S U M E X

STANFORD UNIVERSITY
MEDICAL EXPERIMENTAL COMPUTER RESOURCE

COMPETING RENEWAL APPLICATION
RR - 00785

BOOK I
RESEARCH PROPOSAL

Submitted to

BIOTECHNOLOGY RESOURCES PROGRAM
NATIONAL INSTITUTES OF HEALTH

June 1, 1977

DEPARTMENT OF GENETICS
STANFORD UNIVERSITY SCHOOL OF MEDICINE
Joshua Lederberg, Principal Investigator

Table of Contents

BOOK I

TABLE OF CONTENTS

BOOK I (continued)

TABLE OF CONTENTS

BOOK I (continued)

TABLE OF CONTENTS

BOOK I (continued)

List of Figures

Table of Contents

BOOK II

TABLE OF CONTENTS

BOOK II (continued)

II. RESEARCH PLAN - BOOK I

This is an application for renewal of a grant supporting the Stanford University Medical EXperimental computer (SUMEX) research resource for applications of Artificial Intelligence in Medicine (AIM). The research plan has been divided into several logical parts:

1) Book I - Resource research objectives and rationale, progress report, and detailed research plans.

2) Book II - Biographical sketches, collaborating project reports and plans, and supporting appendixes.

3) Budget - First year budget detail, five-year budget summary, and budget explanation and justification.

1 BACKGROUND AND PROPOSED WORK

1.1 OVERVIEW OF OBJECTIVES AND RATIONALE

The SUMEX-AIM project is a national computer resource with a dual mission: 1) the promotion of applications of artificial intelligence (AI) computer science research to biological and medical problems and 2) the demonstration of computer resource sharing within a national community of health research projects.

In the body of this proposal, we offer definitions and explanations of these efforts at several levels of detail to meet the needs of reviewers from various perspectives. For this overview, we give only a brief summary of our recent accomplishments, present status and expectations for the requested term of the renewal, the five years beginning August 1,1978.

Definitive funding of the SUMEX-AIM resource was initiated in December 1973. The principal hardware was delivered and accepted in April 1974, and the system became operational for users during the summer of 1974. The present renewal is therefore written from a perspective of just short of three years of experience in attempting to develop and serve the user community for the resource.

The original SUMEX proposal was an outgrowth of two lines of endeavor at Stanford that had been supported by the Biotechnology Resources Program. The ACME project (Advanced Computer for MEdical Research), 1965-72, had introduced the innovation of interactive time-shared computing to the medical research community at the Stanford Medical Center. Based on an IBM 360/50 with mass core storage, this system was notable for the ease with which physicians and scientists, previously inexperienced with computers, were able to learn a variety of applications with minimal help from professional programmers. With the further development of the technology, and the rationalization of computer support functions at Stanford, this system was eventually integrated with the university-

wide time-sharing service. While ACME had some shortcomings as a production (contra development) tool many of our colleagues at the medical school still look back regretfully at having lost it as a medical-school-dedicated system tuned to their special needs. The second line, the DENDRAL project, is a resource-related project connected with applications of artificial intelligence to problems of molecular characterization by analytical instruments like mass-spectrometry, gas-chromatography, nuclear magnetic resonance, and so on.

In 1972 we applied to NIH for the establishment at Stanford of a next generation computer resource to supplant ACME for applications for which the university-wide facility was inadequate. The DENDRAL project was the central source of this initiative; several others entailing real-time instrumentation as much as AI needs were also specified. During the subsequent 18 months, we entered a phase of protracted review and negotiations with BRP and its advisory groups, from which emerged the policy determination that resources of this scope were best justified if they could be functionally specialized, but geographically generalized. The emerging technology of computer networking opened an opportunity to demonstrate this model in a way that could serve both local and national needs. With all of this in mind, we were happy to undertake the responsibility of such a demonstration, which seemed important as a step in community-building as well as in providing the computing resources so urgently needed for our own and others' research efforts. In many respects it would have been far more convenient to focus on our own requirements, but the satisfaction of these seemed both infeasible and too limited an aspiration in the face of the suggested opportunity.

Three years is hardly long enough for a conclusive determination of the success of such a model, though we can fairly take pride in the diligence and technical competence with which we have responded to the community responsibilities mandated by the terms of the award. An important element in satisfying those responsibilities was the establishment of a mutually satisfactory management structure, on which we report in further detail below. Good will and common purpose are of course the indispensable ingredients, and we are grateful to have been able to offer this service in a congenial framework, and at the same time to be able to support our local computing research needs.

Our technical task has been achieved: to collect and implement an effective set of hardware and software tools supporting the development of large and complex AI programs and to facilitate communications and interactions between user groups. In effect, users throughout the country can turn on their own teletype or CRT-display terminals, dial a local number, and logon to SUMEX-AIM with the same ease as if it were located on their own campus -- and have access to a specialized resource unlikely to be matched nearby. From the community viewpoint, we have substantially increased the roster of user projects (from an initial 5) to 11 current major projects plus a group of pilot efforts. Many of these projects are built around the communications network facilities we have assembled; bringing together medical and computer science collaborators from remote institutions and making their research programs available to still other remote users. As discussed in the sections describing the individual projects, a number of the computer programs under development by these groups are maturing into tools increasingly useful to the respective research communities. The demand for production-level use of these programs has surpassed the capacity of the present SUMEX facility and has raised the general issues of how such software systems can be optimized for production environments, exported, and maintained.

The principal thrust of this renewal proposal is to sustain the momentum of SUMEX-AIM, both as a facility and as a community, during a period of rapid change in the technology and economics of computers. For reasons that will be justified in more detail, we do not plan further major expansion of centralized hardware at SUMEX, believing that growing community needs should now be met as justified at distributed nodes. It is difficult to make firm predictions of the technological changes that will present themselves during the period of the grant, but it may be that some conversion of the system will be necessary if only to keep pace with the software exchanged with cognate communities.

More concretely, our objectives for this next grant term include:

1) Maintaining the vitality of the AIM community of projects. This will entail scrutiny of old and new projects in what is approaching a steady-state of maximum capacity, and improving the efficiency with which developmental programs can be furnished to medical research groups.

2) Continued computational support for the AIM community based initially on our existing KI-10 facility. We expect the computing hardware technology to change substantially in the next few years with the availability of both more powerful and smaller and cheaper machines. Additional large-machine resources may still be necessary to meet the growing needs of the community during this period. As already stated, this kind of growth should be implemented at sites other than Stanford, but can be embraced by the same management structure as governs SUMEX-AIM. We plan to study these new technological alternatives affecting our central facility and to attempt to maintain software compatibility for our dual KI-10 system. Only should this prove untenable or grossly inefficient will we consider a hardware conversion to a more directly compatible implementation.

3) Continued work to improve system software and communication facilities for community interactions and the dissemination of programs. This will include advantageous connections to emerging communications networks and administrative efforts to exploit community expertise and sharing in software development.

4) Core research work to explore ways of exporting complex AI programs including new language support (MAINSAIL), specialized satellite computer systems, the use of networks for software dissemination and maintenance, and examinations of more operationally efficient implementations of AI programs. We will continue to work closely with the XEROX-PARC group, which remains primarily responsible for maintaining INTERLISP.

5) Core research work to attempt to generalize and document AI tools that have been developed in the context of a number of individual application projects. This will include work to organize the present state-of-the-art in AI techniques and tools through the AI-Handbook effort and the development of generalized software packages for the acquisition, representation, and utilization of knowledge in AI programs. These packages will facilitate the exploration of new areas of application of these tools.

1.2 SIGNIFICANCE

Viewed in the narrowest definition of a biotechnology resource, SUMEX-AIM is justified by the technical capabilities it offers for the pursuit of research using advanced computer applications relevant to the NIH mission. The progress reports of the various user projects speak for themselves in the diversity and pertinence of the work accomplished. We do not underestimate (and share as a grave responsibility) the overall investment charged to the resource; but this is quite reasonable when apportioned over the whole range of projects. The shared resource is plainly far more economical than any alternative method of providing comparable facilities to such a range of users distributed over the country.

Similar considerations apply to a variety of other kinds of research hardware. Unique to the computer is the extent to which shared hardware contributes to methodological cooperation; what in this context we call software compatibility. This follows from the unparalleled complexity of computer programs as process-specifications. What other techniques are or can be formulated as recipes of 100,000 or more instructions, each of which must be faithfully executed or the whole system will collapse? Yet we know that a great deal of our knowledge, e.g., in medical diagnosis, may prove to be of similar complexity when explicitly and formally expressed. We infer that many fields of scientific inquiry will have to use similar methods of exchange of critical commentary; that the electronic communications of computer programs is a prototype for the maintenance of other knowledge bases essential for the fabric of a complex and demanding society. The computer is at one time the node of a knowledge-sharing network, and the device for verifying the consistency and pertinence of the updates and criticisms that the users remit. Thus we can view our resource as exemplifying a technology that induces a new social organization of scientific effort (we would not be the first to recall Gutenberg; and to view ourselves as analogs of some of the early experiments with the use of the print medium for journals and academies.) From this perspective, it is quite fitting that the initial grant that established SUMEX-AIM was attended by so much preoccupation with managerial design, not ordinarily the favorite occupation of scientific types.

Several concrete illustrations of the encouragement of dynamic criticism that enhances the robustness of shared knowledge can be elicited from current projects (see Section 6 on page 41 in Book II), apart from the most familiar instances of sharing of software over the computer networks. The MYCIN rule bases, and the text of the AIHANDBOOK are continuously updated by critical users and reviewers. In fact, the text of various parts of this proposal went through dozens of iterative revisions, with comments from many interested groups, within the several weeks that were dedicated to its preparation. Another, and one of the most interesting examples, was the experimental use of the CONGEN program (See the DENDRAL progress report on page 42 in Book II) in a graduate class in advanced organic chemistry taught by Professor Djerassi. Each of 25 students scanned the recent literature for claims of new structures whose proofs were deemed to be interesting or dubious or both. Five examples were selected for exhaustive reexamination by the students. In each case, the published proof was found to be defective when it was checked by CONGEN -- alternative structures having been overlooked by the authors that still gave good fits to the given data. These and several comparable examples of asserted scientific fact are being more carefully reexamined in the authors' laboratories in response to the

program's refutations. In due course, we believe this kind of mechanized checking of "proofs" of chemical structures will be a routine part of the peer-review critical function of the editorial staff of the journals. These advances are facilitated by the tight internal cohesion of argument in structural organic chemistry, compared to other scientific fields -- precisely why this scientific domain was the one chosen for our initial work on applied AI.

The technical and sociological implications of our program are in fact elaborated throughout this proposal. By contrast, this may be the place to digress with some more personal observations (in the voice of the principal investigator) about the need for scientists to attend more self-consciously to the process of science itself, and to the political questions of social choice that are part of the accountability of science, to offer due return for value received.

Although SUMEX-AIM is rooted in the sub-discipline of "Artificial Intelligence" we understand and share the discomfort that many bystanders have in trying to give it a precise definition. It might have been preferable to think of "knowledge-engineering" as the thread that links almost all of our projects. This has connotations that might recall "data-base-management"; and we should not disparage the role that efficient systems for retrieving complex data will have in our effort. But our task is not usually to maintain a telephone-directory with yellow pages, but instead to gather, test and validate a hierarchy of generalized rules that operate both on each other, and on data of the kind that are the province of the information-retrieval subdiscipline. The development of the computer programs to perform these operations is the software-science part of our effort. Behind it is necessarily a new level of focussed inquiry into the rules of scientific inference in detail that could only be cross-checked by interaction with the machine.

We are traversing a time when the very justification for basic research is under critical, often even hostile scrutiny. Many quarters are asking such questions as "How much of the health progress of the past 30 years can be attributed to advances in knowledge connected with NIH-supported research?" Are our institutional arrangements and patterns of funding really the most appropriate for the most efficient 'transfer of technology' from the basic laboratory 'to the bedside'?" Less often raised by external critics is, "To what extent does the present system support the most fundamental innovations within science itself; or does it inevitably focus overwhelming support on the most obvious, transparent questions and discourage more revolutionary kinds of inquiry?"

Within the NIH directorate, it has been stipulated that "Currently, within the research community, formal processes are lacking to assure systematic identification and evaluation of clinically relevant research information, and its effective transfer to the health care community...."

It is not always popular to insist that these questions must be faced up to -- that basic science cannot indefinitely subsist on unconfirmed faith as to its promise. Furthermore, it is easy to show that many short-term advances have arisen from the most pragmatic kinds of investigation: empirical screening for antibiotics or antidiuretics has undoubtedly generated more life-saving therapeutic products than the most sophisticated molecular biology, up to the

present moment. Indeed, salt-water, intelligently administered, has been one of the great life-savers of the recent era! On the other hand, I hold that it would be tragic to undermine the enormous long range potential of basic insight without a deeper analysis of the process by which knowledge and insight move from basic science into clinical problems; and we just might find some ways to improve the system without wrecking it!

These remarks should be taken as exposing a philosophical preoccupation rather than as the design of a research program. They do relate to efforts like the MOLGEN project, which include a great deal of focussed introspection on the intellectual substance of scientific inquiry. It would be premature to claim that computer programs per se will soon be delegated the major responsibility for "systematic identification of relevant knowledge", although they can already play a very helpful role in assisting human intelligence to correlate bibliographic data, and in other ways. However, the very process of implementing an "applied philosophy of science", which is the principal forework of developing a domain for the application of knowledge-based AI, is exactly the kind of formal systematization called for in these renewed efforts to facilitate technology transfer to health care. Longer range success in our AI research will be as important in helping us understand what we are doing as scientists and diagnosticians as in providing mechanical assistance to these ends.

Although our substantive efforts are mostly concerned with the "micro-problems" of scientific or clinical inference, there may be more important treasures in a macro-perspective on the integration of knowledge in medicine. My own most important laboratory accomplishments have all concerned the discovery of new problems, and the bringing together of previously disparate disciplines, rather than the solution of extant puzzles -- the discovery of sex in bacteria, better viewed as the marriage of genetics and bacteriology is perhaps the least controversial instance. I believe that it is reasonable to expect that the systematization of biomedical knowledge, to which computer AI will make an indispensable contribution, is an important side effect of these investigations in knowledge-engineering; and that this will lead in turn to the recognition of holes in the overall fabric that badly need patching.

We have too little theory of the practice of science to offer more than case studies at this time -- I have been spending some time in collaboration with a historian and sociologist in trying to achieve a better understanding of the dynamics of discovery of bacterial recombination, and found there is more to the context of that story than my own ingenuity. But it is also very difficult to reconstruct such events without critical recordings of the incidents as they occur -- recordings we are learning how to make in the MOLGEN work. [** Copies of a working paper illustrating this are available on request. **]

To turn to a more clinically urgent arena, it is somewhat dismaying to recall that it took 35 years from Beadle and Tatum's discovery of nutritional mutants in Neurospora to the beginnings of the biochemical genetics of such important situations in man as atherosclerosis. I do intend to initiate some inquiry as to the inevitability of delays of that kind, which seem retrospectively absurd. We will not get analytically persuasive or policywise sound determinations of such questions without more attention to the underlying process of scientific inquiry than unselfconscious scientists are customarily wont to indulge in.

This kind of speculation can also be translated into concrete research programs, which in turn may evoke some new principles. Kidney-stones are an unlikely arena of concern for someone of my particular scientific background: but a number of issues have emerged in consultations with some of my colleagues in the Stanford Division of Urology. There has been substantial evidence for some time of a significant genetic factor in chronic recurrence of stones. This does not seem to be correlated with overall rates of calcium oxalate excretion; indeed one must focus on the stone as a pathological form of crystal aggregation -- much larger quantities of calcium oxalate are passed as microcrystals by normal individuals. Several workers have identified mucopolysaccharides in the matrix of these stones, and some have speculated about their possible role as initiators or cements in stone formation. On the other hand, geneticists have long known that blood-group substances, (mucopolysaccharides!) appear in the secretions, including the urine, of the Se/se and Se/Se [Secretor] genotypes; although saliva is the preferred sample for diagnosis. Still another worker, a pathologist, has remarked on the occurrence of mucopolysaccharide concretions in the tubules near the renal papillae of Se/se subjects. To the best of my knowledge, these disciplinary nuggets have been privately and separately held, and there has been no effort to study their possible interconnection. A survey is now underway at Stanford to test a possible statistical association of Secretor and blood group type with stone recurrence.

These suggestions were arrived at through interpersonal discourse, experts from different disciplines being able to furnish provocative data points when prodded by a more general inquiry. Could one imagine a more general problem-generator that could arrive at similar conclusions? Perhaps so -- one could parse through the medical subspecialties, or through significant diseases, to ask more systematically if they had been scrutinized from the perspective of, say, biochemical genetics. And this raises many other hypothetical inputs to a combinatorial-generator of potential, new interdisciplines. One hastens to add, that most of the rotely drawn intersections will be meaningless or empty -- enough perhaps that the whole game may end up looking quite silly. However, the problematics of the game have not been explored, and to that extent, there is a pilot project here that I intend to pursue. Its practical feasibility will depend in part on the briskness with which relevant data can be fetched from the literature and from other experts, and I will be exploring possibilities of on-line access to bibliographic databases 1) to help support this effort, and 2) to suggest further research efforts in the use of AI techniques for bibliographic inquiry in ways that may be pertinent to macro-policy of research management.

1.3 BACKGROUND AND PROGRESS

1.3.1 PROGRESS SUMMARY

This progress summary covers the period from December 1973, when the SUMEX-AIM resource was initially funded, through April 1977. During this period we have met all of the defined goals of the resource:

i) We have established an effective computing facility to support a nation-wide community of medical AI research projects including connections to two computer communication networks to provide wide geographical access to the facility and research programs.

ii) We have actively recruited a growing community of user projects and collaborations. The initial complement of collaborators included five projects. This roster has grown to eleven fully authorized projects currently plus a group of approximately six pilot efforts in various stages of formulation. Recruiting efforts have included a public dedication and announcement of the resource, NIH referrals from computer-based project reviews, direct contacts by resource personnel and on-going projects as well as contacts through the AIM workshop series coordinated by the Rutgers Computers in Biomedicine resource under Dr. Saul Amarel.

iii) We have established an AIM community management structure based on an overseeing Executive Committee and an Advisory Group to assist in recruiting and assessing new project applications and in guiding the priorities for SUMEX-AIM developments and resource allocations. These committees also provide a formal mechanism for user projects to request adjustments in their allocated share of facility resources and to make known their desires for resource developments and priorities.

iv) SUMEX user projects have made good progress in developing more effective consultative computer programs for medical research; one of the major goals toward which our AI applications are aimed. These performance programs provide expertise in analytical biochemical analyses and syntheses, medical diagnoses, and various kinds of cognitive and affective psychological modeling.

v) We have worked hard to build system facilities to enable the inter- and intra- group communications and collaborations upon which SUMEX is based. We have a number of examples in which user projects combine medical and computer science expertise from geographically remote institutions and numerous examples of users from all over the United States and occasionally from Europe experimenting with the developing AI programs. The SUMEX staff itself has had good success in establishing such sharing relationships on a system level with other research groups and has many examples of complementary development and maintenance agreements for system programs.

vi) We have made numerous improvements to the computing resource to extend its capacity, to improve its efficiency, to enhance its human interfaces, to improve its documentation, and to enhance the range of software facilities available to user projects.

vii) We have begun a core research effort to investigate alternatives and programming tools to facilitate the exportability of user and system software. This is just now producing a "machine-independent" implementation of the ALGOL-like SAIL language which will run on a range of large and small machines and provide a language base for transferring programs.

viii) We have supported community efforts in the more systematic documentation of AI concepts and techniques and in building more general software tools for the design and implementation of AI application programs. These have included a Stanford AI Handbook project comprising a compendium of short articles about the projects, ideas, problems, and techniques that make up the field of AI.

1.3.2 DETAILED PROGRESS REPORT

The following material covers in greater detail the SUMEX-AIM resource activities over the past 3.5 years. These sections attempt to define in more detail the technical objectives of our research community and include progress in the context of the resource staff and the resource management. Details of the progress and plans for our external collaborator projects are presented in Section 6 on page 41 (in Book II).

1.3.2.1 DEFINITION OF TERMS AND OBJECTIVES

Artificial Intelligence is a branch of computer science which attempts to discern the underlying principles involved in the acquisition and utilization of knowledge in reasoning, deduction, and problem-solving activities (1). Currently authorized projects in the SUMEX community are concerned in some way with the application of these principles to biomedical research. The tangible objective of this approach is the development of computer programs which, using formal and informal knowledge bases together with mechanized hypothesis formation and problem solving procedures, will be more general and effective consultative tools for the clinician and medical scientist. The exhaustive search potential of computerized hypothesis formation and knowledge base utilization, constrained where appropriate by heuristic rules or interactions with the user, has already produced promising results in areas such as chemical structure elucidation and synthesis, diagnostic consultation, and mental function modeling. Needless to say, much is yet to be learned in the process of fashioning a coherent scientific discipline out of the assemblage of personal intuitions, mathematical procedures, and emerging theoretical structure of the "analysis of analysis" and of problem solving. State-of-the-art programs are far more narrowly specialized and inflexible than the corresponding aspects of human intelligence they emulate; however, in special domains they may be of comparable or greater power, e.g., in the solution of formal problems in organic chemistry or in the integral calculus.

An equally important function of the SUMEX-AIM resource is an exploration of the use of computer communications as a means for interactions and sharing between geographically remote research groups in the context of medical computer science research. This facet of scientific interaction is becoming increasingly important with the explosion of complex information sources and the regional specialization of groups and facilities that might be shared by remote researchers. Our community building role is based upon the current state of computer communications technology. While far from perfected, these new capabilities offer highly desirable latitude for collaborative linkages, both within a given research project and among them. Several of the active projects on SUMEX are based upon the collaboration of computer and medical scientists at

---

(1) For recent reviews to give some perspective on the current state of AI, see: (i) Winston, P.H., "Artificial Intelligence", Addison-Wesley Publishing Co., 1977; (ii) Nilsson, N.J., "Artificial Intelligence", Information Processing 74, North-Holland Pub. Co. (1975); and (iii) a summary by Feigenbaum, E. A., attached as Appendix I, page 202 (see Book II). An additional overview of research areas in AI is provided by the outline for an "Artificial Intelligence Handbook" being prepared under Professor Feigenbaum by computer science students at Stanford (see Appendix II on page 225 in Book II).

geographically separate institutions; separate both from each other and from the computer resource. The network experiment also enables diverse projects to interact more directly and to facilitate selective demonstrations of available programs to physicians and medical students. Even in their current developing state, we have been able to demonstrate that such communication facilities allow access to the rather specialized SUMEX computing environment and programs from a great many areas of the United States (even to a limited extent from Europe) for potential new research projects and for research product dissemination and demonstration. In a similar way, the network connections have made possible close collaborations in the development and maintenance of system software with other facilities.

1.3.2.2 FACILITY HARDWARE

Based on the AI mission of SUMEX-AIM, we selected a Digital Equipment Corporation (DEC) model KI-10 computer system for our facility. This selection was based on 1) hardware architectural and performance features, 2) available software support relevant to AI applications, 3) price versus performance data for the system, and 4) the scope of the user community from which we might expect to draw collaborators and share software. This choice has proved highly effective.

The current system hardware configuration is diagrammed in Figure 1 on page 14. It is the result of a number of augmentations over the past 3 years to meet the capacity needs of the growing SUMEX-AIM project community. Our initial configuration consisted of a KI-10 processor, core memory (192K 36-bit words @ 1 microsecond), swapping storage (1.7M words @ 8 msec average rotational latency and 2 microsecond/word transfer rate), file storage (40M words), magnetic tapes, DEC tapes, terminal line scanner, and line printer. Our network connections are discussed in Section 1.3.2.4 on page 20.

This system reached prime-time saturation by fall of 1974. Since many of our medical and other professional collaborators cannot adjust their schedules to match light computer loading during the night-time hours, the prime-time responsiveness is crucial to being able to support medical experimentation with developing programs and to allow community growth. We have taken active steps to transfer as much prime-time loading as feasible to evening and night hours including shifting personnel schedules (particularly for Stanford-based projects), controlling the allocations of CPU resources between various user communities and projects, and encouraging jobs not requiring intimate user interaction to run during off hours by developing batch job facilities. Despite these efforts, prime-time loading has remained quite high, particularly with the growth of the number of user projects.

A similar congestion has persisted in the on-line file space we have been able to allocate to user projects. Again we have implemented controls to try to assure effective use of available space and to encourage use of external file storage facilities such as the ARPANET Data Computer and other computer sites. Nevertheless, the interactive character of SUMEX use, the large AI program files, and the extensive use of SUMEX for collaborator communications have continuously raised file space demands beyond those we could meet.

We have proposed a number of hardware configuration augmentation steps to the Executive Committee to cost-effectively provide additional capacity. These were based on analyses of predominant system bottlenecks and enhancement steps feasible within available budgets. The enhancements approved by the committee and implemented include:

1) Add 64K words of core memory and 20M words of file storage (11/74)

2) Add second KI-10 CPU for dual processor operation (5/76)

3) Add 256K words of core memory and upgrade file system to higher volume, lower cost technology (recently approved by NIH and the AIM Executive Committee with implementation in progress)

A plot of effective CPU capacity as a function of continuing investment is shown in Figure 2 on page 15 and displays the cost-effectiveness of our sequential augmentations. At the present time our hardware configuration has grown about as much as is cost-effective. Additional growth would entail significant redesigns of the system including upgrades of existing hardware. Contemplating such future expansion also raises the issues of compatibility with newer hardware technologies being announced. These provide advantages in speed, cost, size, and maintainability. Such a complete upgrade is not envisioned in the immediate future as a number of interesting new product announcements are expected over the next 1 or 2 years that could substantially affect such an upgrade strategy. Our plans in this direction are discussed in more detail under the proposed resource plans for the continuation period (see Section 3.1 on page 62).

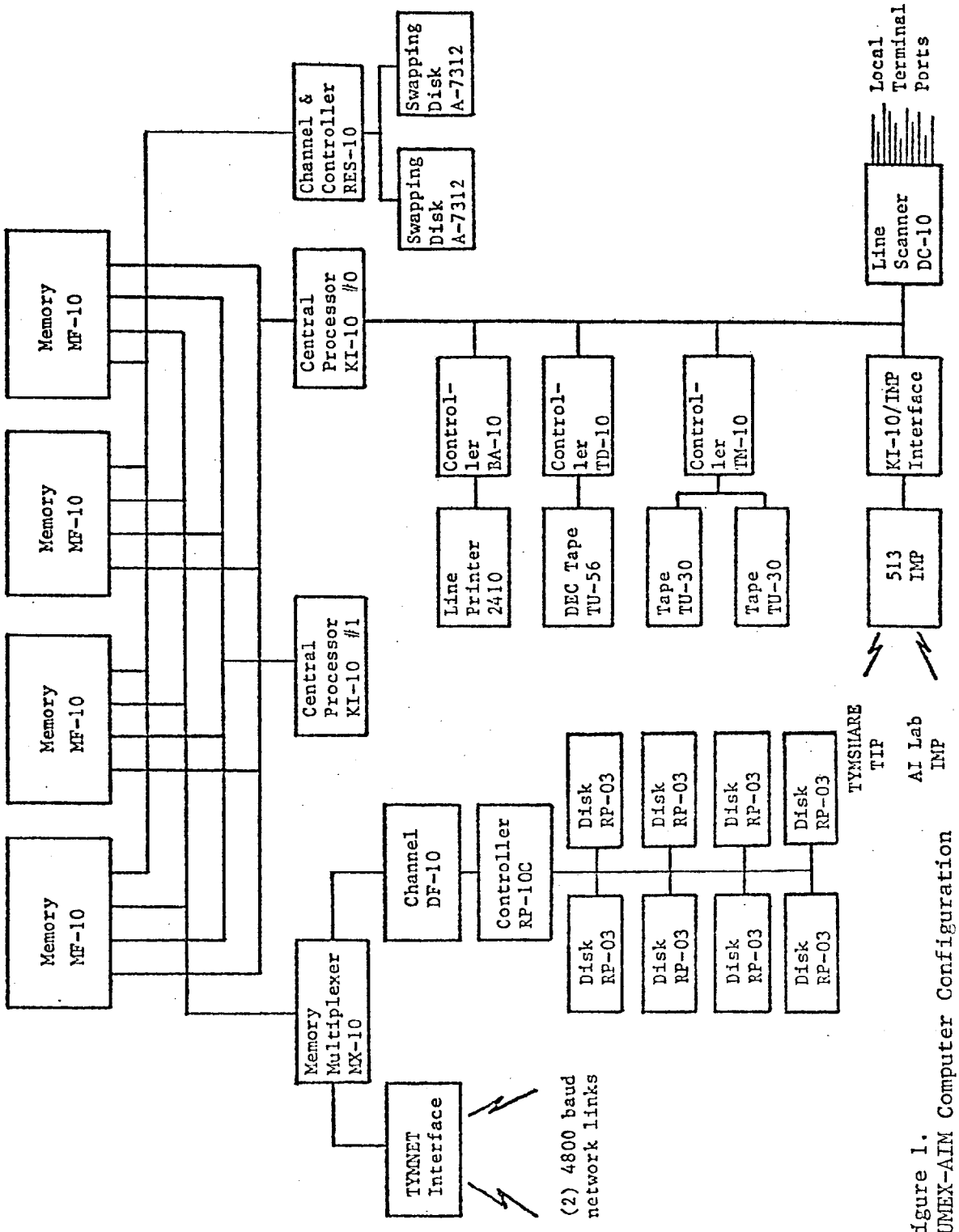
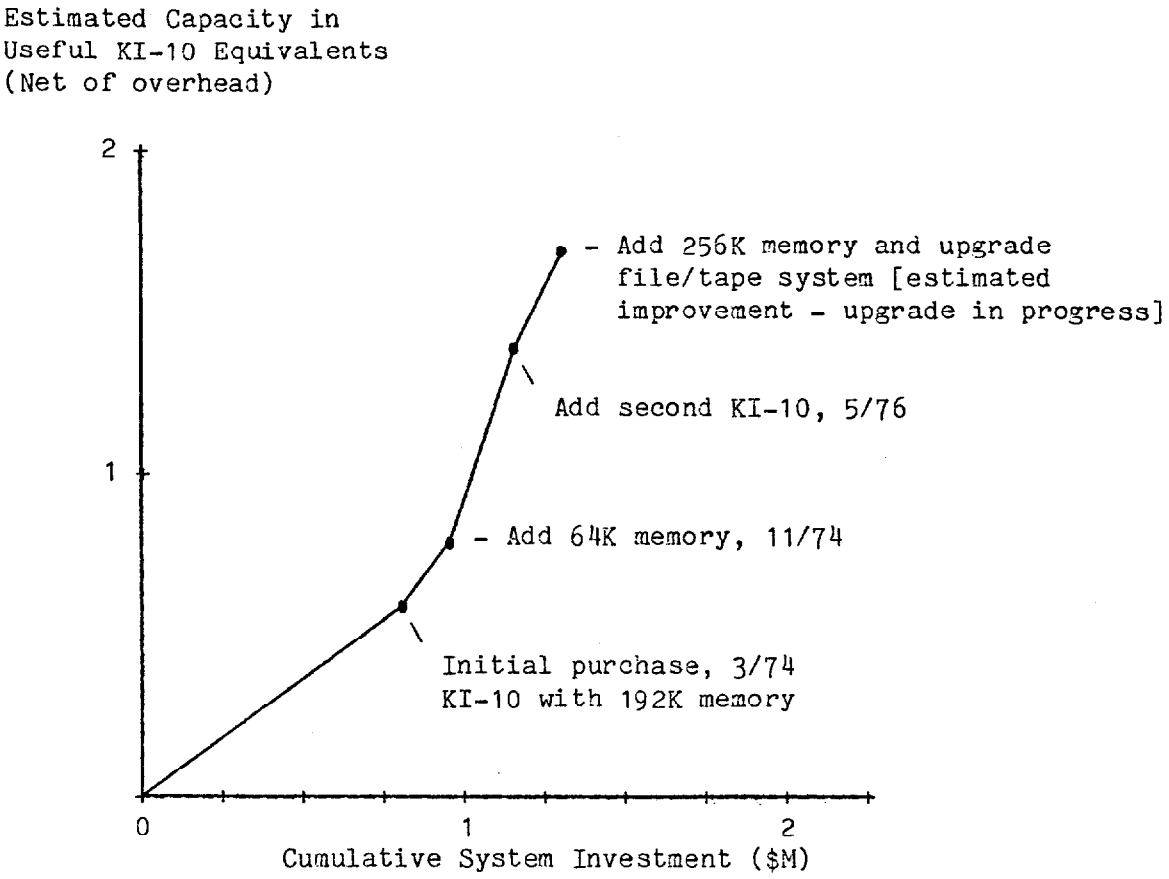


Figure 1.
SUMEX-AIM Computer Configuration

Figure 2. Cost-effectiveness of SUMEX Augmentations



This plot illustrates the incremental increases in computing capacity achieved as a function of cumulative investment in the SUMEX-AIM facility. The higher slope of the curve after the initial investment illustrates both the substantial investment in peripheral devices (file system, tapes, communications, etc.) and the trend toward lower memory prices. The largest impact in terms of PDP-10 memory price reductions occurred around the time of adding the 64K increment in November 1974. Since then processor prices have stayed relatively stable and memory prices have dropped less dramatically. It should be noted that semi-conductor memories have not yet made a big in-road in the PDP-10 market; this technology is where the more recent memory price reductions have occurred.

The original purchase of 1 KI-10 with 192K of memory for about $800K performed with about 60% efficiency under peak load. Adding the 64K of memory for $75K brought the efficiency up to about 85%. Then adding the second processor for $200K increased throughput to about 1.3-1.4 KI-10 equivalents. This step represents about a 59% increase in throughput for a 20% increased investment. A proposal has been approved recently by the AIM Executive Committee and NIH to augment core memory by 256K words. This augmentation would increase throughput to about 1.7 KI-10 equivalents for another $100K; this would be a 26%

throughput increase for 8% additional investment. As part of the proposed memory augmentation we plan to upgrade the file and tape systems as well to relieve file space congestion and increase system operations efficiency. Including the net cost of the file/tape upgrade in these figures (purchase price less resale of existing equipment) raises the proposed additional investment to $160K and the fractional increase from 8% to 13%. Of course, the disk upgrade affects CPU throughput only indirectly in that the increased speed reduces contention, particularly when moving head swapping is necessary. It contributes primarily to supporting the growing on-line file needs of the projects.

Figure 3. Capacity and Loading Increase with Dual Processor Augmentation

	1-PROC OP'N 1/76 - 4/76	2-PROC TRNS'N 5/76 - 8/76	2-PROC OP'N 9/76 - 12/76	2-PROC OP'N 1/77 - 3/77
Peak Ld Ave	4.8	5.6	6.0	6.6
Peak Jobs	30.2	33.3	34.7	38.1
% Overhead/ Processor	18.1	31.1	33.2	31.9
Total CPU Hrs/Mo	304.4	384.9	534.0	520.1

This table presents system usage data averaged over several months preceding, during, and after installation of the SUMEX-AIM dual processor system in order to show real changes in peak loading capacity and computing resources delivered. The first three rows of data are derived from monthly diurnal loading data and reflect average prime-time peak loading conditions (daily peak usage figures are often considerably higher, but those shown better represent gross trends). The last row gives average total monthly CPU hours delivered during the various periods.

With the common criterion that users have pushed both the single and dual processor systems to the limits of useful work in terms of prime time responsiveness, it is clear that the second processor has substantially increased throughput ("tolerable" peak load average up 38%, number of jobs up 26%, and delivered CPU hours up 71%). At the same time the overhead burden per machine has risen from 18 to 32%, principally in the category of I/O wait (total scheduler time and time waiting for a runnable job to be loaded in core). An additional factor, not explicitly shown in these data (because we only have a 1 msec clock), is the added time spent at interrupt level servicing drum swapping. This adds another 10-15% estimated overhead.

We feel these increased overhead figures can be reduced roughly to the single processor levels by adding more memory, thereby effectively recovering about 40-50% of the capacity of a KI-10 processor. A proposal is now pending with the AIM Executive Committee for this augmentation and we expect it to be implemented within the funding ceiling of the current grant.

1.3.2.3 SYSTEM SOFTWARE

In parallel with the choice of DEC PDP-10 hardware for the SUMEX-AIM facility, we selected the TENEX operating system developed by Bolt, Baranek, and Newman (BBN) as the most effective for our medical AI applications work. TENEX was the only available demand-paged system to support simultaneous large address space users, offered the INTERLISP language for LISP-oriented program development, and was well integrated with the ARPANET facilities which provide an excellent base for our community sharing efforts. This choice has proven a very effective one in that the productivity of the TENEX community in AI research has been highly advantageous to us (2).

The original BBN TENEX was written for a hardware-modified KA-10 system. This version of the system required a substantial amount of work to accommodate the relatively limited paging facilities of the KI-10 to run effectively. These early phases also included substantial monitor work to incorporate the TYMNET memory-sharing interface which connects us to the TYMNET and to integrate the high speed swapping storage. We have made numerous enhancements to the monitor calls and corrections of bugs to develop a highly reliable and effective operating system for our community work.

We continue to work to improve the efficiency of the system and its effectiveness in allocating valuable resources. For example we have modified the handling of user page tables so that the expensive procedure of clearing page tables and setting them up to run time-shared users could be minimized. This involved creating a pool of page tables which could be allocated to currently running users and could be kept available without setup overhead. We also implemented a system for migrating dormant pages from our fast swapping storage to moving head disk. This preserves the use of this limited resource for the currently active jobs.

We have implemented a form of "soft" CPU allocation control in the monitor, assisted by a program which adjusts user percentages for the scheduler based on the dynamic loading of the system. The allocation control structure works based on the scheduler queue system and takes account of the a priori allocation of CPU time and that actually consumed. Our TENEX uses a hierarchy of five queues for jobs ranging from highly interactive jobs requiring only small amounts of CPU time between waits to more CPU intensive jobs which can run for long periods without user interaction. These interactive queues (text editting, etc.) are scheduled at highest priority without consideration of allocation percentages. If nothing is runnable from the high priority queues, the CPU-bound queues are scanned and jobs are selected for running based on how much of their allocated time has been received during a given allocation cycle time (currently 100 seconds). If no such jobs are runnable, then those that have received their allocation of CPU time already are scheduled based on how much they are over

---

(2) It should be noted that DEC has recently adopted a form of TENEX (TOPS-20) as their choice for future system marketing. They have made improvements in a number of areas of the monitor and subsystem software but have also shown an increasing tendency to make changes to the TOPS-20 system that impair compatibility with older TENEX systems. The long-term impact of this trend toward incompatibilities with the coming DEC "standard" is discussed in more detail on page 62.

allocation and how long they have waited to be run again. This system is not a reservation system in that it does not guarantee a given user some percentage of the system. It allocates cycles preferentially, trading off a priori allocations with actual demand but does not waste cycles. This allocation control system is still in an experimental state and we are attempting to evolve the "best" policies with the AIM Executive Committee for dividing the system fairly and effectively among the various communities of users.

During the spring of 1976 we implemented a dual processor version of TENEX as the most cost-effective way to increase our processing capacity. In order to upgrade to the new KL-"n" technology, we would have had to replace most of the equipment that had been purchased initially. For the cost of an additional processor and 8 man-months of intensive software development we were able to increase our CPU capacity by 75%. We have an additional 40% equivalent of a KI-10 processor which can be made available by increasing memory to reduce our swapping contention. The dual processor system that has evolved is running quite reliably. It treats the two machines in an almost symmetric manner. The only difference is that one of the machines has all of the I/O equipment attached to it. They both schedule jobs independently and share the rest of the non-I/O-device monitor code. The areas of the monitor involving the management of resources and jobs which cannot be manipulated by both machines simultaneously are protected by a system of locks. We have made some measurements indicating that overhead for lock waits is less than 10%. The overall increase in capacity provided by the processor upgrade is illustrated in Figure 3 on page 17 which measures key loading parameters in the periods before and after the dual processor installation. Observing the delivery of DEC's high-performance KL-TENEX systems over the past 6 months, it seems clear that for the investment, we made the best choice for the community by implementing the dual processor upgrade. We hope to augment the memory soon to finish exploiting the capacity this extra machine provides and to remove some non-linearities remaining in system swapping performance.

Now that the dual processor system has stabilized, we are undertaking another assessment of system performance to be sure we have removed residual and correctable inefficiencies. This study is on-going now.

Finally, over the past year we made several substantial improvements in the "GTJFN" monitor call which interactively acquires handles on file names specified by the user. These extensions allow for more general "wild card" specifications and interactive help in deciding between and searching for existing file name alternatives. They also give the user much more flexibility in designating groups of files and therefore in structuring his data.

With a working dual processor system, the current implementation of allocation controls in our system, the diverging path of the DEC TOPS-20 system, the termination of active BBN TENEX development, and the unique complications of the KI-10 paging system, we have not made any concerted effort to upgrade our TENEX system to the latest BBN release (1.34). The advantages of such an upgrade are not overwhelming in face of the complicated conversion (KI paging, dual processor, special swapping device handler, TYMNET service routines, local JSYS's, etc.) and resulting system unreliability for some period.

Another area of software development is in the EXECutive program which is the basic user interface to manipulate files, directories, and devices; control job and terminal parameter settings; observe job and system status; and execute public and private programs. This work improves system accommodation to users and provides more convenient and useful information about system and job status. Through such features as login default files, directed file search path commands, mail notification, help facilities, better file archival and retrieval commands, and flexible status information, we have tried to make it easier for users to work on the SUMEX-AIM machine.

1.3.2.4 NETWORK COMMUNICATION FACILITIES

A highly important aspect of the SUMEX system is effective communication with remote users. In addition to the economic arguments for terminal access, networking offers other advantages for shared computing such as uniform user access to multiple machines and special purpose resources, convenient file transfers for software sharing and multiple machine use, more effective backup, co-processing between remote machines, and improved inter-user communications. Over the past year we have been substantially aided in exporting the MAINSAIL system through our network connections. Because of the developmental nature of the language at present, it is important that we have close interactions with the user community and that we be able to effectively perform bug fixes and upgrades. Since MAINSAIL by its nature involves operations on a variety of machines and since our access to example systems cannot be entirely local, the network connections to Rutgers, the Stanford AI Lab, and Stanford Research Institute have been invaluable. It would be considerably more difficult to export MAINSAIL and communicate with users via tapes and mail.

We have based our remote communication services on two networks - TYMNET and ARPANET. These were the only networks existing at the start of the project which allowed foreign host access. Since then, other commercial network systems (notably TELENET) have come into existence and are growing in coverage and services. The two networks to which we are currently connected complement each other; the TYMNET providing primarily terminal service with very broad geographical coverage and unrestricted user access, and the ARPANET having more limited access but providing a broader range of communication services. Together, these networks give a good view of the current strengths and weaknesses of this approach.

Users asked to accept a remote computer as if it were next door will use a local telephone call to the computer as a standard of comparison. Current network terminal facilities do not quite accomplish the illusion of a local call. Data loss is not a problem in network communications - in fact with the more extensive error checking schemes, data integrity is much higher than for a long distance phone link. On the other hand, networking relies upon shared community use of telephone lines to procure widespread geographical coverage at substantially reduced cost. However, unless enough total line capacity is provided to meet peak loads, substantial queueing and traffic jams result in the loss of terminal responsiveness.

TYMNET:

Networks such as TYMNET are a complex interconnection of nodes and lines spanning the country (see Figure 4 on page 24). The primary cause of delay in passing a message through the network is the time to transfer a message from node to node and the scheduling of this traffic over multiplexed lines. This latter effect only becomes important in heavily loaded situations; the former is always present. Clearly from the user viewpoint, the best situation is to have as few nodes as possible between him and the host - this means many interconnecting lines through the network and correspondingly higher costs for the network manager. TENEX in some ways emphasizes this conflict more than other time-sharing systems because of the highly interactive nature of terminal handling (e.g., command and file name recognition and non-printing program commands as in text editors or INTERLISP). In such instances, individual characters must be seen by the host machine to determine the proper echo response in contrast to other systems where only "line at a time" commands are allowed. We have connected SUMEX to the TYMNET in two places as shown in Figure 4 so as to allow more direct access from different parts of the country. Based on delay time statistics collected during the previous year from our TYMSTAT program, the response times are scarcely acceptable. When delay times exceed 200-300 milliseconds, the character printing lag problems become noticable with a full duplex, 30 char/sec terminal. In the past these times have been particularly bad in New York with peak delays approaching 3 seconds one way! Other nodes have shown uniformly high readings as well. These data were reflected in the subjective, but strongly articulated, comments of many of our user groups.

We have had numerous meetings with TYMNET personnel to try to ease these problems and have instituted reroutings of the lines connecting SUMEX-AIM to the network. Also local lines to more strategic terminal nodes have been considered for users in areas poorly served by the existing line layout. TYMNET has also made some upgrades in the internal connectivity and speeds with which data is switched within their node clusters. These changes seem to have had some beneficial effects in that delay times have improved and user complaints have subsided.

We will continue to pursue improvements in TYMNET response but user terminal interactions such as used in TENEX programs are not realized in the time-sharing systems offered by most other TYMNET users and hence are not supported well by TYMNET. TYMNET has implemented 1200 baud service in 7 major cities over the past year. Unfortunately many of our users are not in these cities so we have only limited experience with the 1200 baud support.

ARPANET:

The ARPANET, while designed for more general information transfer than purely terminal handling, has similar bottleneck problems in its topology (see the current geographical and logical maps of the ARPANET in Figure 5 and Figure 6 on page 25). These are reduced by the use of relatively higher speed interconnection lines (50 K baud instead of 2400 - 9600 baud lines as in TYMNET) but response delays through many nodes become objectionable eventually as well.

Consistent with the agreements with ARPA when we were granted network access initially, we are enforcing a policy to restrict the use of the ARPANET to users who have affiliations with ARPA-supported contractors and system/software interchange with cooperating TENEX sites. The administration of the network passed from the ARPA Information Processing Techniques Office to the Defense Communications Agency as of July 1975. At that time policies were announced restricting access to DoD-affiliated users. We have restricted the facilities for calling from SUMEX out to other sites on the ARPANET to authorized users. This also protects the SUMEX-AIM machine from acting as an expensive terminal handler for other machines - this function is better fulfilled by dedicated terminal handling machines (TIPS). In general, we have developed excellent working relationships with other sites on the ARPANET for system backup and software interchange - such day-to-day working interactions with remote facilities would not be possible without the integrated file transfer, communication, and terminal handling capabilities unique to the ARPANET.

We take very seriously the responsibility to provide effective communication capabilities to SUMEX-AIM users and are continuously looking for ways to improve our existing facilities as well as investigate alternatives becoming available. We have done preliminary investigations of the TELENET facilities that have been rapidly expanding this past year. BB&N has hooked one of their TENEX systems up to TELENET and whereas we did not have the same quantitative tools we have for measuring response on the TYMNET, we observed TELENET delays at least as long as those encountered on TYMNET. We did the reverse experiment by using long distance telephone to connect from the TELENET node in Washington, D.C. to the SUMEX machine in California and observed the same sort of delays reaching several seconds per character. The TELENET has many attractive feature in terms of a symmetry analogous to that of the ARPANET for terminal traffic and file transfers and being commercial would not have the access restrictions of the ARPANET. However, until the network throughput improves we would not get substantial benefits from connecting to it.

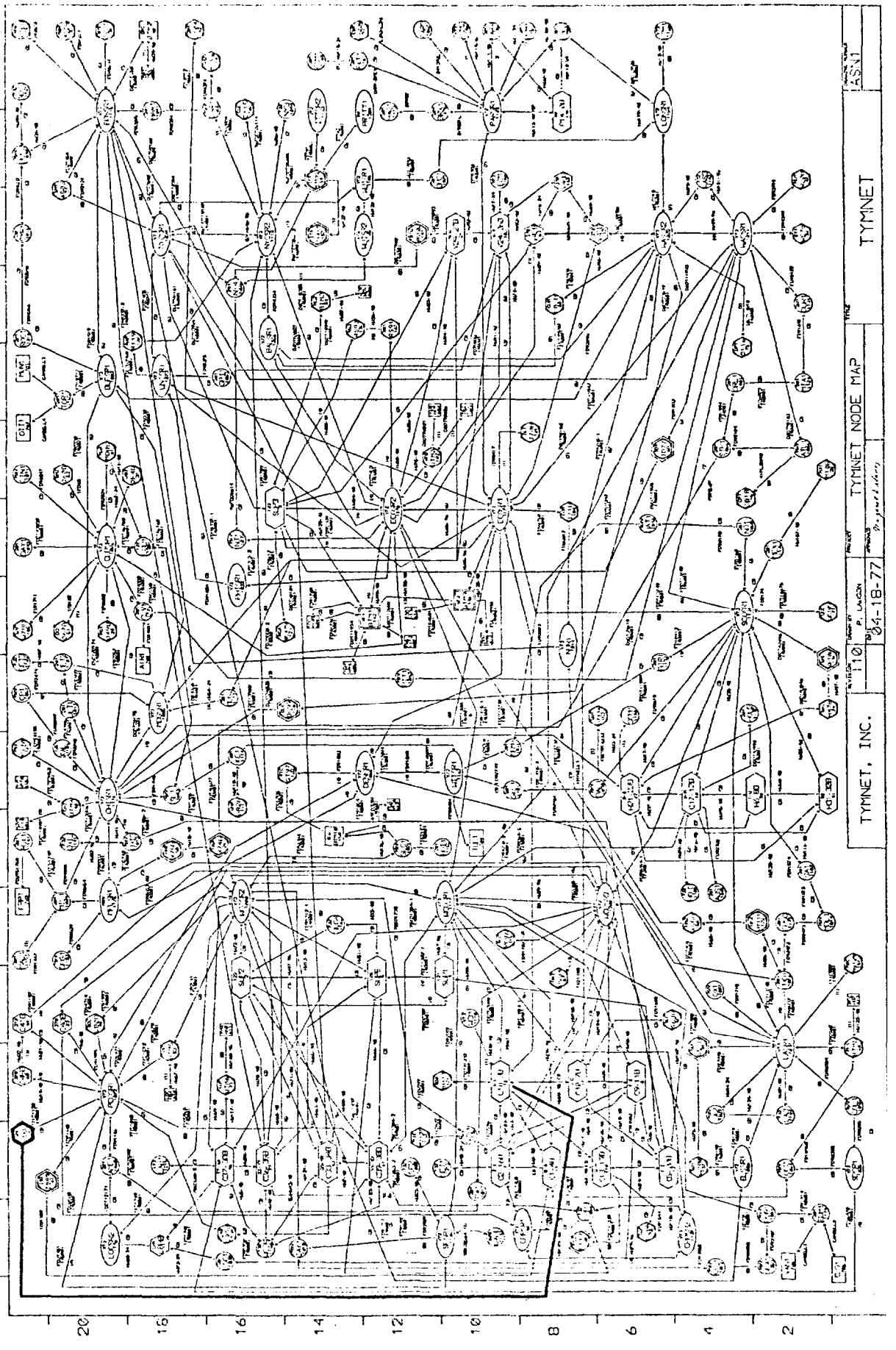


Figure 4. TYMNET Network Map

Figure 5.

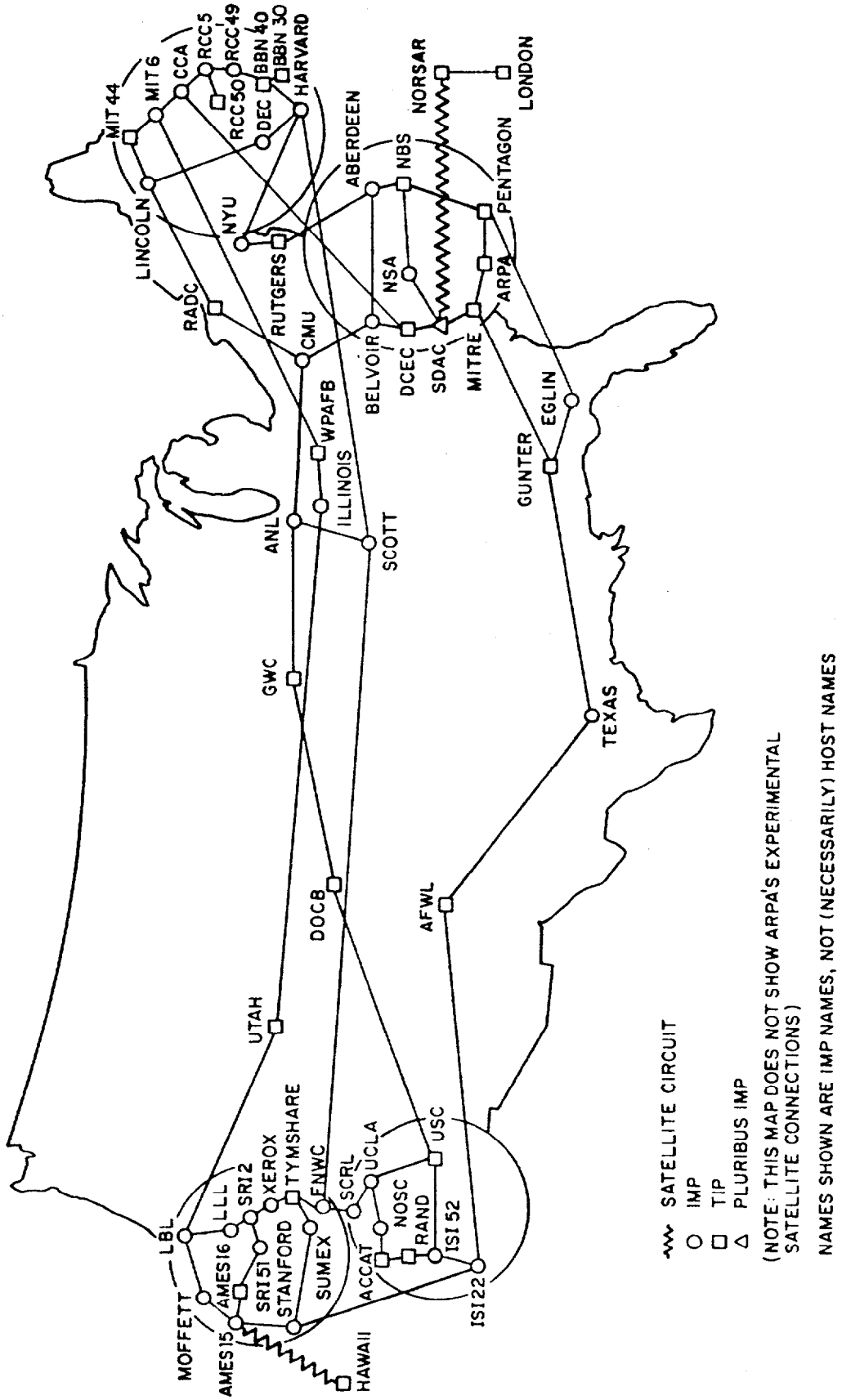
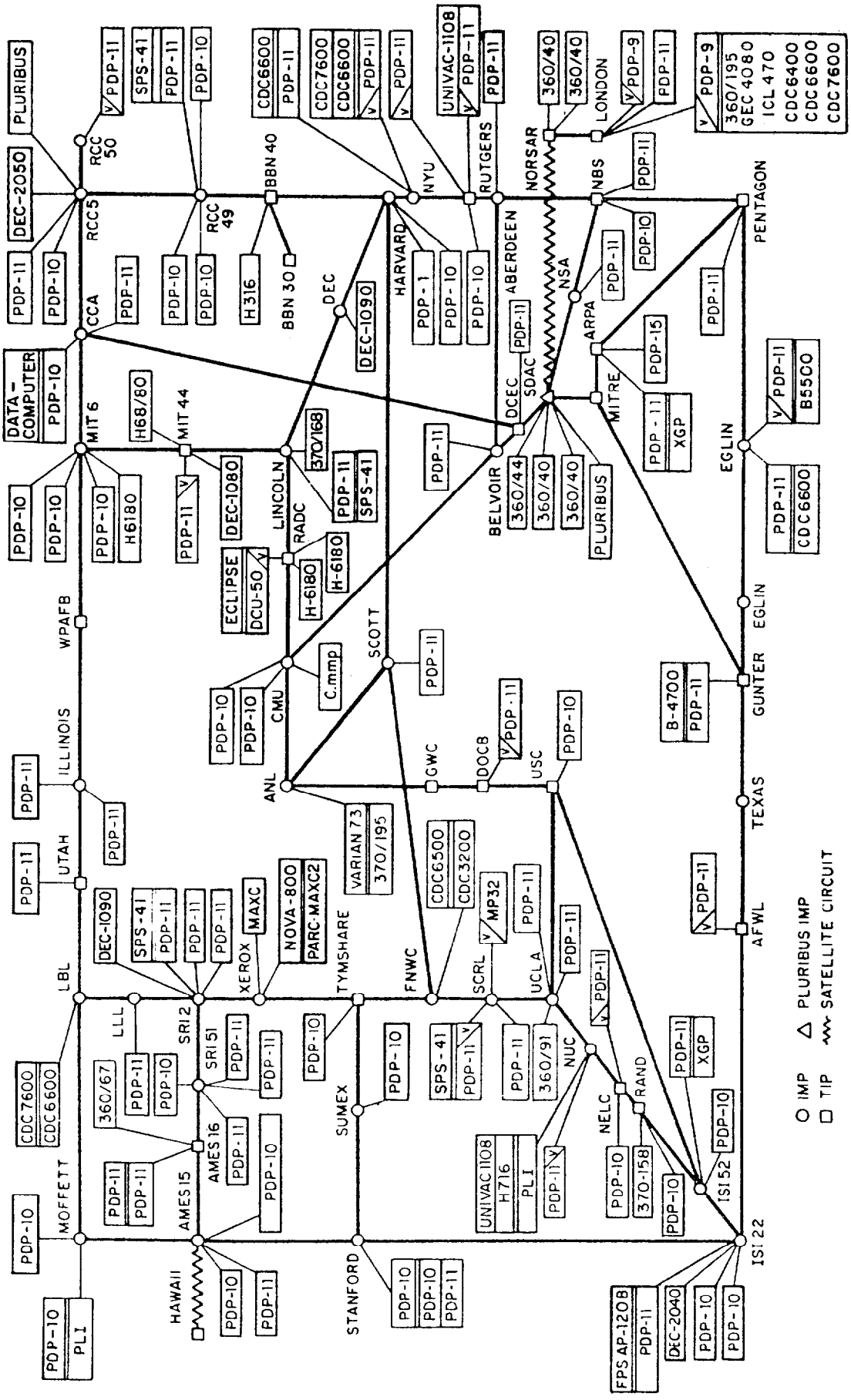

ARPANET GEOGRAPHIC MAP, APRIL 1977
MIT44
MIT6
CCA
RCC5
RCC49
RCC50
DEC
BBN40
BBN30
HARVARD
LINCOLN
NYU
RUTGERS
RADC
CMU
ABERDEEN
NBS
NORSAR
LONDON
PENTAGON
ARPA
NSA
BELVOIR
DCEC
SDAC
MITRE
EGLIN
GUNTER
WPAFB
ILLINOIS
ANL
SCOTT
GWC
TEXAS
DOCB
AFWL
UTAH
LBL
LLL
SRI2
XEROX
TYMSHARE
ENWC
SCRL
UCLA
USC
NOSC
RAND
ISI52
ACCAT
SUMEX
STANFORD
SRI51
AMES16
AMES15
MOFFETT
HAWAII
ISI22
SATELLITE CIRCUIT
IMP
TIP
PLURIBUS IMP
(NOTE: THIS MAP DOES NOT SHOW ARPA'S EXPERIMENTAL SATELLITE CONNECTIONS)
NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

Figure 6.
ARPANET LOGICAL MAP, MARCH 1977



(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

1.3.2.5 SYSTEM RELIABILITY AND BACKUP

System reliability has remained high over the past years; excellent under stable hardware and software conditions and degrading temporarily during debugging and development periods and during periods of difficult hardware problems. In general we take the system down for approximately 50 hours per month for scheduled hardware maintenance, file backup, and other maintenance. In addition we average from 10 to 15 hours per month in unscheduled downtime. During particularly difficult hardware or software difficulties we must absorb substantially more downtime.

1.3.2.6 PROGRAMMING LANGUAGES

Over the past years we or members of the SUMEX-AIM community have continued to maintain the major languages on the system at current release levels, have TENEXized several languages to improve efficiency, and have investigated a number of issues related to the efficiency of programs written in various LISP implementations and the exportability of programs. These issues are becoming increasingly critical in dealing with AI performance programs which have reached a level of maturity so that substantial, non-developmental user communities are growing. The following summarizes general accomplishments and the following section discusses in detail the work this past year in designing a machine-independent ALGOL-like system (MAINSAIL).

LISP Efficiency:

There has been an on-going debate among a number of projects over the best language to choose for developmental implementation of the various AI programs. The key issues include ease and flexibility of conceptual representation of program functions and objects, interactive debugging support, efficiency, and exportability. To date the predominant language choice for AIM research has been LISP and more particularly INTERLISP. These issues are important because they influence the time required to develop new AI programs and subsequently the incremental load placed on the SUMEX machine when in use. We recently attempted an evaluation of INTERLISP and ILISP including the relative efficiencies of the two languages and the level of assistance the language systems provide the user in developing programs. The tests were based on an implementation of a subset of REDUCE (a symbolic algebra manipulator). The results of several iterations in program refinement by experts in the respective languages were that the runtimes for the two versions were quite comparable (far less than the factor of 5-10 disparity predicted by ILISP enthusiasts). A more disquieting result was the substantial difference in runtimes depending on how particular functions were coded IN THE SAME LANGUAGE. It is apparent from the results that factors of 10 differences in time can result from a superficial implementation - expert programming insight is essential to efficient program performance. This is not a real surprise in that it is true of programming in any language - the problems may be increased by such a rich language as INTERLISP with such a wide array of

ways to do the same thing but with little guidance as to the relative costs. It has proven very difficult to quantify the "rules" for good programming. Mr. Masinter and Mr. Phil Jackson attempted to document good INTERLISP programming habits and issued a bulletin for SUMEX users.

A further impact of these data is that it is very difficult to simultaneously develop a new AI program and make the implementation highly efficient. With the iterations required to develop the conceptual design of the program, it is difficult to ensure its efficiency. This may lead to the need to reimplement the program after the basic development stabilizes to increase efficiency while still accommodating convenient and orderly further development. Such reimplementation may or may not be best done in LISP - this will depend on many factors including the nature of the program data structure requirements and anticipated further development efforts.

MAINSAIL Progress

SUMEX, in its role as a nationally shared computer resource, is an appropriate vehicle for the development of software unbound by the underlying machine environment. We have a built-in community of program developers acutely aware of the significance of providing their work to a broader base of users. This intersection of hardware capability, software expertise, and dedication to resource sharing presents a unique opportunity to promote a system designed for program sharing.

The MAINSAIL (3) project has three closely related goals:

1) Provide an integrated set of tools for the creation of efficient portable software on a variety of computer systems, and provide support and continued development of these tools in a form compatible across all implementations.

2) Study innovative approaches to portability, both hardware and software, and develop such approaches into effective tools.

3) Promote the development and distribution of portable software, advise and assist in its design, and evaluate its applicability.

By portable software we mean computer programs which may be executed on a variety of machines with few, if any, alterations. MAINSAIL itself will provide the initial example of portable software, since all of the system is written in the MAINSAIL language except for those parts which are determined by the host environment (hardware, instruction set, operating system, etc.). Even these parts are embedded within MAINSAIL.

---

(3) The MAINSAIL (MAchine-INdependent SAIL) language is derived from SAIL, a programming language developed at Stanford University's Artificial Intelligence Laboratory. It is not compatible with SAIL, since SAIL was designed for a PDP-10 with TOPS-10, and hence contains machine-dependencies. However it has retained the basic attributes of SAIL as an extended ALGOL-like language. A summary of some of the features of the MAINSAIL language and their relationship to other languages is given in Appendix III on page 231 (see Book II).

There is a key distinction between MAINSAIL's approach to portability and the "classical" approach characterized by languages such as FORTRAN, ALGOL, LISP, COBOL and BASIC. These languages attempt to adhere to a single syntax standard which is separately implemented for each different computer system. Invariably these implementations have differences which preclude the creation of a program which is accepted by all. It is difficult, if not impossible, to define a language standard which is unambiguous and at the same time sufficiently comprehensible to provide the basis for compatible implementations. Furthermore, many implementors yield to the temptation to provide "enhancements" to the standard which immediately introduces machine and system dependencies.

MAINSAIL, on the other hand, provides a single system (written primarily in itself) which is employed at every site. This is made possible by its ability to compile itself into code for a variety of machines. Only the compiler's code generators and the runtime operating-system interfaces need be rewritten for each implementation. These parts of MAINSAIL are at a level which has already been defined by the machine-independent parts, and do not affect the language from the user's viewpoint. Thus the "language standard" has been reduced to a "semantic standard" which is surrounded by machine-independent software.

It remains to be seen whether the temptation to augment the language with machine-dependencies (for purposes of ultimate efficiency or to take advantage of particular local system features) can be overcome. Herein also lies the biggest "price" to be paid for exportability. The code emitted from the MAINSAIL compiler can be (and is, based on tests to date) at least as efficient as that from many machine-dependent compilers. On the other hand, special machine or operating system features that cannot be uniformly implemented may provide local optimizations at the cost of exportability or vice versa. We cannot effectively measure the extent of this cost at this stage.

DEVELOPMENT APPROACH

We do not underestimate the difficulty in obtaining the cooperation of a community which will span a wide variety of applications and hardware/software systems. If MAINSAIL is to obtain widespread use, it is crucial that it have an effective and credible base of support. The initial parts of MAINSAIL are just about ready for limited distribution. We want to maintain close supervision of this distribution, and insure that systems labelled as MAINSAIL are not altered without our approval. In this regard we are pursuing legal channels to safeguard the integrity of MAINSAIL software. We plan to take MAINSAIL through an orderly progression of development, and to avoid casual distribution with no provision for a solid base of maintenance and future growth.

REVIEW OF PROGRESS TO DATE

MAINSAIL has been under development for almost three years now. Beginning with an initial goal of converting the PDP-10 SAIL compiler to generate code for a PDP-11, several versions had been implemented on a PDP-10 and a PDP-11, and the groundwork had been laid for extending the system to a wider variety of machines. The current version was begun in August of 1976.

Early versions of MAINSAIL attempted to maintain close compatibility with the original SAIL, but in surveying a wider variety of machines (especially mini-computers), we concluded that this compatibility could be maintained only at the expense of portability. It was felt that MAINSAIL could contribute more by providing a truly portable system. Thus we began redesigning MAINSAIL, rebuilding from previous implementations. This effort has resulted in a new version which is still under development, and is now being tested on several systems.

Initial implementations of the current design are for DEC PDP-10's with the TENEX operating system and with the TOPS-10 operating system. The TENEX version is being tested at SUMEX and has been installed at one other TENEX site (Stanford - IMSSS). The TOPS-10 version was developed at SUMEX by using TENEX facilities which provide compatibility with TOPS-10. The Rutgers University PDP-10 facility was chosen for external testing since it is a standard TOPS-10 system, and can be accessed from SUMEX over a network. MAINSAIL is now undergoing preliminary testing there. A modified TOPS-10 version has been set up on the Stanford AI-lab's PDP-10, but also has not been open to general use.

Little additional work will be necessary to make the TENEX version execute on a DECSYSTEM-20 since TOPS-20 is derived from TENEX. However, some time will be needed to take full advantage of the extended instruction set of the KL-10. Two sites are available for TOPS-20 development: the LOTS facility at Stanford; and a machine at SRI, close to Stanford and accessible over a network. Both of these sites have expressed an interest in using MAINSAIL.

The PDP-11 has been chosen as the first mini-computer to be implemented. Code generators have been written for it but not debugged. Several variants of these code generators will be necessary to cover the full PDP-11 family.

MAINSAIL interfaces to three PDP-11 operating systems (RT-11, RSX-11 and UNIX) are now under development. All of these operating systems are available to the MAINSAIL project on PDP-11's at Stanford. RT-11 will be the first to be implemented. The mix of instruction sets, operating systems and configurations will be a good test of MAINSAIL's ability to provide a compatible implementation, even across this one family of computers. We expect the PDP-11 systems to be operational by this summer.

1.3.2.7 STANFORD AI HANDBOOK PROJECT

The AI Handbook is a compendium of short articles (3-5 pages each) about the projects, ideas, problems and techniques that make up the field of Artificial Intelligence. Over 150 articles have been drafted by researchers and students in the field, on topics ranging in depth from "Augmented Transaction Networks" (ATN's) to "An Overview of Natural Language Research", and covering the entire breadth of AI research: search, robotics, speech understanding, real-world applications, etc. An outline of the current contents of the handbook is given in Appendix II on page 225 (see Book II).

During the Spring of 1976 the final push for drafting new articles was completed, with some 60 articles produced by students during that quarter. Since then the process has begun of rewriting the various chapters of the Handbook to produce coherent manuscripts from the original work of five to ten authors. This effort involves rewriting articles for accuracy and completeness as well as integrating the 15 to 25 articles in a section into an editorially uniform and readable document. An editor has been added to the project team who will be responsible for maintaining a consistent format and style in the Handbook.

When completed, each chapter will be reviewed by experts in the appropriate research area before it is released to the public. At present, the chapter on Natural Language research is completed and being reviewed, and we expect that the sections on Search, Speech Understanding, Representation of Knowledge, and Automatic Programming will be completed during the next two months. During the Fall of 1977 the first seven chapters of the handbook will be published in preliminary form. Meanwhile, the handbook is already available to cooperative experts and critics on-line via the SUMEX-AIM network connections. We are considering maintaining the handbook on-line, with occasional hard-copy editions, and believe this method of "publication" may be a prototype for other encyclopedic monographs.

1.3.2.8 USER SOFTWARE AND INTRA-COMMUNITY COMMUNICATION

In addition to the system and language software development efforts of SUMEX, we have assembled or developed where necessary a broad range of utilities and user software. These include operational aids, statistics packages, DEC-supplied programs, improvements to the TOPS-10 emulator, text editors, text search programs, file space management programs, graphics support, a batch program execution monitor, text formatting and justification assistance, and magnetic tape conversion aids. We have also developed a number of user information assistance programs such as a "WHOIS" facility to recover names and affiliations of users and a "HELP" facility to locate on-line documentation of interest through key word searches.

Of major importance for our community effort is the set of tools for inter-user communications. We have enhanced the message sending and manipulation programs to better integrate text editting facilities for easier message preparation and reading. We have also developed a unique "bulletin board" system to deal with informal notes, thereby bridging a functional gap between formal system documents and private messages communications between individual users. The bulletin board system provides an informal and dynamic base for information about system facilities, lore, bugs, etc. or can provide a means for intra-project communication and coordination.

The system has been in operation for more than one year and has been exported to IMSSS (Stanford's other TENEX site) and USC-ECL. We have also proposed that the next generation of ARPANET information services provide for bulletin board-like facilities. At SUMEX-AIM there are 10 bulletin boards, 8 of which are project-specific. The main system bulletin board currently contains more than 140 bulletins under 85 topics covering system status announcements,

explanations of recent crashes, hardware troubles and monitor upgrades, new developments, bugs, and little-documented features of our programming languages and utilities. Project bulletin boards have been used for notices and minutes of meetings, references to and abstracts of papers, coordination of on-going developments, vacation schedules, documentation and announcements of various kinds.

Current Bulletin Board features include:

Multiple bulletin boards (public, private, general, specific, etc.).

Topics and subtopics (separated by periods) may be nested to any depth.

Expire dates for each bulletin, after which they are removed automatically.

Interest-list-of-topics for each user allows him to be notified about new bulletins he is interested in and to ignore others.

Users notified when new bulletins arrive, by running BBCHECK (the bulletin-board MAIL CHECK) or by mail.

Help and browsing facilitated in a variety of ways (? can be typed anywhere, general and command-specific help provided).

Command structure modelled after the TENEX EXEC, with conscious attention to human-engineering.

Companion program BBREAD is a bulletin-board READMAIL.

Companion program BBNEWS types out a directory listing of any new bulletins.

1.3.2.9 DOCUMENTATION AND EDUCATION

We have spent considerable effort to develop, maintain, and facilitate access to our documentation so as to accurately reflect available software. The HELP and Bulletin Board systems have been important in this effort. We have limited manpower for user assistance. In general, users are responsible for their own software development and maintenance. The SUMEX staff, however, (including Lederberg and Rindfleisch) share the responsibilities for system level assistance to users, tracking down bugs, reviewing user suggestions, etc. The terminal linking facilities of TENEX have been valuable tools to assist remote user groups and also for system users to communicate with each other. With the recent initial release of the MAINSAIL system on selected machines, we are becoming increasingly involved in describing MAINSAIL and advising user projects in its possible applications.

1.3.2.10 SOFTWARE COMPATIBILITY AND SHARING

At SUMEX-AIM we firmly believe in importing rather than reinventing software where possible. At SUMEX many avenues exist for sharing between the system staff, various user projects, other facilities, and vendors. In the past

without communication networks, the system vendor served as the focal point for distribution of most software to user sites. Since the process of distributing tapes (and particularly of handling bug reports and user suggestions) was very slow, it was common for sites to take a version of a program and then modify and maintain it locally. This caused a proliferation of home-grown versions of software. Similar impediments have existed to the dissemination of user software. User organizations like SHARE and DECUS have helped to overcome these problems but communication is still cumbersome. The advent of fast and convenient communication facilities coupling communities of computer facilities has the potential of making a major difference in facilitating inter-group cooperation and to lower these barriers.

The TENEX sites on the ARPANET have been interacting increasingly with each other to develop new software systems. This functions effectively to build communication around the network and promote a functional division of labor and expertise. The other major advantage is that as a by-product of the constant communication about particular software, personal connections between staff members of the various sites develop. These connections serve to pass general information about software tools and to encourage the exchange of ideas among the sites. Certain common problems are now regularly discussed on a multi-site level. We continue to draw significant amounts of system software from other ARPANET sites, reciprocating with our own local developments. Interactions have included mutual backup support, hardware configuration experiments, operating system enhancements, utility or language software, and user project collaborations. We have been able to import many new pieces of software and improvements to existing ones in this way. Examples of imported software include the message manipulation program MSG, TENEX SAIL, TENEX SOS, INTERLISP, the RECORD program, ARPANET host tables, and many others. Reciprocally, we have exported our contributions such as the drum page migration system, KI-10 page table efficiency improvements, GTJFN enhancements, PUB macro files, the bulletin board system, SNDMSG enhancements, our BATCH monitor, etc. The most recent example of this cooperative use of networks is in the preliminary export of MAINSAIL.

1.3.2.11 RESOURCE MANAGEMENT

PHILOSOPHY OF MANAGEMENT

The tidiest way to administer a national resource would be by subcontract to a fee-compensated, neutral agent. This would still have to involve a governing body that could speak to the technical and quality-control interests of the served constituency. Appropriate in some circumstances, this model would separate the administration of a resource from active research and development. An approach expected to foster greater creativity is to couple the resource with an active user-center. This of course can lead to manifest conflicts of interest that must be addressed and avoided if the resource is to be fairly available on a regional or national basis.

As indicated in the introduction, our proposal for the latter approach was followed by searching negotiations over a management plan that would be sensitive to these considerations. The bureaucratic procedures, much as they have to be

spelled out, are almost the last items that need to be specified for such a plan. Far more important is a charter that spells out the underlying objectives and responsibilities of the program, and which establishes incentives, resources, and obligations for proper performance. We believe the plan that was negotiated and implemented has all of these ingredients, and has made the design of the procedural framework a matter of simple common-sense logic from these premises. It will be plain that the convergence of local self-interest, and peer and contractual responsibility offers the best assurance that the programmatic goals will be respected, and simplifies the tasks of surveillance and accountability.

The self-interest part of this equation stems from our original motivation in requesting the resource: the need for specialized computing facilities to support intense, interdisciplinary studies in applications of AI at Stanford University Medical School. Comprising several departments (Genetics, Medicine, Computer Science and Chemistry), and interwoven projects (e.g., DENDRAL, Heuristic Programming, MYCIN, MOLGEN) and principal faculty (Professors Lederberg, Feigenbaum, Djerassi, Cohen, and Buchanan), a substantial body of research that has progressed and evolved over many years would be sacrificed if such a resource were not available. Successful, stable collaborations of this scope are not readily found. This history both depends upon and contributes to the doctrine of resource-sharing that underlies the SUMEX-AIM effort.

One premise of the management plan was therefore the charter allocation of half the user-available capacity of the SUMEX facility to the Stanford complex of projects, subject to a local committee chaired by Professor Lederberg.

The acceptance of this principle clearly defines the local benefit of the resource, minimizes anxiety and conflict-of-interest, and en suite enables the local group to respond quite objectively to the allocations that are made by an Executive Committee for the "national" or non-Stanford aliquot (see "Executive and Advisory Committee Organization" below). Another important contribution to the success of the plan is the welcome participation of an NIH-BRP representative on the Executive Committee. What would be inappropriate meddling, in the conduct of a narrower research project funded by NIH, is a communication channel and source of detached judgment that has been invaluable in expediting the innumerable decisions about which NIH must and should be consulted in the week-to-week business of the resource. The efficacy of this principle, as is appropriate to acknowledge here, has been validated and enhanced by the style and energy that Dr. William Baker has brought to this task.

That the "national" community should be conscientiously cultivated for the most efficacious use of its aliquot, and that further growth of facilities should in due course be distributed, are further inferences from the charter principles.

Finally, the recognition in the charter that SUMEX-AIM was not merely a retail-store for computer cycles, but the means of building a community, was a necessary basis for the morale of the whole operation. Some of these matters were addressed further in the section on SIGNIFICANCE (see Section 1.2 on page 4). The remainder of this section will now speak to the way in which these responsibilities are handled bureaucratically.

ORGANIZATION AND PROCEDURES

The SUMEX-AIM resource is administered within the Genetics Department of the Stanford University Medical School, Professor Lederberg's "main office", though he also holds appointments in the Computer Science Dept. and the Human Biology program. Its mission, locally and nationally, entails both the recruitment of appropriate research projects interested in medical AI applications and the catalysis of interactions among these groups and the broader medical community. User projects are separately funded and autonomous in their management. They are selected for access to SUMEX on the basis of their scientific and medical merits as well as their commitment to the community goals of SUMEX. Currently active projects span a broad range of application areas such as clinical diagnostic consultation, molecular biochemistry, belief systems modeling, mental function modeling, and instrument data interpretation (see Section 6 on page 41 in Book II). We have pondered the possibilities of a fee-for-service approach to allocation of the resource. We believe that this would be inappropriate for an experimental system of such national scope, whose pricing structure would have to be revised almost on a week-to-week basis to fairly respond to evolutionary changes in the system. This would also pose problems of accountability for the transfer of funds from one institution to another. Our present policy of non-monetary allocation control, which we propose to continue for the next term, of course accentuates our responsibility for the careful selection of projects with high scientific and community merit.

EXECUTIVE AND ADVISORY COMMITTEE ORGANIZATION

As the SUMEX-AIM project is a multilateral undertaking by its very nature, we have created several management committees to assist in administering the various portions of the SUMEX resource. As defined in the SUMEX-AIM management plan adopted at the time the initial resource grant was awarded, the available facility capacity is allocated 40% to Stanford Medical School projects, 40% to national projects, and 20% to common system development and related functions. Within the Stanford aliquot, Dr. Lederberg has established an advisory committee to assist him in selecting and allocating resources among projects appropriate to the SUMEX mission. The current membership of this committee is listed in Appendix V (see Book II).

For the national community, two committees serve complementary functions. An Executive Committee oversees the operations of the resource as related to national users and makes the final decisions on authorizing admission for projects. It also establishes policies for resource allocation and approves plans for resource development and augmentation within the national portion of SUMEX (e.g., hardware upgrades, MAINSAIL development priorities, etc.). The Executive Committee oversees the planning and implementation of the AIM Workshop series currently implemented under Prof. S. Amarel of Rutgers University and assures coordination with other AIM activities as well. The committee will play a key role in assessing the possible need for additional future AIM community computing resources and in deciding the optimal placement and management of such facilities. The current membership of the Executive committee is listed in Appendix V (see Book II).

Reporting to the Executive Committee, an Advisory Group represents the interests of medical and computer science research relevant to AIM goals. The Advisory Group serves several functions in advising the Executive Committee; 1) recruiting appropriate medical/computer science projects, 2) reviewing and recommending priorities for allocation of resource capacity to specific projects based on scientific quality and medical relevance, and 3) recommending policies and development goals for the resource. The current Advisory Group membership is given in Appendix V (see Book II).

These committees have actively functioned in support of the resource. Except for the meetings held during the AIM workshops, the committees have met by telephone conference owing to the size of the groups and to save the time and expense of personal travel to meet face to face. These telephone meetings, in conjunction with terminal access to related text materials, have served quite well in accomplishing the agenda business and facilitate greatly the arrangement of meetings. Other solicitations of advice requiring review of sizable written proposals are done by mail.

We will continue to work with the management committees to recruit the additional high quality projects which can be accommodated and to evolve resource allocation policies which appropriately reflect assigned priorities and project needs. We hope to make more generally available information about the various projects both inside and outside of the community and thereby to promote the kinds of exchanges exemplified earlier and made possible by network facilities.

NEW PROJECT RECRUITING

The SUMEX-AIM resource has been announced through a variety of media as well as by correspondence, contacts of NIH-BRP with a variety of prospective grantees who use computers, and contacts by our own staff and committee members. The number of formal projects that have been admitted to SUMEX has more than doubled since the start of the project; others are working tentatively as pilot projects or are under review.

We have prepared a variety of materials for the new user ranging from general information such as is contained in a brochure (see Appendix VI in Book II) to more detailed information and guidelines for determining whether a user project is appropriate for the SUMEX-AIM resource. Dr. E. Levinthal has prepared a questionnaire to assist users seriously considering applying for access to SUMEX-AIM (see Appendix VII in Book II). Pilot project categories have been established both within the Stanford and national aliquots of the facility capacity to assist and encourage projects just formulating possible AIM proposals pending their application for funding support and in parallel formal application for access to SUMEX. Pilot projects are approved for access for limited periods of time after preliminary review by the Stanford or AIM Advisory Group as appropriate to the origin of the project.

These contacts have sometimes done much more than provide support for already-formulated programs. For example, Prof. Feigenbaum's group at Stanford has initiated a major collaborative effort with Dr. Osborn's group at the Institutes of Medical Sciences in San Francisco. This project in "Pulmonary Function Monitoring and Ventilator Management - PUFF/VM" (see Section 6.4.6 on

page 197 in Book II) originated as a pilot request to use MLAB in a small way for modeling. Subsequently the AI potentialities of this domain were recognized by Feigenbaum, Nii, and Osborn who have submitted a joint proposal to NIH and have a pilot status at present.

The following lists the fully authorized projects currently comprising the SUMEX-AIM community (see Section 6 in Book II for more detailed descriptions). The nucleus of five projects that were authorized at the initial funding of the resource in December 1973 are marked by "<*>".

National -

1) Acquisition of Cognitive Procedures (ACT); Dr. J. Anderson (Yale University)

<*> 2) Higher Mental Functions Project; K. Colby, M.D. (University of California at Los Angeles)

3) INTERNIST Project; J. Myers, M.D. and Dr. H. Pople (University of Pittsburgh)

4) Medical Information Systems Laboratory (MISL); J. Wilensky, M.D. and Dr. B. McCormick (University of Illinois at Chicago Circle)

<*> 5) Rutgers Computers in Biomedicine; Dr. S. Amarel (Rutgers University)

6) Chemical Synthesis Project (SECS); Dr. T. Wipke (University of California at Santa Cruz)

Stanford -

<*> 1) DENDRAL Project; Drs. C. Djerassi, J. Lederberg, and E. Feigenbaum

2) Large Multi-processor Arrays (HYDROID); Dr. G. Wiederhold

3) Molecular Genetics Project (MOLGEN); Drs. J. Lederberg, E. Feigenbaum, and N. Martin

<*> 4) MYCIN Project; S. Cohen, M.D. and Dr. B. Buchanan

<*> 5) Protein Structure Modelling; Drs. J. Kraut and S. Freer (University of California at San Diego) and E. Feigenbaum (Stanford)

As an additional aid to new projects or collaborators with existing projects, we provide a limited amount of funds for use to support terminals and communications needs of users without access to such equipment. We are currently leasing 6 terminals and 4 modems for users as well as 4 foreign exchange lines to better couple the Rutgers project into the TYMNET and a leased line between Stanford and U. C. Santa Cruz for the Chemical Synthesis project.

STANFORD COMMUNITY BUILDING

The Stanford community has undertaken several internal efforts to encourage interactions and sharing between the projects centered here. Professor Feigenbaum organized a seminar class with the goal of assembling a handbook of AI concepts, techniques, and current state-of-the-art. This project has had enthusiastic support from the students and substantial progress made in preparing many sections of the handbook as reported earlier. An outline of the material being prepared can be found in Appendix II on page 225 (see Book II). Several examples of completed articles are given in Appendix I on page 202 (see Book II).

A second community-building effort was a mini-conference on AI held at Stanford in January 1976. This 3 day series of meetings featured presentations by each of the local projects and comparative discussions of approaches to current problems in AI research such as knowledge representations, production system strategies and rule formation, etc. Weekly informal lunch meetings (SIGLUNCH) are also held between community members to discuss general AI topics, concerns and progress of individual projects, or system problems as appropriate as well as having a number of outside invited speakers.

AIM WORKSHOP SUPPORT

The Rutgers Computers in Biomedicine resource (under Dr. Saul Amarel) has organized a series of workshops devoted to a range of topics related to artificial intelligence research, medical needs, and resource sharing policies within NIH. Meetings have been held for the past two years at Rutgers and another is planned for this summer. The SUMEX facility has acted as a prime computing base for the workshop demonstrations. We expect to continue this support for future workshops. The AIM workshops provide much useful information about the strengths and weaknesses of the performance programs both in terms of criticisms from other AI projects and in terms of the needs of practicing medical people. We plan to continue to use this experience to guide the community building aspects of SUMEX-AIM.

RESOURCE ALLOCATION POLICIES

As the SUMEX facility has become increasingly loaded, a number of diverse and conflicting demands have arisen which require controlled allocation of critical facility resources (file space and central processor time). We have already spelled out a policy for file space management; an allocation of file storage is defined for each authorized project in conjunction with the management committees. This allocation is divided among project members in any way desired by the individual principal investigators. System allocation enforcement is implemented by project each week. As the weekly file dump is done, if the aggregate space in use by a project is over its allocation, files are archived from user directories over allocation until the project is within its allocation.

We have recently implemented system scheduling controls to attempt to maintain the 40:40:20 balance in terms of CPU utilization (see page 18). The initial complement of user projects justifying the SUMEX resource was centered to a large extent at Stanford. Over the first term of the SUMEX grant, a substantial growth in the number of national projects was realized. During the same time the Stanford group of projects has matured as well and in practice the 40:40 split between Stanford and non-Stanford projects is not ideally realized (see Figure 8 on page 43 and the tables of recent project usage on page 45). Our job scheduling controls bias the allocation of CPU time based on percent time consumed relative to the time allocated over the 40:40:20 community split. The controls are "soft" however in that they do not waste computer cycles if users below their allocated percentages are not on the system to consume the cycles. The operating disparity in CPU use to date reflects a substantial difference in demand between the Stanford community and the developing national projects, rather than inequity of access. For example, the Stanford utilization is spread over a large part of the 24-hour cycle, while national-AIM users tend to be more sensitive to local prime-time constraints. (The 3-hour time-zone phase shift across the continent is of substantial help in load-balancing.) For the present, we propose to continue our policy of "soft" allocation enforcement for the fair split of resource capacity. If necessary to assure proper apportionment, we can implement a pie-slice reservation system to more rigidly control the allocations.

Our system also categorizes users in terms of access privileges. These comprise fully authorized users, pilot projects, guests, and network visitors in descending order of system capabilities. We want to encourage bona fide medical and health research people to experiment with the various programs available with a minimum of red tape while not allowing unauthenticated users to bypass the advisory group screening procedures by coming on as guests. So far we have had relatively little abuse compared to what other network sites have experienced, perhaps on account of the personal attention that senior staff gives to the logon records, and to other security measures. However, the experience of most other computer managers behooves us to be cautious about being as wide-open as might be preferred for informal service to pilot efforts and demonstrations. We will continue developing this mechanism in conjunction with management committee policy decisions.

1.3.2.12 SUMMARY OF RESOURCE USAGE

The following data give an overview of SUMEX-AIM resource usage. There are five sub-sections containing data respectively for 1) monthly CPU time consumed, 2) resource usage by community (AIM and Stanford), 3) resource usage by project, 4) recent diurnal loading data, and 5) Network usage data.

MONTHLY CPU TIME CONSUMED

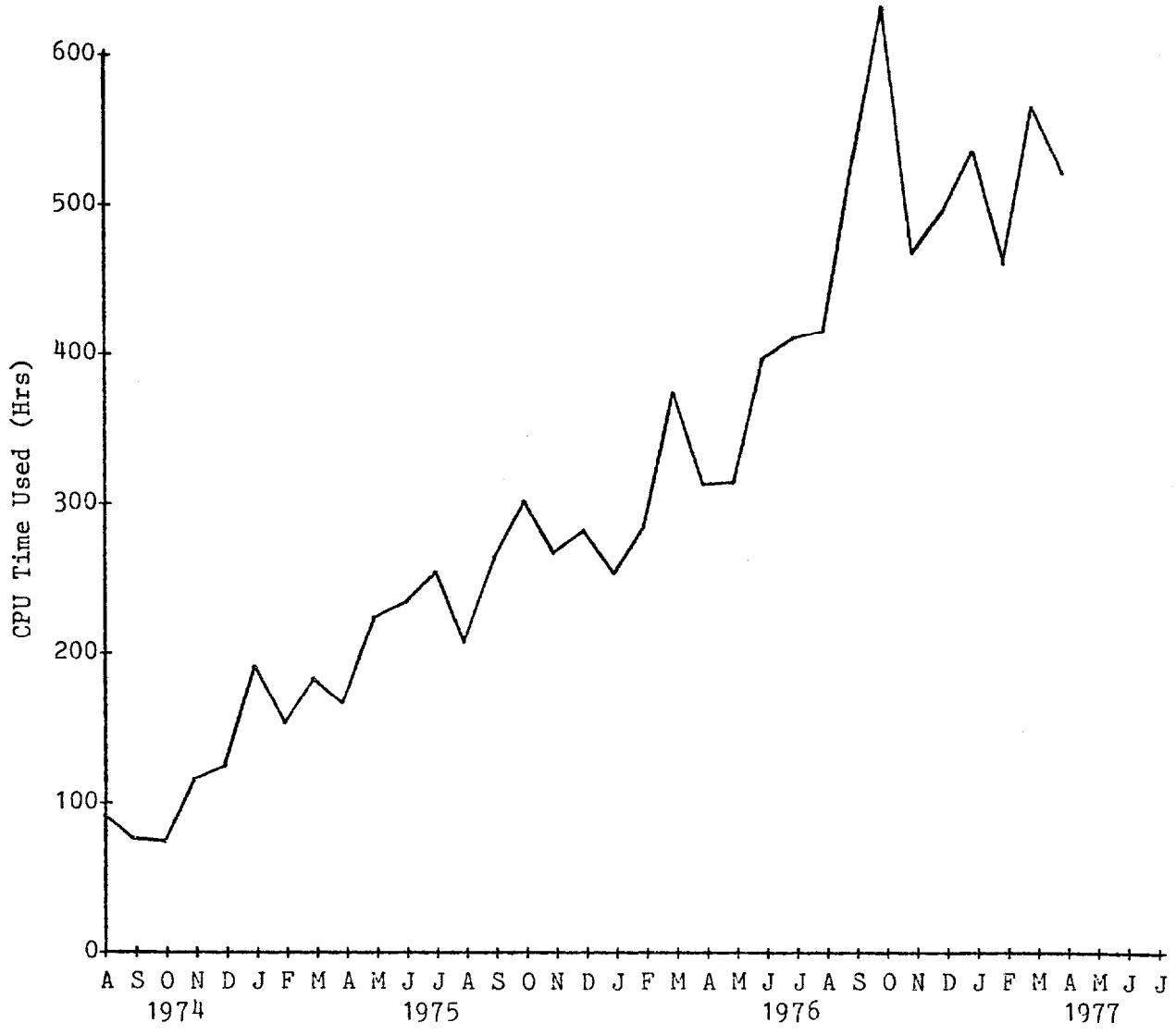


Figure 7. Monthly CPU Time Consumed

RELATIVE SYSTEM LOADING BY COMMUNITY

The SUMEX resource is divided, for administrative purposes, into 3 major communities: user projects based at the Stanford Medical School, user projects based outside of Stanford (national AIM projects), and common systems development efforts. As defined in the resource management plan approved by BRP at the start of the project, the available resource in terms of CPU capacity and file space will be divided between these communities as follows:

Stanford	40%
AIM	40%
Staff	20%

The "available" resources to be divided up in this way are those remaining after various monitor and community-wide functions are accounted for. These include such things as job scheduling, overhead, network service, file space for subsystems and documentation, etc.

The monthly usage of CPU and file space resources for each of these three communities relative to their respective aliquots is shown in the plots in Figure 8 and Figure 9. It is clear that the Stanford projects have held an edge in system usage despite our efforts at resource allocation and the substantial voluntary efforts by the Stanford community to utilize non-prime hours. This reflects the development of the Stanford group of projects relative to those getting started on the national side and has correspondingly accounted for much of the progress in AI program development to date.

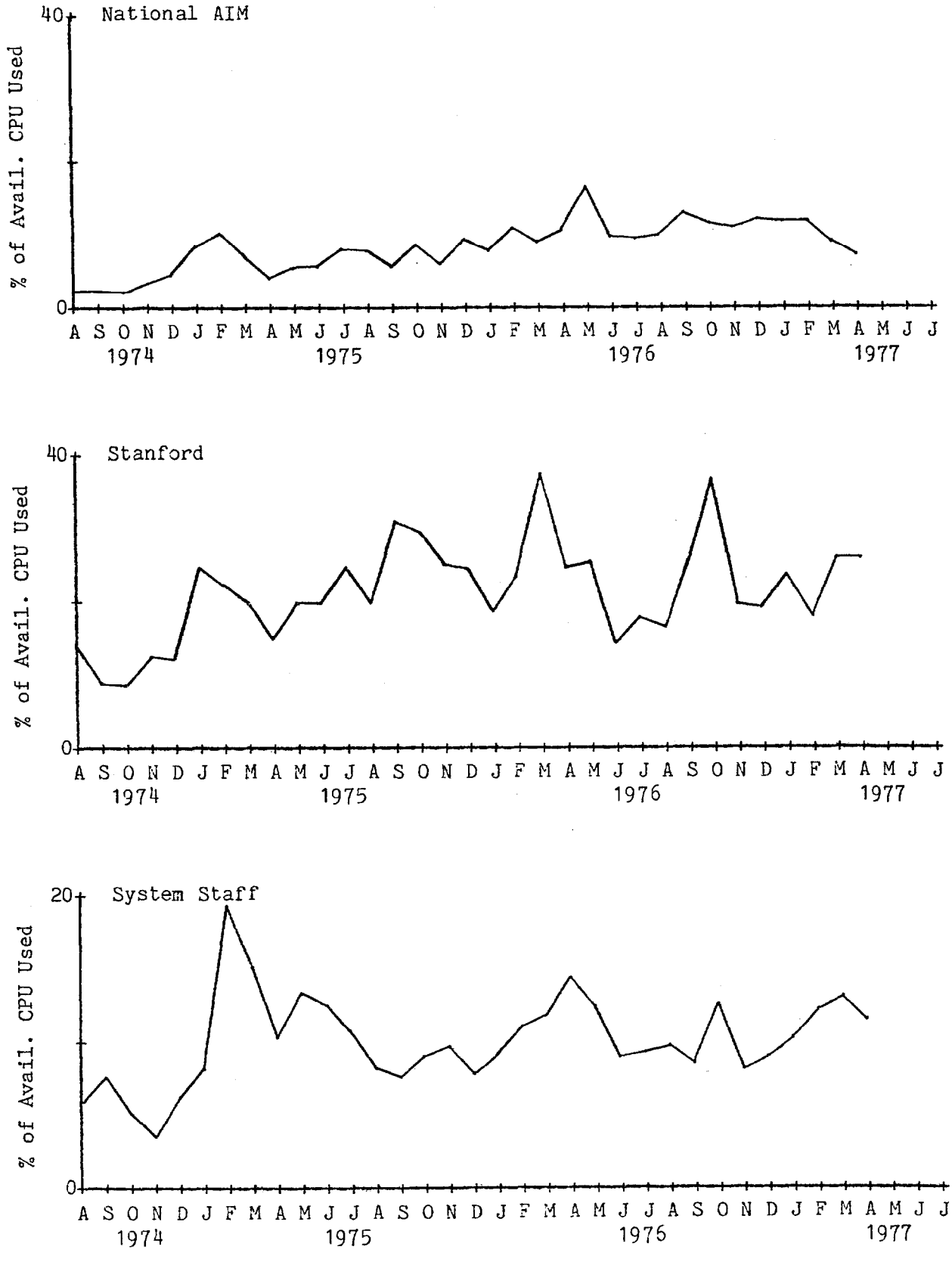


Figure 8. CPU Usage by Community

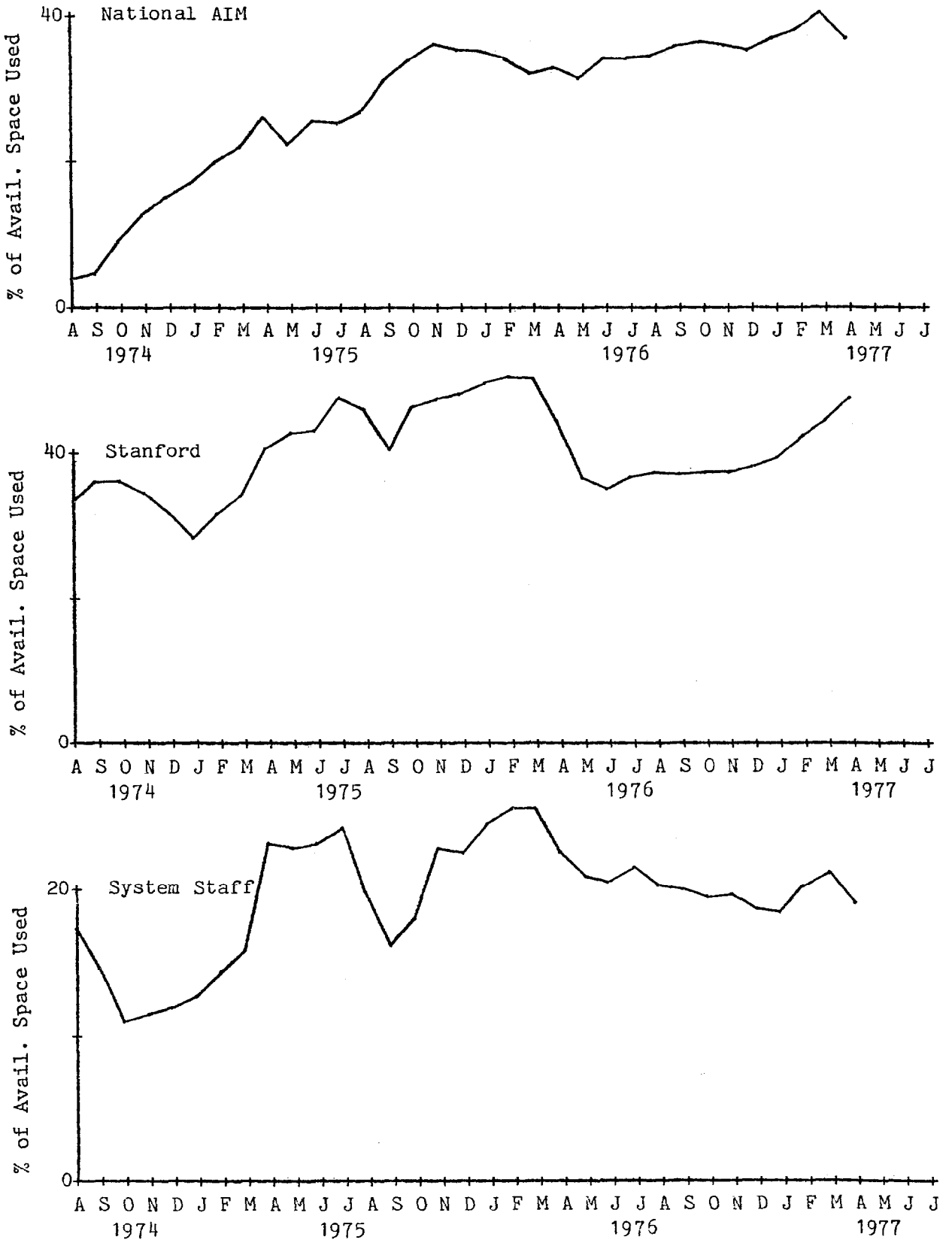


Figure 9. File Space Usage by Community

INDIVIDUAL PROJECT AND COMMUNITY USAGE

The table following shows cumulative resource usage by project in the past grant year. The data displayed include a description of the operational funding sources (outside of SUMEX-supplied computing resources) for currently active projects, total CPU consumption by project (Hours), total terminal connect time by project (Hours), and average file space in use by project (Pages, 1 page = 512 computer words). These data were accumulated for each project for the months between May 1976 and April 1977. Again the well developed use of the resource by the Stanford community can be seen. It should be noted that the Stanford projects have voluntarily shifted a substantial part of their development work to non-prime time hours which is not shown in these cumulative data. It should also be noted that a significant part of the DENDRAL and MYCIN efforts, here charged to the Stanford aliquot, support development efforts dedicated to national community access to these systems. The actual demonstration and use of these programs by extramural users is charged to the national community in the "AIM USERS" category, however.

RESOURCE USE BY INDIVIDUAL PROJECT

STANFORD COMMUNITY	CPU (Hours)	CONNECT (Hours)	FILE SPACE (Pages)
1) DENDRAL PROJECT "Resource Related Research Computers and Chemistry" NIH RR-00612-08 (3 yrs. 1977-80) ARPA DAHC-15-73-C-0435 (2 yrs. 1977-79)	1181.64	19657.56	13058
2) HYDROID PROJECT "Distributed Processing and Problem Solving" ARPA DAHC-15-73-C-0435	40.92	924.49	239
3) MOLGEN PROJECT NSF MCS76-11649 NSF MCS76-11935 (2 yrs. 1976-78)	85.61	2487.73	1853
4) MYCIN PROJECT "Computer-based Consult. in Clin. Therapeutics" HEW HS-01544 (2 yrs. 1977-79) NSF (2 yrs. 1977-79)	410.87	5640.75	6688
5) PROTEIN STRUCT MODELING "Heuristic Comp. Applied to Prot. Crystallog." NSF DCR 74-23461 (2 yrs. 1977-79) ARPA DAHC 15-73-C-0435	159.80	2894.19	2477
6) AIHANDBOOK PROJECT	26.46	464.42	639
7) PILOT PROJECTS (see reports in Section 6.3 in Book II)	327.67	5919.33	3506
COMMUNITY TOTALS	2232.97	38988.47	28460

NATIONAL AIM COMMUNITY				
1)	ACT PROJECT "Acquisition of Cognitive Procedures" NIMH MH29353 ONR N0014-77-6-0242	57.02	1195.84	986
2)	HIGHER MENTAL FUNCTIONS "Computer Models in Psychiatry and Psychother." NIH MH-27132-02 (2 yrs.) UCLA NPI Gen. Res.	206.03	2680.16	2198
3)	INTERNIST PROJECT (DIALOG) "Computer Model of Diagnostic Logic" BHRD MB-00144-03 (3 yrs.)	205.20	2721.26	3535
4)	MISL PROJECT "Medical Information Systems Laboratory" US-PHS-MB00114-03 (3 yrs.)	9.27	380.05	876
5)	RUTGERS PROJECT "Computers in Biomedicine" NIH RR-00643-05 (3 yrs.)	139.63	2433.43	10862
6)	SECS PROJECT "Chemical Synthesis"	308.96	4374.03	4515
7)	AIM PILOT PROJECTS (see reports in Section 6.4 in Book II)	40.91	1326.56	1558
8)	AIM Administration	11.13	383.22	1762
9)	AIM Users	56.89	672.35	362
		---	---	---
	COMMUNITY TOTALS	1035.04	16166.90	26654

SUMEX STAFF AND SYSTEM

1) Staff	903.07	23198.86	11919
2) Miscellaneous	80.87	2508.98	1721
3) Operations	1505.50	63113.94	32382
COMMUNITY TOTALS	2489.44	88321.78	46022
RESOURCE TOTALS	5757.45	143977.15	101136

SYSTEM DIURNAL LOADING VARIATIONS

The following figures give a picture of the recent variations in diurnal SUMEX system load, taken during March 1977. The plots include:

Figure 10 - Total number of jobs logged in to the system

Figure 11 - Percent of total CPU time used by logged in jobs (maximum is 200% for dual processor capacity)

Figure 12 - Percent of total CPU time consumed as overhead; I/O wait, core management, scheduling, etc. (maximum = 200%)

Figure 13 - Balance set size (number of jobs in core)

Figure 14 - Number of runnable jobs (whether or not in core)

The abscissa for these plots is broken into 20 minute intervals throughout the day. The ordinate for each interval is the average of all the daily measurements for that interval over the weekdays during March 1977. A daily measurement for a given 20 minute interval is in turn an average of the appropriate statistic sampled every 10 seconds. Since these plots display overall average data, they give representative illustration of the general characteristics of diurnal loading. There are, of course, substantial fluctuations in the quantities measured from day to day as well and for some, also on time scales shorter than the intervals displayed in the figures. For example in Figure 14, the number of runnable jobs (equivalent to the system "load average") shows a fairly smooth curve peaking at 6.7 jobs. On both a scale of minutes and from day to day, however, the number of runnable jobs will vary from only a few to 12 or more. This fluctuation is not shown in these average plots but also plays a role in the responsiveness of the system.

In the heading of each plot are shown range statistics for the measurement over various parts of the day. Range data include the minimum value "Low", average value "Ave", and maximum value "High". The first line of the heading gives the range over the whole day and on succeeding lines, "Prime Time" covers 6:00-18:00 Pacific time and "Non Prime Time" covers the remaining night time hours.

It can be noted in Figure 12 that the current overhead level for the dual processor system is quite high (about 33% per processor). This is because of the limited memory size (256K words) we currently have and the resulting increase in swapping interrupt rate and I/O wait time. We have a proposal pending with the AIM Executive committee to augment our memory which should reduce this overhead down to our earlier single processor levels (about 15-20% per processor).

Figure 10. Average Diurnal Loading (3/77): Total Number of Jobs

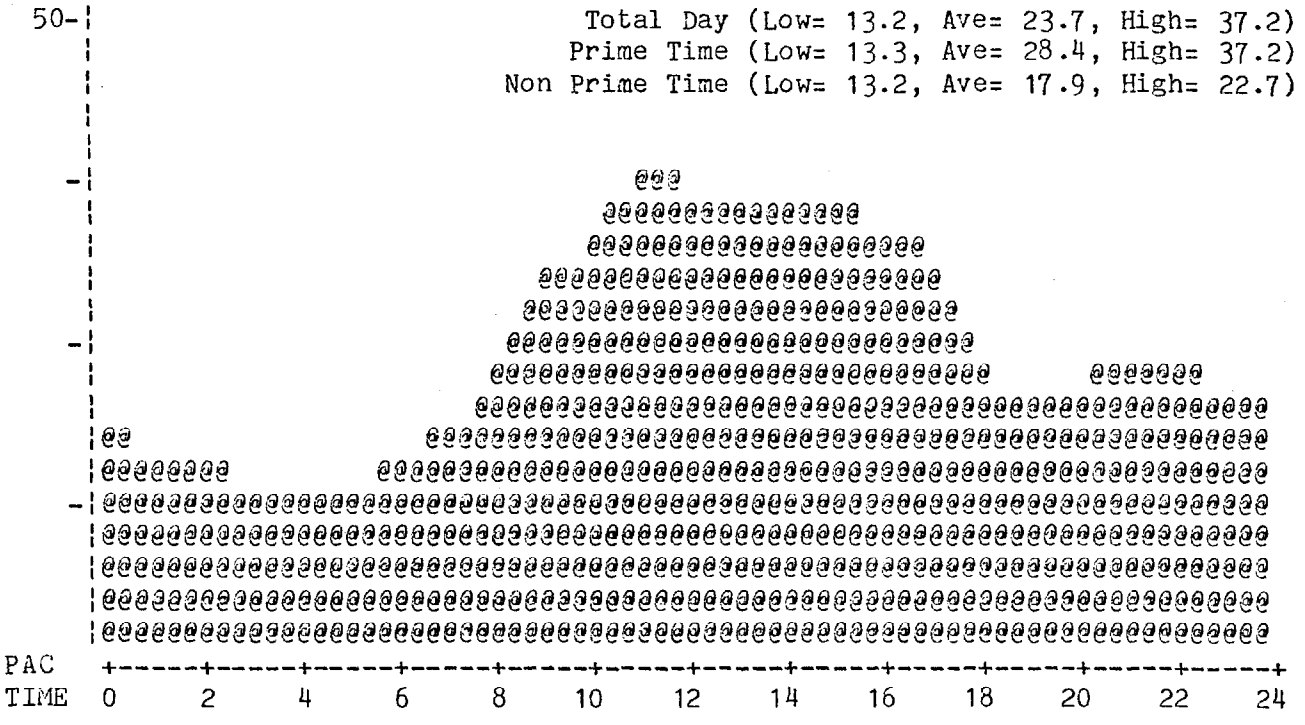


Figure 11. Average Diurnal Loading (3/77): Percent Time Used

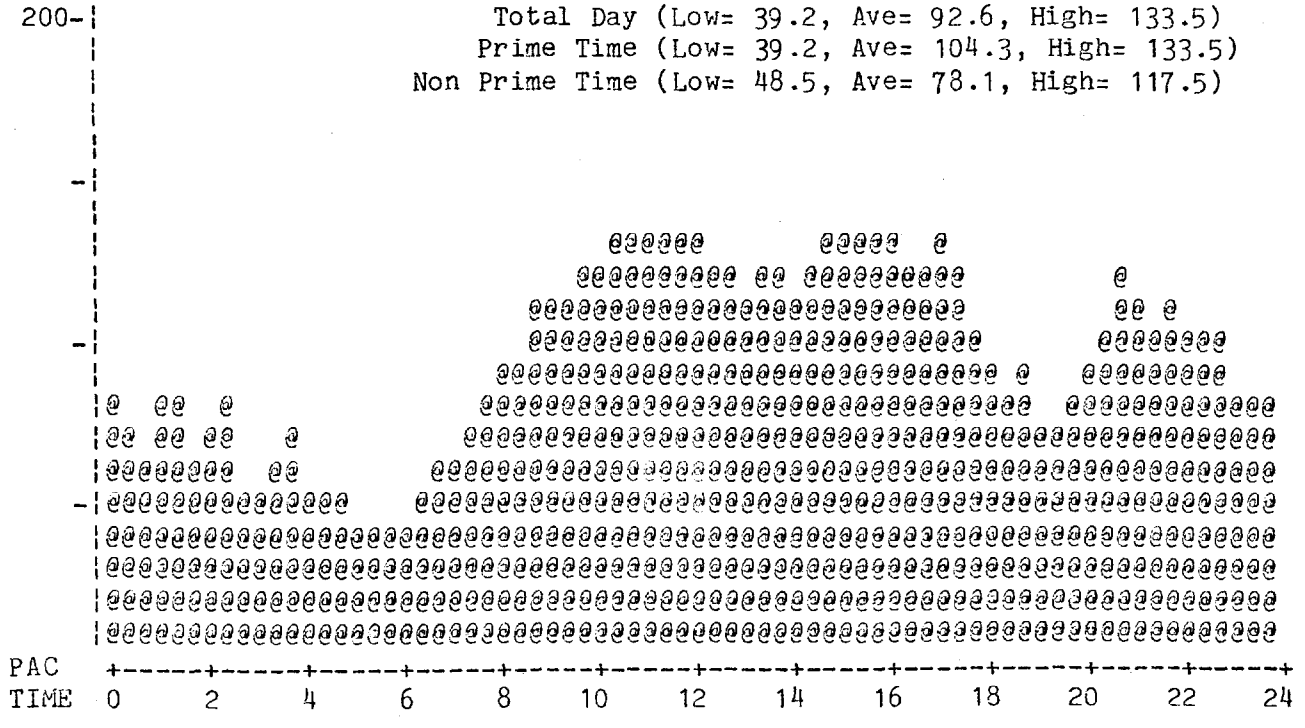


Figure 12. Average Diurnal Loading (3/77): Percent Overhead

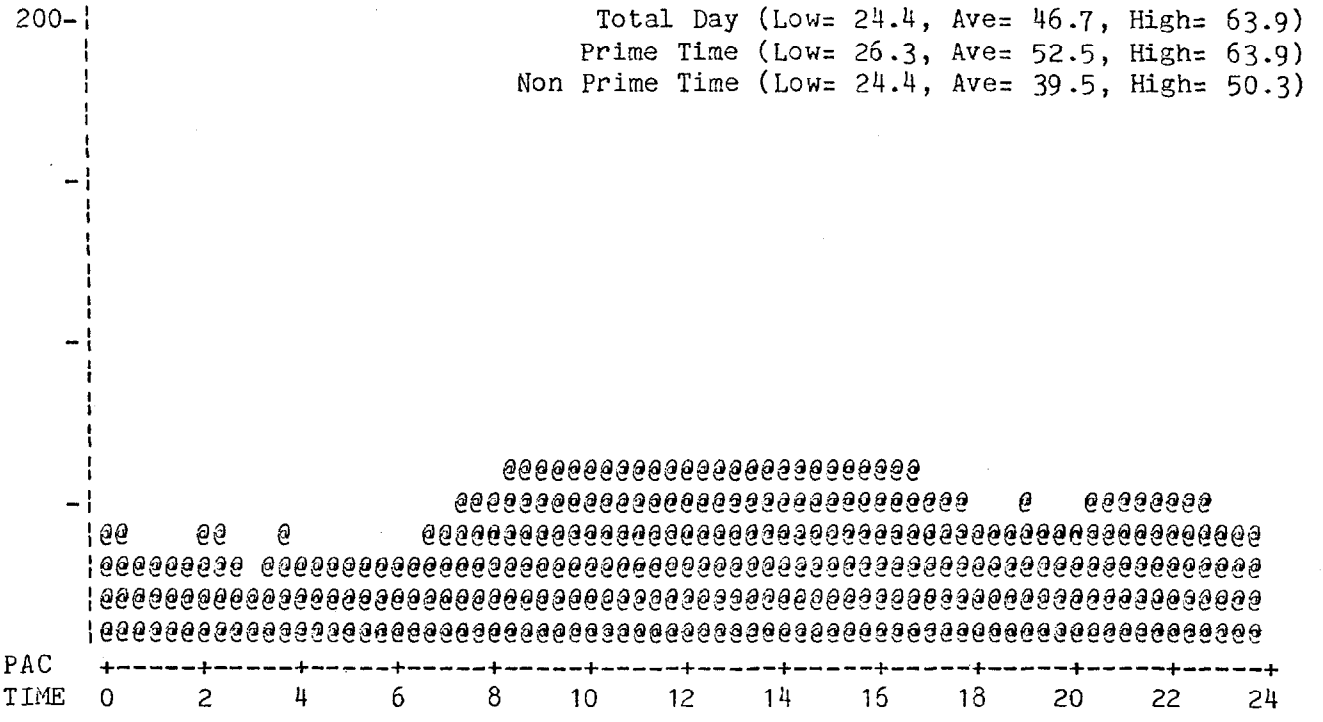


Figure 13. Average Diurnal Loading (3/77): Balance Set - Jobs in Core

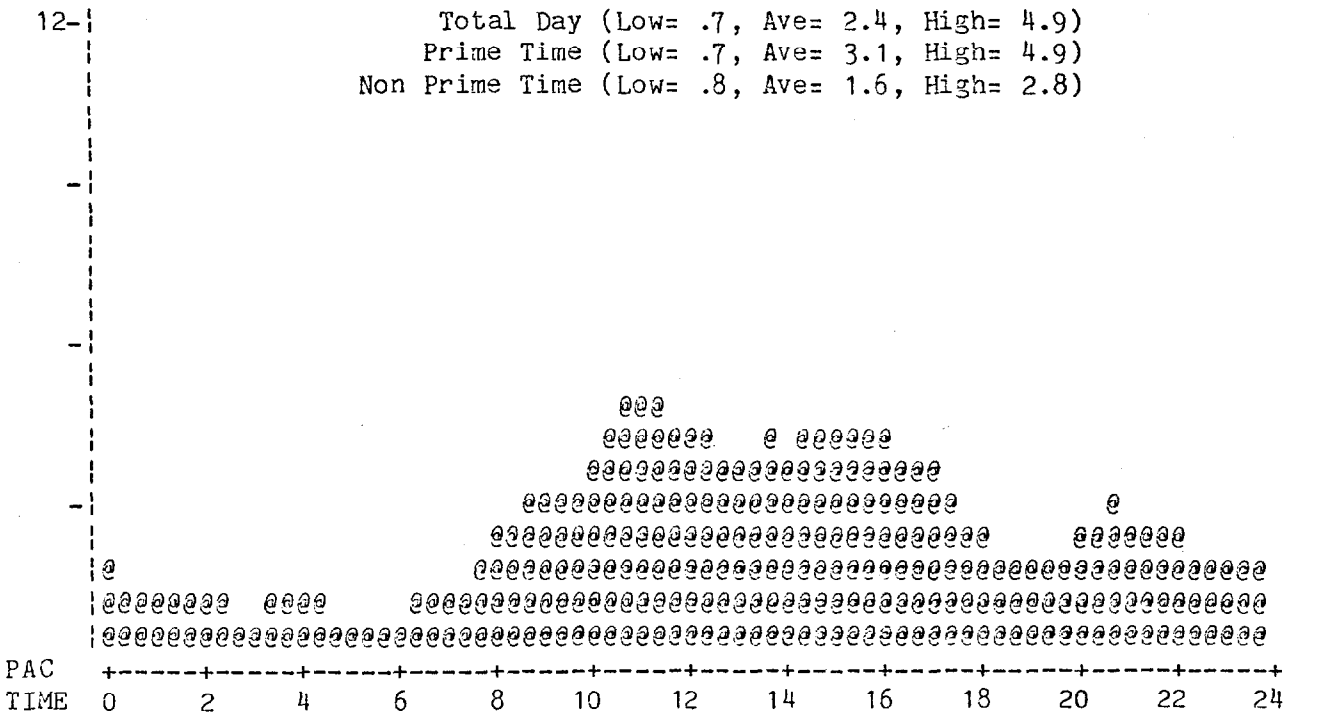
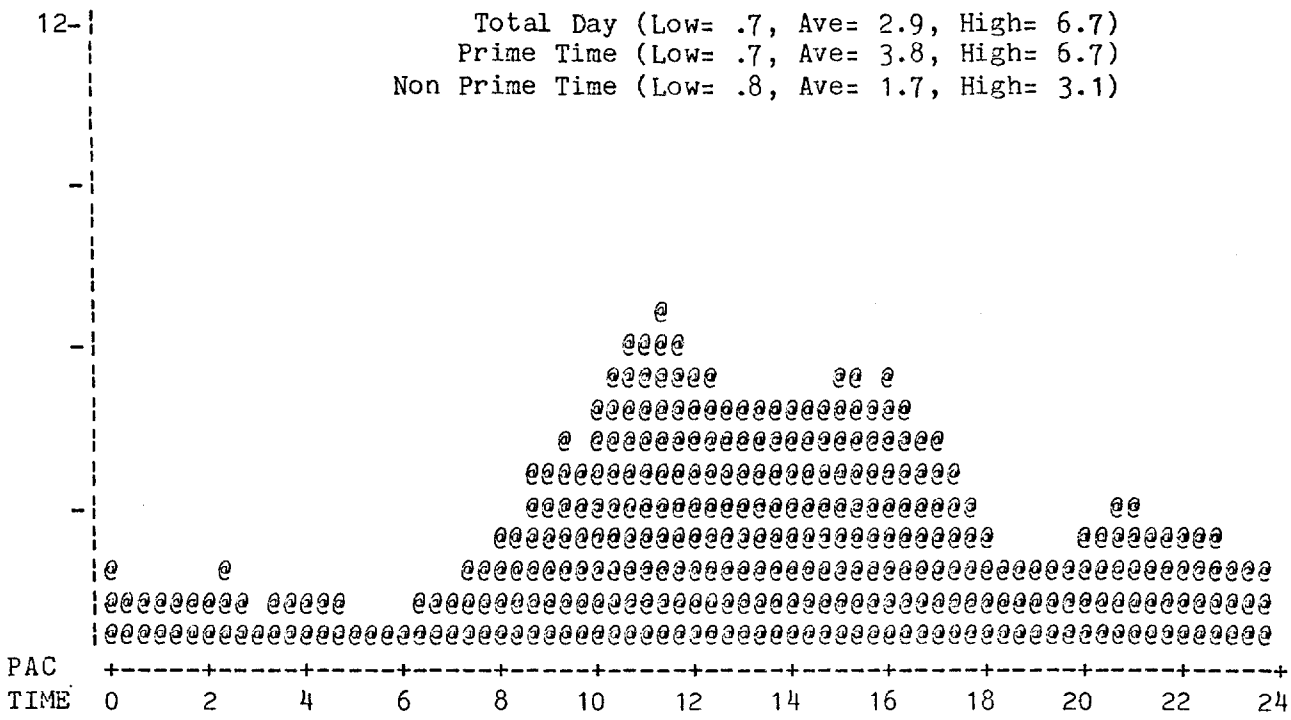


Figure 14. Average Diurnal Loading (3/77): Runnable Jobs



1.3.2.13 NETWORK USAGE STATISTICS

NETWORK USAGE PLOTS

The plots in Figure 15 show the major billing components for SUMEX-AIM TYMNET usage. These include the total connect time for terminals coming into SUMEX and the total number of characters transmitted over the net. The ratio of characters received at SUMEX to characters sent to the terminal is about 1:12 over our period of usage. Also shown for recent months is a plot of ARPANET connect time which tracks the corresponding data for TYMNET usage fairly closely. No data for "character" transmission is available for ARPANET since file transfers and terminal traffic use different byte sizes and these data are not resolved and maintained for the ARPANET.

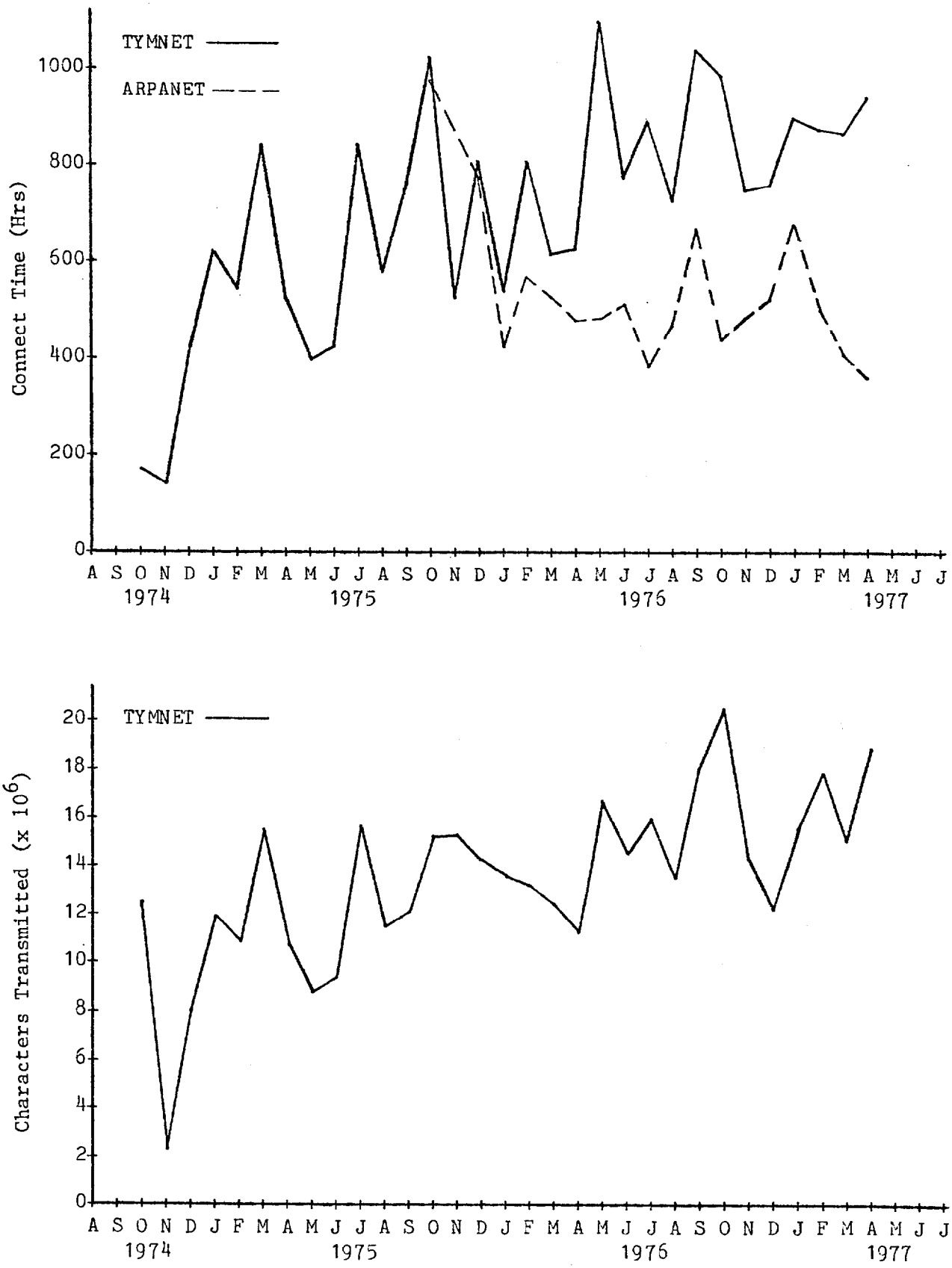


Figure 15. TYMNET and ARPANET Usage Data

1.3.2.14 PUBLICATIONS

The following are publications for the SUMEX staff and have included papers describing the SUMEX-AIM resource and on-going research as well as documentation of system and program developments. Publications for individual collaborating projects are detailed in their respective reports (see Section 6 on page 41 in Book II).

[1] Carhart, R.E., Johnson, S.M., Smith, D.H., Buchanan, B.G., Dromey, R.G., and Lederberg, J, "Networking and a Collaborative Research Community: a Case Study Using the DENDRAL Programs", ACS Symposium Series, Number 19, COMPUTER NETWORKING AND CHEMISTRY, Peter Lykos (Editor), 1975.

[2] Levinthal, E.C., Carhart, R.E., Johnson, S.M., and Lederberg, J., "When Computers Talk to Computers", Industrial Research, November 1975

[3] Wilcox, C. R., "MAINSAIL - A Machine-Independent Programming System," Proceedings of the DEC Users Society, Vol 2, No 4, Spring 1976.

Mr. Clark Wilcox also chaired the session on "Languages for Portability" at the DECUS DECsystem10 Spring '76 Symposium.

In addition as reported earlier, a substantial effort has gone into developing, upgrading, and extending documentation about the SUMEX-AIM resource, the SUMEX-TENEX system, the many subsystems available to users, and MAINSAIL. These efforts include a number of major documents (such as SOS, PUB, and TENEX-SAIL manuals) as well as a much larger number of document upgrades, user information and introductory notes, an ARPANET Resource Handbook entry, and policy guidelines (see Appendix VI, and Appendix VII in Book II). Publications for individual user projects are summarized in the respective reports (see Section 6 in Book II).

1.3.2.15 RESOURCE STAFFING HISTORY

PROFESSIONAL PERSONNEL (YEARS 01-04)

Name	Title of Position	(*) % of Effort	Period of Appointment
Lederberg, Joshua	Principal Investigator	10	10/1/73 - present
Rindfleisch, Thomas	Facility Manager	100	10/1/73 - present
Levinthal, Elliott	AIM Liaison	22	12/1/73 - present
Cower, Richard	System Programmer	100	6/24/74 - 6/15/77
Crossland, James	System Programmer	100	8/6/74 - 1/16/76
Gilmurray, Frank	System Programmer	100	6/1/77 (tent. start)
Heathman, Michael	System Programmer	100	10/1/73 - 8/15/75
Lieb, James	System Programmer	100	7/1/74 - 11/14/75
Reiss, Steven	System Programmer	100	10/1/73 - 7/31/74
Sweer, Andrew	System Programmer	100	1/19/76 - present
Tucker, Robert	System Programmer	100	6/1/77 (tent. start)
Schulz, Rainer	System Programmer - IMSSS	61	2/1/74 - present
Roberts, Ronald	System Programmer - IMSSS	50	2/1/74 - 7/31/74
" "	" " "	52	5/1/75 - 7/31/75
Smith, Robert	System Programmer - IMSSS	50	5/1/75 - 7/31/75
Quam, Lynn	Syst. Prog. - Cardiology	50	3/1/76 - 5/31/76
Johnson, Suzanne	Applications Programmer	100	7/22/74 - present
Smith, Nancy	Applications Programmer	100	3/25/74 - 8/20/76
Kahler, Richard	User Consultant	100	12/1/75 - present
Jackson, Phillip	User Support Specialist	100	11/18/74 - 7/28/75
Wilcox, Clark	Syst. Prog. - Res. Asst.	63	3/25/74 - present
Veizades, Nicholas	Electronics Engineer - IRL	50	10/1/73 - present
Nozaki, Thomas	Electronics Engineer - IRL	6	5/1/74 - present

(*) The figures shown give the % of effort during the respective periods of employment.

2 SPECIFIC AIMS

The following outlines the specific objectives of the SUMEX-AIM resource during the follow-on five year period. Note that these objectives cover only the resource nucleus; objectives for individual collaborating projects are discussed in their respective reports (see Section 6 on page 41 in Book II). We break our research aims into the categories 1) resource operations, 2) training and education, and 3) core research.

2.1 RESOURCE OPERATIONS AIMS

The broad objectives remain to provide an effective computing facility with extensive network access to support the community of projects developing AI applications in medicine. This goal includes the limited dissemination of these programs to outside research groups to provide the necessary feedback from actual research applications for effective program development. Specific aims include:

1) Continue the building of a community of projects applying AI techniques to medical problems including improving mechanisms for inter- and intra-group collaborations and communications. We plan to extend the existing AIM community management structure to accommodate justified growth in computing resources at other sites including a close collaboration between nodes on such a "resource network" and a meaningful division of responsibilities and regional expertise. To minimize administrative barriers to the community-oriented goals of SUMEX-AIM, we plan to retain the current user funding arrangements; user projects will fund their own manpower and local needs and will actively contribute their special expertise to the SUMEX-AIM community in return for an allocation of computing resources under the control of the AIM management committee structure. There will be no "fee for service" charges for community members. While AI is our defining theme, we may entertain exceptional applications justified by some other unique feature of SUMEX-AIM essential for important biomedical research.

2) Provide an effective computing resource to support the development and research dissemination of large and complex computer programs for a broad range of medical AI applications. This will include the continued development and refinement of the existing resource and the development and implementation of a plan for the upgrade of current hardware to the emerging next generation when justified by community, technical, and economic advantages.

3) Provide effective and geographically accessible network communication facilities to the SUMEX-AIM community for effective remote collaborations and to allow external users to experiment with available AI programs. We also plan to demonstrate the utility of network communications for scientific collaboration, in selected cases which do not interfere with our primary mission, to groups in other areas of computer science related to medicine. The ONET collaboration (see the Rutgers Resource progress

report on page 144) illustrates the value of these facilities apart from the AI programs themselves.

2.2 TRAINING AND EDUCATION AIMS

Our goals during the follow-on period for assisting new and established users of the SUMEX-AIM resource are a continuation of those adopted for the first grant term. Collaborating projects will provide their own manpower and expertise for the development and dissemination of their AI programs. The SUMEX resource will provide community-wide support and will work to make resource goals and AI performance programs known and available to appropriate medical scientists. Specific aims include:

1) Provide documentation and assistance in interfacing users to resource facilities and programs. We will continue to exploit particular areas of expertise within the community for developing pilot efforts in new application areas.

2) Continue to allocate "collaborative linkage" funds to qualifying new and pilot projects to provide for communications and terminal support pending formal approval and funding of their projects. These funds are allocated in cooperation with the AIM Executive Committee reviews of prospective user projects.

3) Provide support for a "visiting scientist" position to allow prospective qualified SUMEX-AIM project investigators or users to spend a term in close contact with on-going research work. The selection of appropriate candidates for this rotating position would be made in cooperation with the AIM Executive Committee.

4) Continue to support AIM Workshop activities in collaboration with the Rutgers Computers in Biomedicine resource.

2.3 CORE RESEARCH AIMS

Our core research efforts will emphasize the generalization and documentation of tools and techniques available for AI research and applications and the examination of alternative approaches for implementing and exporting large and complex AI performance programs. These efforts will be important community-wide to facilitate the investigation of new application areas and to meet the demand, beyond SUMEX-AIM capacity, for external users to be able to run developed AI programs conveniently. Fortunately, we have independent funding from various agencies for research activities that overlap the core-research

opportunity, e.g., CONGEN, MOLGEN, Heuristic Programming Project, and DENDRAL mass spectrometry. Specific aims include:

1) Continue to encourage community efforts at organizing and developing AI techniques by supporting projects such as the AI Handbook, special language developments (e.g., KRL), and other projects community members may propose to contribute.

2) Explore the generalizations of AI tools for knowledge acquisition, representation, and utilization; reasoning in the presence of uncertainty; strategy planning; and explanations of reasoning pathways. This effort will attempt to extract and generalize some of the best concepts and functional capabilities developed in the context of particular projects (e.g., DENDRAL, MYCIN, MOLGEN, etc.). The objective is to evolve a body of software packages that can be used to more efficaciously build future knowledge-based systems and explore other medical AI applications.

3) Explore AI software implementation and export mechanisms such as network communication systems, machine-independent languages, and special purpose computer systems. This will include the continued development of the MAINSAIL system and the investigation of microprogrammable machines specialized for target languages or satellite general purpose machines capable of running existing systems. Even the present level of computer capacity is not sufficient to meet the demands of a number of our projects. The DENDRAL CONGEN program is a good example where the potential for effective application to real biochemical structure determination problems is close but it simply takes too long to run problems that are really interesting. Therefore new approaches to computing are needed that may involve parallel processing, multiple small machines, or new developments from commercial vendors such as very much cheaper analogs of the PDP-10 that could be run in a more nearly dedicated mode.

3 METHODS OF PROCEDURE

This section details our plans for SUMEX-AIM goals during the next five year period. As indicated earlier, objectives and plans for individual collaborating projects are discussed in Section 6 on page 41 (see Book II). In general SUMEX-AIM will retain its community orientation in formulating and implementing a resource for AI research in medicine. We have had good success at integrating the tools and expertise of on-going active research efforts where possible and building on these where extensions or innovations are necessary. This orientation has proved to be an effective way to build the current facility and community and we expect it to be equally productive during the next period. We have assembled a growing community of projects which contribute to SUMEX-AIM resource goals and have at the same time come to depend on SUMEX for computing support and as a means of interacting with collaborators. We plan to continue our commitment to providing effective support to this community of projects.

This opportunistic approach also places constraints in synchronizing particular advances with our community needs. We are presently facing demands for increased computing resources as well as for effective methods for exporting mature AI performance programs. At the same time a new generation of hardware and firmware systems is just becoming available. These will have a large impact as a means to meet our goals, providing economic and technical advantages while minimizing redesign and reprogramming requirements. The anticipated timing for the announcement of a new generation of general purpose machines that might run AI software using existing operating systems and language support with substantially reduced capital investment is one to two years off. Such systems could be used to export software packages intact or to incrementally augment central resources like SUMEX. A similar situation exists for special purpose microprogrammable machines which can be tailored to particular language needs for increased throughput and efficiency. We aim to respond in a timely fashion to take advantage of this emerging technology but until concrete details are publically available, we can only describe our basic objectives and general design possibilities.

Thus the following description of research plans concentrates on software issues in planning for assimilation of the new technologies with the expectation that hardware announcements one to two years hence will impel careful reconsiderations of our strategies. Detailed budgets for computing hardware conversions are only approximate pending more detailed information on pricing. Our approach is to describe the research concept and gross estimated funding required, for review of these objectives at this time. We will further refine and elaborate the details of these plans during the first one to two years of the grant and submit them through the AIM Executive and Advisory Committees and the NIH Biotechnology Resources Program Office for approval prior to implementation.

3.1 RESOURCE OPERATIONS PLANS

3.1.1 SYSTEM HARDWARE AND MONITOR PLANS

As discussed in the progress section and supported by collaborating project reports, we have implemented an effective computing resource to support AI applications to medical research. We have augmented the present system to increase its effective capacity as far as we economically can to meet community needs. We do not propose any substantial changes either in scope of the existing resource or in its capacity. Other members of our community have proposals pending for other regional centers which may be justified on their own merits and the needs of the AIM community. We support the development of such regional expertise and specialization where justified which may allow a more coherent adaptation of a particular facility's resources to the needs of a subset of the AIM community. For example, a substantial group of biochemical structure analysis projects has grown up (DENDRAL, Chemical Synthesis Project, Protein Structure Project, and Molecular Genetics Project) as well as a group of medical diagnostic projects (MYCIN, Rutgers ONET, and INTERNIST as well as several pilot efforts). If regionalization becomes indicated, AIM facilities could be reoriented to serve the special needs of these research and target communities via separate systems, while maintaining close administrative and informational ties. We cannot predict the funding support such new facilities might receive but we will cooperate fully in getting them started and in assuring effective management for the benefit of the overall AIM community.

Our own facility has operated at capacity since early in our present grant term owing to the continuing maturing of on-going projects and the recruitment of new users, despite the periodic augmentation. As indicated earlier, our present hardware cannot be augmented further without upgrades to major mainframe and memory components. This should be done only after optimizing with respect to available new systems which are scheduled for announcement in the next year or so. There have been a number of recent relevant announcements but these machines have not yet been of a capacity or economic advantage to warrant immediate upgrade (indeed our decision to develop the dual KI-10 processor system was made on the basis of optimum cost-effectiveness within current technology and budgets). Furthermore, these systems are being sold packaged with relatively expensive memory and file storage and future releases may allow a more cost-effective mix of components from multiple vendors.

Our hardware design is now approximately five to six years old and will be twelve years old by the end of the follow-on 5 year grant term. The economics and technical performance of the newer systems, the evolving software gaps from inherent backward incompatibilities, and the reliability and maintainability of our existing equipment will pose new opportunities and problems. They may point to a strong rationale for an upgrade of the SUMEX-AIM system to meet the needs of the AI community we are supporting. The costs of this new generation of hardware will represent a progressively smaller part of the overall effort, compared to human resource inputs, especially if user participation is fairly weighted.

The TOPS-20 system DEC is currently marketing is derived from TENEX but already, DEC has made changes which cause incompatibilites with earlier systems. Many of these are in the direction of improved system performance (file system redundancy, system call enhancements, etc.) while others are of less obvious value (file naming conventions, message file formats, etc.). Whatever the reason, DEC's TOPS-20 system will likely dominate future system purchases and will increasingly diverge from ours. This causes a larger burden in our pursuit of software sharing and will affect the ease with which we can cooperate with other potential AIM network nodes. To avoid effective isolation, we will have to maintain effective compatibility. DEC has no plans for making TOPS-20 run on KI-10's and it is not likely others will undertake this within the currently strict licensing restrictions and DEC's motivations to sell KL-10's. Our apparent alternatives are to upgrade to some KL-"n" system when this product line matures and fills out so a proper choice can be made or to progressively modify our current system to remain as compatible as possible. A hardware conversion would likely cost at least $500,000 (based on current prices, but presumably much less as time passes) while system modifications for compatibility will entail 1-2 additional people per year in software effort. The cost of the latter approach must also include a measure of user community investment to circumvent unavoidable residual incompatibilities. The choice for optimum return will depend on the timing of major price declines for a given hardware capability, and on the way that cognate facilities evolve and participate in sharing software burdens.

We do not expect these trade-offs to be clear before 1979. We tentatively propose to expend the man-effort required to maintain compatibility between our existing system and TOPS-20 so long as this remains tenable. We budget initially one person for this purpose and add an additional programmer at the middle of the grant term. If this approach proves too costly and ineffective, we may propose reallocating these funds for a hardware conversion. Such a contingency would be thoroughly reviewed with AIM management committees and the NIH-BRP before finalizing a plan or requesting additional funding.

In the meantime we plan to reevaluate the performance of our existing system to wring out any remaining inefficiencies for more effective community support. The dual processor system has stabilized nicely and with the memory augmentation we are implementing, we will have taken advantage of all of the obvious sources of inefficiency. We will rereview the detailed operation of the facility to try to uncover remaining areas of cleanup. Recent measurements show that a high percentage of available time (80-90% in one recent test) is spent in various system routines which provide the rich set of monitor calls available through the TENEX system. It is therefore important to optimize the efficiency of the most widely used calls.

We also plan as part of this investigation to examine alternative strategies for managing memory allocations to running jobs. This will include attempting to minimize paging overhead by preloading job working sets to better utilize and overlap swapping I/O with other activities rather than waiting for page faults to read in pages on demand. We will also consider giving some program control over working set definition.

3.1.2 COMMUNICATION NETWORK PLANS

Networks remain centrally important to the research goals of SUMEX-AIM. We have had good success at meeting the geographical needs of the community during the early phases through our ARPANET and TYMNET connections. The major problems focus on terminal interaction delays through relatively slow or congested network facilities. In the next year or so TYMNET will be announcing their upgraded network (TYMNET II) which may offer additional advantages for our community such as higher terminal speeds, more dynamic terminal routing, and inter-host communications. If additional AIM servers are implemented, it will be important to coordinate their network access with that of SUMEX for effective user interactions and system collaborations.

During this same period ARPANET may be undergoing similar redesigns and possible further specialization to defense needs. In parallel, the TELENET facilities are evolving rapidly and whereas they offer a symmetric service for file transfer and terminal traffic, character delays are currently too high to warrant connecting immediately. We expect to retain our present connections over the early phases of the follow-on grant and to evaluate new upgrades as they become available. The specific goals for this upgrade will be improved terminal support and effective file transfer mechanisms available community-wide, particularly to interact with other AIM nodes.

3.1.3 SOFTWARE SUPPORT PLANS

We will continue to maintain the system, language, and utility support software on our system at the most current release levels, including up-to-date documentation. We will also be extending the facilities available to users where appropriate, drawing upon other community developments where possible. We rely heavily on the needs of the user community to direct system software development efforts. Two specific areas we plan to pursue are extensions to the bulletin board system and improved facilities for managing and organizing collections of related information as for example, program libraries and documentation, bulletin board or message files, collections of user profile information, etc. Bulletin board extensions will include improved facilities for searching for relevant information, associating a given bulletin with multiple topic labels, and more effectively apprising users of new information of interest. We are also examining extensions of the TENEX file system syntax and design to allow better logical organization and access to groups of file information. This may include facilities to define a hierarchical data structure, a"file system within a file", to name and manipulate logically related but independent pieces of information. A number of programs use ad hoc directories to access segments of information. We would hope to better standardize and improve such tools.

3.1.4 COMMUNITY MANAGEMENT PLANS

We plan to retain the current management structure that has worked out well for the recruitment and review of new projects and the guiding of resource policy formation. We expect the Executive and Advisory Committees to play a continuing important role in advising on priorities for facility evolution and on-going community development efforts such as MAINSAIL in addition to their recruitment efforts. The composition of the Executive committee will grow as needed to assure representation of major user groups and medical and computer science applications areas. The Advisory Group membership rotates with each member serving one to two years and spans both medical and computer science research expertise. We expect to maintain this policy.

The AIM workshops under the Rutgers resource have served a valuable function in bringing community members and prospective users together. We will continue to support this effort in terms of the Stanford community participation and providing a computing base for workshop demonstrations and communications.

3.2 TRAINING AND EDUCATION PLANS

We have an on-going commitment, within the constraints of our staff size, to maintain a high level of documentation of the evolving software support on the SUMEX-AIM system and to provide user help facilities such as the HELP and Bulletin Board systems. These latter aids are the best way we can assist resource users to find the information they need when they need it to solve access problems. Since much of our community is geographically remote from our machine, these on-line aids are indispensible for self help. We will also provide on-line personal assistance to users within the capacity of available staff through the SNDMSG and LINK facilities.

We allocate funds in our budget to continue the "collaborative linkage" support initiated during the first term of the SUMEX-AIM grant. These funds are allocated under Executive Committee authorization for terminal and communications support to help get new users and pilot projects started.

We also have requested support for a "visiting scientist" position which will allow selected prospective investigators to gain first hand experience by visiting on-going projects such as at Stanford. We feel this can serve an important role in catalyzing the development of new application areas and in disseminating the AI programs and techniques developed within the SUMEX-AIM community. The selection of appropriate individuals will be coordinated with the AIM committees as well.

Finally, we will continue to actively support the AIM workshop series in terms of planning assistance, participation in program presentations and discussions, and providing a computing base for AI program demonstrations and experimentation.

3.3 CORE RESEARCH PLANS

3.3.1 GENERALIZATION OF AI TECHNIQUES

The SUMEX-AIM facilities have made it possible to explore many of the frontiers of Artificial Intelligence research within the context of specific systems of medical relevance. Among those issues are the acquisition, representation and utilization of knowledge (both formal and judgmental), reasoning under uncertainty, explanation of a program's reasoning steps, and strategy planning. During the next period we wish to extract some of the best concepts and programming techniques from the specific programming systems, demonstrate their generality by incorporating them into other working programs, and design and implement packages which can be used to construct other high performance, knowledge based systems.

The five projects described below are proposed as basic core research in support of the various AIM community projects applying the techniques of AI research to biomedical problems. References for this material can be found on page 76. Because these projects are extensions of on-going work, we are able to generalize from existing programs without requesting support for maintenance or development of the programs themselves. This is another example of the synergistic community interactions of the SUMEX-AIM resource.

3.3.1.1 DESIGN OF KNOWLEDGE-BASED CONSULTATION SYSTEMS

Objective

Recent work has suggested that one key to the creation of intelligent systems is the incorporation in programs of large amounts of task-specific knowledge. We intend to develop (i) methods of using large stores of expert knowledge as a foundation for computer-based reasoning, and (ii) methods of facilitating the knowledge transfer from human experts to computer programs. We believe that this will lead to principles that may help turn the art of building large systems into more of a science, and thus aid other investigators who are building large knowledge-based systems. To do this, we will work on a number of problems involving knowledge representation, accumulation, management, and use, in the context of a software "laboratory" designed to facilitate the construction and use of large knowledge bases.

Motivation

Some of the earliest work in artificial intelligence centered around the attempts to create generalized problem solvers. Work on programs like GPS [Newell72] and theorem proving [Nilsson71], for instance, was inspired by the apparent generality of human intelligence and motivated by the belief that it might prove possible to develop a single program applicable to all (or most) problems. While this early work demonstrated that there was a large body of

useful general purpose techniques (such as problem decomposition into subgoals, and heuristic search in its many forms), these techniques did not by themselves offer sufficient power for high performance.

Recent work has instead focussed on the incorporation of large amounts of task specific knowledge in what have been called "knowledge-based" systems. Rather than non-specific problem solving power, knowledge based systems have emphasized high performance based on the accumulation of large amounts of knowledge about a single domain.

A second successful focus in work on intelligent systems has been the emphasis on the utility of solving "real world" problems, rather than artificial problems fabricated in simplified domains. This is motivated by the belief that artificial problems may prove in the long run to be more a diversion than a foundation for further work, and by the belief that the field has developed sufficiently to provide techniques that can aid working scientists. While artificial problems may serve to isolate and illustrate selected aspects of a task, solutions developed for those selected aspects often do not generalize well to the complete problem.

There are numerous current examples of successful systems embodying both of these trends, systems which apply task-specific knowledge to real world problems. They include efforts at symbolic manipulation of algebraic expressions [Macsyma74], speech understanding [Lesser74], chemical inference [Buchanan71], and interactive consultants in a few specific areas [Pople75, Shortliffe76].

While all of these systems display an encouraging level of performance, however, two fundamental problems remain. First, assembling the knowledge base for each of these is a difficult, continuous task that has in most cases extended over several years. Second, the result of this effort is typically a system with an impressive level of performance, but only within a sharply limited domain of application. High performance has been achieved at the cost of generality and man-years of work in knowledge base construction.

But if programs require large stores of knowledge for high performance, can we take a step back and discover powerful and broadly applicable techniques for accomplishing this transfer of knowledge? That is, can we discover ways of facilitating the communication, management and use of large amounts of task-specific knowledge? The result would be an intelligent system whose generality arose from access to the appropriate human experts, and whose power was based on the store of knowledge it acquired from them.

Two central themes of the proposed work are facilitating knowledge base construction and improving the generality of the reasoning programs that use the knowledge base. We intend to employ a computer system based on broadly applicable techniques for knowledge encoding and use, and couple it with powerful techniques for accomplishing the transfer of knowledge from human experts to computer programs. The foundation for the computer system will be provided by the domain independent core of the Mycin system [Shortliffe76, Davis77]. This will be the basis for a software "laboratory" in which we can examine the relevant issues of knowledge representation, accumulation, management, and use. By setting this work in the context of a specific, existing body of software, a number of a very general issues become focussed into specific questions. Since

the program that constitutes our "laboratory" has been demonstrated to have a strong degree of domain independence, the results of this work will be widely applicable.

This should produce a new form of generality. Unlike GPS, we do not offer one program which can solve problems in any domain. Rather, we offer the foundation for a system, along with a methodology for instantiating that system in any one specific domain. The foundation and methodology provide a framework for the expression, management, and use of domain specific knowledge, to make this instantiation task a reasonable one. It is there in the foundation and the methodology that our generality lies, not in the final performance program which results.

3.3.1.2 ATTEMPT TO GENERALIZE (AGE) PACKAGE

The objective of this research is to isolate inference, control and representation techniques from previous knowledge-based programs; reprogram them for domain independence; write a rule-based interface that will help a user understand what the package offers and how to use the modules; and make the package available to SUMEX users, other research groups engaged in knowledge-based systems development, and the general scientific community.

Detailed Discussion:

The goal of this new effort is to construct a computer program to facilitate the building of knowledge-based systems. The design and implementation of the program will be based primarily on the experience gained in building knowledge-based systems at the Heuristic Programming Project in the last decade. The programs that have been built are: DENDRAL[Buchanan71], meta-DENDRAL[Buchanan72], MYCIN[Shortliffe76], AM[Lenat76], HASP[Nii77], Protein Structure Modeler[Engelmore77], and MOLGEN[Stefik77] (the latter two currently under development). Initially, The AGE program will embody methods used in our programs. However, the long-range objective is to integrate methods and techniques developed at other A.I. laboratories. The final product is to be a collection of useful "building-block" subprograms, combined with a knowledge-based front-end that will assist a user in constructing knowledge-based programs. It is hoped that AGE can speed up this process and facilitate transfer of the technology by: (1) packaging common AI software tools so that they do not need to be reprogrammed for every problem; and (2) helping people who are not knowledge-engineering specialists to write knowledge-based programs.

Two Specific Research Activities of the AGE Effort are:

1. The isolation of techniques used in knowledge-based systems. It has always been difficult to determine if a particular problem-solving method used in a knowledge-based program is "special" to a particular domain or whether it generalizes easily to other domains. In the currently existing knowledge-based programs the domain-specific knowledge and the manipulation of such knowledge using AI techniques are often so closely

coupled that it is difficult to make use of the programs for other domains. We need to isolate the AI techniques that are general to determine precisely the conditions for their use.

2. Guiding users in the application of these techniques. Once the various techniques are isolated and programmed for use, an "intelligent front end" is needed to guide users in their application. Initially, we assume that the user understands AI techniques and knows what he wants to do, but that he does not understand how to use the AGE program to accomplish his task. The program at this stage of the development will need to have the basic tools coupled with a package to guide the user in applying these tools. A longer-range interest involves helping the user determine what techniques are applicable to his task. That is, we assume that the user does not understand the necessary techniques of writing knowledge-based programs. Some questions to be posed are: What are the criteria for determining if a particular application is suited to a particular problem-solving framework? How do you decide the best way to represent knowledge for a given problem?

   There are some smaller, but by no means trivial, questions which also need answering. Is there a "best way" to write production rules which would apply to many task domains? Is there a data representation that would cover many tasks? What is the best way to handle differences in the ability of the users of the AGE program?

Research Plan:

The AGE program will be developed along two separate fronts, both of which are divided into incremental development stages. The first of these fronts is the development of the ability to help build many different types of knowledge-based programs (the "generality" front). The second front is the development of "intelligence" in the interaction between the user and the AGE program; i.e. moving from dialogues on "how to use the tools in AGE" to "what tools to use" (the "how-to-what" dialogue front). The proposed development plan contains the following stages:

a. Generality: The development of a program package that will enable the user to build "HASP-like" knowledge-based programs characterized by the integration of multiple sources of knowledge, multi-level representation of solution hypotheses, opportunistic problem-solving methods, and explanation capability of the reasoning steps. The HASP-like paradigm has been used to solve problems of interpreting large amounts of digitized physical signals, but can also be extended to problems of processing large amounts of symbolic data.

   Dialogue: The development of dialogue to show the user how to utilize the packaged components in AGE to build HASP-like programs. The interactive capability will be limited to: specifying how to build multi-level hypothesis structure; how to write production rules to represent domain knowledge; and how to use various techniques available for opportunistic hypothesis formation.

b. Generality: Supplement the ability to build HASP-like programs with a capability to build MYCIN-like goal oriented programs.

Dialogue: Same level of dialogue capability with additional ability to discuss how to chain rules and how to specify the necessary parameters for the context tree.

c. Generality: Same level as for b., i.e. ability to build HASP-like, MYCIN-like or combination of HASP- and MYCIN-like knowledge-based programs.

Dialogue: Begin to extract from the user some key characteristics of the task, and using that information begin to suggest appropriate knowledge representation and problem-solving techniques for the user's task. This interactive capability will be limited to the generality level at this point in the AGE development.

d. Test phase: Test the usefulness of the AGE system by developing an application program in some task domain. (a) An application program will be chosen from among on-going program development efforts within our own project or within the SUMEX-AIM community. An application will be chosen whose primary task is that of interpreting large amounts of symbolic data or described signal data. (b) Collect specific knowledge needed for the application program and begin to develop the program using the AGE system.

3.3.1.3 PLAN PACKAGE

The PLAN package is oriented toward the representation of plans-of-action and toward an expert's knowledge of the best problem solving strategies to employ in his domain. A feature of the package is its ability to make inferences on components of planning and strategy rules so that new plans and strategies can be constructed readily from previous ones. The representation will allow the manipulation of various "levels of detail" of plans and strategies. The package will be made available as previously mentioned in connection with AGE.

Detailed Discussion:

Before starting a technical presentation of the ideas for the Plan Package, it is worth highlighting some of the issues which motivate its development.

a. How can a variety of types of domain actions be accommodated in a knowledge base?

b. How can a variety of types of strategy and control knowledge be incorporated in a knowledge base?

c. How can a variety of types of problem solving states be expressed and manipulated by the system?

d. How should plans be represented?

e. How can the problem statements for a variety of types of problems be acquired?

f. How does the expression and representation of problem solving states relate to the expression of the domain and strategy knowledge?

The Plan Package consists of two major entities -- the Planning Network and the Strategy Package. The Planning Network is a set of software which manages the representation of the plans created during the problem solving process. When a problem is acquired from a user, it is represented as an initial planning network. Problem solving takes place as the active strategy rules manipulate the planning network to create solutions. The Strategy Package itself is discussed in the next section.

Since the planning state knowledge is important for the expression of strategy in the Plan Package, it is worthwhile exploring briefly the nature of this knowledge. It is useful to consider the planning network as being composed of three parallel planes -- the solution plane, the planning plane, and the focus plane. These planes contain (1) the solution steps (domain rule applications) and world states, (2) the planning and design steps and (3) the focus of attention knowledge respectively. All three planes of the network are built dynamically during the problem solving process. Different types of nodes in the network correspond to the different components of the problem solving process.

A number of issues have been raised about the management of strategy knowledge.

a. How should strategies be expressed?

b. How can strategy information be assimilated so that the system will use it appropriately when designing or explaining solutions?

c. How can a knowledge based system assist a domain expert in structuring and expressing his ideas about strategy?

Means-ends analysis is one of the simplest ideas in the current stock of methods for problem solving. As such, it should exist as a standard strategy in a strategy package of artificial intelligence techniques to be used as needed. The current state of artificial intelligence, where a researcher must re-code Means-ends analysis any time he wishes to use it is akin to a carpenter forging a new hammer for each job.

One approach for making an instance of Means-ends analysis available as a tool would be to provide a packaged program which accepts arguments for the various components of Means-ends analysis (e.g. a difference table, difference function, etc.). The alternative being proposed here is a system which uses schemata to drive the strategy acquisition process and which can guide a user through the details. The goal is to create a supportive environment for the painless testing of fairly high level strategies. Such a system should be able to draw on its knowledge base to provide assistance in casting a problem into a Means-ends framework.

In summary, other systems have stumbled over the expression of more complex forms of domain and strategy rules and have been limited to solving a single kind of problem. We propose extending this work by developing what we have termed the Plan Package. The Plan Package consists of two major components - a schema-based representation for the problem-solving states termed the Planning Network and a schema-based representation for domain rules and strategies termed the Strategy Package. The Planning Network will provide a representation for a variety of types of problem solving so that the problem solving system will be able to solve more than one type of problem. The Strategy Package will provide a set of standard artificial intelligence strategies in the form of schemata, which may be instantiated into strategy rules when they are supplied with the particulars of domain knowledge. These schemata will facilitate the acquisition of tailored strategies by guiding a user a step at a time through the particulars of the acquisition process.

The Plan Package will be developed and tested in the domain of molecular genetics as part of the MOLGEN project. It will be further developed and extended to other domains as a test for generality as part of the AGE project.

3.3.1.4 HEURISTIC KNOWLEDGE ACQUISITION

Automatic Rule Formation Methods

Given a body of data from which rules are to be formed, together with a basic approach to rule induction, there remains a range of ways in which the data may be utilized, which differ in the degree of parallelism involved in the examination of instances. At one extreme are methods in which rules are formed and refined in a sequence of steps, each step involving the examination of one new instance. At the other extreme are methods which involve a single-pass rule formation process, using all available data. There are, of course, many intermediate possibilities. We propose to investigate, within the Meta-DENDRAL framework, whether some of these methods are optimal in the sense of yielding rules of comparatively high quality with the expenditure of comparatively little computing effort. It is hoped that the investigation will lead us to some general insights concerning the optimal utilization of data in automatic rule formation.

Research Plan:

a. Develop and implement one or more procedures for updating an evolving set of rules on the basis of newly examined data. These procedures will make use of existing capabilities of the RULEGEN and RULEMOD programs, and will make possible the implementation of a variety of schemes for data utilization, as described above.

b. Select and implement a representative subset of the class of data utilization schemes indicated above, and test their performance in the application area of mass spectrometry.

c. Describe in a technical report these experiments, their results, and the lessons learned.

Rule Acquisition via Dialogue

Since large stores of knowledge appear to be required for high performance, the process of accumulating that information should be made as easy as possible. The fundamental question here is, how can we make it easy for the expert to tell the system what he knows about the domain. Some initial steps in this direction are described in [Davis76], which reports on the use of what has been labelled "meta-level knowledge" as a basis for establishing communication between the system and an expert. In the simplest terms, meta-level knowledge refers to giving the system the ability to "know what it knows", and can support a wide range of useful abilities.

The basic approach developed there relies on the notion of knowledge acquisition in the context of a shortcoming in the knowledge base. That is, rather than simply asking an expert to "explain all he knows about the field", we allow him to challenge the system with difficult problems and observe its behavior. If he indicates at some point that the system has made a mistake, there is available a large amount of contextual information which can aid in the process of knowledge explication and communication. Thus rather than asking "What is there to know about this domain?", we can say "Here is a problem on which you claim the system made a mistake. Here is the knowledge it used to reach its answer. Now WHAT IS IT THAT YOU KNOW AND THE SYSTEM DOESN'T that allows you to avoid making that mistake?"

This appears to be an effective approach to the problem, since it creates a well defined context, allowing the expert to focus his attempt to describe his knowledge of the domain, and provides the system with a set of expectations about the content of the new knowledge it is going to receive. Both of these offer significant advantages in helping to build up the knowledge base.

Working from this foundation, we plan to extend these ideas to provide a powerful system for knowledge acquisition. Currently, for example, the scope of the context is limited to a particular error in the knowledge base during a particular session with the expert. It ought to be extended to provide a wider perspective, so that the system could form more sophisticated expectations about a particular tutor, thereby making communication between them more effective. Thus rather than forming expectations concerning only the shortcoming presently under examination, for example, the system might be able to consider also the past several shortcomings, in an attempt to detect a broader "theme" in the knowledge it was acquiring.

There ought also to be more effective control over its use of context. The system is currently too "single-minded", in that it holds tenaciously to any expectations it may have formed. There should be a way of indicating to the system that it has formed incorrect assumptions, and that it should "sit back and observe" for a while until it can get "reoriented".

Dealing with large knowledge bases also requires a range of auxiliary capabilities that assist the expert in keeping track of and organizing his work.

Together these constitute a "scratch pad" of sorts that allows him to annotate his new additions, mark existing rules that may need further work, or perhaps examine selected parts of the knowledge base to find areas that may presently be weak. All of these should be aimed at making it possible for the expert to extend his work over several sessions without loss of continuity, and to keep track of both changes that are required and work that has been done, no matter how large the knowledge base may eventually grow to be.

3.3.1.5 GENERAL EXPLANATION SYSTEM

The function of an explanation capability is to permit the user or builder of a knowledge based system to determine:

1. in general, how the system solves problems or uses information;
2. retrospectively, how the system solved a particular problem;
3. interactively, how and why the system came up with its current answers.

The success of the explanation capability for the MYCIN rule based system indicates the usefulness of this capability in debugging the system and in making it easier for a user to learn and believe the system's operations. To make it easier to build explanation capabilities for future knowledge based systems, including systems whose knowledge is embedded in procedures, we intend to construct a system which will provide explanations for a wide class of problem solvers.

Given the appropriate trace of a program's decisions and states, and a model of its problem solving process, it should be possible to answer a variety of well constrained but informative questions about program operation, in general or in a specific run. The aim of this research is to determine what sorts of traces and process models are needed to support selected types of explanations in several classes of knowledge based problem solvers. When the requirements for a class are determined, we intend to implement a general explanation facility to provide the selected explanations for programs in that class. Such a facility should be made useful for several classes of problem solver.

The steps of the research will include:

1. Choose the types of problem solvers to which the explanation system will be applied;
2. Select example knowledge based systems of each class (e.g. protein structure modelling as an example of event/model driven hypothesis formation systems);
3. For each system selected, determine questions to be asked, and what information, such as traces and process descriptions, are needed to answer them;

4. Implement a facility which accepts descriptions of problem solver class and enables the user to ask the questions for that class about an example system;

5. Investigate new kinds of explanation capabilities -- for example, how a program's operation might be meaningfully summarized for several kinds of users, such as domain experts and programmer/system designers.

References for this section

[Buchanan71] Buchanan B G, Lederberg J, The heuristic DENDRAL program for explaining empirical data, IFIP, 1971, pp 179 - 188.

[Buchanan72] Buchanan B G, et.al., Heuristic theory formation: data interpretation and rule formation, in Machine Intelligence 7, (Meltzer & Michie, eds), pp 267-292, 1972.

[Davis76] Davis R, Applications of meta level knowledge to the construction, maintenance, and use of large knowledge bases, (thesis), AI Memo 283, Stanford University, July 1976.

[Davis77] Davis R, Buchanan B, Shortliffe E, Production rules as a representation for a knowledge-based consultation program, Artificial Intelligence (to appear, Jan 77).

[Engelmore77] Engelmore, R, and Nii, H Penny, A Knowledge-Based System for the Interpretation of Protein X-Ray Crystallographic Data, 1977.

[Lenet76] Lenat, D, AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search, Ph.D. Thesis in Computer Science, 1976.

[Lesser74] Lesser V R, Fennell R D, Erman L D, Reddy D R, Organization of the HEARSAY II speech understanding system, IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-23, February 1975, pp 11-23.

[MACSYMA74] The MACSYMA reference manual, September 1974, The MATHLAB Group, MIT.

[Nii77] Nii, H. Penny and Feigenbaum, E, Rule-based Understanding of Signals, presented at Workshop on Pattern-directed Inference Systems, 1977.

[Newell72] Newell A, Simon H, Human Problem Solving, Prentice-Hall, 1972.

[Nilsson71] Nilsson N J, Problem Solving Methods in Artificial Intelligence, McGraw Hill, 1971.

[Pople75] Pople H, Meyers J, Miller R, DIALOG, a model of diagnostic logic for internal medicine, 4IJCAI, pp 848-855. (the system has recently been renamed INTERNIST).

[Shortliffe76] Shortliffe E H, Computer-based clinical therapeutics: MYCIN, American Elsevier, 1976.

[Stefik77] Stefik, M and Martin, N, A Review of Knowledge-Based Systems as a Basis for a Genetics Experiment Designing System, 1977.

3.3.2 SOFTWARE EXPORT ALTERNATIVES

Over the past few years, a number of the programs being developed by SUMEX-AIM projects have reached a developmental maturity where we need to consider ways of meeting the demands to make them operationally available to a larger user community and to export them where appropriate to other sites. Current examples of such programs include the CONGEN biochemical structure elucidation program, the SECS chemical synthesis analysis program, and the MYCIN, ONET, and INTERNIST medical diagnosis programs. Our present PDP-10 facilities are quite insufficient for meeting the operational needs of this growing group of users, even if providing this level of service were within the SUMEX-AIM mandate.

These programs have been written in a variety of source languages (principally various dialects of LISP or SAIL) and are characterized by very large address space requirements. The development medium for these programs at Stanford has been the PDP-10 TENEX environment and the choice of language made to facilitate development and representation of logical program concepts. In contemplating the export of such programs, several points seem relevant:

- Development is continuing on the programs to extend their conceptual framework and operational effectiveness. This implies that there must be a low threshold between developmental versions of the programs and operational ones during this phase and that the implementation environment of the programs must be conducive to both.

- Because of the complexity of the programs, it is likely that their maintenance and upgrade should be centralized. This implies a convenient means of receiving user feedback and of providing program updates.

- Because of the address space requirements for these programs (even after possible rewrite for increased efficiency), it does not appear reasonable to export them via 16-bit mini computers where unwieldy overlay structures would be required to circumvent the addressing constraints.

- The target community for these types of programs will be fairly heterogeneous. Users may include academic research groups, industrial houses, hospitals, and educational institutions. One can expect the native computing resources in these various user sites to cover a wide range of hardware and operating systems, not all existing PDP-10's. We cannot expect many users interested in the programs to be able to set up a full-scale PDP-10 site capable of running them.

We have been considering a number of mechanisms for exporting such software. These include a) implementing individual programs on machines which could be accessed by interested users over some (commercial) network, b) implementing or (reimplementing existing) individual programs in an appropriate language which is "machine independent" and thereby could be run on a user's existing computer given some minimum size, or c) making the programs available on an exportable machine (PDP-10 or its more cost-effective descendants) which is compatible with the existing programs and the centralized PDP-10 facilities used for continuing development.

3.3.2.1 NETWORK ACCESS

There is a growing number of uses of computer networks for program dissemination ranging from business accounting and modeling packages available from commercial vendors to attempts to consolidate research tools such as a collection of mass spectral library search and analysis programs (see for example S. R. Heller, G. W. A. Milne, and R. J. Feldmann, "A Computer-based Chemical Information System", Science, Vol. 195, Number 4275, page 253, 1/21/77). The existing network connections at SUMEX are well-configured for experiments within our capacity on this means of disseminating software. For many such programs, this seems to be well-suited for export; and indeed Heller reports 162 current user groups subscribing to his Chemical Information System. However, unless the network machine runs the same operating system and language in which the program was developed, a conversion would be required and perhaps at the same time a barrier would be established between the continued development of a program and its operational use. This appears to be the case for at least one proposal for a network-available version of our CONGEN program. The DENDRAL project has undertaken a very laborious conversion of CONGEN from its native LISP implementation to one in MAINSAIL to achieve a level of exportability for lack of other immediately available mechanisms. Other aspects of this approach involve security and privacy. Some of the data used with these programs are sensitive (patient records, or private, unpublished information on chemical structures, etc.). Having such a public access as over a network can create problems in protecting these data; and individual user groups may prefer to run the programs on machines which are under their local control. Finally, since many of these tools are in the research domain, it is not clear that they would be cost-effective in a commercial environment.

3.3.2.2 MACHINE-INDEPENDENT LANGUAGE IMPLEMENTATION

An ideal which has been long sought for program sharing is to develop languages with "universally" accepted standards and which are implemented in machine independent ways so that programs running on one machine environment will run in another with a minimum of conversion effort. This of course involves both language implementation and application program implementation concessions to achieve effective machine independence. We are working on a machine independent version of the SAIL language called MAINSAIL now to experiment with these sorts of issues. Our detailed plans for MAINSAIL development are given below including the possibility of special microprogrammed machines which may most economically and efficiently run MAINSAIL. Practically speaking, the machine-independent language approach is best-suited to the design of new program systems; and in the particular case of MAINSAIL, to those that can be effectively expressed by means of an ALGOL-like language. For existing programs, an extensive conversion would be required. We are still exploring the full range of implications of language choice for AI programs such as are being developed on SUMEX but it is likely that MAINSAIL cannot be a universal substitute for the full range of languages (including LISP) useful for these programs in both operational use and on-going development. MAINSAIL is nevertheless a definitive step toward understanding the requirements, advantages, and costs of machine independent systems. It may offer a useful base for implementing all or parts of new systems as well as for the ultimate reengineering of existing systems as they become fully operational.

3.3.2.3 EXPORTABLE (PDP-10) SYSTEM

An alternative view is that with the dramatic downward plunge of hardware costs, the costs of software development should play a larger and larger role in determining software/hardware optimizations. An attractive solution involves a PDP-10-like machine which could run the existing software intact and which could be made available for a reasonable cost to interested user (or network) groups. Since the machine could run the native operating system and language in which the program was developed, the initial conversion would be minimized and future developments (either conceptual or for improved efficiency) would be readily incorporated. Furthermore, a given user group could (perhaps with a change of microcode or system) run programs from various PDP-10 environments. By using network communication facilities, such satellite machines could retain contact with central development efforts, share files or data bases where appropriate, and provide a means for cost-effective incremental expansion by adding more such satellite machines or upgrading to a larger PDP-10 configuration when usage justifies. In this sense, this option is really a variant on the first network option using a more flexible hardware capability which can adapt better to individual program and development group/user community needs.

This approach may be best suited for this intermediate stage in AI program development where continued research and improvement is going on while extensive operational access is demanded. An economical export by this means defers the need for reprogramming until the design is fully stabilized and ready to be "cast in concrete". Nevertheless, even if the host machine is very inexpensive, in the long term if a factor of 10 improvement in speed or the number of users supported is possible by reprogramming, then a reimplementation will likely be warranted eventually as development tapers off and more and more users demand efficient production runs.

3.3.3 EXPORTABLE MACHINE PLANS

Because of the already large effort that has gone into other existing software systems we are attempting to export, the "exportable machine" option may offer a substantial advantage in minimizing conversion efforts, maintaining contact with program development groups, and offering a cost-effective way for even relatively small groups to use these programs. This is particularly important in just moving from the strictly developmental phase into a combined development/refinement/operational stage.

For our purposes, such a machine could be either a hardware-designed PDP-10 or a microprogrammed emulation of this machine. As a tentative functional configuration we would like the machine to perform at about the speed of a KI-10 with several users including:

- PDP-10 instruction set and "BB&N" paging facilities
- at least 256K logical address space
- 256K physical memory size (36 bit words, < 1 microsecond cycle)
- memory interface for swapping device and small file system including at least 200M bytes of disk storage
- facilties for about 16 terminals
- 200-300 lpm printer
- slow tapes
- some kind of external bus interface (I/O bus, UNIBUS, etc.)
- facilities for network communication connections

The cost for such a system (CPU, memory, and minimal peripherals) should ideally be in the range of $50,000 - $100,000. This may be below the initial announcement price for such machines but should represent realistic longer term pricing possibilities. A number of vendors may be working on the planning stages of such a machines which could be announced within the next 18 months. We budget for an initial version of such a machine at $200,000 based on very general pricing estimates (noting also that no vendor announcement has been made). The detailed alternatives and plans for this acquisition will be reviewed with the AIM management committees before implementation.

The detailed requirements for integrating such a machine into the SUMEX-AIM resource are also necessarily vague since this will depend on needed operating system and user support changes to accommodate the reduced size and perhaps different memory management system (paging). These changes may also reflect themselves in modifications for the language support underlaying the programs we want to export. We expect to track these developments closely during the first year of the follow-on grant and to formulate a plan for acquiring such a machine for experiments in packaging our AI programs for export. We will only be able to assess the required level of system software work when the details of the vendor systems become known. The budgetary details are discussed in the "justification" section of the five year budget plan.

These kinds of machines may also offer an effective way to incrementally expand the capacity of facilities like SUMEX and we will review them in this context as well (see the discussion of facility hardware upgrade plans on page 62). The main issues arising in coupling such satellite systems to the central facility as independent machines involve managing a distributed file system,

convenient terminal routing, and allocating users between machines. These are all manageable problems within existing technology such as we employed in developing the initial dual processor implementation. Since we are operating on fully amortized hardware, the indicated time table is driven by the real costs of system software modernization and compatibility of maintenance. Local users will be less injured by persevering with dated systems than a wider community to which software must be efficaciously exported in a contemporaneous idiom.

3.3.4 MAINSAIL DEVELOPMENT PLANS

The on-going MAINSAIL development effort was described earlier as part of our detailed progress report. A summary of language features can be found in Appendix III on page 231 (see Book II). This section summarizes the planned directions for future MAINSAIL developments. These efforts have two complementary thrusts: 1) development as a programming system and research tool and 2) demonstration of implementations for additional target systems. The first area is independent of what machines are used as hosts and seeks to explore the design ramifications, programming techniques, and advantages and costs of machine independence. The second area addresses the acquisition of practical experience in the export and use of MAINSAIL on real systems and the issues involved in gaining user acceptance of MAINSAIL as a programming tool.

3.3.4.1 DEVELOPMENT MANAGEMENT

In the early phases, the design for MAINSAIL was developed by Mr. Wilcox with a range of community inputs collected in relatively informal exchanges. These have included discussions with the designers of the SAIL language, studies of other languages (PASCAL, ALGOL-60/68, and SIMULA in particular), comments on our preliminary design documents from interested groups, presentations and discussions at several DECUS symposia, and community experimentation and critique of evolving MAINSAIL implementations. Our network connections have been invaluable in this regard, providing access to our documents, allowing rapid responses to suggestions, and providing a means for network collaborators to experiment with MAINSAIL on their own machines as implementations have become available. As MAINSAIL achieves a more operational status and we receive feedback from a larger community, we will reexamine many of these initial design decisions based on criteria of generality and effective portability as well as community acceptability. In this process we will formalize our user community contacts to take better advantage of their suggestions for system evolution and for effective system maintenance. We will, of course, provide a mechanism for reporting community comments (most easily done via networks) and may organize workshops or participate in other meetings to disseminate and discuss MAINSAIL. The AIM Executive Committee will play a key role in advising about development plans and making priority trade-offs within our limited available resources.

3.3.4.2 LANGUAGE DEVELOPMENT

Interrupts: We are currently investigating the implementation of both deferred and immediate interrupt facilities for MAINSAIL to give the ability to stop a program in the midst of execution, communicate with an interrupt-driven i/o device, or synchronize cooperating processes. A key issue is how to coordinate interrupt control transfers with on-going dynamic memory and storage management. This is particularly critical for immediate interrupts as may be

needed for real time applications. It may be necessary to restrict the range of language facilities available during such interrupts. We will continue these studies and implement appropriate interrupt handling support.

Concurrency: The current implementation of MAINSAIL has been designed with concurrency in mind, and appears to provide a solid base. We must complete the definition of the role of concurrency in MAINSAIL, then specify a set of primitives needed to support concurrency. There will then be an efficient implementation of these primitives including a convenient and flexible user interface.

Minimize runtime checking: Much of the code produced for runtime checking could be eliminated if the compiler "understood" more about the program. We propose to give MAINSAIL the ability to verify that certain conditions are met within the program so that more checking can be done at compiletime, and less at runtime. This involves exploration of what features MAINSAIL should include to allow the programmer to help in this process.

LEAP: LEAP is a facility in SAIL which provides an associative data store to allow the retrieval of data based on the partial specifications. We have encountered a number of prospective MAINSAIL users who have used and feel a need for LEAP. We plan to investigate the most useful features in LEAP which should be incorporated into MAINSAIL. It should be pointed out that many of the facilities of LEAP can easily and efficiently be coded in MAINSAIL using RECORD's.

3.3.4.3 COMPILER DEVELOPMENT

Increase speed of compilation: There is much room for improvement in the speed of compilation. The current version was designed for flexibility rather than efficiency. Most important is a close look at the symbol-table lookup, for that is where (the first pass of) the compiler spends most of its time.

Improve error detection and recovery: The compiler's error detection and recovery is now rather primitive. In general the entire edit-compile-debug loop should be streamlined for user convenience. We propose the utilization of a text editor as an integral part of compilation, so that MAINSAIL can automatically switch between compiling and user editing.

Machine-Independent code optimization: The first pass of the compiler produces an intermediate language which is the same for all target machines. This intermediate language is simply a recoding of the source file into an assembly-like language which reflects the properties of MAINSAIL. Various machine-independent transformations could be carried out on this intermediate text to translate it into an equivalent but more efficient representation of the source program.

Machine-dependent code optimization: The MAINSAIL code generators, themselves being MAINSAIL procedures, can be more readily written to utilize

complicated algorithms and data structures if necessary to generate efficient code. At present, the primary hurdle to a thorough analysis of the intermediate code by the code generators is the lack of a "look ahead" facility. We propose adding to the second pass the ability to build a machine-independent structure, on the procedure level, which can be interrogated by the code generators prior to generating code for a procedure. This would allow the code generators to make decisions based on a global knowledge of a procedure.

3.3.4.4 RUNTIME DEVELOPMENT

The runtime system is composed of modules which support the code generated for a user module. A single small module, called the kernel, is permanently resident, while all other modules are swapped as necessary. The modularity of the runtime system is what allows MAINSAIL to run in a small address space.

Optimize system modules: To a large extent, the efficiency of the system modules determines the efficiency of user programs. Thus it is well worth our time to optimize these modules. We propose to develop some modules which measure system performance. These would also be made available to users to help them evaluate their programs. A profile of a program, reporting how many times each statement is executed, is also proposed.

The primary use of these performance measurements will be for the tuning of memory allocation, swapping and garbage collection. MAINSAIL is largely independent of the exact strategies utilized, thus providing much leeway in working with alternate approaches. These algorithms need to be separately tuned for each implementation.

Virtual data space: MAINSAIL now supports the swapping of control sections, which could be considered a form of virtual control space. We are interested in studying whether this same form of support can be extended to data. Now that MAINSAIL can support a virtually unlimited control space (by breaking the program into modules), an implementation will be limited primarily by the amount of data which must be resident. We propose to add facilities to the language which allow the user to help structure the data so that it can be efficiently moved between memory hierarchies.

Support data operations: Machines which do not directly support the data types which MAINSAIL offers will need additional support modules. In particular, we need to write machine-independent modules to perform arithmetic on long integers, reals and long reals.

Runtime certifier: We will need a runtime certifier, i.e., a set of modules which give new MAINSAIL implementations a thorough workout, comparing the results with those obtained from running MAINSAIL on other machines. We have been using the compiler for this purpose, but it does not exercise all facilities of MAINSAIL, e.g., real and long real.

3.3.4.5 DEBUGGING SYSTEM DEVELOPMENT

We feel an effective and integrated debugging system will play a key role in the utility of MAINSAIL. Our goal is to provide interactive debugging capabilities comparable to those of INTERLISP which can significantly increase programming productivity. The combination of comprehensive debugging facilities with efficient production execution will help bridge the gap between program development and operational use.

The basic approach involves the integration of the now distinct phases of source text editing, compilation and execution. An internal representation of the program will be maintained which can serve a variety of purposes. This representation will be interpreted during debugging so that MAINSAIL can monitor execution and interact with the user in a manner which reflects the program structure. Errors can be corrected by editing this structure, and execution continued with no need for recompilation. Program text can be generated from the structure in a standard format, including the original variable names.

Machine code can be generated from this same structure, and compiled and interpreted code intermixed during execution. This provides fast execution of debugged modules along with interpreted execution of modules under scrutiny. Interpreted execution will allow for the interrogation of variables, setting and removal of break points, procedure trace, and single stepping. We plan to integrate these capabilities with a display terminal under the control of an editor, though the debugger will also operate from a hard-copy terminal. A split-screen facility will allow the program text to be viewed during execution along with any output from the program.

There are a number of difficult problems to be resolved concerning the relationship between the original source text (if any) and its internal representation which may be edited during debugging. Unlike LISP, the MAINSAIL syntax requires a significant amount of compilation before it can be put into a form which can be interpreted with reasonable efficiency.

3.3.4.6 DOCUMENTATION PLANS

Language manual: The currently available documentation for MAINSAIL consists of a preliminary language reference manual. It will be rewritten and expanded to be useful to users unfamiliar with SAIL.

Runtime manual: We will also provide a runtime manual which explains what happens during program execution. This information can be enlightening when designing a program, though its primary purpose is to document the machine-independent runtime system. This manual will also be necessary for the implementation of MAINSAIL on a new machine.

Code generation manual: A third manual, the code generation manual, will describe how to write code generators. This involves a description of the intermediate code, and how it is presented to the code generators. The goal is to

describe the code generation process in sufficient detail to allow any user to write a complete set of code generators. In this way the burden of implementing MAINSAIL on new machines can be dispersed.

System implementation manual: The system implementation manual will describe how to write the machine-dependent parts of the runtime system. This manual will describe what procedures need be written, and the data structures and other procedures with which they interact. It will also describe all the parts of MAINSAIL, how they fit together, and how to build a new system.

3.3.4.7 MAINTENANCE AND DISTRIBUTION PLANS

The maintenance and distribution of MAINSAIL could easily overwhelm us if we do not carefully plan for it. This is a good opportunity to bring someone else into the project, since it presents the chance to become familiar with the inner workings of the system.

Local experts: Each site must have a local expert who can repair errors in the machine-dependent portions and make patches to the machine-independent parts prior to receiving a new version which incorporates the changes. Another role for the expert would be that of liaison between the local user community and SUMEX. Questions and bug reports should first be directed to the local contact, and then directed to SUMEX in a form standardized across all sites.

SUMEX liaison: As MAINSAIL begins to be used at a number of sites, we would expect the number of inquires from potential users to rise to the point where it could require an inordinate amount of time from the developers. We propose that an additional person be hired at SUMEX as a liaison for MAINSAIL. This individual must be capable of fixing bugs and generally keeping current versions of the system healthy. The liaison will keep in touch with the local experts, and pass to them any necessary updates. This involves making tapes and sending them through the mail; editing the documentation, overseeing its printing and distribution; responding to inquiries from potential users; consulting with new users concerning program design (but not actually writing user's programs); and new user orientation.

3.3.4.8 PLANS FOR ADDITIONAL IMPLEMENTATIONS

The current implementations are for the PDP-10 and PDP-11. These give us experience on medium and small scale machines. We plan to hold off on introducing additional implementations until we have received sufficient feedback from these. It appears that the orchestration of parallel implementations on a wide variety of machines will rival the technical problems.

We have surveyed a large number of computer systems while designing MAINSAIL. Most of these are known to us only through manuals, so that further study will be necessary to determine how well a particular system could support MAINSAIL. Among the machines surveyed are: IBM (360/370, Series/1), CDC (6000 Series, 7600), UNIVAC (1100 Series), Texas Instruments (990), Honeywell (Level 6), Varian (V70), Hewlett-Packard (3000 and 2100), Data General (NOVA, ECLIPSE), Interdata (16 and 32 bit series), SEL (32), Harris (Slash series), Burroughs (B1700) and MODCOMP. We plan to keep abreast of new computer announcements, since we are in the position of relatively easily providing software for emerging hardware.

Choices for target systems will be based on user demand and priorities established in consultation with the AIM management committees. We are projecting approximately two man-months to create a new implementation, though this will vary according to how well the target machine and operating system fit MAINSAIL, and the availability of a target system during the early design iterations. Additional time will be required to actually install the implementation at the target site, have it thoroughly tested, distribute documentation and make it generally available. There are, of course, problems in developing MAINSAIL for a machine to which we have no access. The code generators and operating-system interface can be written independently of the target machine, but the debugging of these will require access for a period of at least a few weeks. It would not be acceptable to implement a machine by sending tapes through the mail. There appear to be four possibilities: access over a network; access to a nearby machine for which MAINSAIL has been implemented; rent or borrow a machine for the duration of the development; emulation of the target machine.

3.3.4.9 MAINSAIL OPERATING SYSTEM PLANS

In the course of designing the operating system interfaces it has become apparent that MAINSAIL needs very little support from any machine-dependent operating system, at least with regard to the execution of a single program. We feel that in many cases we could provide our own stand-alone version of MAINSAIL for single-job environments. Technology seems to be pointing in the direction of less expensive computers which can be dedicated to a single user at a time, and these would be the initial target of our operating system.

In the context of a single-job system, MAINSAIL's primary need is a file system and device drivers. Once our primitive operating system is written in MAINSAIL, it should not be difficult to add monitor commands and utilities such as file manipulation. Of course the MAINSAIL operating system would be special purpose in that it would support a single language, with everything designed around that language. The main elements of our operating system would be the compiler, a text editor, the MAINSAIL runtime system, and the additional modules to support the file system and i/o.

MAINSAIL does not need a linker, overlay system, or loader (the swapping of modules takes care of those). Additional components of the system could simply

be added as new modules. A goal would be to design an open-ended operating system kernel which could be extended by the user as desired.

3.3.4.10 MICROCODED MAINSAIL MACHINE PLANS

We have thus far been discussing the achievement of portability by making MAINSAIL fit existing machines. If the reason for portability is understood as the desire to provide an economically viable way of distributing software, then another approach is to make the hardware fit MAINSAIL, and distribute the hardware along with the software.

We propose to design an "optimal" representation of MAINSAIL code for emulation by a microprogrammable computer; to purchase a suitable computer for MAINSAIL emulation; to implement MAINSAIL and the supporting microcode on this computer; and to evaluate the resulting system to determine the economic and technical feasibility of distributing such an integrated hardware-software programming environment. Details of our plans are given in Appendix IV on page 235 (see Book II).

We expect considerable improvement over implementations for existing machines which have been accommodated to less than optimal, and in some cases quite poor, instruction sets. Many benefits accrue from such an approach, and it is likely that microcoded hardware, specialized to a particular language or application, will play an increasingly important role in the development and operational use of future software systems. We expect a microcoded MAINSAIL to outperform other MAINSAIL implementations in much the same way that DELtran (a "directly executable language" (DEL) implementation for FORTRAN II) outperforms FORTRAN II(4). Initial measurements show that the DELtran representation is less than one fifth the size of the code generated by the FORTRAN-H optimizing compiler, and executes about five times faster.

MAINSAIL is perhaps better suited to the emulation approach than FORTRAN because of the locality of reference provided by procedures, records and modules. A preliminary DEL has already been designed for MAINSAIL, but further work is necessary before we can predict (or demonstrate) size and execution comparisons with standard implementations.

This work will complement the on-going implementations of MAINSAIL on conventional hardware. Thus we will be in a unique position to compare the two approaches.

The combination of a microprogrammed machine with the MAINSAIL operating system could result in a system optimized for the execution of MAINSAIL programs. As hardware costs continue to fall we see this approach as a realistic way of providing a powerful system at a low price. We are interested in determining

---

(4) See Hoevel, L. W. and Flynn, M. J., "The Structure of Directly Executed Languages: A New Theory of Interpretive System Support," Stanford Digital Systems Laboratory, Technical Note No. 108, Stanford University, March 1977.

whether a "soft" machine of this sort can be provided cheaply enough to serve as a basis for the export of software which presently requires extensive hardware facilities.

3.3.4.11 DEVELOPMENT OF PORTABLE SOFTWARE

We would like to see a collection of portable programs developed in MAINSAIL both to serve as examples of portable software, and to provide support to those sites which begin to rely on MAINSAIL as the primary programming resource. Such software development will also help us debug MAINSAIL, familiarize the programmers with it, and spread its use. We are aiming for the complete support of a stand-alone MAINSAIL implementation which is aligned with developing hardware trends, i.e. video displays and compact, relatively inexpensive computers and peripherals.

We do not now have the facilities to implement all of this software at SUMEX, and thus expect to collaborate with others in its design and implementation. It is imperative that the software be portable except possibly for certain well-defined modules which need support outside MAINSAIL (e.g., special device support).

Display editor: A MAINSAIL text editor is at the core of a number of planned developments. Our interest is centered around a display-oriented editor because of its clear superiority over hard-copy editors. The TV-EDIT program now in use at Stanford and a few other sites is an excellent base of development, especially since it is written in SAIL. We would like to see additional features added to what TV-EDIT now possesses. Our intended applications for compilation and debugging require a split-screen facility, and a multi-file capability. It must direct all communication with the display through a display package, as described below. This separates the editing functions from the display functions, so that the editor is independent of the display and hence can be used with a variety of displays.

Display package: A display package is necessary as part of the editor, and is also important as a package for use by other programs. The display package will accept standard commands to control a display terminal. It must be smart enough to simultaneously maintain several areas on the screen. Such a package will be machine-independent (as much as possible), but have terminal-dependent modules which feed the terminal hardware commands to effect the machine-independent commands. It should be able to drive a hard-copy terminal as if it were a limited display terminal.

Graphics package: Similar to the display package is a graphics package for drawing pictures on a graphics display device. This package would allow for the description of pictures, the choice of display device, and the display of the pictures. This package would be machine-independent and display-independent. The OMNIGRAPH system developed by Sproull at NIH may form the basis for this package.

Document preparation: A simple document preparation program would serve as

the "back end" to the display editor. We feel that much of the work of current document programs could be provided by the editor in a form providing instant feedback. Thus the primary purpose of the document program would be to provide global processing, e.g., to generate a table of contents or index, and fill in symbolic references with appropriate chapter or section numbers.

Math and statistics packages: MAINSAIL currently has a mathematics package with trigonometric and logarithmic functions. These functions need additional testing for accuracy, and should be augmented with other functions, e.g., a random-number generator. There is also a need for a statistics package.

4 AVAILABLE FACILITIES

The existing SUMEX-AIM computer and communications configurations have been described in earlier sections. The number of personnel to support this follow-on work will remain at approximately the same level as before so no additional office space will be required. We anticipate no changes will be needed for the machine-room facilities.