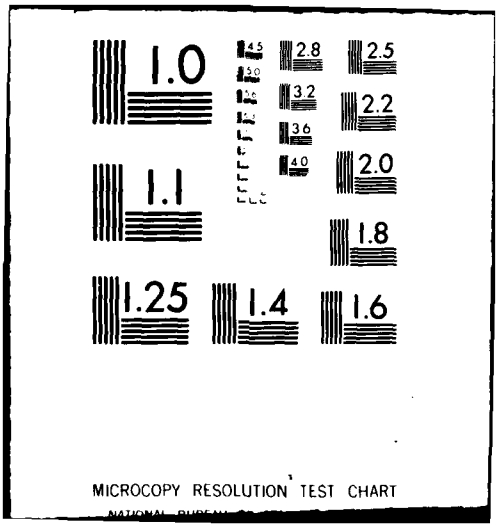
AD-A110 248 RAVEN SYSTEMS AND RESEARCH INC ATLANTA GA F/G 5/1
MICRO RESOURCE ESTIMATION RESEARCH PROJECT. PHASE IV.(U)
DEC 81 DAAK70-78-D-0052
UNCLASSIFIED NL

END
DATE FILMED
2-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART

LEVEL II

AD A110248

DTIC ELECTE FEB 1 1982 S D H

MICRO RESOURCE ESTIMATION RESEARCH PROJECT PHASE IV

Prepared for the U. S. Army
Institute for Research in
Management Information and
Computer Sciences

Contract No. DAAK70-78-D-0052

December 1981

DTIC FILE COPY

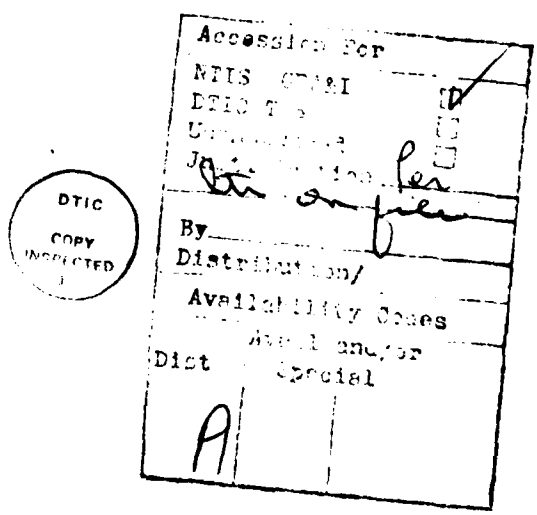
Raven Systems & Research, Inc.

Atlanta, Georgia

The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

Micro Resource Estimation Project: Phase IV

Final Report



Micro Resource Estimation Project: Phase IV

Final Report

Microestimation is a technique by which system analysts predict the manpower requirements needed to make changes to computer programs. During the past three years, Raven Systems and Research has conducted several studies related to microestimation, and it has recently developed a prototype Automated Microestimation System (AMS). This report is provided to accompany the delivery of the AMS. It is both a description of the new system and a summary of research to date.

This report consists of four parts:

- Part I describes the history of the microestimation study, including significant conclusions.
- Part II describes the AMS, its development and overall design and operation.
- Part III discusses questions related to interfacing AMS with several other systems either currently in existence or under development.
- Part IV discusses several general problems related to microestimation, brought to light as a result of the entire microestimation study. Part IV also makes recommendations for further development and research related to the AMS and microestimating in general.

Part I: The Microestimation Studies

The United States Army Computer Systems Command (CSC) has as its responsibility the development and maintenance of Standard Army Multicommand Management Information Systems (STAMMIS). These systems are administrative in nature and are distributed worldwide to computer centers throughout the Army. These systems support functions such as supply (SAILS), civilian payroll (SIDPERS), and miscellaneous fund accounting (STANFINS). The systems are standardized in the sense that programs are distributed in object code form only, and the local user has no authorization to make changes to the systems.

All systems changes to STAMMIS are accomplished by the mechanism of a System Change Request (SCR). In practice, SCR's are originated at a variety of sources and are channeled through the Proponent Agency to CSC for "impacting." "Impacting" consists of using the microestimation technique to estimate the number of manhours (and hence the cost) required to implement an SCR. At this point, two things are worth noting: 1) An SCR may affect several programs. A data format change, for example, may propagate itself through an entire system. 2) At any given instant, most Proponent Agencies have more outstanding SCR's than a CSC programming team can implement within a reasonable length of time. Hence, on a regular basis, the Proponent Agency and the programming team cooperatively develop System Change Packages (SCP's) which are then programmed by CSC

personnel and then broadcast to the field installations. An SCP is a collections of SCR's.

The microestimate, then, is a crucial part of the entire system change process. It provides the Proponent Agency with decision-making information (what SCR's to put in an SCP) and it helps the CSC system manager estimate manpower needs for system changes.

The microestimate (sometimes called a "Form 50") is fairly simple to make. The details are fully covered in the appropriate CSC manuals, but in general, the estimator 1) determines which program functions will be affected by the SCR, 2) makes an estimate of the complexity of those affected functions, and 3) places that information, (plus information on turnaround time, developmental status of the system, and programmer resources) into a standatd equation which, when solved, produces a manpower estimate.

Raven's study of the microestimation technique began in 1979. At that time, CSC Support Group Lee appeared to be having trouble with its microestimates. The evidence of this trouble was in the form of missed deadlines, and it was felt that the manpower estimates (and hence the microestimating technique) were at fault. In its first task (MICRO I), Raven was asked to validate and, if necessary, suggest changes to the Form 50 and/or the microestimation procedure.

The MICRO I study reached several significant conclusions:

- Actual manpower estimates for system change packages (SCP's) are usually not directly related to the microestimates made when impacting the SCR. Upon examination, the reasons for this are quite simple. When an SCP is assembled, it is common for several SCR's to impact each program in the system. To the extent that they can be programmed, debugged, and tested simultaneously, development time is reduced. Put another way, three changes accomplished on the same program at one time require less effort than they would if each were done individually.

  An additional reason why microestimates are not generally used as management planning tools is that SCR's frequently affect more than one program. In practice, programs are assigned to individual programmers or programming teams. As a result, an SCR is not a meaningful unit of work at the management level. In fact, SCP's are managed by program, not by SCR.

- Review of a representative group of completed Form 50's indicated that over half were completed incorrectly. While Raven personnel did not attempt to verify the actual factors selected by the estimators, over half the estimates contained actual computational errors. These errors frequently doubled the microestimate.

- Finally, the MICRO I study revealed that the actual manpower estimates used by managers,regardless of their sources, were relatively accurate. However, they consistently tended to be slightly low. In most cases, however, the sources of the estimates were unclear. Words like "SWAG" and "programmer estimate" were disturbingly common in interviews with persons responsible for estimating.

Aside from the fact that microestimation was a locally-mandated practice and also was felt to be useful in assembling SCP's, the technique appeared to have intrinsic merit as a approach to manpower estimation. Since it had not been possible (For the reasons outlined above) to validate microestimation directly, personnel at the School of Industrial and Systems Engineering, Georgia Institute of Technology, were engaged to develop a design for a formal study to validate the Form 50. This activity (MICRO II) resulted in a set of formal study guidelines, utilized by Raven in MICRO III.

The mechanical details of the MICRO III study are irrelevant here. For details, the reader is invited to consult Raven Technical Paper TP-4 (December, 1980). However, the results are quite important to the microestimation process. These results were as follow:

- o There is a formally defined and unambiguous process for making program-by-program estimates based on Form 50's (SCR's). This process, once the SCR's are identified, is quite mechanical and is amenable to computerization. The process is called "compositing", was developed at Ft. Lee, and is described in detail in TP-4, referenced above.

- o The composite estimating procedure is quite accurate. In a study of one SCP, the correlation between manpower estimates on fourteen programs and the actual manpower required was 0.89. These fourteen program estimates encompassed 8 SCR's, each affecting from 1 to 10 programs. Consequently composite estimating, based on microestimating, is a useful management tool for managing the development of SCP's.

One of the problems with composite estimating is that the technique, while simple, requires a cinsiderable amount of paper shuffling and hand calculation. The 8 SCR SCP described above involved a total of 34 individual Form 50's. In addition, at least one other alternate SCP was considered, involving stll more microestimates. It requires one to two days to develop a composite estimate by hand. If several SCP alternates are considered, a week or more of hand calculation may be required.

The development of a composite SCP is a crucial step in the management process. Since SCP broadcast deadlines are relatively inflexible, it is important to develop SCP's that 1) can be developed within the time frame defined, and 2) do not place an undue burden on one menber of the programming team while others are relatively unpressured. In addition to meeting these managerial needs, the ideal SCP must also come as close as possible to meeting the priority demands of the proponent agency, which usually has more SCR's in the queue than it is possible to implement.

These conflicting needs are resolved at what is known as a "scrub session." A scrub session is essentially a negotiation between representatives of a proponent agency and CSC at which proposed SCP's are developed. A scrub session is usually followed by a period wherein the CSC programming team reviews the SCP and resolves any possible problems. Eventually, the SCP is finalized, approved by the proponent agency, and accepted by the CSC programming team.

It is clear that any estimating tool which can facilitate the realistic assembly of SCP's is useful in the management process. Current techniques involve "seat of pants" knowledge of how programs and SCR's interact, followed by program-by-program estimating (compositing) prior to final SCP approval. Compositing may be formal (as described above) or may simply involve programmers and analysts making individual estimates.

Both methodologies require considerable time and hence reduce the exploration of a variety of SCP options. If rapid development of composite SCP estimates were possible, not only would manpower be saved, but more effective allocation of programming resources would result.

Part II: The Automated Microestimation System

In February, 1981, Raven Systems and Research received a Task Order from the Army Institute for Research in Management Information and Computer Science to (1) develop an automated microestimating system (AMS) and (2) study the feasibility of several different interfacing options related to the AMS and other AIRMICS and CSC-sponsored manhour accounting and/or management systems. This section describes the development of the AMS and discusses its functional characteristics. Accompanying this report is a User's Guide and System Documentation for the AMS which are incorporated herein by reference. The User's Guide contains complete operating instructions for AMS, including color illustrations of actual screen displays. The system documentation contains functional hierarchy diagrams and system operating characteristics.

The design of AMS began with several known needs and constraints. They were:

- The system would utilize currently available color graphics technology embodied in the CSC-owned Chromatics CG-1999 machines.
- The system would utilize distributed processing approaches in a manner similar to that used by the already-developed Automated Project Management System (APMS).
- To the extent possible, user-friendliness would be enhanced by the use of color displays, user prompts, powerful edit checks, and well-designed screen layouts.
- The system should allow users to rapidly and easily compute Form 50's, file those Form 50's, and delete and modify them as needed.
- The system should allow the used to easily select a set of SCR's composing and SCP, compute a composite estimate on the SCP, and display the results in a manner useful to a system manager. The system should allow for the "saving" of SCP's so that they could be passed to other systems or reviewed and modified at will.

With this set of criteria in mind, Raven personnel began by developing a set of informal functional specifications. These specifications were essentially "screen layouts," describing in detail what each AMS screen would look like and how it would interact with the user.

It was recognized at the outset that the computational aspects of the AMS, while lengthy, were essentially trivial. On the other hand, file handling was extremely complicated. The reasons for this have to do with the retrieval requirements of the system, viz:

- In order to modify a Form 50, the system must be able to directly access the form, modify it, and replace it in the file on a real-time basis.

- In order to compute a total SCR impact from a set of Form 50's (one Form 50 per program impacted by the SCR), the system must sequentially and exhaustively access each Form 50 belonging to that SCR. Since a number of SCR's are included in an SCP, this process must be repeated several times every time an SCP is displayed.

- In order to compute a total SCP impact, the system must access Form 50's in program order. (The previous computation requires SCR order.) The two requirements are mutually exclusive. In conventional data processing terms, the system requires two files of Form 50's; one sorted by program within SCR; the other sorted by SCR within program. The situation is rather like asking for a telephone book to be sorted in both name and telephone number order.

While relational database systems offer excellent solutions to these kinds of problems, hardware requirements made such an approach impossible. It would be necessary to program the AMS in Fortran IV, using existing PDP-11 direct file access facilities. As a result, the system design called for a total of five files, only three of which contain user-prepared data. The other two files merely contain pointers for referencing back and forth.

Upon completion of the general functional specifications, Raven personnel presented the screen diagrams and an oral description of system functions to the contract officer and the selected Ft. Lee personnel. The only major change was a request to be able to embed complexity factors in the program file in order to precompute them when a particular SCR impacted a specific program fuctional area. While it was not possible to accommodate this request in the final implementation, the program file was redesigned so that the appropriate fields would be available for further expansion.

Following the specification phase, programming on AMS went forward in a conventional manner. The programming effort had three significant characteristics.

- Since the system was to be distributed, simultaneous detailed design and programming took place on two different machines in two different languages. The Chromatics CG-1999 was programmed in BASIC; the AIRMICS PDP-11/70 was programmed in FORTRAN IV. In general, it was decided that the FORTRAN program would maintain the files and perform the SCP computation. The BASIC program would perform SCR computations, all edits, and all screen routines. There was a conscious effort in the AMS design to build more intelligence into the Chromatics machine than the previous system developed by Raven (APMS) had required. This was in order to reduce communication to a minimum.

As a result, the tasks of each programmer on the project were clearly defined. In large measure, each could program independently. The Fortran programmer could program file maintenance routines with no knowledge of how the Chromatics was editing input data; it was only necessary to know the data would be "clean." The BASIC programmer could design edits with no need to know the details of how the data would be shared or used.

Once the basic modules were completed, however, the nature of the programming task changed. While data formats were never a problem, handshaking and message protocols were. For example, what was the appropriate (distributed) response when the user requested the addition of a nonexistent SCR to an SCP? Problems of this nature required close collaboration between the two programmers and extensive system testing.

- Finally, it should be mentioned that the system development effort was somewhat hampered by the fact that the Chromatics terminal was not located on the contractor's site. While there are obvious overall advantages for AIRMICS in having the terminal located at the Georgia Tech campus, distributed program testing becomes a burden when contractor personnel must make trips across town whenever a testing session is needed.

The development of the AMS did, however, proceed with no major problems. The completed system has two basic "screens." One is a duplicate of a Form 50. The user can step through the form in a variety of ways. All computations are performed automatically by the Chromatics Terminal. When the user is satisfied with the results, he may touch a "post" target with his light pen to file the Form 50 away on the Form 50 file.

The second screen is the heart of the microestimating system. It is used to prepare composite estimates of SCP's. The user is confronted with two simultaneous displays: one displays SCR names and the names and impact hours of the major programs affected by the SCR; the other displays, by programmer, the effort expected on the total SCP. The user may use the light pen and keyboard to add and delete SCR's. As he does so, both displays constantly change to reflect the status of the SCP. At any given point the user may save the SCP for review at a later time, or choose to work on another SCP already stored in the file.

Screen photographs and accompanying user instructions are provided in the AMS User's Guide accompanying this report. The AMS currently represents a working, usable prototype system for rapid and accurate preparation of SCP's. As such, it provides the project manager with a powerful tool for developing planning options when SCP's are developed.

Part III: Interfacing Questions

Raven personnal were tasked to examine the feasibility of several different options relating to interfacing the AMS with various systems. These systems were:

A. The Automated Project Management System (APMS) currently being tested by the Georgia Institute of Technology.

B. The Project Management System (PMS). PMS is a commercial project management system (PAC II) used by some elements at CSC-Support Group Lee to plan and monitor programming projects.

C. The SCR Accounting System. This system is used by CSC to monitor and report on the status of each SCR which it has received. This system is essentially a reporting system rather than a management system.

In addition to examining the interfaces between AMS and the three systems named above, Raven was tasked with studying the feasibility of bringing AMS up on the Ft. Belvoir 3033 computer operating under TSO.

Each of these four issues will be addressed separately in the sections below:

A. Interface between AMS and APMS.

Compatibility between any two systems depends on two elements:
(1) The system software and hardware requirements must be sufficiently similar so that they may function on the same physical system.
(2) The two systems must share enough common data elements to make interfacing of the two systems worthwhile.

In the case of the AMS/APMS interface, software and hardware requirments of the two systems are essentially the same. Both were, in fact, developed on the same system; they share common hardware and software requirements; and they both were initially developed by the same programming team using the same distributed design philosophy.

The data structures of the two different systems are, however, quite different. AMS uses several direct access files and a very fine data structure: each Form 50 is represented by a single record. APMS, on the other hand, uses a single direct access file with each network represented by a single, elaborately structured record. While this causes no direct problems, it does require that an interchange program be developed to transfer AMS records to the APMS. Such a program would accept as input an SCP name, extract the relevant Form 50's from the F5Ø file, attach the relevant resource from the program file, and reformat the information into a partial APMS record.

APMS requires a number of data items for successful operation. The major ones are:

1. Resource (manpower) requirements data.

2. Resource availability data.

3. Activity sequence data (precedence relationships).

Of the three data classes listed, only the first is supplied by AMS. However, resource requirements data constitute more than 50% of the total data input for APMS. Therefore, the interfacing of AMS with APMS appears to be an attractive endeavor.

B. Interface between AMS and PMS (PAC II).

PMS requires four basic data elements.

1. Resource requirements data.

2. Resource availability data.

3. Activity sequence data.

4. Activity priority data.

Basically, AMS is able to supply resource requirements data only to the PMS. However, this information again accounts for more than 50% of the data requirements of PMS. Therefore, on a data element basis, the interfacing of AMS and PMS appears worthwhile.

Hardware and software compatibility, however, is another matter. There are several fundamental questions in this area which will be addressed in Section D. Only if these questions are successfully resolved will an AMS/PMS interface be a realistic possibility.

C. Interface between AMS and the SCR Accounting System.

Interface between AMS and SCR Accounting System is subject to the same hardware and software compatibility constraints as the AMS/PMS interface. These will be discussed in Section D.

In addition, AMS can supply only one data element to the SCR Accounting System (Resource requirements). Since the SCR Accounting System is primarily concerned with an SCR's status (broadcast date, package membership, etc.), the projected resource requirement is a minor part of its data input requirements. This data item is routinely handled and posted through current administrative channels. As a result there appears to be no valid reason for attempting to interface AMS and the SCR Accounting System.

D. Compatibility between AMS, as currently implemented, and IBM 3033 operations under TSO.

It is clear that if AMS or APMS are to become viable fielded systems they will have to be made operational on one or more of CSC's large computers. This is true for two reasons. First, the AIRMICS PDP-11 is not designed for, nor is it capable of supporting the files and computing requirements necessary to support AMS on a widescale basis. Second, while communications access to the AIRMICS PDP-11 is very limited, every major CSC element has convenient and direct access to at least one of the command's large computers. Bringing AMS up on the 3033 under TSO, then, would appear to be a top priority.

In order to test the feasibility of bringing up AMS under TSO, Raven personnel went to Ft. Belvoir and partially installed the system on the 3033 located there. Rather than test with AMS, Raven prepared all tests with APMS, an older system with similar operating characteristics.

The APMS consists of thirty-two separate Fortran modules, linked through the overlay structure available in IAS. These modules were offloaded onto an unlabeled 1200 BPI ASCII tape at the AIRMICS computer center and then read onto the Ft. Belvoir 3033. Reading the tape onto the 3033 was uneventful. The only minor problem was that it was necessary to bypass label processing on the transmittal tape due to the fact that the PDP-11 program makes no labels on the tape.

After installing the APMS modules as TSO files, Raven personnel compiled each module, listing all compile errors. It was anticipated that there would be only minor problems, since APMS is written in very low-level Fortran. This proved to be the case. All programs compiled with a total of only three error types.

1. IBM Fortran does not accept BYTE as a data type. This data type is used frequently in APMS and AMS.

2. IBM Fortran does not allow incrementing a DO loop by a negative number. (Used to index "backwards" through an array.)

3. IBM Fortran does not support embedded OPEN and CLOSE statements. Such information must be provided by JCL.

All three of these compile errors are considered minor. It is anticipated that an error-free compile of APMS would require no more than a week's work, with AMS requiring only one or two days.

After compile, Raven personnel debugged a sub-module of APMS and attempted to link it. No problems were encountered. However, since the architecture of the PDP-11 requires that both AMS and APMS have an extensive overlay structure, a complete linkage of either system would be somewhat more complex. This is for the following reason.

In order to build a successful overlay structure, it is sometimes necessary to put duplicate (large) subroutines into several "leaf" nodes of an overlay tree rather than in the "root" module. Since only one leaf module resides in core at one time, the PDP-11 overlay generator has no trouble resolving these duplicate subroutines. However, when the entire system resides in core at one time, address ambiquity occurs and the linkage editor is unable to resolve the duplicate addresses. In order to correct this it will be necessary to remove all duplicate subroutines from the system, a nontrivial job. It is estimated that such a task would require approximately 8 working days for APMS, no more than five on AMS.

Finally, While the current file structure of AMS can be supported in a TSO environment, a much more practical approach would be to use the currently available IDMS database management system. Such an approach would require extensive reprogramming of the AMS, but would result in considerably reduced response times, plus the ability to access considerably larger databases. It is estimated that reworking the AMS to utilize IDMS file support would require a minimum of 10 working days.

In summary,

A. Interface between AMS and APMS utilizing the PDP-11 system is both feasible and practical.

B. Interface between AMS and PMS (PAC II) is feasible in a TSO environment.

C. Interface between AMS and the SCR Accounting System is feasible, but would be of little practical benefit.

D. Placing AMS in a TSO environment is relatively simple if the current file structure is maintained. If a database management system is used, extensive reprogramming would be required.

E. All feasible interfaces presuppose an "extraction" program which will access the AMS files and reformat the appropriate data for the other systems. There is no feasible approach to direct file sharing given the idiosyncrasies of the various systems involved.

Part IV. Recommendations and Conclusions

Since MICRO IV was not a research project per se, the recommendations and conclusions which follow grow out of practical experience rather than statistically validated research. They do, however, reflect the careful thought and extensive experience of Raven personnel during the three years of the Microestiamtion project.

A. Microestimation is a valid tool for predicting manpower requirements for programming SCR's. When used with a compositing technique, it is useful for developing and managing entire System Change Packages.

B. It has not been shown by this microestimation project that microestimation is a valid technique for estimating requirements on large, developmental systems. We view with skepticism any proposal which suggests that microestimating be used in such an environment.

C. The AMS represents a prototype system for developing composite estimates on a rapid, validated basis. It is recommended that further work be done on this system, aimed at refining the man-machine interactions.

D. It is recommended that the refined system developed in (C) above be installed on Command-wide large-scale systems and that the appropriate interfaces between AMS and PMS (PAC II) be established.

E. At such a time as APMS has been validated as an appropriate management tool, it is recommended that it also be installed on Command-wide large-scale systems and interfaced with AMS.

END

DATE
FILMED

2-82

DTIC