MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES
3 4456 0279138 9

ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

ORNL/TM-10558
(CESAR-87/44)

Analysis of Tasks for Dynamic Man/Machine Load Balancing in Advanced Helicopters

C. C. Jorgensen

CENTRAL RESEARCH LIBRARY
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A03; Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-10558
CESAR-87/44

Engineering Physics and Mathematics Division

ANALYSIS OF TASKS FOR DYNAMIC MAN/MACHINE LOAD BALANCING IN ADVANCED HELICOPTERS

C. C. Jorgensen

Date Published - October 1987

Research performed under
DOE Interagency Agreement #40-1729-86
Between Oak Ridge National Laboratory
and the
AAAI Program Office of NASA Ames Research Center

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
operated by
Martin Marietta Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES
3 4456 0279138 9

TABLE OF CONTENTS

ABSTRACT v

EXECUTIVE SUMMARY vii

I. LOAD BALANCING 1

I.1. Load Balancing Problems 5
I.2. Optimization Using Neural Networks 10
I.3. Conclusions Regarding Load Balancing 11

II. TASK STRUCTURING FOR DYNAMIC REALLOCATION 13

II.1. Functional Decomposition 13
II.2. Diagnostic Tests for Poor Functional Decompositions 16
II.3. Impacts for Traditional Task Analysis 17
II.4. Data Separation 18
II.5. Communication 20
II.6. Conclusions about Decomposition 21

REFERENCES 25

ABSTRACT

This report considers task allocation requirements imposed by advanced helicopter designs incorporating mixes of human pilots and intelligent machines. Specifically, it develops an analogy between load balancing using distributed non-homogeneous multiprocessors and human team functions. A taxonomy is presented which can be used to identify task combinations likely to cause overload for dynamic scheduling and process allocation mechanisms. Designer criteria are given for function decomposition, separation of control from data, and communication handling for dynamic tasks. Possible effects of n-p complete scheduling problems are noted and a class of combinatorial optimization methods are examined.

EXECUTIVE SUMMARY

The Army Aircrew Aircraft Integration program (AAAI) at NASA Ames Research Center seeks to advance the state of the art in design work stations for future Army helicopters. One part of this effort is the early consideration of man/machine interactions; in particular, interactions among autonomous human and intelligent machine design elements which may influence training or personnel requirements. The proposed work station must surpass existing facilities since future rotocraft will make extensive use of intelligent machine support systems. The shift of decision making responsibility from pilot to aircraft highlights problems requiring fundamental advances in our understanding of dynamic task description, symbiotic man/machine behavior, and training technology. The selection of which combat factors will have the greatest impact on future designs remains an open issue, but some likely sources of problems can be identified. Five areas which will have a major impact are environmental instability, sustained operational requirements, incomplete tactical knowledge, communication, and feedback:

1. Environmental Instability

Future battlefield environments will change rapidly. Failure to respond quickly to controllable conditions will result in ripple effects across highly integrated weapon systems because enemy responses and friendly defenses are likely to be automated. Availability of crew resources cannot be guaranteed and task tradeoffs to machines may have to be made dynamically depending upon the limitations of equipment.

2. Sustained Operations

Once initiated, rapidly evolving battle scenarios will require continuous control and decision making activity. Attempts to delegate task functions may result in discontinuous pilot workload depending upon the capability of the intelligent subsystems to hand off tasks and the amount of pilot intervention needed.

3. Incomplete Tactical Knowledge

Although many aspects of a mission can be anticipated during the design process, the full range of task performances cannot be known in advance since an adaptive and competitive struggle is involved. Thus apriori task allocations will have to be based largely on a design analysis of known human and system limitations rather than mission foreknowledge.

4. Communication

Battlefield information and its assessment during a mission will be demanding but also incomplete and inaccurate. A pilot will have to filter information, and human and machine perceptual limitations will

result in a simplified internal model of the true state of the battlefield. Automated machine-intelligent components will also filter data but may operate using a different world model. The degree of overlap must be sufficient for mutual communication as well as accurate enough for task handoff. Both the pilot and the machine will require compatible data interpretation and judgment processes.

5. Indeterminate or Delayed Feedback

Pilot and machine actions can influence the same environment that provides task performance cues. Long delays or diluted effects of actions may provide asynchronous behavioral cues, non-correlated cues, or no cues at all. Thus the world models of Item 4 will need to provide task cues when external cues are not available. The generation of such models pose a formidable problem for a machine intelligent entity.

Recognizing that design issues such as those previously mentioned currently draw on a very limited research base, the AAAI program office of NASA Ames initiated a basic research contract with Oak Ridge National Laboratory to examine dynamic interaction effects in greater detail. In particular, ORNL proposed to study whether useful design insights could be gained through a comparison between human-to-human interactions and analogous functions involving linked processors in distributed computing systems. One computer used as a basis for this comparison involved a class of architectures referred to as MIMD or multiple instruction multiple data path machines. Typical of this class are the INTEL iPSC and NCUBE hypercube supercomputers.

This research selected two topics for detailed study. The first involved the optimum allocation of functions and resources between human and machine systems, i.e. the "load balance problem". The second was a determination of methods which might be used to more precisely define the tasks to be balanced, their flow of control, and the handling of data used for decision making. Because the human factors and computer science communities have different perspectives, a significant amount of translation was needed to recast common definitions and solutions in a form amenable for use by AAAI. This report addressed the areas in the following way. In the first half of the report, a definition of load balancing is given and a mathematically inspired taxonomy for categorizing multiprocessor balancing situations is summarized. The purpose of the taxonomy is to permit a designer to classify potential helicopter task allocation strategies using their similarity to classical scheduling theory problems. Next, specific cases are studied because they most resemble situations which may occur in a human machine interaction. Solutions are considered which have been successfully used in load balancing that required combinatorial optimization. Potential advantages of simulated annealing and neural networks are discussed in this context.

The second half of the report deals with dynamic task decomposition. Preliminary guidelines are proposed for decomposition in load sharing environments and requirements for decomposition are identified. These include the separation of decisions from data, information hiding, functional binding, function coupling, and communication dependencies. Three AAAI design issues are discussed incidentally during the development of the topics: communication management, sensor data interpretation, and autonomous adaptability. Briefly summarized issues are:

1. Communication Management

The primary motivator for optimum man/machine load balancing is to minimize communication while maximizing the distribution of workload across available processing resources. If a future helicopter's task allocation mechanism is not human, it must be capable of communication with both human and machine intelligent programs. Underlying knowledge representations must absorb new information, express knowledge, and solicit battlefield data in human compatible formats (this is to permit a pilot to override any machine actions if needed). Degrees of human machine communication are possible. For example, Greenstein and Revesman (1986) consider implicit communication in which probable human actions are conveyed to the computer through a model of the human's action strategy. This has been shown to be effective in improving performance even if the capability of the model used is quite limited.

2. Data Interpretation

Numeric and perceptual data often needs to be interpreted for the pilot in qualitative ways to minimize overload. In the case of machine intelligent subsystems, a transformation table must be developed to permit an autonomous machine to link raw sensor data to task actions. Yet data reliability must at some time be judged by both the human and machine components. The judgment process should include consistency and plausibility checks based on an evolving mission history, previous experience, and factual knowledge. Incomplete or unreliable data may permit multiple hypothesis to be developed. Branching decision and action possibilities will have to be managed because under appropriate circumstances there may be a combinatorial explosion of action choices resulting in NP hard or P hard task scheduling requirements.

3. Autonomous Adaptability

Intelligent machines should be capable of adapting to their environment without direct intervention by the pilot. Such capabilities imply adaptive learning, extendibility of functions, rapid responsiveness, and the ability to be interrupted by unanticipated events. Such a system is currently beyond the state of the art. Therefore, in addition to basic research on machine learning, intelligent software limits should be considered as an integral part of design (much as skill and ability impacts will be considered for the human).

I. LOAD BALANCING

We begin our study of man/machine task sharing with a definition of computer load balancing:

> Load balancing may be defined as a distribution of tasks among multiple processors so as to minimize a performance metric while simultaneously minimizing interprocessor communication (e.g., Kleinrock 1985).

The most common metric used is execution time although others are used such as expenditure rates for resources. Generally, load balancing is divided into static load balancing and dynamic load balancing. Static load balancing assigns tasks to what appears AT THE TIME to be the best processors. Tasks are not moved once initiated under an assumption that the result of their execution will not cause a fundamental shift in the work load on other processors. Distribution of tasks to other processors occurs only when new tasks are introduced. Barhen (1985) states that for precedence constrained tasks this type of load balancing represents the current state of the art.

If task execution results in a change in load necessitating movement of operations to other processors, then the problem involves dynamic load balancing. For example, if a pilot begins a combat function and must shed tasks, the handoff of some or all tasks would constitute a dynamic load balancing requirement. In these cases, load balancing could occur at any time and involve varying numbers of processors depending upon the complexity and resource demands of the task.

Mathematical analysis of static load balancing problems is documented in a variety of references on job shop scheduling (e.g., Coffman 1976). Such analysis has led to models of typical scheduling

situations (Gonzalez 1977). They are usually grouped into taxonomies reflecting a correspondence of job processing configurations with their computational complexity. Graham et. al. (1979) provides a particularly useful taxonomic scheme composed of a set of generic job variables and three multi element fields: alpha, beta, and gamma are used to define processing configurations.

In the following discussion, this taxonomy has been recast to correspond more closely to a human task scheduling environment. We begin with a description of six generic factors usually applicable to all jobs in a system:

(a) Number of tasks.

(b) Processing times on one or more processing units, (a single task may not be executable on only one processor and each processor may require different times for the same job).

(c) The time during which a task can be performed (e.g., selected intervals).

(d) A due date when a task must or should be completed.

(e) A weight giving the importance of the task relative to other tasks that could be executed.

(f) A cost function associated with the completion times for a particular task.

Given values for the generic factors, the processor architecture (or man/machine mix) is then further limited by values in three fields: alpha, beta, and gamma. The Alpha field reflects machine resources or in the case of AAAI, a mix of pilot and machine capabilities (we shall hereafter refer to this as the pilot/intelligent-machine mix or PIM). There are two parts to an alpha field. The first describes the structure:

(a) A single unit such as one pilot or one computer.

(b) Identical parallel PIMs where each unit performs each task in the same amount of time.

(c) Uniform PIMs where each unit type can perform the same tasks but may have different execution rates.

(d) Unrelated PIMs each unit performs at its own rate.

If a single task has multiple steps, a second half of an alpha field is used to specify the type of "shop" required to sequence the task steps:

(a) Open shop all steps must be performed on the same processor in a fixed time, their order is immaterial.

(b) Flow shop all steps performed on the same processor but order is important (e.g., a fixed pilot procedure).

(c) Job shop steps can be performed on different processors but they must still be completed within a fixed time.

The second or Beta field specifies job or task restrictions. We have selected four of the most common:

(a) Job splitting whether tasks can be interrupted and resumed later.

(b) Resource usage what resources a task consumes and whether a single resource can be shared.

(c) Precedence constraints - whether task connections are derived from directed graphs, trees, or are unrelated (this will be discussed in greater detail below).

(d) Availability constraints for tasks, also called release criteria.

The third or Gamma field specifies evaluative measures for assessing PIM performance and are used to define optimization functions. Some measures often used include:

(a) Completion times relative to a schedule (e.g., PERT charts).

(b) Ideal completion times versus actual time (lateness).

(c) Unit penalties if completion time is greater than some criteria.

The above taxonomy permits different mixes of PIMs, task restrictions, and resources to be formally stated using a mathematical shorthand and then identified as members of mathematical problem classes having known computational complexity (e.g., Garey and Johnson 1979). For example, if a PIM design was specified using the above as:

generic = none

alpha = 1 pilot

beta = precedence

gamma = max lateness

or written another way:

PIM = {1P,prec.,L}

The problem would be to minimize the amount of time a pilot falls behind schedule in an precedence constrained serial task where only the pilot performs the task functions. If the problem were instead:

PIM = {unrelated,precedence,sum-completion}

The goal would be to minimize the total time to complete a job on a variable mix of pilot and intelligent machines where preemptive task allocation was allowed. The computational complexity of this type of problem is currently unknown but is conjectured as NP hard. Unfortunately, this configuration is very similar to what is anticipated for possible AAAI designs. The strong implication from comparison with similar mixes is that advanced helicopter designs will have to use optimization methods currently not practical for real time use. If they cannot, a designer must carefully structure task handoff capabilities to avoid a combinatorial scheduling explosion and a resulting system overload. It is not well understood how a designer could anticipate all

such tradeoff situations in advance, but we will now begin to propose some possible approaches to minimize the problem.

I.1. Load Balancing Problems

System designs involving PIMs will inevitably require many tradeoffs involving task handling, resource management, and performance. Rouse (1981) provides an excellent overview of many problems involved in human computer interactions. This report will not duplicate that work but an interested reader is advised to study it for its discussion of dynamic versus static task tradeoffs and human performance models. Techniques that have helped minimize analogous problems encountered in homogeneous multiprocessor computing systems will be emphasized in this report.

To begin, a method is required to avoid the problem Dijkstra (1968) called "lockout". This has also been called the mutual exclusion problem. When a task is split into pieces and distributed among asynchronous processors, it may happen that two or more of these components want to share the same data. If one task component changes the shared data base, errors in partial calculations performed on other processors can occur from temporal sequence effects. A commonly used but ill advised "fix" for this problem is to tag task components so they must wait for other tasks to be completed. However, the waiting period then "locks out" computation on other parallel channels and the system performance slows down to the pace of the blocking element. Dijkstra proposed that three adjustments should be made to the way tasks are divided to prevent this occurrence:

(a) At any given time, tasks should be synchronized so only one task can perform subtasks assessing a previously defined "critical area" of code involving shared data operations.

(b) Data sharing must be defined so stopping task execution outside a critical area will have no impact on any other process.

(c) Every task that needs to execute using a critical area must eventually be allowed to do so.

Clearly, structuring tasks to avoid lock out in serial subtasks will place restrictions on the flexibility of task allocations within a helicopter design. The most obvious implication is that subtasks effecting common data should be performed by the same PIM element if possible.

A second problem involves scheduler loading. In addition to lock out, consideration must be given to the global information used to control selection of tasks for process allocation. One key question concerns how much information is required. Typically, a designer response for many automated systems has been to make all possible information available and rely on the human to filter it out. Hiltz and Turoff (1985) studied human performance for high information bandwidth systems and identified six common but non-optimal human coping strategies to avoid such overload.

(a) They failed to respond to inputs.

(b) They degraded the precision of responses.

(c) They stored data in some temporary way (e.g., notes).

(d) They filtered input.

(e) They recoded input into another form (e.g., labels) or

(f) They just locked out selected information systems.

Although these strategies sometimes worked for a one processor system (the human), most would clearly fail if systems were highly

interdependent. A good example is a command and control network which reduces large amounts of targeting data. In these situations the most common mitigation approach is data filtering. The result (as might be anticipated) is often increasing propagation of incorrect or redundant communication across the network. In purely human teams, human flexibility usually permits a variety of compensation responses as on-the-job experience increases. AAAI designers however will have to address the problem of how an intelligent machine could show similar adaptability. There are some simple machine examples using concepts of linear adaptive filters (e.g., Widrow's work on adaptive antennas) that have been used effectively with raw sensor data. Higher order information processing such as expert systems is another question although certain applications of fuzzy reasoning may be helpful.

A third problem is that information that is irrelevant when it is first filtered may turn out to be critical after the mix of tasks is changed during a dynamic reallocation. Information must be retained for some operations while discarded for others. Although no final solutions exist at present, some designer strategies that may be helpful for minimizing such filtering effects are:

(a) Summarize data.

(b) Increase the rate of information feedback for high risk interactions.

(c) Use categorical sorting for information.

(d) Manipulate message length limitations based on context.

(e) Use dated message purging.

(f) Provide periodic system reminders for key information categories.

Other parts of designs in which humans are information should minimize tasks vulnerable to proven biases (Schwartz, Kullback, and Shrier, 1986). These include:

(a) Confusing covariation with correlation of events.

(b) Confusing consistency in non-independent data sources with greater reliability of data.

(c) Being unduly influenced by extreme values in information (e.g., exaggerated battlefield reports).

(d) Removing uncertainty by ignoring facts.

(e) Finding illusory correlations to support theories.

(f) Improper use of negative information.

(g) Confusing a cue's label with its actual information value.

(h) Poor use of sequential data.

As stated above, there is also a potential for combinatorial explosions when attempting to load balance between a human operator and a machine. Most task scheduling programs such as those used for business planning assume that task demands will remain static, at least as far as load is concerned and that the best criteria to determine schedule optimization is overall job time (e.g., a PERT chart). With a human/machine hybrid however, several other factors need to be taken into account. First there is the question of authority. Not only are processors in a PIM mix likely to differ in capability, but machine components must also remain subordinate to human initiated control. Thus if tasks are handed off to a machine subsystem and contain decision points at which human control authority would be exercised, the machine will have to pause to communicate with the pilot. In this way authority level should become an additional factor in equations used to balance task load.

Further, the throughput of a machine processor can be precisely specified because the performance characteristics of its hardware are known. However as its software becomes more "intelligent" with large numbers of decision branches, recursion, and deductive or inductive reasoning, responses may become very difficult to predict. The prediction process itself may take more time than that which would be lost if a non-optimal schedule was selected. A human element manifests still greater variability. This makes precise prediction of human performance risky at best. It is for this reason that many simulations use random variables to provide reaction times for unknown or poorly understood processes. Unless a functional decomposition of man/machine tasks is unusually successful, it is probable that scheduling will also have to include stochastic elements for the approximation of load balancing times. Such calculations will impose further burdens on scheduling computation methods.

Several additional factors would have to be added to generate a complete human/machine load balancing equation but go beyond the scope of the present paper. These include workload estimation (Casper, Shively, and Hart, 1986), physiological and perceptual latencies, fatigue, task learning level, error probabilities (Chambers and Nagel, 1985), multi-sensor processing capabilities, task durations, precedence graph structure, task interruptability, job periodicity, deadline criticality, and resource availability schedules. Two other factors, communication and task decomposition, will be considered later in this paper. First however, we should consider what methods are available to implement a task scheduler.

I.2. Optimization Using Neural Networks

Given that it is possible to construct a cost function which relates the appropriate factors for human/machine loads, what techniques might serve as options for an AAAI system designer? If we assume for the moment that task decomposition methods provided in the next section will not remove all combinatorial scheduling instances, we are left with the need for techniques capable of providing at least "good" estimates of the solution for n-p complete problems. Graph models and integer 0-1 programming models (e.g., Chu, Holloway, Land, and Efe, 1980) expand in polynomial time and are clearly not suitable. Heuristic models can provide approximate solutions to the task allocation problems and execute more rapidly in time critical or high dimensional cases, but they may lack sufficient precision. An example of this approach is the module clustering algorithm (Efe 1982). Control flow graphs could be constructed to represent PIM interrelationships and then used to calculate communication costs but may not be suitable for highly dynamic environments unless task modules can be decomposed in very specific ways.

What may ultimately have to be used are the slower techniques of combinatorial optimization. One important class is based on a process called simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983) and fast simulated annealing (Szu 1986). Both of these methods take advantage of an analogy between combinatorial optimization in a configuration space (in this case it would be the set of tasks to be assigned to processors) and the stable energy states of atoms undergoing a thermodynamic cooling process such as in the annealing of metals. Recently, the algorithm has been implemented through the application of

a powerful computational technique called simulated neural networks (for a review see Jorgensen and Matheus, 1985). Barhen, Toomarian, and Protopopescu (1987) have combined the two techniques in order to increase the speed of solution convergence which under classical simulated annealing is very slow. Their implementation focused on the scheduling of tasks on a hypercube MIMD computer. The methodology also enabled the efficient use of multiprocessor scheduling for mobile intelligent robots under potentially time critical missions. The method has so far only been evaluated for static load balancing (i.e., the tasks and their dependencies were known apriori) but it is being extended to dynamic load scheduling (Gulati, Barhen, and Iyengar, 1987). Although much faster than existing annealing methods, it may still not be sufficiently rapid for AAAI use. A final decision as to scheduling techniques must await further evaluations. Consequently for the near term we feel it is necessary to emphasize AAAI use design reconfigurations that minimize and hopefully avoid combinatorial explosion even though to do so will mean a loss in potential aircraft performance. Consequently the second part of this report will examine methods through which such reconfigurations may be effected.

I.3. Conclusions Regarding Load Balancing

What conclusions can be drawn from the above discussion that have the most immediate impacts for the AAAI program? First, it is highly probable that without careful limitations on which task handoffs will be allowed, a helicopter design using a machine co-pilot concept will be combinatorially explosive and hence will probably not be capable of effective real-time management by a centralized scheduler. At present,

designs should emphasize task groupings that facilitate flexible task reconfigurations in contrast to largely static task and function allocations currently used by training decision makers. Current methods will probably not provide a designer with enough constraints to accurately specify an optimal human/hardware mix.

Second, if a load balancing scheduler is to be used it should be careful to avoid task interactions caused by communication and sensor data shared resources, and biases introduced by human data filtering and decision making.

Third, although some methods for dealing with combinatorial optimization do exist they have been used largely for static problems, are too slow for real time, or are still under development. Off the shelf methods are not available for PIM problems and designers will have to consider dynamic task reallocation carefully to avoid introducing unanticipated effects during heavy combat loading of the pilot and software.

Computer load balancing in MIMD machines is facilitated by a separation of data from control. This is a more subtle distinction when human tasks are involved. Human knowledge often mixes control information (such as procedures and heuristics) with parametric data for specific situations (instantiations). Computer code oriented toward serial processors can safely mix the two but unfortunately PIM load balancing requires clearer distinctions to be drawn. Consequently, the next section focuses on some insights gained from a study of how computer languages structure their code so they are suited for dynamic task reallocations. It also begins a preliminary analysis of the extent to which these ideas might be transferred to human task analysis.

II. TASK STRUCTURING FOR DYNAMIC REALLOCATION

II.1 Functional Decomposition

In 1972, D. L. Parnas wrote a seminal paper on the criteria to be used in decomposing systems into modules. One of the most influential concepts he proposed was called information hiding. The basic idea was that to maintain usability in the face of a changing environment, situation specific details should be excluded from procedures. These details should be stored in separate areas accessed as data. Parnas argues that even the structure of the data itself might in turn be partitioned. Global tasks or functions were then defined to be "hidden" or distributed into many small independent pieces so as to minimize the impact of a single changed variable on an overall control structure. Effects of change were minimized by limiting the range of impact of any single operation. To implement the idea it is was important to determine how tasks could interact. Thus a taxonomy for task dependency was required. Brodie (1984) detailed one useful dependency concept called functional strength in his discussion of the underlying ideas used in building the FORTH language. We begin with his definition of functional strength.

> Functional strength is a measure of the uniformity of purpose of all activities occurring within a task.

Such a definition sounds nice but to be usable it must also be linked to tests that can be applied by an analyst in practice. One simple test to determine if a function is "strong" was arrived at independently by training developers in describing a well formed task; namely, "can you describe it in a single sentence?" Fortunately, computer science concepts of functional strength have also studied the

problem and identified other forms of interrelationship or "binding" that may also be present when an ideal criteria of uniformity are not met. These should, in principle, be useful for training developers to detect poorly structured tasks.

We will elaborate on other binding types bearing in mind that what computer science calls a function is usually what training developers call a task and the operations in a function are what developers call subtasks.

There are five main types of functional binding that have been identified:

1. Coincidental binding: subtasks are lumped together because they often occur simultaneously (recall in the first section of this report the human bias toward coincident correlation).

2. Logical binding: related parts of a task require a flag (value) in order to select operations (this type of situation means a decision process is nested within the task).

3. Temporal binding: the only real relationship between the parts of a task is that they must occur at the same time.

4. Communication binding: parts of a task all refer to the same data set when they are active (remember we observed the potential risks of such connections when a function was decomposed in multiprocessing in the concept of "lock out").

5. Sequential binding: parts of a task are related by the fact they become the input or output of other parts.

Each of these bindings effects the decomposition of groups of tasks. If the effects can be minimized then the hiding concept of Parnas can be applied to partition tasks for maximum dynamic flexibility. Presumably, the designer would then gain greater opportunity to maximize overall system performance.

A second useful concept comes from the literature on structured software design (Yourdon and Constantine, 1979). This deals with the

external relationships groups of functions can have to each other. This concept called "function coupling" may be defined as follows:

> Function coupling is a measure of how functions influence the behavior of each other.

Function coupling is important because individual tasks having external coupling are restricted in how they can be decomposed. This is in contrast to functional strength where the emphasis is on internal relationships within task operations. The concept provides some criteria which could be applied to identify task situations which would influence PIM load balancing. Four types of coupling are as follows:

1. Code modification coupling: One task actually changes the code (or in the case of human tasks the procedural steps) of another.

2. Control cue coupling: One task controls flags or decision cues used by another task.

3. Data coupling: One task passes data other than control data to another task. (Such local data coupling is better than global data coupling. Generally this type of coupling is acceptable for dynamic task handoff procedures and can be included in a PIM design).

4. Parameter pass coupling: This may well be the nicest type of function coupling because only values are passed as arguments when called.

There are also factors which influence each of these types of coupling and how important it is to modify a particular task. These include the number of connections linking tasks and whether they include internal influences, the complexity of the data exchange going on, i.e. the number of items being passed, the type of information, and how long the tasks remained coupled during a changing scenario. Some guidelines for decoupling tasks include standardizing intertask connections, task grouping, and localization of task effects.

With the above concepts of information hiding, functional strength, and coupling, what are tests which an AAAI designer should apply for task decomposition in PIM's? Ignoring the clearly static tasks, the following are proposed as part of a design procedure.

II.2. Diagnostic Tests for Poor Functional Decompositions

We begin with questions that can be used to identify poorly written task lists (possibly provided to a designer by a second party outside the design process) and follow them by rules which may help if further task decomposition must be made by the designer. Although coming from the computer literature, the similarities to guidelines for human task analysis are interesting (e.g. Rogoff 1987).

1. Does the task description have to be a compound sentence?
2. Must it use time indexed words such as first or next?
3. Does it have a nonspecific object following the verb?
4. Does it use a general verb-like "initialize" which implies multiple functions are actually going on?
5. Can it be rewritten to produce minimum redundancy with other tasks?

We should also provide principles to AAAI designers for decomposition if tasks fail the above questions. Below are principles that have proven valuable in structuring distributed computer codes:

1. Be requirement driven, determine functional components based on their actual need to execute.
2. Group all functions involved in communicating data in a common interface area.
3. Define all shared data in terms of objectively measurable terms.
4. Look for areas most likely to be effected by change.
5. Look for areas having the most impact on other tasks.

6. Make sure tasks do not depend upon fluctuating system states (if possible).

II.3. Impacts for Traditional Task Analysis

We should also consider the impact the above considerations may have on traditional task analysis and how these concepts are similar or different from existing procedures. In effect an additional step is being added to task analysis. Hopefully, tasks decomposed into dynamically insensitive pieces will be easier to learn and result in improved man/machine designs. This process is not easily categorized in a traditional taxonomic sense. Data on task sequence, hierarchy, and input/output are usually kept separate from task duties. Situation specific data will become separated from tasks but force task changes in specific situations (e.g., a mission scenario).

Traditional task analysis usually places an artificial hierarchical decomposition on task dependencies whether or not a scenario implies it. Current techniques used for task analysis thus cause a distortion of training in at least three ways:

1. It implies that training courses (and associated pilot learning) should follow a hierarchical structure.

2. It implies levels in artificially imposed task hierarchies can and should be separated into instructional modules. (This discourages recognition of utilization patterns common across multiple contexts).

3. It implies personnel requirements vary due to different hierarchical levels that may not exist in the real world problem domain.

A second question is: "Given the use of object oriented language in the AAAI models are the above requirements readily captured by object oriented programming?" The answer is a qualified no. The "object" of object oriented programming is a portion of code that can be invoked by

a single name. To paraphrase Brodie (1984), such code can perform more than one function. To select a particular function, an object is invoked and passed a set of parameters. Regarding information hiding, object oriented programming involves a similar philosophy because program change is much easier but there are also differences:

1. Objects often contain complex branching structures to determine which of many functions they should perform. The highly modularized task decomposition approach proposed above separates control data from tasks and invokes functions directly.

2. Objects usually are written as self-contained entities, thus they duplicate code. An approach which atomizes operations such as that proposed above facilitates redundancy reduction and thus the recognition of common elements between tasks (for example, as in the tight compilation characteristics of FORTH code).

3. An object is defined to work with a predetermined set of cases. It is difficult to add new scenarios whereas functionally independent units are easily extensible. In the case of human tasks, this might imply a way to identify which new tasks can be added to a job with minimum retraining.

As can be seen, if AAAI is to use the full power of functional decomposition it must be able to separate data from procedures. Thus, it is also important to determine what methods of data decomposition might be available.

II.4 Data Separation

We can think of at least two instances where a PIM would be directly impacted by the effective separation of task functions and control data: first, executing a previously trained procedure on new equipment, and second attempting to switch between different operational modes as conditions change (as might occur when a task was handed off between two PIM elements of different computational power.) First, as we did for function decomposition, it is useful to define questions that

might be used to assess the impact of required information on the allocation of data used by dynamic functions. The following are suggested:

1. What is the probability that data will be reused?
2. What is the probability of procedural changes?
3. What is the smallest set of interacting data?
4. Can structures and algorithms that share knowledge about how they collectively work be grouped?
5. Can redundancy be minimized?

Although these questions are helpful in identifying which elements of data could be clustered into smaller units, they say little about how to best represent information used by a decomposed task. Several frameworks have been proposed each of which has strengths and weaknesses. Structured english offers precision but can often be too hard to read quickly when human data demands are high. Decision trees do a nice job of representing logical branching but can be very hard to simplify as witnessed by standard task analysis. It appears at present that decision tables may be the best compromise. Some of their features include a clear graphic representation, ability to easily transfer to machine coding schemes, and a structure that permits backtracking of task calls. What the best graphic display method is for the use of decision table data or whether a tabular structure may best be left invisible to the pilot is an open question.

We have now considered how tasks can be decomposed, how information may be represented in data structures, and how to find which tasks may be interrelated. One remaining area not discussed so far concerns communication requirements during load balancing. The final section will consider that problem.

II.5. Communication

Symbiotic integration of man and intelligent machine has an assumption that effective communication can take place between all entities involved. In the case of humans, such communication requires team or group cooperation. The study of team performance enhancement has been a major concern of organizational psychology for many years and has been recognized as a difficult topic to quantify because of individual differences. As complex as these interactions are for humans however, people have certain common factors which at least permit interactive attainment of shared goals. PIMs however create a new situation.

One problem is how to assure a common frame of reference. Like a science fiction character attempting to communicate with totally alien species, there is no guarantee that different sensing and cognitive mechanisms of pilots and machines share common frames of reference. For example, imagine trying to communicate the different Eskimo words for snow (significant for an arctic seal hunting culture) to an Australian aborigine whose environment is desert sand. Or consider trying to cooperatively hunt BISON bombers with a silicon based life form (a PIM computing element?) which has microsecond responses and sees in the infrared spectrum.

As mentioned in the introduction, it is adaptability which offers a research path to overcome such large mismatches between men and machines. In scenarios where environments can be anticipated and common task requirements can be designed into an intelligent machine, the need for adaptive intelligence can be minimized. But, as the nature of team interactions become complex, such as where PIMs share work and

authority, required levels of intelligence and adaptability must be considered carefully. Central to both is a machine that learns about its PIM partners and communicates to obtain information supporting that learning. A tradeoff occurs in that communication is usually resource intensive and runs counter to the second tenet of load balancing which is to minimize communication.

As discussed above, one way this problem is handled in multiprocessing systems is to remove control structures from functions and embed them in shared data bases. The main factor forcing control structure isolation is exponential complexity caused by multiple decision branches. That complexity tends to make procedures rigid and unadaptive. In particular, control structures that depend upon locally computed data force sequentiality while control structures with many logical branches become serially linked and hence subject to the binding and coupling described above.

Whether one can separate control structures from procedures to minimize communication in human tasks is still a speculative issue at present. However an AAAI designer might try the following techniques:

(a) Replace decisions with tests for values that can be looked up in an external data base (much like a blackboard logic).

(b) Drive control decisions from external decision tables rather than if-then conditionals in task procedures.

(c) Group rare decisions together. Separate them from other functions. Examine whether components of the special case can be categorized using previously defined areas.

II.6. Conclusions About Decomposition

We now draw conclusions about the topic of task decomposition and passing tasks between multiprocessor systems. We began with benefits such an approach has for handling of an environment with rapid changes

such as those in the Executive Summary. First, environmental change under a strategy of maximum functional decomposition impacts only a limited subset of the total set of job tasks and should not ripple across the entire interactive PIM structure.

Second, functional decomposition ideas have already proved useful for multi-programmer development of large integrated software systems. In these projects, code is focused on a broad problem objective even though written autonomously by many different programmers. Many of the underlying code management techniques also seem suited to a reversal of the usual process where predetermined man/machine interactions must be divided into cooperative but independent units.

Third, there are some possible guidelines that could be used to obtain such properties. Summarized by their general area the proposed ideas were:

(a) Implicit function calls;

(b) Implicit data passing;

(c) Direct access of memory (possible with machine systems but requiring data base access or external memory for humans such as a dynamically changing displays);

(d) Tasks defined in terms of immediately executed operations; and

(e) Communication not as data passing but rather command passing.

The latter occurs because in a functional decomposition approach, communication does not exist merely to pass data, because data calls are implicit. Rather communication enables tasks to be performed. Communication loads are IMPLICIT in such a task definition. The hidden limitation of current communication methods where passing raw data between men and machines occurs is that procedures that require detailed

advanced planning become counterproductive. What is usually needed most in battlefield environments is flexibility. Traditional methods of procedural specification force rigidity because they are not organized with change as a central concept.

A fourth point concerns where the control of task scheduling should reside. In MIMD machines the scheduler is part of the computer and the channel for all data flow. It interprets data in exactly the same way as the processors, and receives the output of the processors prior to any interface with the outside world. In contrast, a human pilot does not receive (in fact may actively not desire) all information coming in from the outside world. He may perceive the world in a focused or biased manner relative to intelligent support systems, may not have direct control over the interface of those systems with the environment, and may or may not choose to interact with supporting hardware and software.

Thus, in order to be queried, world models used by intelligent subsystems must be compatible with the world model of the user. This is a special type of communication problem and focuses attention upon the data exchanged between the pilot and the machine support system as well as the location of that data (in the pilot's mind, an external data base, residence on a particular processor, or embedded in computer code). This report primarily stressed the minimization of communication through restructuring of tasks; specifically, how decision branches might be separated from algorithms, and algorithms from data in such a way as to facilitate dynamic allocation of tasks.

In the first section we presented a taxonomic structure for describing design situations. Categorization of PIMs used a three field

classification scheme composed of: (1) task data (e.g., number of operations, processing times, release dates for activities, importance weightings, and open shop, flow shop, or job shop environments); (2) relationships among task units like preemption, precedence, and processing time constraints; and (3) optimality criteria (which specified the cost functionals defining the optimization).

The result of a taxonomic analysis was that the dynamic man/helicopter interface problem was probably of a mathematical class that was n-p complete (although this was conjectured not proven). That is the scheduling decision problem as defined by the most probable architecture for asynchronous, non-homogeneous, real time processes in a dynamic environment was combinatorially explosive. Optimized global task allocation would require methods capable of solving such problems. Unfortunately the existing computational methods are limited. One current research approach that showed promise was the use of simulated neural networks. We caution however about near term limitations of the approach and suggests that until the technology is fully developed the pilot/helicopter task domains may initially be better served by constraining allocations by limiting schedule conflicts, i.e., hard divisions of labor defined on functionally decomposed tasks.

It is clear that many aspects of human/helicopter interaction present a challenging problem for AAAI. What is equally clear from an analysis of problems encountered in multiprocessor scheduling is that many insights remain to be gained from a formal study of communication structures, data flow, and functional decomposition. A workstation for future helicopter designs provides an excellent testbed for that analysis.

REFERENCES

1. Barhen, J., "Combinatorial Optimization of the Computational Load Balance for a Hypercube Supercomputer," Proceedings of the Fourth Symposia on Energy Engineering Sciences, Argonne National Laboratory (1986).

2. Barhen, J., "Load Balancing in Hypercube Multiprocessors via Simulated Annealing," CESAR Research Report, Oak Ridge National Laboratory, 1985.

3. Barhen, J., J. R. Einstein, and C. C. Jorgensen, "Advances in Concurrent Computation for Machine Intelligence and Robotics," Proceedings of the Second International Conference on Supercomputing, Vol II, pg. 84-97 (1987).

4. Brodie, L., "Thinking Forth," Prentice-Hall (1984).

5. Casper, P. A., R. J. Shively and S. Hart, "Workload Consultant: A Microprocessor Based System for Selecting Workload Assessment Procedures," Proc. of the International Conf. on Systems Man and Cybernetics (October 14, 1986).

6. Chambers, A. B. and D. C. Nagel, "Pilots of the Future: Human or Computer," Computer (1985).

7. Chow, Y. C. and W. H. Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System," IEEE Trans. Comp., Vol. C-28 (1979).

8. Chu, W. W, L. J. Holloway, M. Lan and K. Efe, "Task Allocation in Distributed Data Processing," Computer (November 1980).

9. Chu, W. W. and M. T. Lance, "Task Allocation and Precedence Relations for Distributed Real-Time Systems," IEEE Transactions on Computers, Vol. C-36, No. 6 (June 1987).

10. Coffman, E. G., "Computer and Job Shop Scheduling Theory," New York, John Wiley and Sons (1976).

11. D'Ambrosio B., P. Raulefs, M. R. Fehling and S. Forrest, "Real-Time Process Management for Materials Composition in Chemical Manufacturing." To appear IEEE Expert (August 1987).

12. Dijkstra, E. W., "Co-operating Sequential Processes," In Programming Languages, F. Genuys Ed., Academic Press, pg. 43-112 (1968).

13. Efe, K., "Heuristic Models of Task Assignment Scheduling in Distributed Systems," Computer (June 1982).

14. Garey, J. R. and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman (1979).

15. Gonzalez, M. J., "Deterministic Processor Scheduling," Computing Surveys, Vol. 9, No. 3. (September 1977).

16. Graham, R. L., E. L. Lawler, J. K. Lenstra and A.H.G. Rinnooy Kan, "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. Annals of Discrete Mathematics, #5, 169 (1978).

17. Greenstein, J. S. and R. E. Revesman, "Two Simulation Studies Investigating Means of Human Computer Communication for Dynamic Task Allocation," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 16, No. 5 (1986).

18. Greenstein, J. S. and S. Lam, "An Experimental Study of Dialogue Based Communication for Dynamic Human Computer Task Allocation," Int. J. Man/machine Studies, #23, 605-621 (1985).

19. Gulati, S., J. Barhen and S. Iyengar, "The Pebble Crunching Model for Dynamic Load Balancing in Homogeneous Computation Ensembles," Submitted to IEEE Transaction on Computers (July 1987).

20. Hiltz, S. R. and M. Turoff, "Structuring Computer Mediated Communication Systems to Avoid Information Overload," Comm. of the ACM, Vol 28, #7 (1985).

21. Jorgensen, C. and C. Matheus, "Catching Knowledge in Neural Networks," AI Expert (December 1986).

22. Kirkpatrick, S., S. Gelatt and M. Vecchi, "Optimization by Simulated Annealing," Science, 220, 671 (1983).

23. Kleinrock, L., "Distributed Systems" Comm. of the ACM, Vol. 28, #11, (1985).

24. Parnas, D. L., "On the Criteria to Be Used in Decomposing Systems into Modules," Communications of the ACM (December 1972).

25. Rogoff R. L., "The Training Wheel," John Wiley and Sons (1987).

26. Rouse, B. R., "Human Computer Interactions in Multi-task Situations," IEEE, SMC-7, No. 5 (1977).

27. Rouse, B. R., "Human Computer Interaction in the Control of Dynamic Systems," Computing Surveys, Vol. 13, No. 1 (1981).

28. Schwartz, J. P, H. H. Kullback and S. Shrier, "A Framework for Task Cooperation within Systems Containing Intelligent Components," IEEE Trans. on Systems, Man, and Cybernetics, Vol. 16, #6 (1986).

29. Szu, H., "Fast Simulated Annealing," in Neural Networks for Computing," J. Denker Ed., American Institute of Physics (1986).

DISTRIBUTION LIST

Internal Distribution

1. S. M. Babcock
2. J. Barhen
3. D. L. Barnett
4. M. Beckerman
5. G. de Saussure
6. C. W. Glover
7. W. R. Hamel
8. J. P. Jones

9-13. C. C. Jorgensen

14. F. C. Maienschein
15. R. C. Mann
16. E. M. Oblow
17. L. E. Parker
18. C. R. Weisbin
19. F. G. Pin
20. A. Zucker
21. EPMD Reports Office
22. Central Research Library
23. ORNL Technical Library - Document Reference Section

24-25. Laboratory Records

26. ORNL Patent Office
27. Laboratory Records - RC
28. J. J. Dorning (consultant)
29. G. H. Golub (consultant)
30. R. M. Haralick (consultant)
31. D. Steiner (consultant)

External Distribution

32. Office of the Assistant Manager, Energy Research and Development, DOE/ORO, Oak Ridge, Tennessee 37831.

33-37. Dr. James Hartzell, NASA Ames Research Center, Life Sciences Building, MS239-2, Moffett Field, CA 94035

38. Dr. Ewald Heer, Heer Associates, Inc. 5329 Crown Avenue, LaCanada, CA 91011.

39-40. Dr. Robin Keesee, Director SRL, U.S. Army Research Institute, 5001 Eisenhower Boulevard, Alexandria, VA 22333.

41. Dr. John O'Hare, Office of Naval Research, Engineering Psychology Program, Code 1142EP, 800 N. Quincy Street, Arlington, VA 22217-5000.

42. Richard Reynolds, Human Factors Division, Code 712, Naval Training Systems Center, Department of the Navy, Orlando, Florida 32813-7100.

43-72. Technical Information Center, Oak Ridge, TN 37830.