DRTT LANGLEY

Semi-Annual Report

Grant No. NAG-1-242

RESEARCH IN COMPUTER SCIENCE

Submitted to:

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Attention: Dr. John N. Shoosmith
ACD, MS 125

Submitted by:

James M. Ortega
Professor and Chairman

Report No. UVA/528209/AM85/106

December 1984

DEPARTMENT OF APPLIED MATHEMATICS

Semi-Annual Report

Grant No. NAG-1-242

RESEARCH IN COMPUTER SCIENCE

Submitted to:

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Attention: Dr. John N. Shoosmith
ACD, MS 125

Submitted by:

James M. Ortega
Professor and Chairman

Department of Applied Mathematics
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF VIRGINIA
CHARLOTTESVILLE, VIRGINIA

Report No. UVA/528209/AM85/106
December 1984

Copy No. ____

This report summarizes work under NASA Grant NAG-1-242 for the period June 1, 1984 to Lecember 1, 1984. During this period, ten graduate students were supported. The students, their major area of interest, Langley contact, University of Virginia faculty advisor, and total period of support are summarized below.

Student	Area	Langley Contact	Advisor	Period
P. Ammann	Software Engineering	E. Senn	J. Knight	1/15/84-
D. Bahler	Data Management	R. Fulton	J. Pfaltz	6/1/82-
S. Brilliant	Software Engineering	E. Senn	J. Knight	6/1/83-
N. Fitzgerald	Concurrent Processing	O. Storassli	T. Pratt	1/15/84-
N. Grine	Software Engineering	E. Senn	J. Knight	5/21/84-8/17/84
C. Luan	Computer Graphics	D. Lansing	W. Martin	5/21/84-
J. Marco	Software Engineering	E. Senn		5/28/84-
E. Poole	Parallel Computing	J. Lambiotte	J. Ortega	6/1/83-
B. Smith	Software Engineering	E. Senn	J. Knight	6/1/83-8/24/84
J. Taylor	Concurrent Processing	J. Lambiotte	T. Pratt	6/4/84-8/17/84

During this reporting period, Eugene Poole and Lois St. Jean, who was previously supported on this grant, completed their masters degrees. The title of their thesis or project is given below.

E. Poole - An M-Step Incomplete Cholesky Preconditioned Conjugate Gradient Method for the CDC Cyber 203/205 Vector Computer

L. St. Jean - Testing Version Independence in Multi-Version Programming

Several other students are expected to complete their degrees by the end of the current semester.

We next give short summaries of the work performed during the reporting period.

Resilient Seeded Errors Via Simple Techniques

Paul E. Ammann, Masters Candidate in Computer Science
John C. Knight, Associate Professor of Computer Science

An experiment has been performed in which simple syntactic alterations are introduced into program text; the experiment was carried out in the spirit of the testing strategy Error Seeding. The mean times to failure for the seeded errors were observed in repeated trials over a fixed range of inputs. The experiment's goal was to determine if randomly placed syntactic manipulations can produce failure characteristics similar to those of the indigenous errors found within the unseeded programs. The seeded errors were found to have a broad spectrum of mean times to failure independent of the syntactic alteration used.

The programs used for the seeding experiment are 27 Launch Interceptor Programs (LIPs) written to identical specifications. Over the course of an extensive N-version programming experiment, the unseeded LIPs have each been run over 1,000,000 randomly generated input cases; consequently a great deal is known about the indigenous errors within the LIPs. The use of functionally identical programs allowed the individual programmer to be removed as a variable from the error seeding experiment.

It is possible to produce errors of arbitrary difficulty to locate with simple error seeding techniques. In addition, several unexpected results indicate that not all of the issues involved in Error Seeding have been addressed. Specifically, certain of the seeded errors were benign; they had no effect on the program's functionality. Other seeded errors actually corrected indigenous errors. These results have clear implications for the testing strategy.

Knowledge Representation for Engineering Design

Dennis R. Bahler, Ph.D. Candidate in Computer Science
John L. Pfaltz, Professor of Computer Science

Mr. Bahler's new representation scheme for the modeling of three-dimensional objects, known as the B-spline cylinder, was refined, and the implementation was enhanced to permit the automatic construction of cylinder faces passing through any arbitrary set of interpolation points. These entities are comprised of interpolations among sets of cubic B-spline curves, and are similar to the generalized cylinders sometimes used in vision processing. Entities defined in this way have proven to be extremely efficient in both storage and computational cost, while being suitable for a variety of applications both in image understanding and geometric modeling. All the common primitives of a constructive solid geometry (CSG) modeler can be represented by this single entity class. In addition, more general free-form objects are also representable by exactly the same scheme and can function as user-defined primitives. Work has begun on the algorithms necessary to enable grouping of entities into more complex compound objects.

Beginning in the late summer and fall, 1984, a number of promising new avenues of research are being explored. Work has now begun on defining a set of spatial and structural predicates over the domain of cylinder-objects, and a resolution-based theorem prover has been implemented for the purpose of employing automated deduction techniques in this domain. This system is now being debugged and refined. Also under investigation is the suitability of using a set of Euler operators together with a specified set of constraints that can then be automatically satisfied at each stage of construction of a primitive or complex object. Preliminary investigation has begun of the design of a deductive database containing geometric and topological information as well as non-geometric data pertaining to the object world.

Analysis of Faults in a Multi-Version Software Experiment

Susan S. Brilliant, Masters Candidate in Computer Science
John C. Knight, Associate Professor of Computer Science

As part of an experiment in n-version programming, twenty-seven students at the University of Virginia and at the University of California at Irvine independently wrote versions of a Pascal procedure in compliance with the same specification. The versions have been tested by comparing their output against that of a "gold" version on one million randomly generated test cases. This research is directed toward identifying and analyzing the errors causing the failures observed during this testing. The types of errors and the relationships among errors causing correlated errors are to be studied. The impact of correlated failures on the reliability of multi-version software units is to be analyzed.

Implementation of a Parallel Programming Environment

Nancy J. Fitzgerald, Masters Candidate in Computer Science
Terrence W. Pratt, Professor of Computer Science

PISCES (Parallel Implementation of a Scientific Computing EnvironmentS) is a computer system intended for the solution of large scale problems in scientific and engineering computation. It is based on the use of MIMD parallel computation to achieve high computation rates. The system includes a programming environment, programming language, operating system, and machine architecture. Because the software provides an abstract "virtual machine" to the user, the precise details of the hardware and lower levels of the operating system software are not of concern to the user.

The base sequential language used in PISCES is FORTRAN. Language constructs have been added which allow the user to initiate tasks, and to communicate with other tasks via "message passing". A prototype system has been developed using UNIX as the underlying operating system. This implementation, which runs on a VAX under UNIX 4.1bsd, simulates the task and cluster level parallelism of the PISCES design on a uniprocessor. By using the UNIX "process" mechanism the tasks appear to be running in parallel, although, no actual parallel execution is possible.

The current PISCES implementation was constructed on the ICASE VAX 11/750 from June 1984 – August 1984.

Symbolic Execution of Concurrent Programs

Virginia S. Grine, Masters Candidate in Computer Science
John C. Knight, Associate Professor of Computer Science

Symbolic execution is a testing and validation method that attempts to test over the entire range of valid inputs to a program. Symbolic execution substitutes symbolic values for the typically numeric inputs. These symbolic values are then manipulated as the numeric values would have been. The output is a series of general formulas rather than numeric values.

The area of symbolic execution of sequential programs has been researched but symbolic execution of concurrent programs has not. Over the past summer, algorithms for the symbolic execution of the concurrency of the programming language Ada were developed. This fall the algorithms were partially implemented. This implementation has proved to be capable of detecting deadlock, livelock, and global variable misuse.

Two Computer Graphics Systems for Visualization of Pressure Distribution and Convective Density Particles

Carol T. J. Luan, Masters Candidate in Computer Science
Worthy H. Martin, Assistant Professor of Computer Science

The first part of the project is to develop a graphics package to demonstrate pressure distribution on airfoils. One dimensional color contours and two-dimensional mesh surfaces are available. Major approaches are McAllister's algorithm for shape-preserving quadratic splines, Coon's bivariate interpolating scheme and Jensen's removal of hidden line algorithm.

The second part of the project is to design a graphics system for visualization of a solid obstacle immersed in a set of convective translucent density particles. Volumes of density particles are renomalized to apply geometric optics. The ray tracing mechanism is replaced by a model of sequences of renormalized planes to avoid the high cost of solving complicated differential equations. The shape of the obstacle surface is recovered by constructing individual microsurfaces from input density data. A Gaussian normal distribution is applied to approximate the light reflection rate on the obstacle surface.

Design of a Source Code Management System

Jerrold L. Marco, Masters Candidate in Computer Science
Edmund H. Senn, NASA Langley Research Center

A system is being designed to centrally control and manage changes to libraries of source code files in software development and maintenance projects. A software project manager will have extensive control over who may make what sort of changes to what files. A history of changes will be maintained for each file, in such a way that previous versions of the file may be regenerated from the current version, thus providing a way to return to an earlier version without the overhead of storing largely redundant files. A further goal is that the system will respond to requests quickly; this has been taken heavily into account in the preliminary design. The package is to run under CDC's NOS.

A statement of requirements has been prepared, and a command language has been designed. A prototype of the command language interpreter is in place, and a prototype of the differencing mechanism is currently being developed. The next step is to produce a detailed design of the system.

Vectorizing Incomplete Conjugate Gradient on the Cyber 203/205

Eugene L. Poole, Ph.D. Candidate in Applied Mathematics
James Ortega, Professor of Applied Mathematics

This research is concerned with modification of existing algorithms and development of new methods used in solving problems involving large sparse matrices on vector computers such as the Cyber 205 at NASA-Langley or experimental parallel computers such as the FLEX computer to be installed at NASA-Langley in January '85. This project has vectorized incomplete Cholesky conjugate gradient for the Cyber 205 and is currently extending the results to a three dimensional model problem. Of particular interest is the multicolor ordering of grid points to achieve the matrix structure desired for long vector operations in the incomplete Cholesky algorithm.

Work has been completed on two model problems and results were presented at the Supercomputer Applications Symposium at Purdue University in October of this year. In addition, during the past summer at NASA, work was begun on a three dimensional simplified space platform model to be used in extending the previous results to a large scale structures problem.

Future plans are to test the new model problem on both the Cyber 205 and the FLEX at NASA-Langley, using the vectorized incomplete Cholesky conjugate gradient method previously developed.

Extensions of the Domain Testing Theory

Brandon Smith, Masters Candidate in Computer Science
John C. Knight, Associate Professor of Computer Science

The theory of domain testing was first developed by Lee J. White and Edward I. Cohen in the late 1970's at Ohio State University. It is a testing method designed to detect errors in the control flow of a computer program by generating test data on and near the boundary of a given path domain in order to uncover errors in that path domain. This work was followed by an analysis performed by a group from the University of Massachusetts led by Lori Clarke. While attempting to apply White and Cohen's domain testing theories to some programs, Clarke's group stumbled upon several difficulties. These problems led them to examine the domain testing strategy and corresponding error measures, and after this examination, two alternative strategies were proposed to improve on the error bound.

Both of these groups made many assumptions and restrictions on the allowable programming language constructs to be used in order for this testing strategy to be reliable. These assumptions and restrictions weaken the domain testing method considerably since they do not allow such common programming constructs as arrays, non-linear predicates, loops, and subroutines. After a thorough analysis of the work previously done with domain testing, it was determined that such restrictions and assumptions are unnecessary. By removing them and extending the domain testing strategy to accommodate for the new programming constructs allowed, the domain testing method is both worthwhile and very valuable for use with most modern programming languages.

Performance Analyzer for the Pisces System

Jeffrey A. Taylor, Masters Candidate in Computer Science
Terrence W. Pratt, Professor of Computer Science

PISCES is a Parallel Implementation of Scientific Computing Environments currently being implemented on the UNIX operating system. It is to be used as a distributed system for large scientific problems and is being written in FORTRAN.

PISCES operates on being able to pass messages for one task to another. These messages are passed through the use of files and are called message heaps.

The performance analyzer's purpose is to evaluate these message heaps and indicate how well the tasks were able to communicate with each other. Several options are available in the analyzer and if one wishes to look at a particular aspect of a run of PISCES, they are able to do so.