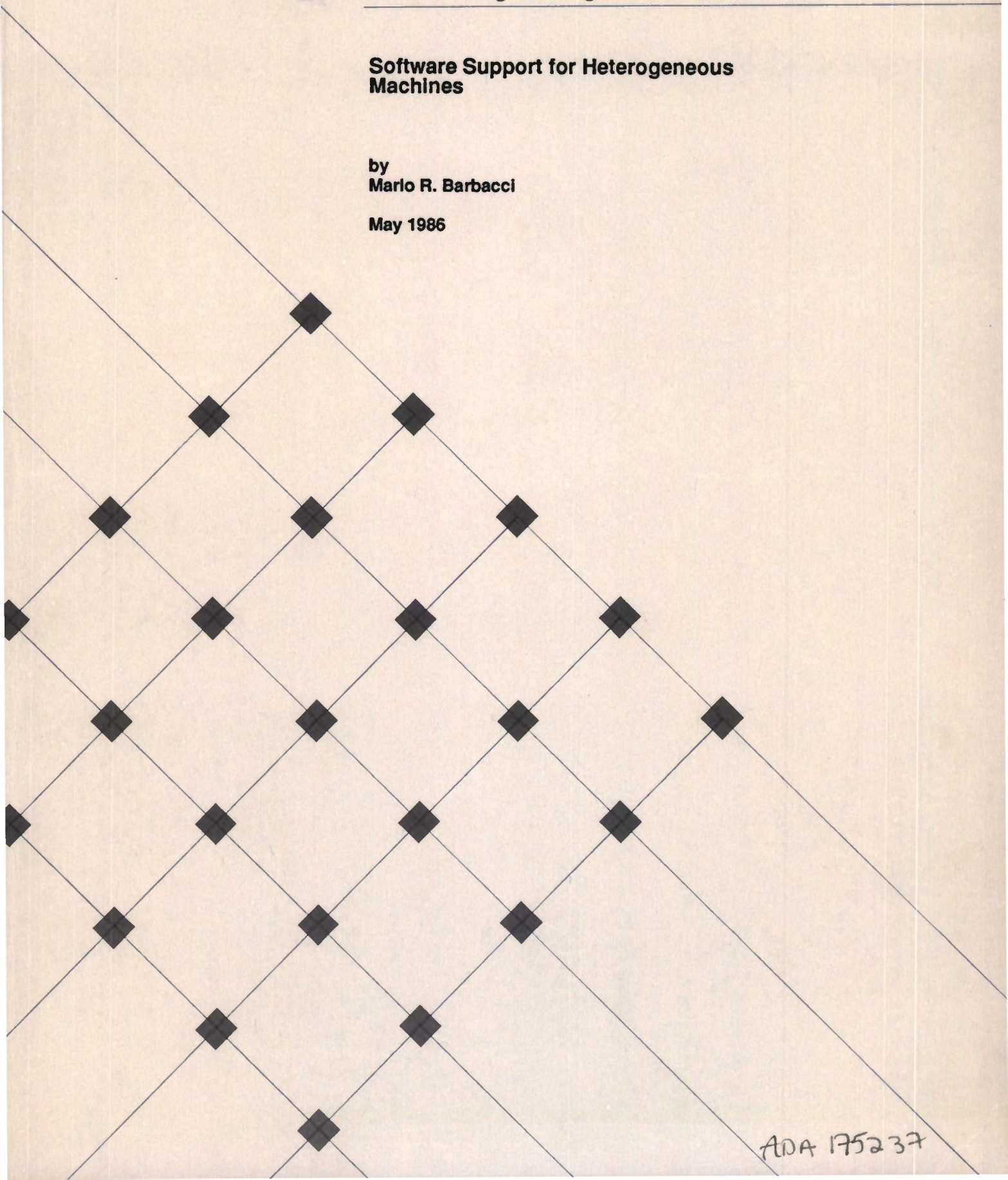
**Technical Report**
**CMU/SEI-86-TM-4**
**ESD-TR-86-210**

Carnegie-Mellon University
Software Engineering Institute

# Software Support for Heterogeneous Machines

**by**
**Mario R. Barbacci**

**May 1986**



**Software Engineering Institute**

**Technical Report**

**ESD-TR-86-210**

**SEI-86-TM-4**

**May 1986**

# Software Support for Heterogeneous Machines

**Mario R. Barbacci**



Carnegie Mellon University Pittsburgh, Pennsylvania 15213

This work was sponsored by the Department of Defense.

This technical report was prepared for the

SEI Joint Program Office
ESD/XRS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

**Review and Approval**

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

Karl Shingler
SEI Joint Program Office
ESD/XRS

# Software Support for Heterogeneous Machines

Mario R. Barbacci

**Abstract.** This paper describes a new research effort carried out jointly between the Software Engineering Institute and the Department of Computer Science at Carnegie Mellon University. The objective of the project is to investigate languages, methodologies, and tools for programming computer systems consisting of networks of heterogeneous processors.

Typical users of these notations (and associated support software) will be the developers of real-time, computation-intensive applications such as those contemplated under the Strategic Computing Initiative. In particular, this research is being conducted in the context of the Autonomous Land Vehicle application, running on the Heterogeneous Machine being developed in the Computer Science Department.

This paper provides some background on the nature of the problem posed by these applications, the opportunities presented by the emergence of heterogeneous machines, and the goals of this project.

## Introduction

We are all familiar with traditional numerical computation applications that were concerned with the accuracy and performance of complex algorithms. They operated on simple data structures (e.g., scalars and arrays) and were implemented in some imperative language (e.g., FORTRAN). The appearance of list-manipulation languages in the late 50's gave rise to symbolic computation applications that were concerned with the manipulation of complex data structures (lists and plexes of different kinds). They were implemented in relatives of Lisp and derivatives of Algol 60. The hardware architectures, however, remained relatively constant for several decades, and a great deal of progress was achieved in the development of useful programming languages and environments.

We are now beginning to build networks of heterogeneous processors whose users are concerned with allocation of specialized resources to tasks of medium to large size, executing concurrently.[1] Heterogeneous machines (e.g., Figure 1) will have general purpose processors, special purpose processors, memory buffers, and switches which can be configured in more or less arbitrary *logical* networks. In addition, these networks will not be static, configured once and for all for a given application. The networks will be able to alter their configuration depending on the needs of a particular application.

The Department of Computer Science at Carnegie Mellon University is building a heterogeneous

---

[1]For our purposes, let's assume that a medium size task granule takes in the order of 100 times a basic synchronization operation. That is, we are not dealing with the minute level of concurrency provided by array processors (e.g., ILLIAC IV) or pipelined functional units (e.g., CRAY) but rather with the scheduling and management of larger chunks of computation, with commensurably larger resource allocation requirements.

machine as a vehicle for research on high-performance computing. Research is also being conducted in vision processing (e.g., landmark recognition) and machine reasoning (e.g., path planning). These applications will depend on the large amounts of computing power that can be delivered by the proposed heterogeneous machine. The research described in this paper addresses the missing link, namely, the development of languages and methodologies for programming the heterogeneous machine to exploit the coarse-grain, task-level concurrency available in the applications. This work is being carried out at the Software Engineering Institute. Exploring this task-level parallelism is a new direction in parallel processing.

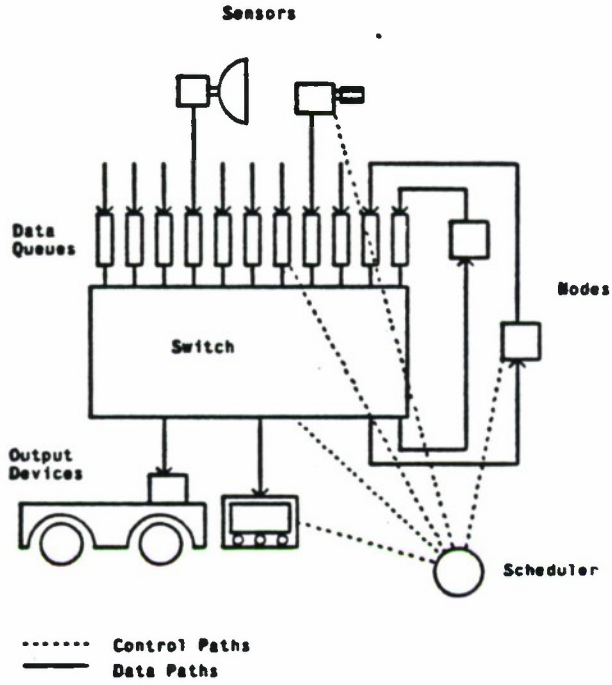



**Figure 1:** A Heterogeneous Machine

## 1. The Nature of the Problem

We expect that users of a heterogeneous machine will rely on libraries of painstakingly developed procedures to accomplish the common operations in their domain of application. On a high performance engine with multiple functional units, pipelines, and register sets, these procedures can be very difficult to write. However, this is within the reach of current compiler technology, and programming these engines is not the showstopper.

The major source of complexity in the applications for which the new heterogeneous machines are being built (e.g., Autonomous Land Vehicle [DARPA 83]) does not come from the basic data operations (these are hidden in the node procedures) or the data structures (usually limited to arrays and records). The complexity comes from the communication patterns between the computing elements required to make effective use of the available resources.

The writers of the application programs (e.g., top of Figure 2) must be familiar with the nature of the tasks performed by the processors (nodes in the graph) and the contents of the data queues (links in the graph) in order to program the applications. In general, the tasks will take different times to complete. The programmer must schedule the arrival of sufficient data to prevent starvation of nodes but not schedule so much that queues overflow. Thus, an expert's knowledge of the application is required to select and connect the right resources to achieve some optimal performance. Applications might have additional requirements that might not be directly expressible in terms of nodes, queues, and links. For instance, one requirement might be that some operation be performed twice as often as some other operation elsewhere in the graph in order to obtain some balanced flow of data. These requirements and constraints are part of the program and must be explicitly indicated.

The efficient use of a heterogeneous machine requires, therefore, support for developing application programs organized as multiple, concurrent, cooperating coarse-grain tasks. The tasks in turn could be more tightly-coupled parallel programs executing on specialized processors such as systolic arrays. These two programming levels can be separated from each other. The writer of a library procedure that performs some basic computation [e.g., convolution, histograms, etc.] does not necessarily know the context in which the procedure will be used. The procedure executes on a processor that consumes data from input queues and delivers results to output queues. By the same token, the developer of the application does not necessarily know the details of the procedures running on the nodes. The procedures are treated like black boxes or primitive building blocks with predetermined, perhaps nominal, performance characteristics.

## 2. The Nature of the Solution

Suppose that the application programs are represented as graphs, with nodes representing the tasks, and links representing data communication. Programming the task associated with a node involves *intra-node* concurrency while making the nodes of the graph to work in parallel is *inter-node* concurrency. The intra-node concurrency problem has been thoroughly studied, and there are reasonably mature techniques for writing useful concurrent programs for intra-node computations on special purpose systems. However, the higher level, inter-node concurrency is not as mature or understood; this is the area of interest to us.

As illustrated in Figure 2, a compiler for a task-level programming language will translate the application program into code for a virtual machine. The target "machine language" will consist of commands to be interpreted by a scheduler node. Typical commands include requests for data movements, data transformations, down-loading code to the computation nodes, invoking task, etc. It is the job of the scheduler to generate the appropriate low level control messages and route them to the processors in the system.

Ideally, neither the language nor the compiler should make assumptions about the structure of the heterogeneous machine; this knowledge should be left to the scheduler. In practice, the programmers may need to know something about the hardware to perform application dependent

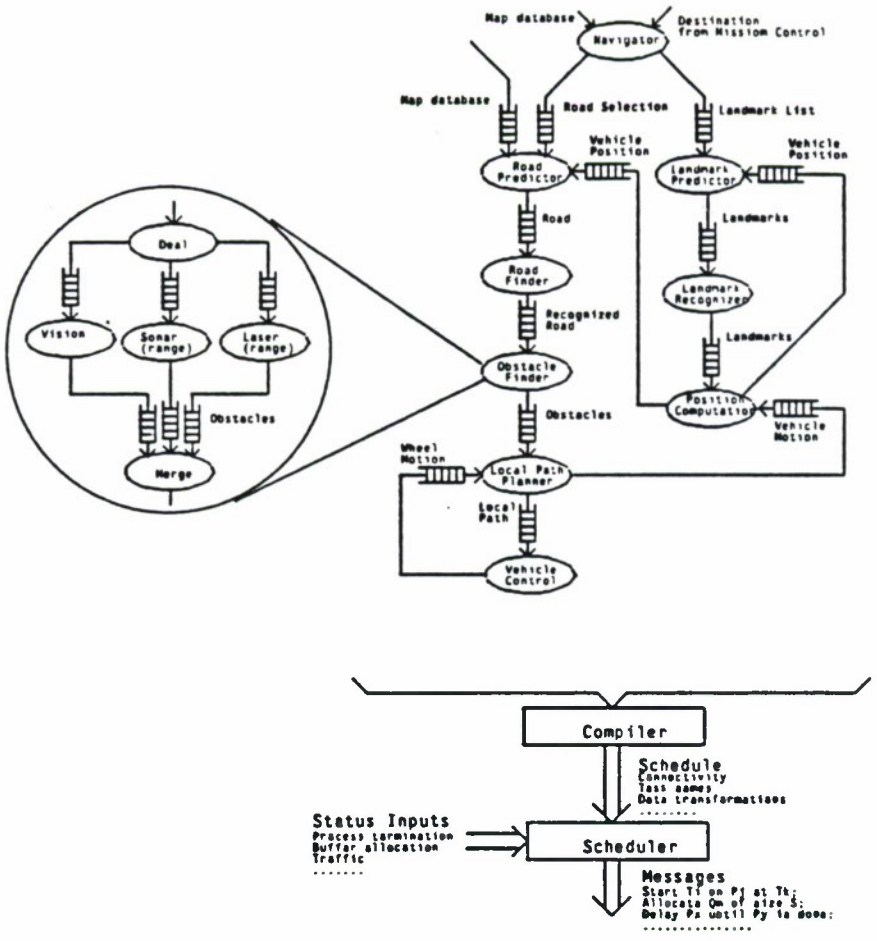



**Figure 2:** Compilation and Execution of a Task-level Concurrent Program

optimizations when they choose to do so. We are dealing, therefore, with (potentially) multiple virtual machine layers, organized in a hierarchy, and implemented by networks of message-passing "smart" resources such as processors, queues, switches, etc. The programmer will develop an application by specifying the operations (i.e., messages) to be carried out by the virtual machine level(s) deemed optimal for the application at hand. The range of abstractions provided by the virtual machines must be available to the programmer; this has obvious implications in the language design.

In this project we will address the following questions: How much information about the computing engines (nodes) and the tasks running on these engines should be visible? How much information about the machine structure (in contrast to the program structure) should be visible? How much information about the data and control communication infrastructure should be visible[2]?

[2]We distinguish between the *control* communication and the *data* communication networks. The control communication is used by the processors and other resources of the heterogeneous machine to exchange messages of various kinds as they schedule (or reschedule) the computation tasks. The data communication network, on the other hand, is used to implement the data flow through the machine, and is likely to have higher bandwidth requirements.

## 3. Project Goals

The objective is to develop a specialized programming language for writing distributed programs with coarse-grain concurrency. Suitable constructs will be included in the language for specifying individual tasks, their attributes, relationships between them, and preferred host processors for task execution. In general, language features will be designed in concert with the intended users and the hardware designers. The users will drive the design of the features needed to express task level programs. The designers will provide information about the hardware capabilities. Since the hardware design is taking place concurrently with the design of the language, the former is likely to be affected by the latter (i.e., language features needed to support the applications might require the implementation of appropriate hardware features).

The task-level, data-flow notation described appears to be a promising start in this direction, especially for signal processing computations where data continuously flow from input nodes to output nodes. There is a reasonably large body of literature on this subject, and we are studying a number of existing language models. Given the nature of the problem, dataflow languages such as ID [Arvind 78a; Arvind 78b] and VAL [Ackerman 79; Dennis 79] come immediately to mind. However, the applications are likely to require more flexibility than a pure dataflow model: We need to specify computations on streams of data as provided in Lucid [Ashcroft 77; Wadge 85]. In addition to the data flow operations, the applications require the specification of task synchronization, task control, and graph reconfiguration under a variety of conditions. To satisfy these requirements notations such as path-expressions [Campbell 74a; Campbell 74b] might be more appropriate.

Programs in the task-level programming language will be compiled into sequences of task invocation and data communication operations. The first part of the problem to be tackled is the definition of the basic language concepts: the operators and operands used to program the machines at the task level, ignoring the languages and support tools needed to program the basic tasks executing in the computation nodes. The design and implementation of the language and associated tools will be an iterative process, developing virtual machines to execute the task level programs. The first version of the system will provide a simple language, allowing for direct control of the physical resources (the lowest level "virtual machine"). The emphasis will be in obtaining early feedback from the users. Later versions of the system will incorporate additional application requirements (e.g.,, perhaps better user interfaces.) and multiple virtual machine layers. The abstractions provided by these machines will allow optimizations at the appropriate levels by both the users and the compiler.

## 4. Conclusions

This effort started in January 1986. In the interim, we have been studying existing language models that could be suitable as starting points for the development of a task-level concurrent programming language. At the same time, we have started the design and Implementation of a prototype heterogeneous machine (to be operational by the Fall of 1987) and are building a simulator to debug both the language and the hardware design.

## References


[Ackerman 79]
W.B. Ackerman, *VAL - A Value-oriented Algorithmic Language Preliminary Reference Manual*, MIT Laboratory for Computer Science, MIT/LCS/TR-218, June 1979.

[Arvind 78a] Arvind and K.P. Gostelow, *Dataflow Computer Architecture: Research and Goals*, Department of Computer Science, University of California, Irvine, TR 113, February 1978.

[Arvind 78b] Arvind, K.P. Gostelow, and W. Plouffe, *An Asynchronous Programming Language and Computing Machine*, Department of Computer Science, University of California, Irvine, TR 114A, December 1978.

[Ashcroft 77]
E.A. Ashcroft and W.W. Wadge, *Lucid, a Nonprocedural Language with Iteration*, CACM Vol. 20 No. 7, July 1977, pp 519-526.

[Campbell 74a]
R.H. Campbell and A.N. Habermann, *The Specification of Process Synchronization by Path Expressions*, University of Newcastle upon Tyne, Computing Laboratory, Technical Report 55, January 1974.

[Campbell 74b]
R.H. Campbell and P.E. Lauer, *A Spectrum of Solutions to the Cigarette Smoker's Problem*, University of Newcastle upon Tyne, Computing Laboratory, Technical Report 63, May 1974.

[DARPA 83] *Strategic Computing*, Defense Advanced Research Projects Agency, October 1983.

[Dennis 79] J.B. Dennis and K.K.S. Weng, *An Abstract Implementation for Concurrent Computation with Streams*, Proceedings of the 1979 International Conference on Parallel Processing, Detroit, Michigan, August 21-24, 1979, pp 35-45.

[Wadge 85] W.W. Wadge and E.A. Ashcroft, Lucid, the Dataflow Programming Language, APIC Studies in Data Processing No. 22, Academic Press, 1985.

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS<br>NONE |
| 2a. SECURITY CLASSIFICATION AUTHORITY<br>N/A | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for Public Release; Distribution Unlimited. |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>N/A | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>CMU/SEI-86-TM-4 | 5. MONITORING ORGANIZATION REPORT NUMBER(S)<br>ESD-TR-86-210 |
| 6a. NAME OF PERFORMING ORGANIZATION<br>SOFTWARE ENGINEERING INST.<br>6b. OFFICE SYMBOL (If applicable)<br>SEI | 7a. NAME OF MONITORING ORGANIZATION<br>SEI JOINT PROGRAM OFFICE |
| 6c. ADDRESS (City, State and ZIP Code)<br>CARNEGIE-MELLON UNIVERSITY<br>PITTSBURGH, PA 15213 | 7b. ADDRESS (City, State and ZIP Code)<br>ESD/XRS1<br>HANSCOM AIR FORCE BASE<br>HANSCOM, MA 01731 |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>SEI JOINT PROGRAM OFFICE<br>8b. OFFICE SYMBOL (If applicable)<br>ESD/XRS1 | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS (City, State and ZIP Code)<br>CARNEGIE-MELLON UNIVERSITY<br>PITTSBURGH, PA 15213 | 10. SOURCE OF FUNDING NOS.<br>PROGRAM ELEMENT NO.: 63752F<br>PROJECT NO.: N/A<br>TASK NO.: N/A<br>WORK UNIT NO.: N/A |
| 11. TITLE (Include Security Classification)<br>SOFTWARE SUPPORT FOR HETEROGENEOUS MACHINES | |
| 12. PERSONAL AUTHOR(S)<br>MARIO R. BARBACCI | |
| 13a. TYPE OF REPORT<br>FINAL | 13b. TIME COVERED<br>FROM MAY 86 TO MAY 86 |
| 14. DATE OF REPORT (Yr., Mo., Day)<br>MAY 86 | 15. PAGE COUNT<br>6 |
| 16. SUPPLEMENTARY NOTATION | |

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB. GR. | |
| | | | |
| | | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

THIS PAPER DESCRIBES A NEW RESEARCH EFFORT CARRIED OUT JOINTLY BETWEEN THE SOFTWARE ENGINEERING INSTITUTE AND THE DEPARTMENT OF COMPUTER SCIENCE AT CARNEGIE-MELLON UNIVERSITY. THE OBJECTIVE OF THE PROJECT IS TO INVESTIGATE LANGUAGES, METHODOLOGIES, AND TOOLS FOR PROGRAMMING COMPUTER SYSTEMS CONSISTING OF NETWORKS OF HETEROGENEOUS PROCESSORS. TYPICAL USERS OF THESE NOTATIONS (AND ASSOCIATED SUPPORT SOFTWARE) WILL BE THE DEVELOPERS OF REAL-TIME, COMPUTATION-INTENSIVE APPLICATIONS SUCH AS THOSE CONTEMPLATED UNDER THE STRATEGIC COMPUTING INITIATIVE. IN PARTICULAR, THIS RESEARCH IS BEING CONDUCTED IN THE CONTEXT OF THE AUTONOMOUS LAND VEHICLE APPLICATION, RUNNING ON THE HETEROGENEOUS MACHINE BEING DEVELOPED IN THE COMPUTER SCIENCE DEPARTMENT. THIS PAPER PROVIDES SOME BACKGROUND ON THE NATURE OF THE PROBLEM POSED BY THESE APPLICATIONS, THE OPPORTUNITIES PRESENTED BY THE EMERGENCE OF HETEROGENEOUS MACHINES, AND THE GOALS OF THIS PROJECT.

| | |
|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐ | 21. ABSTRACT SECURITY CLASSIFICATION<br>UNCLASSIFIED, UNLIMITED DISTRIBUTION |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>KARL H. SHINGLER | 22b. TELEPHONE NUMBER (Include Area Code)<br>412 268-7630<br>22c. OFFICE SYMBOL<br>SEI JPO |

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE