TECHNICAL REPORT 81-01

Quarterly Technical Report:

Defense Microcomputing in the 1980s: Problems & Research Priorities

LEVEL II

James F. Wittmeyer, III

CSM INC
Computer Systems Management, Inc.
1300 WILSON BOULEVARD, SUITE 102 • ARLINGTON, VIRGINIA 22209

(12)

TECHNICAL REPORT 81-01

Quarterly Technical Report:

Defense Microcomputing in the 1980s:

Problems & Research Priorities

by

James F. Wittmeyer, III

Computer Systems Management, Inc.
1300 WILSON BOULEVARD, SUITE 102
ARLINGTON, VIRGINIA 22209

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: CSM-81-01	2. GOVT ACCESSION NO.: AD-A094 076
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): Defense Microcomputing in the 1980s: Problems & Research Priorities.	5. TYPE OF REPORT & PERIOD COVERED: Quarterly Technical Rept. 10/1/80 - 12/31/80 / 1 Oct - 31 Dec 80
	6. PERFORMING ORG. REPORT NUMBER: --
7. AUTHOR(s): James F. Wittmeyer, III	8. CONTRACT OR GRANT NUMBER(s): MDA903-80-C-0155, ARPA Order-3829
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Systems Management, Inc. 1300 Wilson Boulevard Arlington, Virginia 22209	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: ARPA Order Number 3829
11. CONTROLLING OFFICE NAME AND ADDRESS: DARPA 1400 Wilson Boulevard Arlington, Virginia 22209	12. REPORT DATE: 12/31/80 (11) 31 Dec 80
	13. NUMBER OF PAGES: 50 (12) 57
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Defense Supply Service-Washington (DSS-W) The Pentagon Washington, D.C.	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Recommended for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Recommended for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microcomputers; interactive system design; technology transfer; software; software psychology

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

Microcomputing in DoD in the 1980s should be informed by creative hardware integration, software psychology, humanistic interactive systems design, and interactive interfacing considerations.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

411586

TECHNICAL REPORT 81-01

QUARTERLY TECHNICAL REPORT:

DEFENSE MICROCOMPUTING IN THE 1980s: PROBLEMS & RESEARCH PRIORITIES

ARPA Order No.:	3829
Contractor:	Computer Systems Management, Inc. 1300 Wilson Boulevard, Suite 102 Arlington, Virginia 22209
Effective Date of Contract:	11/5/79
Contract Expiration Date:	9/30/81
Contract No.:	MDA903-80-C-0155
Principal Investigator:	Mr. James F. Wittmeyer, III (703) 525-8585
Contract Period Covered:	10/1/80 - 12/31/80
Short Title of Work:	Microcomputing in the 1980s

This research was sponsored by the Defense Advanced Research Projects Agency under ARPA Order Number 3829; Contract Number MDA903-80-C-0155; and Monitored by DSS-W. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either express or implied of the Defense Advanced Research Projects Agency or the United States Government.

SUMMARY

This quarterly technical report covers the period from October 1, 1980 to December 31, 1980. The tasks/objectives and/or purposes of the overall project are connected with the design, development, demonstration and transfer of advanced computer-based systems; this report covers work in the microcomputer systems design area. The technical problems addressed include the evolution of microcomputing, microcomputing logistics, and user psychology. The general methods employed include interactive system design principles identification and examination, substantive area identification, and design, development, demonstration, documentation, and transfer research priority specification. Technical results include the specification of five primary research areas: creative hardware integration, humanistic interactive system designing, interactive interfacing, software production psychology, and unconventional documentation and transfer. Future research will present some specific microcomputer systems designed to solve myriad analytical problems.

TABLE OF CONTENTS

FIGURES

1.0 INTRODUCTION

In the 1970s computer-based solutions to decision, forecasting, training, and information management problems by and large took but a few forms. Initially they were geared to information overload and "accounting" problems; decision-making, forecasting, and training problems, as well as those which were primarily analytical rather than procedural, were relatively neglected. The emphasis was on information collection, processing, storage and retrieval. In fact, the very first production systems occurred in two totally unrelated areas: budgeting and large scale data processing connected with weapons development. Only later did the role of the individual forecaster, decision-maker, planner, evaluator, trainer, and information manager--in analytical contexts--become the focus of serious research and development (R&D).

It is also important to note that the legacy of 1960s computing was comprised of large, "macro" systems supported by relatively inflexible software. Consequently, there was, in the early 1970s, a fundamental incompatibility between the available hardware/software and the kinds of requirements which underlie individual cognitive operations.

In the mid-1970s, however, "decision support systems," "user-oriented" data base management systems, and a potpourri

of analytical algorithms were developed, tested, and applied to many cognitive operations and tasks. Minicomputers and (clumsy) microcomputers also proliferated the computing environment during the mid-1970s.

In the late 1970s enormous changes occurred in the applications sphere. More and more DoD agencies and offices, through impressive R&D efforts, began to augment their problem-solving prodedures via computer-based and computer-assisted means.

As we enter the 1980s, however, more enormous changes are unfolding. The revolution in hardware has made it possible to compute most anywhere, anytime--and inexpensively; and the software revolution is about to compliment users in ways only imagined in the 1960s.

During the mid- to late-1970s the DARPA/CTO-conceived Demonstration and Development Facility (DDF) concept was a reflection of the enormous changes which occurred during the period. Succinctly, the DDF was minicomputer oriented and served its community well as DARPA/CTO-supported contractors developed, tested, demonstrated, and transferred a whole host of minicomputer-based decision, forecasting, training, and information management systems. But times are changing. Clearly, the 1980s will see a major shift away from minicomputer-

based systems to those which are portable, personal, and (therefore) distributed.

This report thus examines this evolution and identifies the problems and research priorities which will emerge in the 1980s. It will also attempt to carve a new role for the DDF.

2.0 FIRST GENERATION PROBLEMS AND PRIORITIES IN THE 1970s

2.1 Problems

2.1.1 Evolution - In the early 1970s computing was plagued by a whole host of problems directly attributable to the still young evolution of computer-based problem-solving. Hardware was by and large unperfected at the minicomputer level and virtually unknown on the microcomputer level. Reliability and maintenance statistics indicate conclusively that reliability was at best fair among minicomputers and at worst non-existent. Since vendors were scrambling to such new (and relatively untested) machines into what was preceived to be a burgeoning new marketplace, and engineers were busy designing second-phase, competitively priced systems, little attention was paid to the end user as a user; instead, emphasis remained strong upon coaxing macrosystem users into the minisystem realm without reference to applicability or reliability. At the same time, macrosystem houses stepped up their efforts to lure users into service bureaus still unpopulated by minicomputers. In terms of the evolution of analytical, and/or problem-solving computing, these early events fragmented the computing population into adversary groups, loosely organized not according to user applications areas, but in terms of how macro- and minisystem designers and vendors envisioned user opportunities (defined almost exclusively, in turn, in terms of sales).

Of course the real losers during this early evolutionary period were the ultimate users who had been promised a new day in computing and were delivered a new set of problems and unrealistic promises. The internal (macro/mini) designer/ vendor power struggle only exacerbated the unhappiness, and ultimately alienated many prospective new users from computing altogether (as substantiated by a survey of corporate EDP managers).

On the software side, things were even more disorganized. For the first time, in the early 1970s many vendors were distributing software to new minicomputer users, software which previously had been developed to run on macrosystems and which was meticulously maintained by on-site and on-call service bureau (timesharing) programmers. The initial impact was predictably unimpressive. Not only were vendors unprepared to deal with software problems which developed after distribution but many were unwilling to do so as well; and since in the early 1970s most hardware vendors were also the software vendors, users had to either acquiesce to the confusion or hire their own cadre of programmers.

On the user interaction side, things were not simply confusing and disorganized but unrelated to on-line problems of orientation, throughput, and--most importantly--relevant

output. Users in many instances were left on their own to modify and (frequently) restructure whole programs. Since software origins could be traced to timeshared macrosystems, low user expectations, and relatively inflexible output, what today we regard as "user-orientation" and "personal computing" were completely unknown to early users. Consequently, many new users were again forced to tailor software to their own particular needs, usually at great expense and with considerable frustration.

2.1.2 Logistics - In addition to all of the above, the early 1970s minicomputer user inherited innumerable problems (from the macrosystem era) with which they were unable to deal. Included here were data management, communications, peripherals, and environmental support problems. Not unlike the amateur photographer who discovers that to take real pictures he must purchase several lenses, a timer, a light reader, and master the worlds of aperture and depth, the new minicomputer user discovered that he needed several tons of air conditioning, a false floor, and a lot of expensive power to compute. Other logistical concerns included maintenance and insurance issues, among many, many others. Competent staff also had to be located, trained, and kept well-fed since the competition for skillful programmers was (and remains) keen.

2.1.3 Psychology - Finally, underlying and exacerbating the evolutionary and logistical problems were a set of attitudes which regarded computing as a nuisance. (Indeed, user ratings of hardware and software systems in the early 1970s were almost universally negative.) While today we frequently hear computing horror stories (e.g., WWMCCS) yesterday they were generated on an almost daily basis.

Part of this attitude problem can be traced to vendor over-sell and the resultant artificially high expectations created in the user community. Part of it can be traced to the generally poor quality hardware and software systems, and part to the natural confusion which results when science outpaces its own arrival. But a good deal of the problem can be traced to the inevitable result of distributing technology which from the outset was never intended to be distributed. Indeed, while the components of macro- and minisystems differ more in scale and capability than in concept, from the prospective of the user the processes of macro- versus minisystem computing are as alien to one another as an instamatic camera is to a 35MM.

2.2 Priorities

The legacy of early 1970s computing was a set of critical priorities which, one by one, began to receive attention.

Since hardware technology continued to advance throughout the 1970s, more and more capabilities began to appear. Display technology, input devices, printers, memory size and type, and communications devices all evolved impressively in the 1970s. Minicomputer software as well improved and proliferated in number and quality; and while standardization problems continued to plague users, output began to approximate user requirements. Since many users had by the mid-1970s developed their own programming staffs (and were psychologically prepared to actually use and even enlarge such staffs), a great many problems began to give way to legitimately higher performance expectations. Also, in the mid-1970s hardware vendors began to improve their software offerings and--most importantly--a whole new kind of software vendor emerged, vendors which produced no hardware but excelled in applications and systems software capable of running on many different hardware systems.

This development not only filled a critical gap but changed the focus for the first time away from basic hardware and software capabilities and concerns to how responsive the software was to the on-line user. As an outgrowth of this development hundreds of user-oriented peripheral manufacturing companies sprung up; and by the mid- to late-1970s color displays, speech input and output devices, and spatial data management systems were proliferating throughout the user community.

2.3 Defense Computing

This whole story simultaneously played itself out in the Defense Department where extremely large macrosystems appeared early. Unfortunately, however, beyond the research laboratories, these macrosystems have been only reluctantly replaced by now relatively efficient minisystems. The reluctance may in part be explained by the peculiar difficulties surrounding government problem-solving (explained, in turn, by cumbersome procurement) and by the inertia which often encumbers all large bureaucracies. In any case, it is safe to say that all government (including the Defense Department) is lagging somewhat behind the general use of minicomputers in industry, a fact which translates into an enormous challenge for government research and development managers.

Fortunately, because of the obvious imperatives connected with national security, the Defense Department (and related agencies and departments) has, through the efforts of effective research and development managers, made impressive progress. Much of this progress, as documented briefly below, has been the results of programs sponsored by the Defense Advanced Research Projects Agency's Cybernetics Technology Office and Division (DARPA/CTO/CTD).

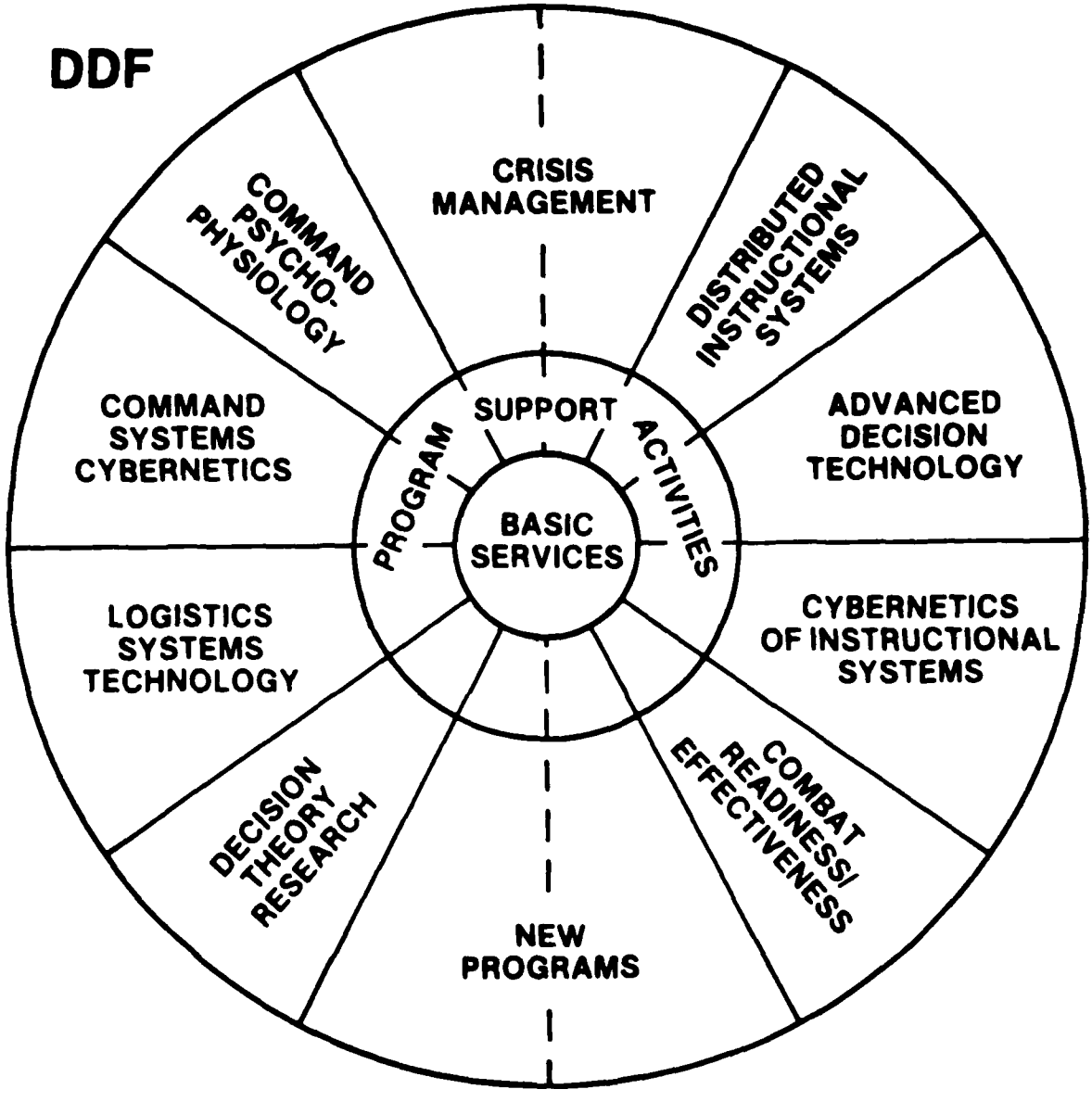
2.4 DARPA/CTO/CTD Computing

In many important respects, DARPA/CTO/CTD has constituted the "technology push" in many parts of the Department of Defense (DoD). At the same time, the same office/division has pioneered the identification of user requirements--and responded in some very unique ways. For example, DARPA/CTO/CTD invented several variant spatial data base management techniques designed to enhance user attitudes toward and effectiveness with mini-computers; over ten Bayesian decision-aiding systems were developed which augment the analytical capabilities of defense decision-makers. Implemented on turnkey mini- and microcomputer systems, these aids have been enormously popular with decision-makers and managers on all defense levels, and have in many instances bridged the gap between analytical decision-makers and analytical computing. Similarly, the Early Warning and Monitoring System (EWAMS) is a good example of a previously non-computerized process which has been (re)incarnated in a very user-oriented computer program. On the information management side, the Adaptive Information Selector (AIS) and the Ultra Rapid Reader (URR) are two excellent programs designed to ease the burdens of C^3 information managers and processors, and the recent work in Advanced Mapping and Tele-conferencing constitute yet two more computer-based systems which will ultimately enhance human performance in DoD.

Nearly all of these programs are minicomputer-based. (Yet, very recently some have been implemented efficiently as microcomputer programs.) Many have been transferred to operational and quasi-operational (or testbed) environments; and many have been successfully tested and evaluated in these and other contexts.

2.5 The DDF and DARPA/CTO/CTD Computing

The DARPA/CTO/CTD Demonstration and Development Facility (DDF) was established in 1977 under funding from DARPA/CTO.[1] It commenced full operation ahead of schedule and completed a highly successful initial program of product demonstration, transfer, integration and development support. During the first year, the DDF was used exclusively for DARPA/CTO-funded Crisis Management research. In fiscal 1979, DARPA expanded the scale of DDF operation to serve researchers working on all CTO programs (as suggested below). DDF also conducted intensive technology transfer efforts in 1979 in support of four selected CTO research programs: Combat Readiness/ Effectiveness, Advanced Decision Technology, Command Systems Cybernetics and Crisis Management. The intensive transfer support of the Crisis Management Program represented a continuation, on an expanded basis, of the DDF activities which met success in 1978.



The basic purpose of the DDF was (and remains) to increase the return on CTO/CTD expenditures by accelerating the creation of high quality, concrete, transferred research results. This has been achieved through:

- Sharing of hardware and software resources;
- Expert support and consultation on computer related matters;
- A central facility for the installation, integration, demonstration and storage of finished computer related research products; and
- The exercise of a capability for technology transfer to, and support of, experimental use of research results.

Some of the specific benefits to DARPA have included:

- Minimization of the development of redundant and/or incompatible hardware/software products;
- Demonstration of multiple research products on a single visit (residing on single menu-driven system), even when those products were developed in diverse independent environments;
- Integration of research products into consolidated packages for well coordinated and coherent demonstrations and transfer; and
- Convenient access for researchers to one another's products and data, by virtue of their initial development at or eventual integration into the DDF.

DDF origins can be traced to several known minicomputing research and development problem areas. In the field of cybernetics technology, for example, in which research results are often embodied in part in computer software or computerized data bases, the usual difficulties of validating research

results through experimental use are exacerbated because computer-based products are diverse, complex, expensive to maintain and difficult to use in their preliminary forms. Worse yet, demonstrable products developed at individual research centers are often never seen by those in DoD who could benefit from them, simply because they lack the time or travel funds to visit the many locations at which contractors do their work. [1]

While research results are widely publicized in journals, the managers and commanders who must be reached are often far more responsive to demonstrations and effective personal contact. This is especially important when the underlying concepts are advanced and difficult to appreciate except when seen in action. Upon initial consideration of this problem, one is tempted to think that the solution is simply to ask researchers to deliver more finished initial products. Then potential users can easily try them, and the additional development required to put them into routine use will be reduced. Unfortunately, this approach is prohibitively expensive. In commercial development of computer software, one well-known rule of thumb says that a finished program product costs roughly 9 times as much to develop as a working prototype.[2] For a fixed level of funding, then, this approach might be expected to reduce significantly the number of programs which ever get to the experimental stage. In any case, this approach does not overcome some of the other obstacles.

Some of the basic technical problems of research product demonstration and transfer which led to the establishment of the DDF, can be summarized as follows. Research products are often:

- Implemented in a variety of languages;
- Implemented on a variety of hardware configurations;
- Saved on a variety of media, in varying formats, at dispersed locations;
- Minimally documented;
- Designed without full awareness of or concern for software engineering practices;
- Not fully or systematically tested; and
- Developed with little or no regard for the operational environments to which they might later be transferred.

The diversity of forms among, and the common defects of, research products are natural results of each researcher employing a direct approach to his or her individual goal. People use available skills, software and machines to get quick initial results. (Indeed, it is precisely such economies that help make the initial proposals attractive to funding agencies.) The problems set in when the best of such results are to be transferred for experimental use. At that point, one faces the initial technical barriers. For example, a primary 1970s target military system was (and still is) a PDP-11 minicomputer on which frequently only COBOL and FORTRAN are maintained,

while the research was done in APL on another computer (with incompatible hardware and software). A few new features are needed to put up even an experimental system, and the original implementor is now working on the development of a new concept. Furthermore, he did not document the code (because he wanted to keep the cost of the initial proposal down) or even fully test the software.

In addition, there are several organizational problems:

- Support of experimental use involves different skills and motivations than development of an initial prototype;
- Maintenance and distribution of documentation, software and data involves a certain amount of overhead, not justified in many small research organizations and not economical for a single project; and
- The form in which a result can be convincingly demonstrated to a fellow researcher is usually not the best demonstration to a pragmatic, distant and skeptical expert in a different field, among many others.

All of these problems can be overcome of course for any given project, with sufficient DARPA attention, imagination, management and resources. The solution to these problems--the DDF--enhanced the quality of research results, increased the number of successful transfers, and improved the overall cost-effectiveness of research expenditures.

Many of the above problems in research product development, integration, demonstration and transfer were simplified through the use of a single concept, embodied in the DDF. That concept was to operate a centralized facility at which the following resources were available:

- Computer hardware and software to support development, testing, and integration of research products;
- Relevant research software and data produced in previous projects;
- Suitable devices and physical facilities for demonstrations (e.g. advanced graphic displays, audio equipment, specialized terminals); and
- Expert hardware and software support staff, familiar with the research areas and aware of existing products.

The most valuable of these resources, the support staff, stood ready to aid in the following critical functions:

- Installation and integration of research products into the DDF;
- Development of research products at the DDF;
- Development of effective demonstrations;
- Assistance in demonstrations for visitors;
- Transfer of research products out of the DDF into the field;
- Support of experimental use of research products; and

- Modification of existing products to enhance their transferability.

The use of a centralized facility and staff dedicated to these functions provided the following advantages:

- A single source for processing, software, data, and consultation in the development of new computer-based research products;
- A single location for the demonstration of multiple products;
- A single source, supporting uniform hardware and software, from which to stage transfers;
- An organization familiar with the mechanics of transfer and the products to be transferred for the support and planning of transfers;
- Elimination of one integration step in some future developments, since these will be planned from the outset for implementation at the DDF;
- Uniform procedures for the documentation, distribution, maintenance, and support of products in experimental use;
- A single source of information to DARPA on the progress, status and problems in various transfer experiments; and
- A well organized library of research products and documentation which itself will assume increasing value as use of the DDF proceeds.

The DDF itself did not assure the development of timely, significant and quality research results. However, it did provide

the means to transfer such results more reliably, more economically, more frequently, and more successfully than other approaches.

Yet, at the core of all DDF activity in the 1970s were several working premises which included the following:

- Minicomputer standardization in the DEC PDP 11/70 minicomputer;
- Software standardization in CULC FORTRAN IV, the UNIX operating system, and "C"; and
- User interfaces aided primarily in minicomputer programming and peripherals, such as large graphic color systems, and large screen color display devices.

As time passed, however, it became apparent that the premises were soon to be overtaken by technological advances, particularly in the microcomputing area and the simultaneous down-sizing (without significant capability degradation) of popular minicomputer systems. And while by and large the operational side of DoD was unaware of such developments, the research and development community was already gearing up for the next generation of defense computing.

Part and parcel to these developments were increased emphases placed upon the analytical user who must train, learn, forecast, decide, evaluate, and manage information in order to

carry out his or her duties. Similarly, new hardware, software, and related information management devices (such as the video-disc) evolved to the point where they could be mixed and matched into totally unique computer-based systems. All in all, then, in the mid- to late-1970s a whole series of events occurred which have forever changed the nature of defense computing; at the same time, a whole new set of problems and priorities will follow us into the 1980s. The next section will examine these problems and priorities and, then in Section 4.0, redefine the role of the DDF for the 1980s.

3.0 SECOND GENERATION PROBLEMS & PRIORITIES IN THE 1980s

3.1 Problems

3.1.1 Evolution - The revolution in microcomputing has had many positive and negative effects upon defense computing. On the negative side, there are problems of lack of standardization, maintenance, reliability, and even availability. Hardware is frequently poorly constructed, poorly documented, and poorly assembled. Moreover, many microcomputer manufacturers frequently develop their systems as components which, when integrated, exponentially increase the level of operational complexity and present the user with increased chances for operational failure. Also, since many new companies are manufacturing peripherals for many microcomputer mainframes, interface problems are often severe.

On the software side, there are no easily transportable languages. There are also countless new languages (and language variations) and a critical demand for microcoders who are well-versed in higher and lower level languages. Since the early microcomputer systems had relatively little memory, data management was all but non-existent. Now, however, through the availability of storage devices like the Winchester disk, data management needs have grown tremendously, but very few hardware or software houses are offering reliable data

base management software. Finally, while applications software abounds for the home computer hobbyist, few analytical programs are available for serious analytical problem-solving. Consequently, users must write their own problem-solving programs.

Yet, all of these problems are at the same time research and development opportunities. The existence of many peripheral options enables developers to mix and match in ways unavailable to macro- and minicomputer systems users. While software is non-standard, the cost of microcomputer programming is generally lower than macro- and minicomputer programming if only because microcomputer programming is much less machine and facilities intensive. Moreover, software flexibility enables users to build systems from ground zero and thereby avoid the inflexibility inherent in many macro- and minicomputer systems. Finally, the physical size and portability of most microcomputer systems enables developers and users to alter their programming workstyles in many positive respects (see the section on software psychology below).

3.1.2 Logistics - The situation today is clearly decentralized, and while there are many advantages associated with a decentralized manufacturing situation, there are advantages also.

One is maintenance. Often it is impossible to arrange for any support at all. Just as often when support is available it is incompetent. Finally, documentation is often unavailable and/or poor. At the root of these problems is the eagerness of the industry to introduce new technology as soon as possible into the field and the "independent dealer," who, not unlike the independent insurance salesman, functions as a clearing-house for any number of vendors whose products may be only vaguely understood.

Fortunately, however, support requirements for micro-computing are simpler and relatively much less expensive than for minicomputing. The number of real users can thus multiply in ways unimaginable just ten short years ago.

3.1.3 Psychology - In the 1970s the attitudes of minicomputer users were quantifiable (and frequently negative). But the attitudes of prospective microcomputer users are much less definable. Many are still suspicious about computing because of disappointments suffered in the 1970s. Others are almost completely uninformed about the capabilities of microcomputers; and still others will become even more confused as the microcomputing technology revolution continues to gain momentum.

Oversell also continues to be a problem, and in DoD many users who have only just begun to feel comfortable with the PDP 11/70, will undoubtedly feel frustrated and even angry about having to tool up once again. Such, then, are the challenges which confront DARPA/CTD as it enters the 1980s.

3.2 Priorities

If the primary computer-based vehicle of the 1980s is to be the microcomputer system then a clear set of research priorities may be identified. They include:

- Creative hardware integration;
- Humanistic interactive system designing;
- Interactive interfacing;
- Software production psychology; and
- Unconventional documentation and transfer.

3.2.1 Creative Hardware Integration - Above all else we must exploit the mix and match opportunities made possible by the exploding peripheral marketplace. We should do this by matching available machine components to real user requirements, capitalizing upon the flexibility which exists--and is likely to continue to exist--in the mainframe and perhiperals areas. But there are other hardware opportunities as well, including those made possible by video and information technology. Hence,

the watchwords of the 1980s will be integration and flexibility.

3.2.2 Humanistic Interactive System Designing - Hardware is only a means to an end; all computer designers and users know all too well that unless a system is comprised of efficient software there is no system at all. Since microcomputer-based systems design is a relatively new DoD research, development, and transfer goal, we should proceed systematically. Hansen, Wasserman, Pew and Rollins, Gaines and Facey, Cheriton, Gebhardt and Stellmacher, Kennedy, Engel and Granda, Palme, Turoff, Whitescarver, and Hiltz, and Sterling all offer excellent suggestions, as outlined below:[3]

HANSEN'S USER ENGINEERING PRINCIPLES
FOR INTERACTIVE SYSTEMS

- First principle: Know the user.
- Minimize memorization:
  - Selection not entry;
  - Names not numbers;
  - Predictable behavior; and
  - Access to system information.
- Optimize operations:
  - Rapid execution of common operations;
  - Display inertia;
  - Muscle memory; and
  - Reorganize command parameters.
- Engineer for errors:
  - Good error messages;
  - Engineer out the common errors;
  - Reversible actions;

- Redundancy; and
- Data structure integrity.

WASSERMAN'S DESIGN OF IDIOT-PROOF INTERACTIVE PROGRAMS

- Provide a program action for every possible type of user input;
- Minimize the need for the user to learn about the computer system;
- Provide a large number of explicit diagnostics, along with extensive on-line user assistance;
- Provide program short-cuts for knowledgeable users; and
- Allow the user to express the same message in more than one way.

PEW AND ROLLINS' DESIGN GUIDELINES FOR INTERACTIVE SYSTEMS

- Know the user population;
- Respond consistently and clearly;
- Carry forward a representation of the user's knowledge base;
- Adapt wordiness to user needs;
- Provide the users with every opportunity to correct their own errors; and
- Promote the personal worth of the individual user.

GAINES AND FACEY'S DESIGN GUIDELINES FOR INTERACTIVE SYSTEMS

- Introduce through experience;
- Immediate feedback;
- Use the user's model;
- Consistency and uniformity;
- Avoid acausality;
- Query-in-depth (tutorial aids);
- Sequential--parallel tradeoff (allow choice of entry patterns); and
- Observability and controllability.

CHERITON'S INTERFACE DESIGN FOR TIME-SHARING SYSTEMS

- Simple. Project a "natural," uncomplicated "virtual" image of the system;
- Responsive. Respond quickly and meaningfully to user commands;
- User-controlled. All actions are initiated and controlled by the user;
- Flexible. Flexibility in common structure and tolerance of errors;
- Stable. Able to detect user difficulties and assist him in returning to correct dialog; never "deadending" the user (i.e., offering no recourse);
- Protective. Protect the user from costly mistakes or accidents, (e.g., overwriting a file);
- Self-documenting. The commands and system responses are self-explanatory and documentation, explanations or tutorial material are part of the environment;

- Reliable. Not conducive to undetected errors in man-computer communication; and

- User-modifiable. Sophisticated users are able to personalize their environment.

GEBHARDT AND STELLMACHER'S DESIGN CRITERIA FOR DOCUMENTATION RETRIEVAL LANGUAGES

- Simplicity:
  - Few keywords--few commands; few keywords; no extra commands or keywords; special cases.
  - Simplicity of input--fast input (with respect to keyboard layout); mnemotechnically sound abbreviations; simple input structure.
  - Short commands--short keywords; little redundancy; avoidance of multiple input; use of default options.
  - Simple commands--simple command structure; simple syntax of commands; correspondence between syntax and semantica; simple dialog structure.
- Clarity:
  - Hierarchical structure--hierarchical structure of the language (commands and subcommands).
  - Functionality--functional separation of commands; no multiple commands for (nearly) the same function; no command with multiple functions; clear elaboration of important special cases.
  - Homogeneity--same structure for all commands; same meaning of keywords within all commands (where admissible); same capabilities in comparable contexts; uniform interpretation of missing parameters.

- Problem orientation--no avoidable technical restrictions or exceptions (caused by data structure, programming considerations, etc.); no avoidable separation into dialog branches; any command is admissible at any point of the dialog.

- Uniqueness:

  - Determinism--every command is fully determined by its operands and preset options.

  - No undefined states--all system states are always well defined (e.g., default options until the user sets new options).

- Comfortable language:

  - Powerful commands--existence of powerful commands that do much in a single step.

  - Flexibility--long and short forms of keywords; multilingual forms; direct and indirect operands; adjustment of the system to the user's knowledge and experience; commands adapted for causal, regular, and professional users; user control of system options (by parameters or presetting).

  - Short dialog--complete commands (including subcommands) and even command sequences can be input at once; new commands (or parts of commands) can be defined by a macro feature (renaming of strings).

  - Full use of data structure--all data structures can be displayed and utilized for searching and browsing.

- Other comfort:

  - Input comfort--rereading of previous input or output after corrections have been made; menutechnique.

- Interruption--dialog can be interrupted at any time (stopped or continued subsequently).

- Output language--clear, short, understandable system messages; output discernible from input; output reusable as input (where appropriate).

- Additional comfort--various software and/or hardware provisions, as: function keys, acoustic signal after output transmission, highlighting and/or underlining, clear output arrangement, editing and clear display of tables, various techniques for browsing forwards and backwards; user's notebook.

- Evidence and reusability:

  - Evidence--evidence of the system state (waiting for input, input, waiting for output, output); acknowledgement of executed commands; periodic messages on delays; warnings about laborious commands.

  - Help functions--help functions providing information on the system state, the presently used function, all functions, structure and contents of data bases, past dialog, possible continuations; dialog protocol.

  - Reusability--former commands and output (or part of output) reusable for input; insertion of former commands into the present one (in particular, in query construction); saving commands for later execution.

- Stability:

  - Error handling--clear messages on severe input errors; error correction (wherever possible) on slight errors (but displaying to the user the system's interpretation); uniform error handling; no severe consequences of short input.

- No compulsory situations--no compulsion to continue the dialog in a fixed way; dialog can be stopped at any point.

- Data security--different passwords for data structure and data itself; missing passwords may be subsequently delivered to the system; on inadvertent trial to use secret data, the system must react as if this data did not exist; such situations must not lead to dialog discontinuation; security requirements for part of the data may not impede use of open data.

TUROFF, WHITESCARVER, AND HILTZ'S HUMAN-MACHINE INTERFACE IN A COMPUTERIZED CONFERENCING ENVIRONMENT

- Forgiveness--ease in repairing errors;
- Segmentation--layered approach;
- Variety--choice of style;
- Escape--break out of danger;
- Guidance--direction and learning; and
- Leverage--flexible, powerful features.

KENNEDY'S GROUND RULES FOR A "WELL-BEHAVED" SYSTEM

- Use terse "natural" language, avoid codes, allow abbreviations;
- Use short entries to facilitate error correction and maintain tempo;
- Allow single or multiple entries to match user ability;
- Maintain "social element" to the communication;

- Permit user to control length of cues or error messages;
- Error messages should be polite, meaningful and informative;
- Give help when requested or when users are in difficulty;
- Simple, logically consistent command language;
- Control over all aspects of the system must appear to belong to the user;
- Avoid redundancy in dialog;
- Adapt to the user's ability; and
- Keep exchange rate in user's stress-free range; user can control rate.

STERLING'S CRITERIA FOR HUMANIZING MANAGEMENT INFORMATION SYSTEMS

- Procedures for dealing with users:
  - The language of a system should be easy to understand;
  - Transactions with a system should be courteous;
  - A system should be quick to react;
  - A system should respond quickly to users (if it is unable to resolve its intended procedure);
  - A system should relieve the user of unnecessary chores;
  - A system should provide for human information interface;
  - A system should include provisions for corrections; and
  - Management should be held responsible for mismanagement.

- Procedures for dealing with exceptions:
  - A system should recognize as much as possible that it deals with different classes of individuals;
  - A system should recognize that special conditions might occur that could require special actions by it;
  - A system must allow for alternatives in input and processing;
  - A system should give individuals choices on how to deal with it; and
  - A procedure must exist to override the system.
- Action of the system with respect to information:
  - There should be provisions to permit individuals to inspect information about themselves;
  - There should be provisions to correct errors;
  - There should be provisions for evaluating information stored in the system;
  - There should be provisions for individuals to add information that they consider important; and
  - It should be made known in general what information is stored in systems and what use will be made of that information.
- The problem of privacy:
  - In the design of a system, all procedures should be evaluated with respect to both privacy and humanization requirements; and

- The decision to merge information from different files and systems should never occur automatically. Whenever information from one file is made available to another file, it should be examined first for its implications for privacy and humanization.

- Guidelines for system design having a bearing on ethics:

  - A system should not trick or deceive;

  - A system should assist participants and users and not manipulate them;

  - A system should not eliminate opportunities for employment without a careful examination of consequences to other available jobs;

  - System designers should not participate in the creation or maintenance of secret data banks; and

  - A system should treat with consideration all individuals who come in contact with it.

3.2.3 Interactive Interfacing - Assumed within systematic designing are the interface issues which frequently determine whether or not a microcomputer system is actually used or made into a conversation piece. Some of the interfacing issues of the 1980s will include:

- The use of keyboards;
- Soft versus hard copy;
- Cursor control devices;
- Audio output;

- Speech recognition;
- Graphic output, input, and interaction;
- Response time; and
- Error handling.

Clearly in the 1980s keyboards--if they are to be used at all--must be redesigned with specific reference to error frequency, ease of use, and appearance. Hard and soft copy displays will have to be integrated in ways which balance the need for permanence and the convenience of reinforcing graphic displays. Hard copy devices must be quieted and soft copy displays must use productively multiple character sets, blinking, multiple intensity levels, black/white reversal (where color does not apply), erasing, insertion, cursor action, scrolling, and multiple guide windows--all with reference to the type of DoD user and the nature (requirements) of the task. Cursor control devices should also be evaluated against job requirements and the applicability of lightpen, sonic pen, mouse, touch sensitive, touch sensitive plate, and joystick devices. Audio output should also come into its own from the microcomputer systems of the 1980s. But such output will not just be confined to speech output. Other sounds can also be used to signal and/or feed back to the user. On the other side of the process, speech recognition systems should, where appropriate, be used to augment and replace keyboard and other

input devices. But we will have to develop continuous speech systems far superior to DRAGON, HEARSAY-I, and HARPY if we are to bridge the gap between cumbersome and productive speech input. Graphic input and (especially) output systems should proliferate in the 1980s since perceptual psychologists have conclusively demonstrated the benefits of imagery over numbers or words. Advice from researchers like Bennett should be studied:[4]

BENNETT'S SET OF GUIDELINES
FOR GRAPHICS SYSTEMS DEVELOPERS

- Arrange text and graphic symbols on each presentation to establish an explicit context for user action;
- When a user process is not known in advance, concentrate on displayable data representations and then design operations to act upon these representations; and
- Design the system to provide an explicit framework for representations. The framework gives a uniformity of structure within which the user can synthesize problem solutions. This framework can be developed even though problems themselves are unstructured.

Response time considerations should also be made paramount. Miller has compiled a list of user activities and response times which are useful but deceiving:[5]

MILLER'S SYSTEM RESPONSE TIME
AS A FUNCTION OF USER ACTIVITY

User Activity	"Maximum" Response Time
Control activation.	0.1 SECOND
System activation.	3.0
Request for given service:	
simple	2
complex	5
loading and restart.	15-60
Error feedback.	2-4
Response to ID.	2
Information on next procedure.	<5
Response to simple inquiry from list.	2
Response to simple status inquiry.	2
Response to complex inquiry in table form.	2-4
Request for next page.	0.5-1
Response to "execute problem".	<15
Light pen entries.	1.0
Drawing with light pens.	0.1
Response to complex inquire in graphic form.	2-10
Response to dynamic modeling.	-
Response to graphic manipulation.	2
Response to user intervention in automatic process.	4

They are deceptive because it may well be that in order to shape user behavior we should deliberately lengthen some response times; in other words, great speed may not always be desirable. Similarly, error handling routines must be developed which are not offensive and which teach (without repremanding) the user.

3.2.4 Software Production Psychology - All of the above presumes the existence of highly talented, dedicated programmers who are as knowledgeable about hardware as they are about software. Unfortunately, virtually every projection available today indicates that throughout the 1980s a critical shortage of programmers will persist. We must therefore maximize the output of those programmers which we do employ. Learning, designing, composition, comprehension, testing, debugging, documentation, and modification capabilities must all be evaluated and improved. Perhaps for the first time, serious programming managers must pay very special attention to the overall programming environment, the components of which include the physical, social and the managerial environments. Of critical importance, however, will be software quality evaluation procedures. Boehm et al. have developed a software "characteristic tree" which if (easily) converted into a multi-attribute utility model and then manipulated through an EVAL model, could provide us with an excellent evaluation tool, as suggested below.[6]

Accessibility. Extent to which code facilitates use of its parts.

Accountability. Extent to which code can be measured.

Accuracy. Extent to which the output produced by code are sufficiently precise to satisfy their intended use.

Augmentability. Extent to which code a can be expanded in computations functions, or data storage requirements.

Availability. Degree to which a system of resource is ready to process data. Availability. MTBF/(MTBF + MTTR).

Communicativeness. Extent to which code facilitates the specifications of inputs and provides outputs whose form and content are easy to assimilate.

Completeness. Extent to which all parts of code are present and developed.

Conciseness. Extent to which excessive information is not present.

Consistency. Extent to which code contains uniform notation, terminology, and symbology within itself, and external consistency to the extent that the content is traceable to the requirements.

Device independence. Extent to which code can be executed on computer hardware configurations other than its current one.

Efficiency. Extent to which code fulfills its purpose without wasting resources.

Human engineering. Extent to which code fulfills its purpose without wasting users' time and energy or degrading their morale.

Legibility. Extent to which function is easily discerned by reading code.

Maintainability. Extent to which code facilitates updating.

Modifiability. Extent to which code facilitates the incorporation of changes.

Portability. Extent to which code can be operated easily and well on computer configurations other than its current one.

Reliability. Probability that an item (device or program, system) will function without failure over a specified time period or amount of usage.

Robustness. Extent to which code can continue to perform despite a violation of the assumptions in its specifications.

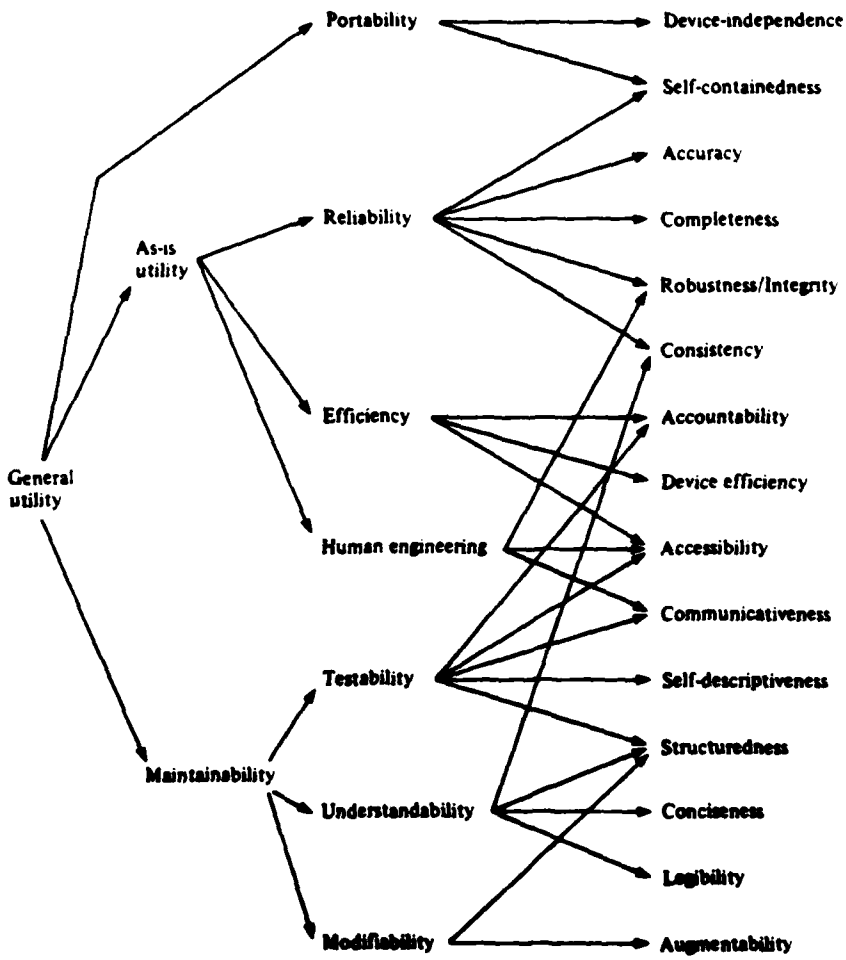
Self-containedness. Extent to which code performs its explicit and implicit functions within itself.

Self-descriptiveness. Extent to which reader of code can determine its objectives, assumptions, constraints, inputs, outputs, components, and revision status.

Testability. Extent to which code facilitates establishment of verification criteria and supports evaluation of its performance.

Understandability. Extent to which purpose of code is understandable to reader.

Usability. Extent to which code is reliable, efficient, and human-engineered.



3.2.5 Unconventional Documentation & Transfer - In order to fully exploit the capabilities of future microcomputer-based problem-solving systems and successfully transfer the systems into operational and quasi-operational DoD environments, we should adopt a series of unconventional strategies. One is the remote video tape-based demonstration. Another is the sampler demonstration, which involves the loaning of systems for indefinite periods of time; and still another involves programming the systems to introduce and explain themselves in a manner not unlike that which is used by manufacturing vendors. Such demonstrations can be of invaluable help to those who must convince others that what they have developed may be of real use.

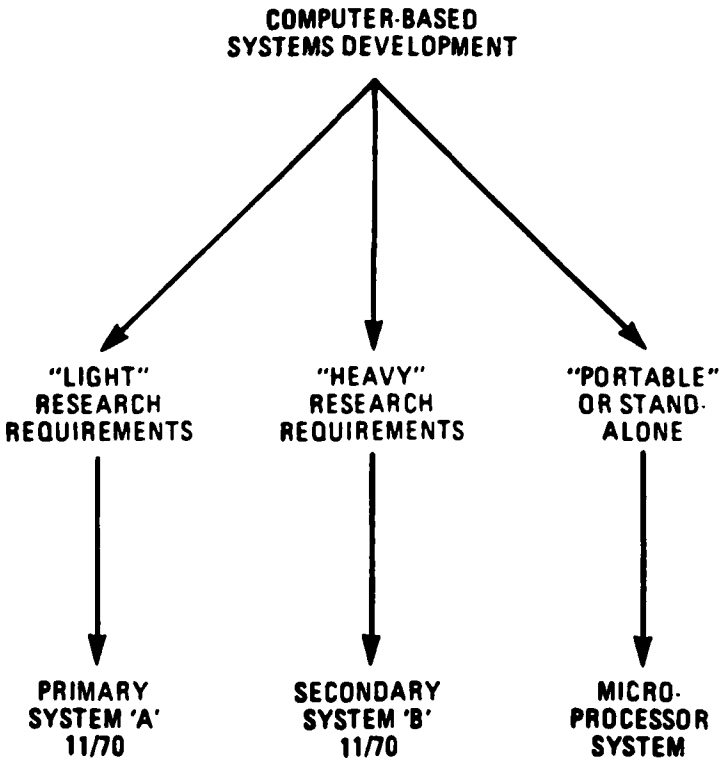
Documentation should also be transformed from the inanimate to the animate. Computer-generated system specifications and functional descriptions can be of immense transfer use, as can on-line users manuals. Similarly, films of documentation can also help to bridge the gap between the developer and the user.

3.2.6 Substantive Priorities - It is important to remember that all research and development should be targeted at some specific problem area(s). As we move into the 1980s the people problems to which DARPA/CTD will respond will be growing in

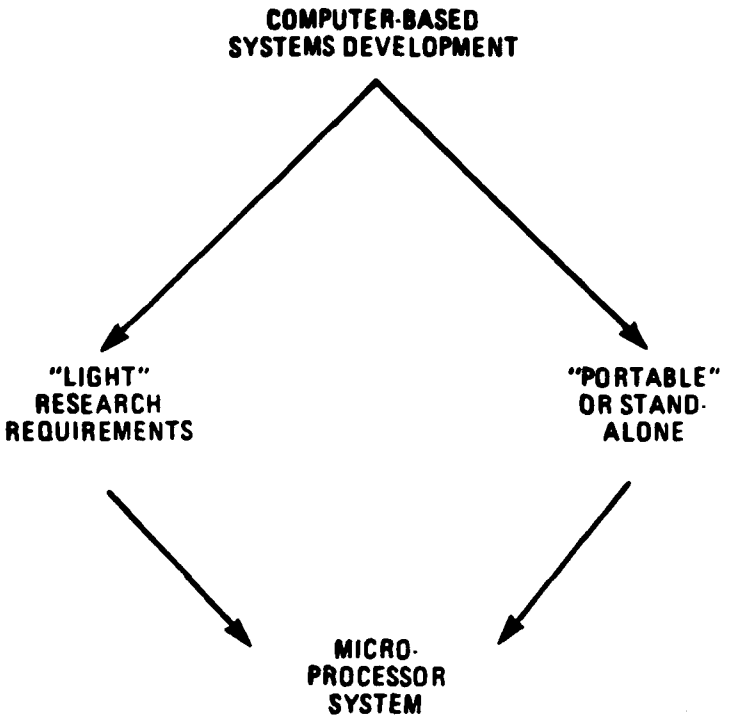
number and severity. Decision-making, forecasting, planning, assessment, evaluation, information management, communications, learning, training, organizational, bureaucratic, personnel, resource allocation, data base construction, software psychology, personal computing, and management problems, among many, many others, will all follow us well into the 1980s (and certainly even beyond). We must therefore orient our new microcomputer systems to these and whatever other critical areas appear.

4.0 THE DARPA/CTD/DDF IN THE 1980s

In the 1970s, the DDF was organized around two PDP 11/70s and several microcomputer systems matrixed around light and heavy and stand alone research and applications requirements, as suggested below.



In connection with this matrix, the design, development, demonstration, transfer, and documentation of computer-based systems was pursued. In the 1980s, we can expect a shift in this configuration, as suggested below.



In all likelihood the microcomputer system will become the primary computer-based system vehicle in the 1980s. It will also be augmented by unique display and data storage devices, such as videodisc, fiche, and large capacity hard disc storage systems. Indeed, intelligent terminals, as we have come to understand them, will evolve dramatically in the 1980s. The DDF's design and development tasks will thus change accordingly in the future. Specifically, we plan to interface videodisk systems to microcomputers, experiment with touch sensitive and other input devices, and develop and test a number of

applications packages, including text editing, mathematical function, storage and retrieval, communications, and UNIX emulative systems.

On the (unconventional) documentation, demonstration, and transfer side, the new DDF will engage in video production and distribution, audio-visual demonstrations, and an aggressive document production and distribution campaign, among other duties. Most importantly, since microcomputer-based systems are easily transportable, we can transfer more easily and frequently than when our systems were minicomputer-based.

5.0 CONCLUSION

FY81 has launched a new chapter in the design, development, demonstration, and transfer of advanced computer-based systems. Essentially, this chapter has been made possible and desirable via the revolution in microcomputing and the associated interfacing of other electronic data storage and display devices. While we are by no means declaring the minicomputer dead, we are clearly stating that the most unique research, development, and transfer opportunities lie with the microcomputer.

6.0 FOOTNOTES

[1]This section relies heavily upon James F. Wittmeyer, III, James J. Allen, Jr., Richard A. Winter, and Christopher F. Herot, Cybernetics Technology Office Demonstration and Development Facility (CTO/DDF) Overview and FY 79 Plan. Cambridge, MA: Computer Corporation of America, October 1, 1978.

[2]See F.P. Brooks, Jr., The Mythical Man-Month. New York, NY: Addison-Wesley, 1975.

[3]See W.J. Hansen, "User Engineering Principles for Interactive Systems," Proceedings of the Fall Joint Computer Conference, 39, AFIPS Press, 1971, 523-532; T. Wasserman, "The Design of Idiot-Proof Interactive Systems," Proceedings of the National Computer Conference, 42, AFIPS Press, 1973; R.W. Pew and A.M. Rollins, Dialog Specification Procedure, Cambridge, MA: Bolt, Beranek and Newman (BBN), Report #3129, 1975; Brian R. Gaines and Peter V. Facey, "Some Experience in Interactive System Development and Application," Proceedings of the IEEE, 63, June 1975, pp. 894-911; D.R. Cheriton, "Man-Machine Interface Design for Time-Sharing Systems," Proceedings of the ACM National Conference, 1976, pp. 362-380; F. Gebhardt and T. Stellmacher, "Design Criteria for Documentation Retrieval Languages," Journal of the American Society for Information Science, 29, July 1978, pp. 191-199; T.C.S. Kennedy, "The Design of Interactive Procedures for Man-Machine Communication," International Journal of Man-Machine Studies, 6, 1974, pp. 309-334; Stephen E. Engel and Richard E. Granda, Guidelines for Man/Display Interfaces, IBM, Report #00.2720, 1975; Jacob Palme, "Interactive Software for Humans," Management Datamatics, 5, 1976, pp. 139-154; M.W. Turoff, J. Whitescraver, and S.R. Hiltz, "The Human Machine Interface in a Computerized Conferencing Environment," Proceedings of the IEEE Conference on Interactive Systems, Man, and Cybernetics, 1978, pp. 145-157; and T.D. Sterling, "Guidelines for Humanizing Computerized Information Systems," Communications of the ACM, 17, 1974, pp. 609-613.

[4]See J.L. Bennett, "The User Interface in Interactive Systems," in C. Cuadra, ed., Annual Review of Information Science and Technology, 7, Washington, D.C.: American Society for Information Science, 1972, pp. 159-196.

[5]See Robert B. Miller, "Response Time in Man-Computer Conversational Transactions," Proceedings of the Spring Joint Computer Conference, AFIPS Press, 1968, pp. 267-277.

[6]B.W. Boehm, Jr., J.R. Brown, and M. Lipow, "Quantitative Evaluation of Software Quality," Software Phenomenology Working Papers of the Software Lifecycle Management Workshop, 1977, pp. 81-94.

7.0 REFERENCES

Bell, D., Programmer selection and programming errors, The Computer Journal, 19, 3, (1974).

Bennett, J.L., The user interface in interactive systems, In C. Cuadra, (Ed.), Annual Review of Information Science and Technology, 7, American Society of Information Science, Washington, D.C., (1972).

Boehm, B.W., J.R. Brown and M. Lipow, Quantitative evaluation of software quality, Software Phenomenology Working Papers of the Software Lifecycle Management Workshop, (August 1977).

Boies, S.J., User behavior on an interactive computer system, IBM Systems Journal, 13, 1, (1974).

Brooks, F.P., Jr., The Mythical Man-Month, Addison-Wesley, New York, (1975).

Brooks, W.D. and P.W. Wilbur, Software reliability analysis, IBM Technical Report FSD 777-0009.

Cheriton, D.R., Man-Machine interface design for time-sharing systems, Proceedings of the ACM National Conference, (1976).

Eason, K.D., Understanding the naive computer user, The Computer Journal, 19, 1, (February 1976).

Endres, A., An analysis of errors and their causes in system programs, In Proceedings, 1975 International Conference on Reliable Software, ACM SIGPLAN Notices, 10, 6, (1975).

Engel, Stephen E. and Richard E. Granda, Guidelines for Man/ Display Interfaces, IBM Poughkeepsie Laboratory Technical Report TR 00.2720, (December 19, 1975).

Gebhardt, F. and I Stellmacher, Design criteria for documentation retrieval languages, Journal of the American Society for Information Science, 29, 4, (July 1978).

Hiltz, S.R. and M. Turoff, The Network Nation: Human Communications via Computer, Addison-Wesley, Reading, Mass., (1978).

Kennedy, T.C.S., The design of interactive procedures for man-machine communications, International Journal of Man-Machine Studies, 6, (1974).

Miller, Robert B., Response time in man-computer conversational transactions, Proceedings Spring Joint Computer Conference 1968, 33, AFIPS Press, Montvale, New Jersey.

Newell, A. and H.A. Simon, Human Problem Solving, Prentice-Hall, Englewood Cliffs, New Jersey, (1972).

Newman, W.M. and R.F. Sproull, Principles of Interactive Computer Graphics, (Second Edition), McGraw-Hill, New York, (1978).

Palme, Jacob, Interactive software for humans, Management Datamatics, 5, 4, (1976).

Pew, R.W. and A.M. Rollins, Dialog Specification Procedure, Bolt Beranek and Newman, Report No. 3129, Revised Edition, Cambridge, Massachusetts, 02138, (1975).

Shneiderman, Ben, Software Psychology, Winthrop, Cambridge, Massachusetts, (1980).

Sterling, T.D., Guidelines for humanizing computerized information systems: A Report from Stanley House, Communications of the ACM, 17, 11, (November 1974).

Turoff, M.W., J. Whitescarver and S.R. Hiltz, The human machine interface in a computerized conferencing environment, Proceedings of the IEEE Conference on Interactive Systems, Man, and Cybernetics, (1978).

Wasserman, T., The design of idiot-proof interactive systems, Proceedings of the National Computer Conference, 42, AFIPS Press, Montvale, New Jersey, (1973).

Weizenbaum, J., Computer Power and Human Reason, W.H. Freeman and Company, San Francisco, California (1976).

Wittmeyer, James F., III, et al., CTO/DDF Overview and FY 79 Plan, Computer Corporation of America, Arlington, Virginia, (1978).