AD-A275 385

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

2

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 11/19/93	3. REPORT TYPE AND DATES COVERED Final 25 Sep 92 - 24 Sep 93

4. TITLE AND SUBTITLE *ADA, Object-Oriented Techniques and Concurrency in Teaching Data Structures and File Management*	5. FUNDING NUMBERS DAAL03-92-G-0413
6. AUTHOR(S) *Dr. Henry G. Gordon*	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) *Kutztown University of PA* *Kutztown, PA 19530*	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P. O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARO 30995.1-MA

DTIC ELECTE FEB 9 1994 S C D

11. SUPPLEMENTARY NOTES
The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This project examined the use of Ada in teaching Data Structures and File Management. The object-based features of Ada, its packaging mechanism, and generic capabilities were used to develop reusable packages for a variety of data and file structures. These packages were used in software applications to illustrate software development for already existing code. Developed packages included singly and doubly-linked list; stacks; queues; general binary search trees; AVL-trees; directed, weighted graphs with traversal; random files; B-tree files; B^+-tree files; and hash files.

94-04277

94 2 08 005

14. SUBJECT TERMS *ADA, Data Structures, File Structures, Object-Oriented, Reusable Code, generic packages*	15. NUMBER OF PAGES 8
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Ada, Object-Oriented Techniques, and Concurrency in Teaching Data Structures and File Management

Final Report

Dr. Henry G. Gordon

November 11, 1993

U. S. ARMY RESEARCH OFFICE

Grant No. DAAL03-92-G-0413

Kutztown University of Pennsylvania

Approved for Public Release

Distribution Unlimited

Introduction

This paper constitutes the final report of a project funded by the Defense Advanced Research Projects Agency program for Curriculum Development in Software Engineering and Ada. The project consisted of using Ada, object-oriented methods and concurrency in teaching data structures and file management techniques. This sequence of courses constitutes the earliest courses in the upper division of the curriculum for computer science majors at Kutztown University of Pennsylvania. It emphasizes the use of abstract data types, modularity, information hiding, genericity, and object-oriented techniques to achieve the development of reusable Ada packages for use in software engineering.

Rationale

At Kutztown University of Pennsylvania, the lower division requirement of courses in computer science ends with an advanced course in data structures and is followed by a course in file management techniques. In order to foster the inculcation of good software development methods which can be used in all of the upper division curricula, we decided to introduce Ada in these centrally positioned courses. Not surprisingly, we have found that students who are well-trained in the use of software development capabilities have had excellent experiences in internships, co-operative employment, and other job-related opportunities. The language Ada provides the ideal medium for the student to experience the development of quality reusable software.

This project naturally divided itself into two parts: Data Structures, and File Management Techniques. In the sections below we will discuss the implementation and results of the development of each of those topics.

Course Descriptions

The courses involved in this project consisted of the standard advanced data structures material followed by a study of file management techniques. The following topics are covered in these courses.

Data Structures

A. A review and implementation in Ada of the elementary data structures examined in CS II. This includes linked lists, stacks, queues, and simple binary trees. Both array and dynamic allocation storage techniques are studied.

B. A study of more advanced storage structures is then pursued. This includes multiply-linked structures, sets, hash tables, maps, advanced tree structures such as AVL and 2-3 trees, and graphs.

C. A study of internal searching and sorting techniques.

File Management

A. A study of the fundamental file processing operations of opening, closing, reading, writing and seeking is followed by an examination of secondary storage and fundamental file structure concepts.

B. A study of the organization of files to improve performance of the fundamental file operations and the activities of searching for and sorting records on a file.

C. A study of sequential, random, indexed, B-tree, B^+-tree and hash files.

Throughout both courses, emphasis is placed on the development of reusable software through the use of abstraction, careful design specification, modularity, information hiding, and genericity. Algorithmic analysis is used to compare the performance of different structures and different implementations of the same structure. Students are assigned programming projects as individuals and as members of a team. Tests are administered at the end of logical sections and a final examination is given.

Course Software Support

The University uses a Unisys U6000 Model 70 multiprocessing computer to support academic computing. The Ada compiler is from Software Leverage, licensed for modification for the U6000 from Verdix Corporation. In addition, OpenAda from Meridian Software Systems was also available on a limited basis to some students and faculty. The use of the U6000 guaranteed unlimited access to computer resources for all students at virtually any time of the day or night.

Implementation

Since most students in the data structures course did not know Ada, the review of the preliminary material on elementary data structures was used as a tool for introducing the language. All of the students knew Pascal, so the PASCAL-like part of Ada was easily understood. The student's prior knowledge of elementary data structures enabled us to concentrate on the features of Ada which were ideally suited to the development of reusable generic components. We used the book *Software Engineering with Ada* by Grady Booch[1] as a background book for this part of the course. We felt that with their prior knowledge of Pascal, the student could use this book as a reference and guide to principles of programming in Ada. This part was supplemented with examples and exercises to build a foundation for later topics in the course. While there are a number of good texts in this area now, we felt that no one of them is the ideal text for this course. We finally chose *Software Components with Ada* by Grady Booch[2]. We felt that most of the material we wanted to cover was in that text and its style would be consistent with the supplementary language text mentioned before. In addition, we agreed with Booch's approach on the specification of ADTs.

In the file management course, we chose to continue to use the non-Ada book *File Structures* by Folk and Zoellick[3]. This was supplemented by readings from *File Structures with Ada* by Miller and Petersen[4].

In the first course, emphasis was placed first on the design and specification of abstract data type. Each specification is compiled separately from the implementation. It may also be tested in the compilation of an application program using that specification before the implementation is even considered. Students developed such packages, perhaps different students doing different packages. In most instances they were tested by small programs that exercised every aspect of the ADT. After several useful packages had been developed, we incorporated them in a significant application which other classes might use. To illustrate, we emphasized the benefit of developing reusable and generic packages by developing a spelling table routine for compilers. When an identifier in a program is encountered by the compiler, it inserts the spelling of the identifier in a spelling table. That spelling table can be visualized as a hash table where collisions are handled by a singly linked list. Each entry in the list contains an information about where the spelling is located in the string pool. The spelling table is constructed from a previously developed generic hash table package and a string pool manager. The example enabled us to illustrate the difference between a type-manager such as a generic linked list package and an object manager such as a string pool, and how each can be used to build a very useful object such as a spelling table with minimal coding effort. The distinction between type-manager and object manager is taken from *Understanding Concurrency in Ada by Ken Shumate[6].*

A package on sets was developed which implemented the idea of set according to familiar Pascal approach. After testing, this package was shared with a graduate class in compiler design which needed a set manipulation package.

Packages on AVL-trees, maps, and graphs were developed. The graph package was used to solve the shortest path problem for airline routing.

The students could also benefit the other way. In a separate course, we were concerned about Ada and non-Ada processes concurrently accessing the same file. We developed a locking package, package Lock-IO, which would allow different types of tasks to access safely the same file concurrently. Students then could use this package in a concurrent file application without having to write the code.

Students developed packages for manipulating random files, indexed files, B-tree files, B^+-tree files and hash files.

Other Activities

In October 1993, we presented the spelling table package at the Fall meeting of the Pennsylvania Association of Computer and Information Science Educators meeting.

In January 1993, a three day seminar for faculty from the other State System Universities was conducted. Sponsored by Meridian Software Systems, each participant was allowed to purchase Meridian's OpenAda compiler at a very special price. The purpose of these activities was to encourage the use of Ada in the Computer Science curricula of the universities in the State System.

Results and Conclusions

The following list summarizes the packages developed during the course of this grant.

singly linked lists
doubly linked lists
stacks
queues
sets
maps
binary search trees
AVL trees
hash table with linear chaining
string pool
graphs
random files
hash files
b-tree files
b^+-tree files

The software can be requested from gordon@acad.csv.kutztown.edu. It maybe available by anonymous ftp in January 1994.

Without a doubt, the students benefited greatly from the experience of developing and using reusable, generic packages. Comments such as "I never have to write or modify another linked list routine, or change a sort" were commonplace. Students immediately see the benefit of developing software in the manner that hardware is developed - using already existing components. The only negative comments were related to the fact that fine points of the language were not always learned because of the fact that this was a course in data structures and we learned "what was necessary to do the job." From a student point of view, the experience was a great success.

On the negative side, anyone contemplating a change to Ada should be warned of the following. Most opposition will come from your colleagues. Be prepared for the lament that "Every one who is doing anything in my area is doing it in C." or whatever their

favorite language is. We are presently debating a change in the language used in the introductory courses. The language choice should be predicated on features that support good software development, reusability, modularity, information hiding, concurrency, object-oriented features, and STANDARDIZATION.
The only language that satisfies them all is Ada.

Personnel

Prof. Linda Day

Dr. Vivian G. Mosca

Dr. Henry G. Gordon

No degrees were obtained during the grant period.

Bibliography

1. Booch, Grady, *Software Engineering with Ada, 3rd Ed.*, Benjamin Cummings, Menlo Park, CA 1994

2. Booch, Grady, *Software Components with Ada: Structures, Tools, and Subsystems,* Benjamin Cummings, Menlo Park, CA 1987

3. Folk, Michael J. and Zoellick, Bill, *File Structures, 2nd Ed.*, Addison Wesley, reading, MA 1992

4. Miller, Nancy E. and Petersen, Charles G., *File Structures with Ada,* Benjamin Cummings, Menlo Park, CA 1990

5. Shumate, Ken, *Understanding Ada with Abstrac: Data Types, 2nd Ed.*, John Wiley & Sons, New York, NY, 1989

6. Shumate, Ken, *Understanding Concurrency in Ada,* McGraw-Hill New York, NY, 1988

7. Stubbs, Daniel F. and Neil W. Webre, *Data Structures with Abstract Data Types and Ada,* PWS-Kent Publishing Co., Boston, MA, 1993

8. Weiss, Mark Allen, *Data Structures and Algorithm Analysis in Ada,* Benjamin Cummings, Menlo Park, CA 1993