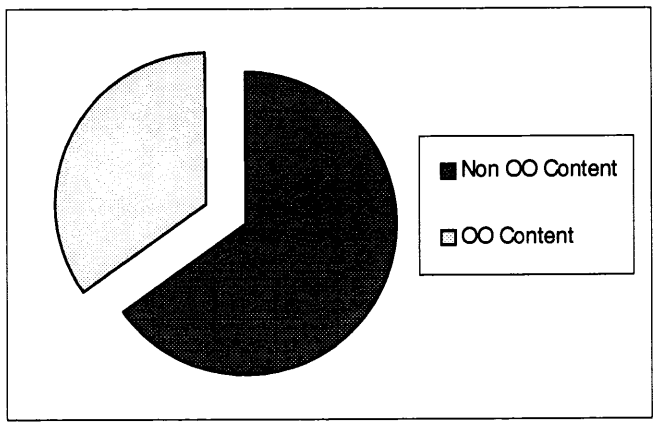

Figure 4.1

Percentage of Courses with Object-Oriented Content

All nine colleges reported offering at least one course containing some degree of object-oriented material. The largest number of courses at any one of the community colleges was six. Figure 4.2 depicts the number of courses containing object-oriented material for all nine colleges.

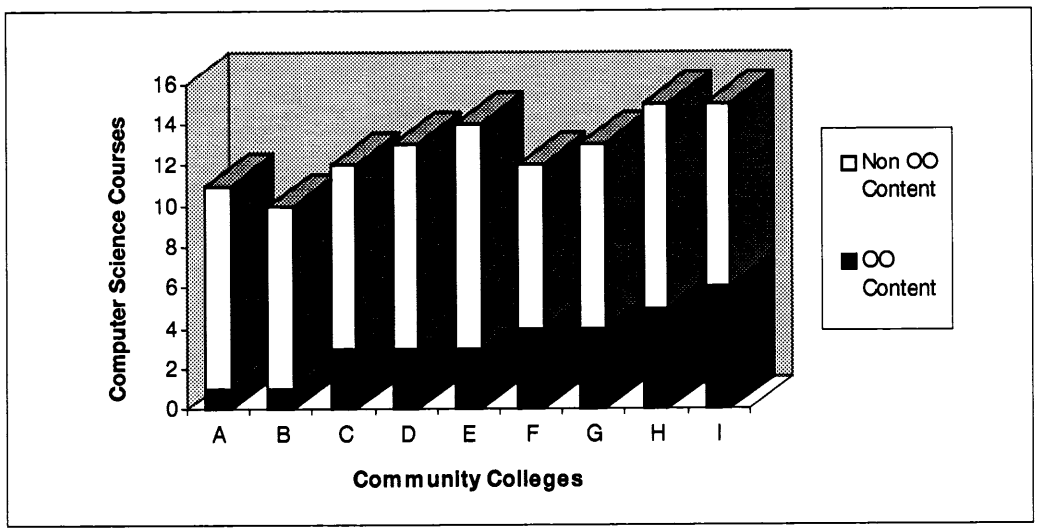

Figure 4.2

Number of Courses containing Object-Oriented Material

A total of thirty courses across all nine community colleges were reported to contain some degree of object-oriented material. The degree of the object-oriented material reported in the courses ranged from 5% to 100%. Fourteen of the thirty courses (48%) contained at least 50% object-oriented material. Eight of the thirty courses (27%) contained 100% object-oriented material. Figure 4.3 depicts the percentage of object-oriented material contained in all courses reported in the survey.

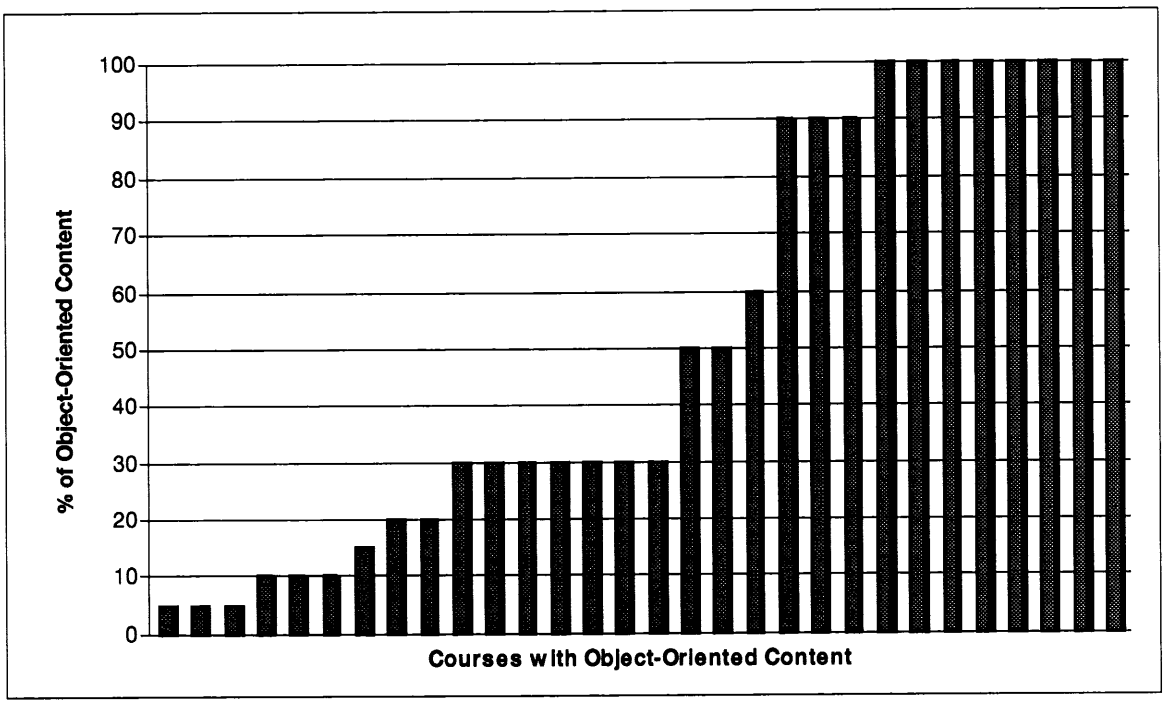

Figure 4.3

Percentage of Object-Oriented Material in Courses

It was noted from the survey that many community colleges offered courses that had similar or exact titles of courses offered at another community college. The content of object-oriented material in these courses differed considerably. For example, one college reported that a course entitled "Programming in C++" contained 100% object-oriented material while another college, offering a course with the same title, reported only 20% object-oriented material. Table 1 maps all thirty course titles reported across all nine community colleges with the percentage of object-oriented material reported for each course.

Table 1

Course Title and Percentage of Object-Oriented Material

Course Title	Percentage of OO Material
Programming in C	5
Data Structures	20
Advanced Programming (C++)	20
Intro to Computer Science	5
Computer Science I	10
Data Structures	15
Intro to Object-Oriented Programming	90
Visual Basic	30
Visual C++	30
Programming in C++	50
Object-Oriented Programming Using C++	100
Programming in Java	100
Data Structures	50
Foundations of Computer Science	5
Programming in C++	100
Programming in Java	100
Advanced Java	100
Computer Science I	100
Intro to Java	90
Computer Science II	100
Advanced Java	90
Visual Basic	10
Object-Oriented Programming and Design	100
Programming in C++	30
Intro to Computer Science I	30
Intro to Computer Science II	60
Programming Languages	30
Database Systems	30
Programming in C++	30
Visual Basic	10

Question two on the survey was an open-form question designed to gain insight as to whether the college plans to develop its computer science curriculum to include more object-oriented programming courses. The college that reported the

most object-oriented courses and the highest percentage of object-oriented content had just revised their computer science AA transfer program. The college that reported the least amount of object-oriented material was in the process of reorganizing the college structure and computer science department and was hopeful of increasing and modernizing their computer science program. The remaining colleges have indicated that they are either in the process of adding new courses addressing object-oriented programming or are increasing the percentage of object-oriented material in their current courses. One college that reported only 5% of object-oriented material in both the "Foundations of Computer Science" and "Data Structures" courses plans to switch to the C++ language in both of those courses. One college reported that the development of object-oriented material in the curriculum is occurring slowly due to the "ramp up speed" of the faculty.

Question three on the survey was an open-form question designed to acquire comments in regard to teaching object-oriented programming at the introductory level. All but one of the respondents agreed that exposing students to object-oriented programming and design is important and should be introduced early in the computer science curriculum. One college reported that so much of their current computer science curriculum is necessary to build a strong foundation in computer science that there may be no room for object-oriented programming without sacrificing the current material.

CHAPTER 5

Summary, Conclusions and Recommendations

Introduction

This chapter concludes the study to assess the current trends in teaching object-oriented programming at the community college level. What courses are being offered in the computer science curriculum at the community colleges? How many of them contain object-oriented material and to what degree? Where are the community colleges heading with respect to offering object-oriented programming courses? The tools used to answer these questions were the questionnaire sent to all community colleges in New Jersey. The first section, Summary of the Findings, provides a synopsis of the study. The next section, Conclusions, state the conclusions based on the findings of this study, and the final section, Recommendations for Further Study, include suggestions for both broadening the scope of the study as well as a follow-up to this study.

Summary of the Findings

The purpose of this study was to assess the current trends in teaching object-oriented programming at the community college level. In particular the study was focused on the computer science curriculum of the community college.

A questionnaire, consisting of three questions, was sent to all nineteen community colleges in the state of New Jersey. Nine of the nineteen surveys were completed and analyzed.

All nine colleges reported offering at least one course containing some degree of object-oriented material. Of the eighty five courses offered at the community colleges which were involved in the study, thirty of the courses (35%) were reported to contain some degree of object-oriented material. Fourteen of the thirty courses (48%) contained at least 50% object-oriented material and eight of the thirty courses (27%) contained 100% object-oriented material.

The content of object-oriented material in these courses differed considerably. Different colleges offer courses with the same or similar title but depending on the college, the degree of object-oriented material in the course varies significantly.

Colleges that reported the most object-oriented content had just revised their computer science curriculum. Most of the colleges in the study have indicated that they are either in the process of adding new courses addressing object-oriented programming or are increasing the percentage of object-oriented material in their current courses. Most of the colleges in the study agreed that exposing students to object-oriented programming and design is important and should be introduced early in the computer science curriculum.

Conclusions

Based on the findings of this study, the following conclusions can be drawn:

1. Community Colleges in New Jersey currently teach object-oriented programming to some degree.
2. The majority of computer science courses in the community college do not currently contain object-oriented material.
3. Computer science courses with the same title vary in the degree of object-oriented content from one community college to another.
4. The migration to an object-oriented paradigm in the computer science curriculum at the community college is occurring slowly.
5. The computer science faculty's experience in object-oriented programming indicates a relationship with the degree of object-oriented material offered in the curriculum.
6. The problem encountered in teaching object-oriented programming after a student has learned procedural programming is likely to continue until the percentage of object-oriented content increases.
7. Community colleges are including more object-oriented content as they re-structure their computer science departments.
8. The need to expand the degree of object-oriented content at the community college is recognized as important.

Recommendations for Further Study

There is a need to replicate this study on a greater scale and a need to repeat it on a regular basis. The computer industry is changing at an alarming rate and the need for qualified programmers graduating college is also increasing. Industry is shifting from a procedural paradigm to an object-oriented paradigm to cut maintenance costs in the software industry. The computer science curriculum must also make this shift to meet industry's demands.

Based on the findings of this study and on the demands of industry, the author recommends that community colleges:

1. Increase their object-oriented computer science course offerings.
2. Increase the object-oriented content of their current computer science courses.
3. Increase the faculty's ability to teach object-oriented material.
4. Coordinate efforts with other community colleges and four year institutions to produce a consistent and reliable computer science curriculum.
5. Keep in touch with industry and how they are using object-oriented technology.

Appendix A

Cover Letter and Questionnaire

Thomas W. Fargnoli
112 West Court
Blackwood, NJ 08012
email: magic@voicenet.com

April 20, 1998

Dear Sir / Madam:

I am a software engineer at Lockheed Martin Corporation and an instructor at Burlington County College in the Computer Science Department. Specifically, I am involved in tracking the software industry's progress as it makes a transition from traditional procedural programming applications to object-oriented applications.

Object-oriented programming offers a new and powerful model for writing computer software. Industry is attracted to this approach for it speeds the development of new programs and, if properly used, improves the maintenance, reusability, and modifiability of software.

One of the main obstacles, however, in a company adopting object-oriented technology is related to the difficulty in *learning* the new technology, even for experienced programmers. This is because object-oriented technology is more than just a way of programming. It is a way of thinking abstractly about a problem using real world concepts, rather than computer concepts.

Consequently, interest in teaching object-oriented programming as an introduction to computer science students, even at the high school level, has increased substantially over the last few years. To this end, I am conducting a survey to help determine the current trend in teaching object-oriented programming at the introductory level.

I am enclosing a questionnaire and ask you to take a few moments to fill out and return in the enclosed stamped self-addressed envelope or, if more convenient, you may email your response to me. The results of this survey will be forwarded to you.

Thank you for your assistance in responding.

Very truly yours,

Thomas W. Fargnoli

Questionnaire
Trends in Teaching Object-Oriented Programming at the Introductory Level

---

Name of College
College Address

Please Respond based on the Course Syllabus

1. Please list all computer science courses offered at your college and next to each course, please approximate the percentage of the course material devoted to object-oriented design and implementation.

	Course Title	% Object-Oriented Material
First Year :	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
Second Year :	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____

2. Does your college plan to develop its computer science curriculum to include more object-oriented programming courses? Briefly explain.

3. Please provide any comments you may have on object-oriented programming in regard to teaching it at the introductory level.

Appendix B

New Jersey Community Colleges

that Responded to the Survey

Colleges that responded to the survey included:

Hudson County College
Jersey City, NJ

Middlesex County College
Edison, NJ

Ocean County College
Toms River, NJ

Mercer County College
Trenton, NJ

Burlington County College
Pemberton, NJ

Cumberland County College
Vineland, NJ

Camden County College
Blackwood, NJ

Gloucester County College
Sewell, NJ

Raritan Valley Community College
Somerville NJ

BIBLIOGRAPHY

Beck, Ken, Apple Computer, Inc., A Laboratory For Teaching Object-Oriented Thinking, (OOPSLA'89 Conference Proceedings October 1-6, 1989, New Orleans, Louisiana and the special issue of SIGPLAN notices Volume 24, Number 10, October 1989 - http//c2.com/doc/oopsla89/paper.html), 1

Bergin, Joseph, "Teaching Object-Oriented Analysis and Design in CS 1" , 1997, http://csis.pace.edu/~bergin/papers/OOAD.html

Fishman, Robert G., and Kemere, Chris F., Object Technology and Reuse: Lessons from Early Adopters, Computer, October, 1997)

Kolling, Michael, Koch, Bett, and Rosenberg, John, Requirements for a First Year Object-Oriented Teaching Language, (SIGCSE Bulletin, Vol. 27, No. 1, March 1995), 173-177

Kolling, Michael and Rosenberg, John, An Object-Oriented Program Development Environment for the First Programming Course, (Proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education, March 1996), 83-87

Montlick, Terry, What is Object-Oriented Software? An Introduction, (Software Design Consultants, 1997 - http:/www.soft-design.com/softinfo/objects.html), 7

NSF Sponsored Workshop, Illinois State University, Object-Orientation Across Undergraduate Computer Science Curricula, Program Grant Number DUE-9455119 (June 3 - June 14, 1996 - http:/www.cs.ilstu.edu/oopoverview.html), 1

Rumbaugh, Blaha, Premerlani, Eddy, Lorenson, Object-Oriented Modeling and Design. (Englewood Cliffs, New Jersey: Prentice Hall, 1991) ix

Traxler, John, School of Computing and Information Technology at the University of Wolverhampton, England, 1994, "Teaching Programming Languages and Paradigms".

Wirfs-Brock, Rebecca and Wilkerson, Brian, Object-Oriented Design: A Responsibility-Driven Approach, (http//www.vpplus.com/obj1.html), 3