PROGRAMMING ENVIRONMENT QUESTIONNAIRE

Technical Report No. 139

Irvine
Programming Environment
Research Center

Department of Information and Computer Science
University of California at Irvine
Irvine, California 92717

(714)-833-6357

Thomas A. Standish, Editor

Authors: T.A. Standish, Ira Baxter, Sherry Cameron, Dave Davis, Hung Do, Gene Fisher, Jerry Hamilton, Steve Hampson, Robert Hartmann, Daphne Hassner, Matt Heffron, Ken Hertzler, Hank Kleppinger, Stephen McHenry, J.R. Meehan, Terry Mellon, Mike Mole, Eric Olson, Paul Palmquist, Bill Rockwell, Roger Smeaton, Frank Tadman, and Greg Taylor.

PREFACE

This questionnaire was produced as a class project for a graduate seminar on programming environments. The class, Information and Computer Science 280B, began on 24 October 1979 and will conclude on 2 December. It has been taught by Professor Thomas A. Standish.

The purpose of the class is to initiate what will hopefully be a continuing study of programming envrionments. A specific goal for such a study is the development of a critical new understanding of what makes environments good, what distinguishes effective ones from ineffective ones, how we can tell when an environment is essentially complete or when it is critically deficient in some aspect, how we can guarantee that the tools in an environment cooperate smoothly together and have consistent interfaces, what sorts of environments support effective life cycle management disciplines, etc.

This a preliminary draft of the questionnaire which we expect will undergo some potentially major revisions. We chose the questionnaire format as an initial vehicle through which to develop a taxonomy of programming environment characteristics. When the first draft of the questionnaire was completed, students began work on the analysis of several well-established existing environments, using the questionnaire as a guideline and basis for the analyses. The results of these analyses will be the subject of a future technical report.

In addition to providing an understanding of the features of the analyzed environments, the analyses will also point out strengths and weaknesses of the questionnaire itself. This information will then be used to produce corrections and revisions to the taxonomy embodied by the questionnaire. The ultimate form of the taxonomy of environment characteristics may be yet another revised questionnaire and/or some other form of taxonmic outline.

CONTENTS

PROGRAM DEVELOPMENT ENVIRONMENT QUESTIONNAIRE

0.0 INTRODUCTION AND INSTRUCTIONS

We are interested in determining the characteristics of a maximally useful and complete program development environment. Your assistance in filling out this questionnaire will be of great value to us.

For the purposes of this questionnaire, we will consider a program development environment to be all those tools and facilities which you use to create, execute, test, modify, maintain, document, and report on the programs which you write. In general, we are interested in those tools and facilities which are available in machine executable form.

The questionnaire is divided into eleven major sections. Section 1 asks some general information questions about your environment. Sections 2 through 10 then ask questions about specific major environment components. Section 11 concludes the questionnaire with some overall qualitative measurement questions. Related categories of questions within the major sections are grouped into subsections.

We have endeavored to supply useful explanatory information in each section to assist the question answering process. We have also included a glossary to define our usage of important technical terms. You may however find that you are totally unfamiliar with the terminology or nature of a question, or you simply do not know the answer. In such cases, please provide one of the following responses as appropriate: "DU" for "Do not understand the question", "DK" for "Understand the question, but do not know the answer", and "NA" for "The question is not applicable to my particular environment".

For the most part, the questions are yes/no or multiple choice format. For some questions, only one choice will be applicable to your environment. For other questions, it may be appropriate to check more than one or all of the choices. When it is not immediately obvious, the question will indicate if one or more than one choice should be checked.

A few of the questions ask for a brief written answer. We have attempted to provide sufficient space for such answers on the questionnaire itself. However, if you require more space for a written answer, please use the back of the page on which the question appears or attach a separate sheet of paper. The questionnaire numbering scheme allows each question to be uniquely identified by section, subsection(s), question number. For example, question 3 of subsection 2 of section 5 can be identified as question 5.2.3. Please use this scheme to identify written answers to questions given on separate sheets.

In some cases, questions or sections of the questionnaire may be answerable in more than one way for your environment. For example, if your environment provides both batch and interactive services, several sections of the questionnaire may require separate answers, one for the interactive facilities and another for the batch facilities. Also, you may regularly use more than one tool for a single program development activity, such as text editing or source program translation. In such cases, you may duplicate the sections of the questionnaire that apply to more than one tool or facility, and answer each duplicate section seperately. We leave to your discression the determination of when to fill out duplicate questions or sections.

Thank you for your help.

1.0 GENERAL INFORMATION

1.1 General Environment Description

There are several general terms that can help to categorized your environment. Some environments may in fact fit into more than one general category. We are interested in particular in how you as a user view your environment overall.

1. What is the name of your environment? (E.g., OS 360, UNIX, APL, LISP.) ____________________

2. Would you describe your environment as

   [ ] interactive
   [ ] batch
   [ ] remote job entry

3. Is your environment

   [ ] Operating system-based (e.g., OS 360, UNIX, MULTICS)
   [ ] Programming language-based (e.g., APL, LISP, BASIC)
   [ ] Networked (e.g., NSW, RSEXEC)

4. If your environment is not programming language-based, what programming language(s) do you use regularly?

   language   % use

   ____________   ________
   ____________   ________
   ____________   ________
   ____________   ________
   ____________   ________
   ____________   ________
   ____________   ________
   ____________   ________

1.2 Respondent Background

This subsection asks about your personal background as it relates to the use of your environment.

1. What is your position or title in connection with the work you do using your environment?

   [ ] Programmer
   [ ] Programmer/Analyst
   [ ] Analyst
   [ ] Maintainer/Tester
   [ ] Manager

[ ] Executive
[ ] Educator
[ ] Student
[ ] Other ______________

2. Approximately how much experience do you have with your environment?

_____ [ ] weeks [ ] months [ ]years

3. What is the name of the institution or firm where you use your environment?

____

1.3 Characteristics of Environment Usage

This subsection asks about how your environment is principally used by you as well as other users, if any.

1. What type(s) of software is produced in your environment? (Check all that apply.)

[ ] scientific [ ] large [ ] small
[ ] business [ ] large [ ] small
[ ] systems tools of a particular hardware vendor (e.g. compilers, operating systems)
[ ] retail packages (e.g. OEM packages, contract work)
[ ] "one-shot" software (e.g., student programs)
[ ] real time
[ ] embedded
[ ] communications
[ ] personal (e.g., home or hobby)
[ ] other ___________________________

2. For what phases of software development is your environment used? (Please number multiple choices, with 1 indicating the most frequently performed.)

[ ] Requirements Analysis
[ ] System Design
[ ] Program Design
[ ] Coding
[ ] Testing
[ ] Maintenance
[ ] Documentation
[ ] Management (of one or more of the above activities)

3. What percentage of the workload in this environment is "production" work? ______

4. Approximately how many users are currently working in this environment? ______

5. Approximately how long has this environment existed in its current configuration?

   ______ [ ]weeks [ ]months [ ]years

1.4 Environment Hardware Configuration

A completely detailed description of your environment's hardware configuration might take many pages. The outline below covers some of the important characteristics of the hardware which supports your environment.

Size or capacity of memory devices should be given in 8-bit bytes. "Number" refers to the number of units of the specified device. For "other" choices please specify: 1) generic or specific name of device, 2) size and performance in appropriate measure, and 3) number of such devices.

[ ] Central Processor

Name(s) ____

[ ] Micro
[ ] Mini
[ ] Midi
[ ] Maxi

[ ] Main Memory

[ ] Core Size ______
[ ] Solid State (e.g., MOS) Size ______
[ ] Other ____

[ ] Disk/Drum Memory

[ ] Fixed-Head Number ______ Capacity ______
[ ] Moving-Head (multi-disc) Number ______ capacity ______

[ ] Cartridge Number ______ Capacity ______
[ ] Floppy Number ______ Capacity ______
[ ] Other ____

[ ] Magnetic Tape Memory

- [ ] Full-Size (e.g., 7 or 9 track) Number ______
- [ ] Smaller (e.g., DECTAPE) Number ______
- [ ] Other ______________________________

[ ] Other Memory

- [ ] Bubble Size ______
- [ ] Laser Size ______
- [ ] Other ______________________________

[ ] Keyboard Terminal Devices

- [ ] Teletype Number ______ Baud ______
- [ ] CRT Number ______ Baud ______
- [ ] Highspeed printing Number ______ Line/min______
- [ ] Other ______________________________

[ ] Line Printers Number ______ Line/min ______

[ ] Graphics Devices

- [ ] Printer/Plotter Number ______
- [ ] Refresh CRT Number ______
- [ ] Storage Tube Number ______
- [ ] Color Number ______
- [ ] Light pen
- [ ] Mice
- [ ] Other Goodies ______________________________

[ ] Paper Products

- [ ] Card Reader Number ______ Card/min ______
- [ ] Card Punch Number ______ Card/min ______
- [ ] Paper Tape Reader Number ______ Char/sec______
- [ ] Paper Tape Punch Number ______ Char/sec______
- [ ] Other ______________________________

[ ] Communication Hardware

- [ ] Network Interfaces
- [ ] Remote Access (including RJE) Number ______
- [ ] Custom ______________________________

[ ] Other ______________________________

____

____

____

2.0 ENVIRONMENT MONITOR

This section of the questionnaire deals with what we call the Environment Monitor. We view the Monitor as that component or subsystem of the environment which handles the user's initial contact with the environment, and through which all other interactions are started. In general, the Monitor is the user's link with the other tools and facilities of the environment. An integral part of the Monitor is its user-communication language. This we call the Monitor Command Language (MCL).

You as a user may view your Environment Monitor and MCL in one or more different ways. For example, if your environment is controlled by a general-purpose operating system, your Monitor environment can be regarded as the user-level or job-control-level, and the MCL is the O.S.'s Job Control Language. If your environment is that of a particular programming language, your Monitor is the "top-level" or command-level of the language, and the MCL is the set of commands used to control program editing, execution, etc.

We have attempted to include many general questions which can be answered independently of your particular user viewpoint. As elsewhere in the questionnaire, certain of the questions may be inapplicable to your environment. If so, answer accordingly as explained in the general instructions given in the introduction. The subsections below will contain further explanatory information as appropriate. This will help you determine whether a particular subsection of questions is applicable to your environment's Monitor.

2.1 Monitor Command Language (MCL)

In many environments, the MCL is simply a set of commands that can be given to the Monitor one at a time for execution. Some environments allow commands to be grouped together to form command "procedures" or "macros" to be executed as a unit by the Monitor. Still other environments provide great flexibility by allowing command procedures to be written using high-level programming features such as conditional or iterative execution. This subsection of questions asks about which of these facilities your MCL has.

1. What is the name of your environment MCL? (E.g. OS/360 JCL, Commodore VI.) ________________

2. Would you describe the MCL as

   [ ] an integral part of the environment operating system
   [ ] a separate, independent component

3. Is the MCL syntax

   [ ] Uniform for all commands
   [ ] Similar to some programming language; if so, which
   ____

4. Which of the following "high-level" language features does the MCL have

   [ ] Variable declarations
   [ ] Type declarations
   [ ] Assignment statements
   [ ] Logical expressions
   [ ] Arithmetic expresions
   [ ] Control structures

      [ ] Branching
      [ ] Conditional
      [ ] Looping
      [ ] BEGIN-END brackets
      [ ] Iteration
      [ ] Recursion
      [ ] Other ________________________________

   [ ] Scoping Rules (i.e. How are MCL identifiers associated with other MCL identifiers or with environment elements?)

      [ ] Static (i.e. determined by position of declarations)
      [ ] Dynamic (i.e. determined by order of execution)
      [ ] Other ________________________________

5. Does the MCL have a facility for assignment of mnemonic names to environment resources (e.g. files, devices, processors)? ______

6. Do MCL commands allow parameters? If so, are they

   [ ] Positional
   [ ] Keyword
   [ ] Mixed
   [ ] Other ________________

7. Are user-defined commands allowed? (E.g. macros, procedures, shells.) If so, are they

   [ ] The same syntax as system-defined commands
   [ ] Parametizable
   [ ] Usable everywhere system-defined commands are

8. Which of the following "convenience" facilities are available in the MCL

   [ ] Command completion (e.g. verbose/quick mode)
   [ ] Spelling korection
   [ ] Monitor-level editing
   [ ] Command undo/redo

9. Are MCL parameter defaults

   [ ] System-definable
   [ ] User-definable
   [ ] Both
   [ ] Overridable
   [ ] Other ______________________________

10. Can a mode attribute (e.g. batch, interactive) be attached to MCL commands? _____

11. Can you in anyway "customize" your MCL environment? _____
    If so, can you

   [ ] Create subset MCL environments
   [ ] Create superset MCL environments
   [ ] Other _______________

2.2 Process Execution, Control, and Scheduling

The term process is used to mean the execution activity associated with a program. In general, the questions in sections 2 through 4 are aimed at processes which come from user programs. The Monitor provides various facilities to control and specify the execution of user programs. The questions in these next three subsections ask about the extent and nature of such facilities.

(See also, section 5 on Program Execution Tools.)

1. Which of the following process scheduling facilities are provided by the MCL

   [ ] Schedule after some elapsed amount of time
   [ ] Schedule at some (real time) clock time
   [ ] Schedule another process

      [ ] Concurrently (as an independent process)
      [ ] As a subprocess
      [ ] In some other scheduling discipline (specify)
      ____

[ ] Schedule as a batch process

[ ] Schedule based on some (future) environment condition (See also questions on "environment inquiry")

[ ] Other ____________________

2. Can priorities be assigned to processes? _____ If so, are they

[ ] Strictly system-assigned

[ ] User assignable

[ ] Numeric in nature (e.g. 1 - 256)

[ ] Level-oriented (e.g. system level, user level)

[ ] Algorithmically determined (e.g. shortest job first)

[ ] Changable by the user

[ ] Other ____________________

3. Can a batch job be divided into "job steps"? _____ If so, can there be

[ ] communication between contiguous steps

[ ] communication between non-contiguous steps

4. When are resources allocated for a batch job

[ ] at the beginning of the job

[ ] when requested

2.3 Process Interaction and Sharing

1. Does the MCL provide an interprocess communication facility? _____ If so, is it

[ ] Between user processes

[ ] Between user and system processes

[ ] Via shared data area

[ ] Via message passing

[ ] Via some other means ____________________

2. Is there an environment inquiry facility? _____ If so, which of the following environment information is available

[ ] System load

[ ] Current users

[ ] Resource availability (e.g. device, file, processor)

[ ] Resource usage (e.g. CPU time, number of I/O's)

[ ] Time/date

[ ] Other ____________________

3. Is the environment designed as a distributed system (e.g., communications coupled processors)? _____ If not, can it support sharing with other environments (e.g. is it in a network)? _____

2.4 Process Protection and Security

1. Does the monitor provide process protection? If so, is it

   [ ] Between user processes
   [ ] Between user and system processes
   [ ] Hierarchial (omnipotent level, high level, dud level)
   [ ] Other ______________________________

2. Does the monitor provide a "failsafe" or "failsoft" facility? (E.g. "graceful" system crashes.) _____

3. Is there an automatic restart facility after failure? (E.g. are user and/or system processes resumed where they were interrupted by the failure?) _____

4. How often does the system fail on the average? (E.g. hourly, daily, weekly.) _______________

5. How long does it stay down on the average? (E.g. hour, day, week.) _______________

6. Are user processes password protected? If so, are passwords

   [ ] user changable
   [ ] available to certain "privileged-class" processes

   (See also Section 3 on the File System for further questions concerning passwords.)

7. Is there a privilege hierarchy over MCL commands? (E.g. are certain commands available only to certain privileged processes?) If so, which of the following classification schemes are available

   [ ] Individual users
   [ ] Groups of users
   [ ] Arbitrary processes
   [ ] Other ______________________________

2.5 Tool Coordination and Control

One of the principle jobs of the Monitor is to control the use of the various environment tools. Some Monitors allow generally sequential tool invocation (i.e., one tool executes after the other). Others allow more flexible tool use, as illustrated by the questions in this subsection.

1. Can all or most tools be invoked by using only their names? (Or, for example, are tools invoked by some command such as "RUN toolname"?)

   [ ] all
   [ ] most

2. Can tools be invoked in a nested fashion? (E.g. can the text editor be invoked from a compiler which was originally invoked from the monitor level?) _____ If so,

   [ ] is such nesting generalizable for all tools
   [ ] are some tools not nestable; if so, list them

   __________ __________ __________ __________

3. Which of the following disiplines does tool nesting follow

   [ ] Stack-oriented (i.e. can only return to most recently previously used tool)
   [ ] Movable-stack-oriented (i.e., can return to any previously used tool with intermediate invocations terminated)
   [ ] Free (i.e., can return to any previously invoked tool)
   [ ] Other ______________________________

2.6 Programming Language/MCL Interface

One often views the Monitor as strictly the "top level" of the environment with which the user communicates. In addition however, user programs need to communicate directly with the Monitor. Such communication is necessary, for example, when a program needs to use MCL process scheduling or file handling commands.

1. Are all MCL facilities available at the programming-level? (I.e. can you in effect invoke all MCL commands from within a program?) _____ If not, which important MCL facilities are not available at the program level?

   ____

2. For those MCL commands which are available at the program level, is the syntax the same as at the monitor level? _____ If not, briefly describe how it differs.

   ____

3. Can values be passed from the MCL to programs? _____ If so, which types

   [ ] Numbers
   [ ] Strings
   [ ] Keyword/value pairs
   [ ] Typeless parameters
   [ ] Other ______________________________

4. Can values be passed from programs to the MCL? _____ If so, which types

   [ ] Numbers
   [ ] Strings
   [ ] Keyword/value pairs
   [ ] Typeless parameters
   [ ] Other ______________________________

2.7 Session Monitoring

A session is that period of time during which activities are performed by a computer in response to requests by a user. In a batch environment, a session is often referred to as a job.

1. Is there a facility which logs session activity (i.e. produces a complete or partial record of all I/O to and from the listing device of the session)? _____

2. Is there a "help" facility? _____ If so , is it

   [ ] Available for all MCL commands
   [ ] User updatable (i.e. can users add new help text)

3. Are there any other "programmer assisstance" facilities available? If so, please briefly describe.

   ____

   ____

2.8 Resource Management

"Resource" is used here as a general term which includes the following entities: files, devices, programs, processors, memory (possibly divided into pages and/or segments). Some environments treat some or all of these resources in a uniform manner. (E.g., the distinction between file and device is transparent to the user.) Other environments may treat each as a distinct entity.

(See sections 5 and 3 on Program Execution Tools and the File System.)

2.9 Error Reporting and Handling

1. Is the format uniform for all types of MCL messages? _____ If not, briefly describe how formats differ for different types of messages.

   ____

   ____

2. Are user-defined error routines allowable? _____

3. Is error message text user-modifiable? _____

4. Can error conditions be trapped at the program level? (E.g, are "on conditions" definable?) _____

2.10 Misc. Utilities and Facilities

1. Which of the following facilities are available

   [ ] Mail
   [ ] User-to-user messages
   [ ] User-to-operator messages
   [ ] Operator-to-user messages
   [ ] Others _______________ _______________

2. Are operator commands embedded in the MCL? _____

3. Can the right to use any (or all) of the operator commands be given to non-operator class users? _____

4. Is there an easily discernable basic/introductory set of commands? _____ If so,

   [ ] how long does it take to learn this set __________
   [ ] can the set be added to in an easy manner _____

5. How long does it take to become an "expert" in this MCL? (Provided one would want to!) __________

6. Does the generality/power of the MCL conflict with its simplicity and ease of learning? _____

3.0 FILE SYSTEM

This section of the questionnaire is concerned with the File System. The File System is that mechanism which is responsible for the long-term storage of programs, data, documents, or any other information that a user might store. Note that we are concerned with how the File System is seen from the environment in which the user works, not from the environments which are potentially accessible. (E.g., if the user environment cannot normally trigger a particular file operation "x", then "x" is not a facility available in the environment). Note also the heavy emphasis on disk technology as the basis for File Systems.

Each possible response to a question has parenthesized hints as to environments which might qualify for that response; these hints will presumably aid users who are unsure of what the proper response should be but can recognize similarites.

3.1 File Structure

User data is stored within some skeletal framework call "the File Structure". This section asks about the framework itself.

1. On what fundamental philosophy is the file system based?

   [ ] All files are randomly addressable streams of bytes (UNIX) or words (MULTICS)
   [ ] All files are randomly addressable blocks of data whose size is dependent on some physical hardware characteristic like a disk sector (PDP-10, TOPS10/20)
   [ ] All files are randomly addressable logical records (independent of any physical hardware characteristic)
   [ ] Each file is accessed by use of a system-defined access method (IBM OS360/370)
   [ ] Each file is accessed by use of user or system defined access methods (Cambridge CAPP, Carnegie-Mellon HYDRA)
   [ ] Other ______________________________

2. What high-level access methods are provided by the file system?

   [ ] Sequential
   [ ] Random access by record number
   [ ] Keyed (B-tree) or Indexed Sequential (record located by content of key field)
   [ ] Hashed (record located by content of key field)
   [ ] Others ______________________________

3. What kinds of data can be stored in files?

[ ] Text (Program sources, documentation, etc.)
[ ] Numeric values stored in a packed form (non-text) manipulable by the programming system? (floating point numbers, integers, etc)
[ ] Program objects (relocatable objects, "core" images)
[ ] Data records containing mixed data objects (e.g., text and numeric data)

4. What is the standard character set used in stored text strings?

[ ] USASCII (most non-IBM systems)
[ ] EBCDIC (IBM systems)
[ ] Hollerith
[ ] Other____

5. What attributes of a file can be obtained by the user?

[ ] File type (access method)
[ ] File size (number of records, disk storage units, etc.)
[ ] Creation date
[ ] Creation time (to within units smaller than the mean time it takes a user to create a new file, e.g., 1 minute)
[ ] Last access time and date
[ ] Last access type (read, write, etc.)
[ ] File version (as in TENEX)
[ ] Owner
[ ] Protection information (protect bits, access list, etc.)
[ ] user who performed last access
[ ] Physical location
[ ] Other____

6. What limits the size of the file?

[ ] Nothing (files can cross hardware and system boundaries, as in fully distributed file system)
[ ] Local system hardware limits (files can span multiple disk drives and/or other units)
[ ] Capacity of single hardware unit (files are limited to a single disk drive)
[ ] Software limitation on space allocated to file retrieval data (e.g., logical sector numbers are limited to 16 bits)
[ ] Other ____

7. What is an "order of magnitude" estimate of maximum file size? (** denotes exponentiation.)

[ ] infinite
[ ] 10**20 (2**64) bytes

[ ] 10**11 (2**36) bytes
[ ] 10**10 (2**32) bytes
[ ] 10**5 (2**16) bytes
[ ] Other ____

8. What is the physical unit of space allocation for a file?

[ ] Block (partial sector) : size = ________________
[ ] Sector: size = ______________________
[ ] Cluster (fixed number of sectors): size = __________
[ ] Extent: A contiguous set of sectors; max size = ___________
[ ] Disk track
[ ] Other______________________

3.2 Operations on Files

Files can be manipulated in many ways by users. Two major categories of operations are common: operations for the retrieval and modification of data within a file, and operations concerned with file as an entity, such as renaming, deleting and so on.

1. What operations can be performed on the content of files?

[ ] Read data; units = _______________
[ ] Write new data; units = ________________
[ ] Modify old data
[ ] Append
[ ] Delete record
[ ] Other operations ____________________________

2. What operations can be performed on files as entities?

[ ] Create a new file
[ ] Delete a file
[ ] Open a file for access
[ ] Name (or Rename) a file
[ ] Allocate space
[ ] Reclaim unused space/reorganize
[ ] Set protection information
[ ] Other____________________________________

3. How are files made available to programs?

[ ] Via explicit OPEN operation
[ ] Via implicit connect (CAPP, MULTICS)
[ ] Other__________________________________

4. Can files dynamically grow? _____ If not, why not?
____

5. At what level can files be shared among multiple users/processes?

[ ] No user/process interlock available at all
[ ] Exclusive access to file
[ ] Record locks are available
[ ] Other________________________________

6. What control does the programmer have over optimization of files transfers?

[ ] Can specify that system should use N buffers with a file
[ ] Can specify where buffers are in user space
[ ] Can specify size of buffers
[ ] Other________________________________

7. Are there any differences between foreground (or interactive) and background (or batch) use of files?

[ ] Entirely incompatible
[ ] no: entirely compatible
[ ] Foreground can get to subset of background
[ ] Other________________________________

3.3 File System Structure

File System Structure refers to the organization of information which keeps track of the files themselves, and typically covers such items as directories, naming conventions, etc.

1. How are files named?

[ ] Alphanumeric identification; Legal character set____________________
[ ] Other; please describe

____

2. How are files that are related designated as such?

[ ] Extension (DEC-Tops 10, .txt, .rel, etc.)
[ ] By type code (all related files have same name, different type)
[ ] By grouping into a common directory (MULTICS)

10. What limits the number of files in a directory?

[ ] Total system storage
[ ] Size of directory
[ ] System implementation limit
[ ] Other________________________________

11. Directories...

[ ] look just like files, and can be read by user programs.
[ ] are special, and only the system can read them.
[ ] are protected from ordinary users.

12. Does the file system provide ordinary device independence? That is, may the user treat files and devices in the same manner?

[ ] Yes, total device independence
[ ] No, different access methods (e.g., Telecommunications Access)
[ ] No, User must write special code.

13. If the system has device independence, how are device-specific operations performed?

[ ] Special system calls for each specific operation
[ ] Device driver watches for special data sequences in Write Data requests (UNIX)
[ ] System supports general Control and Status calls; device drivers have special entry points for these calls
[ ] Not possible
[ ] Other________________________________

14. How is access to files on other systems obtained?

[ ] Transparent networking (ARPA RSEXEC)
[ ] Limited access (transaction processing, etc.)
[ ] File is copied to local site before used
[ ] Generally not done
[ ] Other________________________________

15. How difficult is it to install special I/O devices or facilities (like a new access method) which the designers of the system had never considered? 1 means "Don't try it, there are already three people in the Rubber room", 5 means "Toyota".) _____

3.4 Security Mechanisms

This section deals with facilities for the protection of data in files.

1. What is the granularity of protection? (Check all that apply)

   [ ] Protection of user account
   [ ] Protection of individual directories
   [ ] Protection of individual files
   [ ] Protection of individual records or parts of files
   [ ] Other ______________________________

2. Access is generally limited to individuals or groups of individuals that all share some common property. Group properties that are "interesting" to the protection system include (check all that apply):

   [ ] System-defined user groups (e.g., group accounts)
   [ ] Members of user-defined access list (a la MULTICS)
   [ ] Privileged groups
   [ ] Single individuals
   [ ] Users that own keys that have been distributed (i.e. capability systems such as CMU HYDRA)
   [ ] Other ______________________________

3. How can access rights of one user be propagated to another?

   [ ] Military style security mechanism
   [ ] Capabilities (CMU HYDRA)
   [ ] User modification of access list
   [ ] User modification of file protection
   [ ] Other ___________________________

4. What file operations may be protected?

   [ ] Read data
   [ ] Write new data
   [ ] Modify data
   [ ] Copy data (does not prevent processing, simply limits data propagation)
   [ ] File deletion
   [ ] File copying
   [ ] File attribute reads
   [ ] File attribute modification
   [ ] Protection controls on file/file contents
   [ ] Other __________________________________

5. What mechanisms are used to protect data?

[ ] Protection bits (Read, Modify, Append bits as on Tops 10) Specify protection: ______________________________

[ ] Access list (MULTICS)

[ ] Matching account numbers or account prefixes

[ ] Capabilities (CAPP or CMU HYDRA)

[ ] Passwords

[ ] on individual files

[ ] on accounts

[ ] on directories

How are passwords stored?

[ ] In encrypted form in standard file

[ ] In special files accessible only to system

[ ] In clear text form in a protected file

[ ] In one-way encrypted form (a non-reversible encryption)

[ ] Operator verification

How? ______________________________

[ ] Encryption (e.g., National Bureau of Standards Data Encryption Standard algorithm)

Encryption type:

[ ] NBS DES

[ ] Public key cryptosystem

[ ] Other ______________________________

Who performs the encryption/decryption?

[ ] User program

[ ] System utility program

[ ] System read and write primitives

[ ] Electronic hardware in peripheral read/write circuits

[ ] Other ______________________________

Does unencrypted data appear in any place other than the user's address space?

[ ] no (what a nice system!)

[ ] Yes -- in buffers internal to system read them

[ ] Yes -- other ______________________________

[ ] Other ______________________________

6. How are the protection mechanisms protected?

[ ] Via one way encryption
[ ] Protection information accessible only to system primitives
[ ] Via the protection mechanisms themselves (capabilities)
[ ] Other ____________________________________

7. How well are users encapsulated?

[ ] Completely (hardware prevents privileged operations and references outside users space; system places no protection control information in user space; user must use system primitives to handle data)
[ ] Other ____________________________________

8. What facilities exist which can sidestep the protection mechanisms

[ ] Privileged user/account
[ ] Privileged programs

Can these programs be used to inspect protected data?

[ ] No (well thought out utilities)
[ ] Yes (after clever programmer subverts utility)
[ ] Yes (via Dump or Display facilities built into privileged programs
[ ] Other ____________________________

[ ] Privileged source of programs (e.g., a privileged directory, for which programs fetched and executed via that directory obtain privilege)
[ ] System operator
[ ] Other ____________________________

3.5 Media

This subsection is concerned with the kind of media on which data may be transcibed.

1. Check off the media types supported directly by the file system.

[ ] Disk/drum (rotating magnetic medium with one or more heads)
[ ] Magnetic tape
[ ] ANSI standard labels
[ ] Nonstandard labels
[ ] Multiple files per tape (DECtape)
[ ] Single file per tape

[ ] Very large storage devices (Terabit memories, storage cells, etc.) Please describe and give capacity and performance

____

____

[ ] Other magnetic media (bubble memories, etc.) Please describe and give capacity and performance

____

____

[ ] Alphanumeric displays with keyboard (CRTs)

[ ] Number of character rows __________

[ ] Number of characters wide __________

[ ] Cursor addressability

[ ] Cursor control keys on keyboard (up down left right)

[ ] Graphics displays: resolution __________

[ ] Raster scan

[ ] Electrostatic (or any graphics system capable of 30Hz or better frame generation rate)

[ ] Storage tube (Tektronix series)

[ ] Color: Number of colors ________________

[ ] Light pen graphics input

[ ] Mouse graphics input

[ ] Cross hair graphics input

[ ] Digitizing table graphics input

[ ] Other graphics input ______________

[ ] Alphanumeric keyboard associated with display device

[ ] Paper tape reader

[ ] Paper tape punch

[ ] Card reader

[ ] Card punch

[ ] Line printers

[ ] Data acquisition devices: (Digital to Analog converters, etc.) Describe

____

[ ] Communications devices networking Describe network in two words or less ______________

[ ] Communications devices for local networks

[ ] Communications for Remote Job Entry

[ ] Communications for general purpose Remote Access

[ ] Computer output microfilm

[ ] Other mass output device (laser printers, etc.) Describe ____

3.6 File Utilities

This subsection is concerned with the standard utilities available for copying, archiving, validating, printing, and performing other operations on files.

1. Check file utilities available

[ ] Copy file to file

[ ] Copy file or device to file or device (A special program or programs under systems which do not have device independence)

[ ] List file on printer

[ ] List file on console

[ ] List directory

[ ] With wildcard?

[ ] By date?

[ ] By other attributes?

[ ] List/change file attributes

[ ] Backup/restore

[ ] Compare

[ ] binary

[ ] text

[ ] language specific options

[ ] Inspect file (file dump)

[ ] Change protection on files, directories, etc.

[ ] File comparison for verification/location of differences

[ ] Other____________________________________

2. Utility programs

[ ] exist as seperately invocable user programs

[ ] are typically coalesced into a large, multi-purpose utility How are the functions partitioned?

____

[ ] exist as privileged programs

3. Archiving

[ ] is available via regular, complete file system backups

[ ] is available via automatic backup (e.g., by demon processes)

[ ] is available via explicit request by user to archive a file

[ ] uses multiple levels of storage hierarchy (files are first moved to slower secondary store, then to tape, and so on over time)

[ ] a file is invisible to user, because it is automatically retrieved when requested
[ ] requires user intervention to restore the file
[ ] requires operator intervention to restore the file
[ ] Other interesting features:____________________

4. Some systems have utility programs that can verify that the file system has not been damaged (e.g., by power failure, runaway program, system bugs, etc.).

[ ] A utility exists to detect damage in the file system
[ ] File system damage is detected by manual methods (e.g., when the system acts funny, damage is verified via a file system inspect utility or guesswork.)
[ ] System warns users of damaged files when encountered
[ ] File system damage is repaired by manual methods (e.g., in "superprogrammer" mode)
[ ] File system damage is repaired automatically by system (and user is simply told that it happened)
[ ] File system damage is repaired via special utility
[ ] File system damage is not ever repaired. (E.g., new file system is prepared and all recoverable files are copied or are restored from archive.)

3.7 Human Engineering in the File System

Many file systems have features whose only real utility is making life more convenient for the user (although some might argue that the mere existence of computers is what makes life hard for the user). This section is intended to determine what human engineering features have been installed in your system.

1. How much does a user have to know in order to use the file system in his environment?

[ ] User knowledge level, 1=low, 5=high

2. Automatic features include:

[ ] completion of unique filenames
[ ] spelling correction on filenames
[ ] version numbering
[ ] Other ______________________________________

3. How robust is the file system? (Give a scale number 1-5 indicating how much user data is lost on the average, 1 means "file data is lost if you sneeze in the same room".)

[ ] with respect to hardware failures?
[ ] with respect to software failures?
[ ] over time with no apparent failures

4. Rate the performance of the file system. (1 means "slower than a snail" and 5 means "violates physical law concerning speed of propagation of light".) ______

5. What does the file system do well?

____

6. What does the file system do poorly?

____

7. If you could make everything lightning fast, what feature would you use more heavily?

____

(See also Section 11 for further qualitative measures of the File System.)

3.8 Performance and Hardware Support

This section collects data concerning the resources needed to implement the file system used in your environment, and the performance obtained.

1. How much does the required hardware cost?

[ ] Under $1,000
[ ] $1,000-$5,000
[ ] $5,000-$10,000
[ ] $10,000-$30,000
[ ] $30,000-$100,000
[ ] $100,000-$500,000
[ ] $500,000-$2,000,000
[ ] The sky's the limit

2. What are the limitations of the number and/or amount of secondary storage that can be handled by the file system? (E.g., max number of peripherals = , can only handle floppies, etc.)

____
____

3. What special or unusual hardware is required by the file system?

____

____

4. Concerning main store requirements for file system operation:

[ ] Amount of buffer space per user required (units of K bytes) _____
[ ] Amount of main storage required by file system itself _____
[ ] Cutoff point below which file system is impractical to use _____

5. What is the mean time to...

[ ] access a directory (i.e., open a file)
[ ] read the next record/block
[ ] position to a new place in a file and read a record
[ ] write a new record

6. Please describe any other performance measure you consider especially appropriate for file systems.

____

____

4.0 TEXT PROCESSING TOOLS

4.1 Defining the Environment

This first subsection asks for your definition of the term "text". For example, text may be characters, strings, program source, documentation, data, or some other type of file used in your environment.

1. What is the name of your text processing tool?

____

2. What is meant by the term "text" for this tool?

[ ] Characters
[ ] Strings
[ ] Data in memory
[ ] Data on bulk storage devices
[ ] Data structures or program contexts (e.g. LISP)
[ ] Other ______________________________

3. What type of files can you edit?

[ ] Program source
[ ] Documentation
[ ] Documentation extracted from program source files
[ ] Object files
[ ] Core image files
[ ] Data files (e.g., containing floating point values)
[ ] Other______________________________

4. Are tool commands included with the text? (E.g., RUNOFF.) _____

Questions 5 through 9 concern the media used by the text management tool. For your answers, use the abbreviations given in the following list:

D = Disk
F = Floppy Disk
MT = Magnetic Tape
CT = Cassette Tape
CR = Card Reader
PT = Paper tape
ALL = All devices supported by the file system.
For others give device name and short abbreviation:

____

____

5. From what types of external media can this tool read text?

____

6. From what types of external media can this tool read text?

7. From what media can the tool accept user input? __________

8. Onto what media can the tool write text? (Also indicate the speed of the device, as for example CR (80 card/min).)

____

____

9. Onto what types of media can the tool write user output? (Again, indicate the speed of the device.)

____

____

10. Is the user allowed to "tab" (either using a TAB key or a predefined character which represents a tab)? _____

11. Are there non-printing characters which may affect the input or display of text (e.g. begin/end underline)? _____

12. Can the tool access other tools or environment facilities? _____ If so, which?

____

13. Does this tool have special features that relate to a specific language (e.g., syntax checking or structure editing)? _____ If so,

[ ] Which language? ____________________
[ ] Can the tool be useful outside of that language? _____

14. If the tool is designed to be interactive, can it also be used in batch mode (e.g., by specifying some command file)? _____ If so, is the command structure the same as for interactive? _____

4.2 User Interface

The questions in this subsection deal with the your perception of the text management tool. This perspective includes, but is not limited to: the device, the form and frequency of the dialogue between the user and the tool, and the representation of the text to the user during the processing session.

4.2.1 Batch Processing Tools -

1. Is the command file for the tool generated on some offline media such as cards or paper tape? _____ If not, briefly explain how it is generated.

   ____

   ____

2. Is there a log produced which shows the commands or operations performed? _____

3. Is there a log which reflects the result of the changes made? _____

4. At the end of the run, are statistics generated which indicate the number of changes made, the current file size, etc? _____

4.2.2 Interactive Tools (using Typewriter Or CRT) -

For this subsection, check all choices that apply; you may check more than one choice for a single question.

1. Is this tool driven by

   [ ] Keystroke (one or a few)
   [ ] Command (longer mnemonic)

2. Is the tool

   [ ] Passive (does not prompt for input)
   [ ] Prompting

3. Is the mode (i.e. passive or active) switchable under user control? _____

4. Is there some line edit mode where either text or commands can be edited by control keys or cursor functions which copy or preserve the old information while making the changes indicated (e.g., TYMSHARE)? _____

5. Can the previous command be re-executed by a single

   [ ] Keystroke
   [ ] Command

6. Is the user allowed to set "tabs" and sequentially reference them (either using a TAB key or some predefined character)? _____

7. If it is a predefined character, can the user choose that character? _____

8. Is there a data entry mode where there is a keystroke validation of data within fields (such as alpha vs. numeric)? _____

9. Can the tool be set up to automatically echo the changed area after every change? _____

10. Can statistics regarding the current editing session be displayed (i.e. number of lines or characters in file, number of buffers, etc.)? _____

4.2.3 Interactive CRT Tools -

1. Is the CRT

   [ ] restricted sequential line output
   [ ] two-dimensionally addressable
   [ ] Storage type
   [ ] Incrementally writeable

2. Is the updated text dynamically displayed? _____

3. Can the granularity (e.g. page size) of the displayed text be altered by the user? _____

4. Is the representation of the displayed text the same as the printed output (i.e., does the display have the same line endings)? _____

5. Can portions of two or more files be displayed at the same time? _____

4.3 Commands

We are interested in finding out what commands your text processor has. For each of the following functions, you should indicate:

1. The parameters and their granularity (e.g. character, line, word, expression, number, location expression (see subsection 4); indicate all that apply).

2. The granularity of the range in which the function may be applied (line, page, entire source text, etc.).

3. The effort required to give this command (e.g. 1 keystroke, a few keystrokes, a whole mini-program).

4. The usefulness of this command for your processor (not needed, nice but rarely used, indispensable).

5. Whether the cursor (the tool's focus of attention) is altered by this command.

Not all these functions may be meaningful for your text processor. (E.g., display makes no sense for RUNOFF, delete makes no sense for DDT).

For your answers, use the abbreviations given in the following menu:
Granularity of parameters
C = Character
L = Line
W = Word
E = Expression
N = Number
LS = Location Specification
Granularity of application
C = Character
L = Line
P = Page
E = Entire source file
Ease of use
1 = 1 Keystroke
F = A Few Keystrokes
P = A Whole mini-program
Usefulness
1 = Not Needed
2 = Nice but rarely used
3 = Handy
4 = Frequently used
5 = Indispensable
Tool's Attention
C = Is changed
N = Remains the in the same place

6. Can you insert text?
Granularity of parameters _____
Granularity of application _____
Ease of use _____
Usefulness _____
Tool's attention _____

7. Can you replace text?
Granularity of parameters _____
Granularity of application _____
Ease of use _____

Usefulness _____
Tool's attention _____

8. Can you delete text?
Granularity of parameters _____
Granularity of application _____
Ease of use _____
Usefulness _____
Tool's attention _____

9. Can you copy text from one place to another?
Granularity of parameters _____
Granularity of application _____
Ease of use _____
Usefulness _____
Tool's attention _____

10. Can you move (copy then delete) text from one place to another?
Granularity of parameters _____
Granularity of application _____
Ease of use _____
Usefulness _____
Tool's attention _____

11. Can you display text?
Granularity of parameters _____
Granularity of application _____
Ease of use _____
Usefulness _____
Tool's attention _____

12. Can you search for text forwards?
Granularity of parameters _____
Granularity of application _____
Ease of use _____
Usefulness _____
Tool's attention _____

13. Can you search for text backwards?
Granularity of parameters _____
Granularity of application _____
Ease of use _____
Usefulness _____
Tool's attention _____

14. If you can search for text, how complex can the search pattern be?

[ ] Fixed character string only
[ ] Wild cards
[ ] Matching with binding of variables
[ ] Full access to the power of the surrounding environment (e.g., can write a search algorithm in some available programming language)

15. Can you retrieve or create auxiliary text while processing the main text?
    Granularity of parameters _____
    Granularity of application _____
    Ease of use _____
    Usefulness _____
    Tool's attention _____

16. Can you access more than one text at once (e.g. multiple windows, file-merges, etc.)?
    Granularity of parameters _____
    Granularity of application _____
    Ease of use _____
    Usefulness _____
    Tool's attention _____

17. Can you edit two or more texts in parallel?
    Granularity of parameters _____
    Granularity of application _____
    Ease of use _____
    Usefulness _____
    Tool's attention _____

18. Can you change more than one text within a single "session"? If so, what information is carried over (parameters, buffers)?
    Granularity of parameters _____
    Granularity of application _____
    Ease of use _____
    Usefulness _____
    Tool's attention _____

19. What information (if any) is remembered between whole editing sessions?

    ____

    ____

4.4 Location Specification

In order to specify that part of the text on which you wish to perform some function (e.g. display, delete) there must be a way to specify locations.

1. Are numbers in any way associated with locations? _____ If so:

    1. Are the numbers attached to the text? _____

    2. What is the numbering convention (e.g. increments)?

[ ] 1
[ ] 10
[ ] 100
[ ] 1000
[ ] Other ____________________

3. Can the convention be changed? _____

4. What happens to the numbers between sessions?

[ ] They remain the same
[ ] They are resequenced
[ ] They are not saved between sessions

2. Is there a cursor or focus of attention of the tool (visual or virtual)? _____ If so:

1. Is it visual or virtual?

[ ] Visual
[ ] Virtual
[ ] Visual is the same as virtual

2. Can you move it directly? _____

3. If so, in what units?

[ ] Characters
[ ] Words
[ ] Lines
[ ] Tab zones
[ ] Pages
[ ] Other ___________

4. Is the cursor "at" or "between" locations?

[ ] At
[ ] Between

5. Can you use the cursor to specify "segments" of text as a parameter to some function (e.g. as a means of specifying location for cutting and pasting)? _____ If so, in what units?

[ ] Characters
[ ] Words
[ ] Lines
[ ] Pages
[ ] Other (specify)________________________________

4.5 Extensibility

In some text processors, the command language can be extended; new commands can be added. If your system permits this, please answer the following questions.

1. Can you create a new command by chaining other commands together? _____

2. Can the new commands be named (i.e. recalled later in a simple manner)? _____

3. Let's call this new command a program. How complex is this new programming language? (Check all that apply.)

   [ ] It can loop
   [ ] Test and branch
   [ ] Bind variables
   [ ] Recur
   [ ] Other ____________________________________

4. Are there functions you can access from within a program that you cannot access by the standard set of commands? _____

5. Can the program itself be considered editable text? _____

6. Can you edit the program locally (without leaving the current text)? _____

7. Can the program modify other editing programs? _____

8. Can it modify itself? _____

9. Can you alter existing commands? _____ If so, briefly describe how any restrictions apply.

   ____

   ____

4.6 Interface with the External Environment

This section considers how well the text processor interacts with the environment within which it resides. Specifically, what interactions with the file system are possible, what interactions with other utilities are possible, and in some cases, how are these interactions accomplished.

1. Does the tool interface with the file system? _____

2. If not, briefly describe how the results of an editing session are saved from one session to the next

____

____

3. If so, which of the following capabilities are used?

[ ] Saving and restoring files
[ ] Other ______________________________

4. Are the commands to access the file system consistent with the other commands available from the within the tool?

______

5. What other environment tools or facilities (if any) can be accessed from within the text processing tool?

____

____

6. Can a file be updated in place or is a modified version produced?

[ ] in place
[ ] new version

7. If copies of the file are made, does the tool

[ ] generate copies as it goes
[ ] make the entire copy before proceeding

8. Does the file system permit the tool to move forward and backward arbitrary distances through the file or are some limits imposed?

[ ] Can't move at all
[ ] Limits imposed
[ ] Arbitrary

9. Is there any time penalty for backing up? ______

10. Does the tool automatically update document control parameters such as version number, revision number, etc?

______

4.7 Output Processing

Some text processors have a form of output procesing. In some systems, these capabilities may be part of the same tool which is used for editing program text. In other cases, it may be a seperate program which is fed a file prepared by another tool which has embedded commands to direct the processing of the text (e.g., RUNOFF).

1. Which of the following features does your text processor have?

   [ ] Automatic table of contents preparation
   [ ] Pagination (i.e. heading, footing, page-numbering, etc.)
   [ ] Forms data processing (i.e. it puts a form on the screen and you fill in the blanks)
   [ ] Merging of text and sequential files
   [ ] Others ______________________________

   ____
   ____

2. Does the tool have "pretty-printing" capabilities? _____

3. Can the display be directed to devices other than the normal display? (E.g., if the display normally goes to the screen of a CRT, can it be directed to a printer?) _____

4. What effect do embedded TAB characters have on output processing on the various devices to which output is normally sent?

   [ ] No effect
   [ ] They tab to the desired column
   [ ] They mess everything up

5. Are there any other "funny" non-printing characters which have an effect on your environment? ______ If so, enumerate them and briefly describe their effect.

   ____
   ____

4.8 Safety Features

This subsection deals with the capabilities provided for saving the user from his/her own error, from the unwitting error of others, from system crashes, etc.

1. What type of "help" facility exists?

[ ] None
[ ] Manuals
[ ] Single Key
[ ] File accessible by the tool

2. Is there any way to back out the last n changes made? ______

3. Is there a "key verify" mode? (I.e., a second user rekeys the same text, but instead of being entered, it is simply compared to the text which was entered the first time?)

______

4. Can the user specify that a file be automatically saved after

[ ] some specified time period
[ ] some specified number of text changes
[ ] other____

5. If the system crashes, how much do you lose?

[ ] Nothing
[ ] Everything since you last saved the file
[ ] Everything
[ ] Other ______________________________________

4.9 Error Checking

1. Does the tool do

[ ] syntax checking or parsing
[ ] spelling checks on documents
[ ] other error checking _______________________

____

5.0 PROGRAM EXECUTION TOOLS

This section of the questionnaire addresses the topic of tools available to determine the characteristics of a program during execution. These tools include debuggers, error checking facilities, and resource monitoring and allocation. We do not include in these execution tools program libraries and other databases that add functionality to programs. The main goal of execution tools is to give the user knowledge and control of what his program is doing and the capability to alter it quickly and easily. In this way the user can interact with the execution of his program usually in the debugging stage.

This section is divided into two subsections, one dealing with debugging facilities associated with languages and a second dealing with facilities associated with the environment monitor. Included with the section on programming languages are object code debugging tools that may be machine dependent. We encourage you to complete this questionnaire for each seperate environment with which you are familiar. For example, if you use several programming languages fill out the section on language dependent features for each.

5.1 Debugging Tools For Specific Programming Languages

This subsection deals with the tools used in debugging programs written in a specific language. These tools often have special features allowing the user to debug in the source code instead of a different language (i.e., usually lower level machine code). Question 5 refers to machine dependent features (i.e., object code) while question 4 refers to source dependent features. Answer the appropriate question and then continue this subsection at question 5. If no such debugging tool exists for a particular environment then answer question one appropriately and answer only the value judgement parts (explained below) of the following questions.

1. What is the name of the programming language with which this debugging tool is used? ____________________

2. Is the debugging facility specifically tailored to this language? ______

   Each of the following questions is broken up into two responses. The first bracketed box should be answered 'Y' or 'N' as to whether or not that particular feature exists in your environment. In the second box put a value judgement as to the usefulness of this particular feature. Use the scale of 5 being very useful to 1 being not useful at all.

3. How are the debugging features implemented (if there is more than one method, please identify each)

[Y or N] [1-5]
[ ] [ ] Loaded with the object program
[ ] [ ] Part of a language system (e.g., APL or LISP)
[ ] [ ] A parent process fork of the program
[ ] [ ] Embedded in the operating system
[ ] [ ] ROM resident
[ ] [ ] Implemented by a separate processor using DMA, interrupts, etc.
[ ] [ ] Other

4. Below is a list of source dependent features (i.e., for data objects which must be referenced by their source name rather than by machine address). Mark each as above, with Y or N answer followed by 1-5 answer.

[Y or N] [1-5]
[ ] [ ] Dump of all data objects
[ ] [ ] Snapshot of selected data objects
[ ] [ ] Trace of execution
[ ] [ ] Set break points
[ ] [ ] Break whenever a particular data object is read
[ ] [ ] Break whenever a particular data object is modified
[ ] [ ] Resume execution at the point of a break
[ ] [ ] Examine source code
[ ] [ ] Modify source temporarily
[ ] [ ] Modify source permanently
[ ] [ ] Modify data objects
[ ] [ ] Produce a traceback of the current execution state
[ ] [ ] Produce a traceback of nested dynamic environments
[ ] [ ] Change the context of the debugger to a different environment (static or dynamic)
[ ] [ ] Return to a previous environment and continue execution
[ ] [ ] Single step execution
[ ] [ ] Monitor concurrent processes
[ ] [ ] Set conditional breakpoints
[ ] [ ] Insert debugging routines at break or trace points
[ ] [ ] Automatically get control after an execution time error
[ ] [ ] Search for a particular data object by value
[ ] [ ] Define error control pathways
[ ] [ ] Evaluate expressions of the source language while retaining control (e.g. BASIC immediate mode or LISP break EVAL command)
[ ] [ ] Other

____

___

5. Each of the features below is machine dependent. Although these features are generally less useful than those in question 4, it is nevertheless common to debug systems written in higher level languages with the use of core dumps and object code modification.

Indicate which of the following are present on your system. Answer this question with the same format as question 3.

[ ] [ ] Dump of main memory
[ ] [ ] Snapshot of selected portions of main memory
[ ] [ ] Set break points in object code
[ ] [ ] Trap all read references to a portion of main memory
[ ] [ ] Trap all write references to a portion of main memory
[ ] [ ] Modify object code
[ ] [ ] Single instruction execution
[ ] [ ] Get control after an execution time error
[ ] [ ] Search main memory for a particular value
[ ] [ ] Other ____

___

6. Would a debugger with all of the features marked 1 in the previous question form a minimal, but still useful debugger? _____ Why or why not?

____

____

7. How is the source referenced for purposes of setting breaks and evaluating expressions?

[ ] Line number
[ ] Search string
[ ] Block
[ ] Label
[ ] Other __________

8. How are data items displayed?

[ ] In the notation of the language, automatically
[ ] In one or more formats specified by the user
[ ] In octal or hexidecimal
[ ] Other ___________

9. Can the debugger be called by other tools? ______ If so, briefly describe the extent of information (i.e., the location and context of an error) that is passed to the debugger.

____

____

10. Can the debugger call other tools? ______ If so, what kind of information does it pass to the other tools?

____

____

5.2 Execution in the Monitor Environment

This subsection deals with features of the monitor, operating system, or command processor that enable the user to obtain information concerning a process's (job's) resources, execution time, etc.

1. How is program execution controled? (If there is more than one way (for example batch and interaction), repeat this section for each.)

   [ ] Batch control deck in a JCL-like language
   [ ] Batch control deck with a syntax similar to a programming language
   [ ] Commands on an interactive terminal
   [ ] Commands on an interactive terminal plus packaged commands which may be defined by the user (MIC, UNIX shell)
   [ ] Controlled by a parent process
   [ ] Other __________

2. In what environment do programs normally execute?

   [ ] Linked in core load including user and library routines
   [ ] As in above but with a runtime system
   [ ] Within a language system such as LISP or APL
   [ ] Part of the operating system
   [ ] Other __________

3. Are the following available to the user

   [ ] Initiate execution
   [ ] Stop execution
   [ ] Continue execution
   [ ] Initiate execution of a concurrent process
   [ ] Control a family of processes directly

[ ] Control a tree of processes indirectly through a root process
[ ] Submit a batch job for execution
[ ] Run a job in the background
[ ] Other __________

Questions 4 though 10 refer to the control of resources other than those provided by the file system. In a batch system, 'the user' would refer to the control deck.

4. Does the user have control over core allocation? _____ If so, how?

   [ ] Selection of a fixed sized partition
   [ ] Selection of an initial core size which may be increased by the program
   [ ] On virtual memory systems, specification of the working set size and maximum address space size
   [ ] Other __________

5. How does the user allocate magnetic tapes to the job?

   [ ] By requesting all drives before program execution
   [ ] The program requests them automatically
   [ ] Other ____________

6. What other devices may be allocated to a job at execution time and how is each requested?

   Device   Method of request

   ___________   _________________
   ___________   _________________
   ___________   _________________
   ___________   _________________

7. Which devices are not available to jobs but are spooled automatically?

   Device

   ____
   ____
   ____
   ____

8. How does the user connect the language device identifier to a physical input/output device?

   ____
   ____

9. Does the user have the ability to specify device characteristics at execution time? _____ If so, how and what characteristics?

____

____

10. What limits on execution can the user set?

[ ] CPU
[ ] Memory
[ ] Usage of devices (e.g., page limits, etc.)
[ ] Other ________________

11. Which of the following features are available at execution time? Are they user selectable? (Y or N)

Available	Selectable	
[ ]	[ ]	Array bounds
[ ]	[ ]	Invalid data detection
[ ]	[ ]	Exec. count of stmts etc.
[ ]	[ ]	Other __________

12. List below where each of the features mentioned in the previous question resides.
Debugger ____________________________________
O.S. ____________________________________
Translator ____________________________________

13. Which of the following are available to the user at execution time?

[ ] Display process status
[ ] Time of day
[ ] CPU Usage
[ ] Kilo-core seconds
[ ] Average memory usage
[ ] Disk reads/writes
[ ] State (queue) of job
[ ] Other device usage statistics __________
[ ] Other ___________

6.0 PROGRAM ANALYSIS TOOLS

Software reliability has become increasingly difficult to achieve as software systems become more complex. Ad hoc debugging techniques fail to detect potential errors while brute force testing practices require inordinate amounts of computer time. These traditional approaches have become ineffective, infeasible, and intolerable. Hence, the need for program analysis tools has arisen.

6.1 Language Considerations

A given set of program analysis tools must be associated with a particular programming language.

1. What is the name of the most predominantly used programming language in your environment?

   [ ] Pascal
   [ ] COBOL
   [ ] FORTRAN
   [ ] BASIC
   [ ] ALGOL
   [ ] LISP
   [ ] APL
   [ ] RPG
   [ ] PL/1
   [ ] C
   [ ] Other ______________________________

2. Does your environment provide for static and/or dynamic analysis of programs written in this language? _____

6.2 Static Analysis Tools

Static analysis tools examine some representation of a program and derive information about the program without performing the action the program specifies.

1. Some powerful metatools which can considerably ease the burden of constructing static analysis tools include parser generators, program-directed graph generators, and data flow graph generators. Does your environment support any of these metatools? _____ If so, indicate their names and briefly elaborate on their use.

2. Indicate the static analysis techniques used in your environment (use "C" if the tool is integrated in the compiler; use "S" if the tool can be separately executed).

Then proceed to answer further corresponding questions.

[ ] Syntax checking (2.1)
[ ] Semantics checking (2.2)
[ ] Cross referencing (2.3)
[ ] Completion analysis checking (2.4)
[ ] Dangerous construct checking (2.5)
[ ] Inter-routine parameter checking (2.6)
[ ] Coding standard enforcement (2.7)
[ ] Symbolic assembly language inclusion (2.8)
[ ] Variable map inclusion (2.9)
[ ] Software Science measures (2.10)
[ ] Cyclomatic complexity measures (2.11)
[ ] Reachability complexity measures (2.12)

1. Does your syntax checker

[ ] provide meaningful error messages
[ ] attempt correction of errors

[ ] automatically
[ ] interactively with the user

[ ] halt after detecting "too many" errors

[ ] how many is "too many"? __________

[ ] optionally transfer control to the editor

2. Does your semantics checker

[ ] provide meaningful error messages
[ ] attempt correction of errors

[ ] automatically
[ ] interactively with the user
[ ] halt after detecting "too many" errors

[ ] how many is "too many"? __________

[ ] optionally transfer control to the editor

3. Does your cross referencing tool flag occurrences of

[ ] variable declaration, read and write references
[ ] routine declaration and references
[ ] list of routines called by a given routine
[ ] list of routines which call a given routine
[ ] summary of routine block numbers
[ ] summary of routine nesting levels
[ ] summary of structured statement nesting levels

4. Which of the following completion analysis checks are made?

[ ] improper loop nestings
[ ] unreferenced labels
[ ] unreferenced data
[ ] unreferenced routines
[ ] unreachable statements
[ ] statements with no successors

5. Which of the following dangerous construct checks are made?

[ ] loops incremented by zero
[ ] undefined branch target labels
[ ] transfers to inside a loop
[ ] loops with untested parameters
[ ] hazardous case statements (e.g., no OTHERS clause)
[ ] missing tag fields in variant records
[ ] excessive GOTOs

6. Does your parameter checking tool check, in the declaration and invocation of a routine, for

[ ] equal number of parameters
[ ] correspondence of pass-by-value and pass-by-reference parameters
[ ] compatibility of types of parameters

7. Do your coding standards

[ ] disallow the dangerous constructs previously listed (Question 2.5)
[ ] vary according to the level of expertise of individual programmers
[ ] require that module size remain small
[ ] limit the nesting level of routines and structured statements

8. Do your merged symbolic assembly language statements include

[ ] relative program counter values
[ ] sizes of the corresponding statements
[ ] source line number of the associated higher level language construct

9. Does your variable map include

[ ] the address (fixed or stack relative) of each variable
[ ] the size of each variable
[ ] indication of whether each variable is directly addressed (local or passed by value parameter) or indirectly addressed (passed by reference parameter)

10. Which of the following Software Science measures are recorded?

[ ] number of different operators
[ ] number of different operands
[ ] total of different operators and operands
[ ] number of operator usages
[ ] number of operand usages
[ ] total occurrences of operators and operands
[ ] program volume (number of addressable units to contain it)
[ ] level of language used, in terms of operators and operands required
[ ] internal quality, in terms of volume and level

11. Do you restrict the lower bound of the cyclomatic complexity interval (the number of independent tests necessary to exercise the outcome of each path at least once) to a managable number? ______

12. Do you use the reachability complexity measure to determine how many places the correct "program state" must be set up before coming through a given line of code? ______

6.3 Dynamic Analysis Tools

Dynamic analysis techniques are those which derive information about a program by examining it during its execution.

1. In order to perform dynamic analysis of a program, compile time options must be used which trigger the insertion of runtime checking code. Indicate which options are recognized by your compiler. Then proceed to answer further corresponding questions.

[ ] Range checking (3.1)
[ ] Variant checking (3.2)
[ ] Pointer value checking (3.3)
[ ] I/O checking (3.4)
[ ] Assertion tests (3.5)

[ ] Traceback ability (3.6)
[ ] Performance probes (3.7)
[ ] Completeness probes (3.8)

1. Is range checking performed for

   [ ] arrays
   [ ] subranges
   [ ] set element expressions
   [ ] over/underflow on expression evaluation

2. Does the variant checking ensure that references to variant parts of records are consistent with the values of their tag fields? ______

3. Does the pointer value checking compare the pointer value to NIL when the pointer is used to access data? ______

4. In case of an unsuccessful I/O operation, does the I/O checking

   [ ] cause the program to be terminated
   [ ] retry the operation in an attempt to get valid data

5. Do you have the capability to insert assertions in programs? ______

6. Which of the following are included in the object code when the traceback option is enabled?

   [ ] routine name
   [ ] absolute block number
   [ ] static nesting level

7. What does your performance probe handler keep track of?

   [ ] number of access to variables
   [ ] number of executions of each routine
   [ ] number of executions of main program
   [ ] number of iterations of each loop

8. Does your completeness probe handler keep track of a usage of paths in the following control structures?

   [ ] IF/THEN/ELSE
   [ ] multi-way branches (CASE)
   [ ] REPEAT loops
   [ ] WHILE loops
   [ ] Other ______________________________

6.4 Existing Systems for Program Analysis

In an effort to avoid the horrendous costs associated with traditional methods of debugging and testing, several software evaluation systems have surfaced in the past few years.

1. Does your environment have access to any of the following software evaluation systems?

[ ] PET Program Evaluator and Tester

[ ] FACES Fortran Automatic Code Evaluation System

[ ] STS Software Testing System

[ ] ACES Automatic Code Evaluation System

[ ] CPA Complexity Path Analyzer

[ ] PACE Product Assurance Confidence Evaluator

[ ] Other ______________________________

7.0 PROGRAM TRANSFORMATION TOOLS

This section of the questionnaire deals with Program Transformation tools. These are tools that take as primary input and produce as primary output representations of computer programs. These tools can be characterized by the form of their input and output representations, i.e., their source and object languages. We have categorized languages as belonging to one of three groups:

1. Metalanguages -- these languages define other languages or actions on languages. Examples of such translators are macro-processors, compiler-compilers and tools for syntax directed translation.

2. Problem Level Languages -- these languages are used to express the problem to be solved at the level of the problem. Included in this category are assembly language, FORTRAN, Ada, data flow languages, decision tables, and so forth.

3. Conceptual Machine Languages -- these languages express the problem to be solved at the level of the execution agent (concrete machine, simulator, interpreter). Included in this category are machine languages, reverse polish notation and the intermediate forms of a program in a multi-pass compiler.

Some translation tools, especially most modern compilers, are composite translators, that is, they are implemented as a series of transformations. A single compiler may contain, for example, translators:

1. from a metalanguage to the problem language (macro processing),

2. from the problem language to an internal problem language (syntax and semantic checking),

3. from the internal problem language to another (or the same) internal problem language (target independent optimization),

4. from the internal problem language to an internal machine language (object generation and target dependent optimization), and

5. from the internal machine language to the language of a concrete machine.

7.1 Metalanguage Translators

Please answer the following questions about the tools in your system that process metalanguage descriptions.

1. List the tools in your system that perform macro-processing as all or a part of their function. If the processor is geared to a specific object language, indicate that language (i.e., many assemblers).

Tool Name   Object Language

____________________   ______________________________
____________________   ______________________________
____________________   ______________________________
____________________   ______________________________
____________________   ______________________________

2. List the tools in your system that are used to aid in synthesizing translators. Briefly characterize the services provided by the tool (e.g., scanner generator, parser generator, compiler compiler, etc.)

Tool Name   Service

____________________   ______________________________
____________________   ______________________________
____________________   ______________________________

3. List any other metalanguage processors in your system. Briefly characterize the main function of each tool.

Tool Name   Function

____________________   ______________________________
____________________   ______________________________
____________________   ______________________________

7.2 Problem Language Translators

Please answer the following questions about the tools in your system that process problem level language descriptions (i.e., "source code").

1. List the problem-level-to-conceptual-machine-level translation tools (e.g., most compilers and assemblers) provided by your system. For each, list the source language and object agent (machine, simulator, etc.) used.

Tool Name   Source   Object

____________________   ____________   ____________
____________________   ____________   ____________
____________________   ____________   ____________
____________________   ____________   ____________
____________________   ____________   ____________

2. List the tools that translate from one problem level language to another problem level language (e.g. FORTRAN to Pascal, FORTRAN to assembly language). For each, list the source and object languages. Indicate if object language statements can be intermixed with source statements in the input to the tool (e.g., intermixed FORTRAN and assembly statements).

Tool Name	Source	Object	Mix?
____________________	____________	____________	____
____________________	____________	____________	____
____________________	____________	____________	____
____________________	____________	____________	____
____________________	____________	____________	____

3. List any tools that translate problem level modules to modified modules in the same language.

   1. Source level optimizers

      ____

   2. Error seeding tools (translators that introduce source level errors into a module in order to evaluate the effectiveness of developed test cases)

      ____

   3. Instrumentation tools (translators that add new statements to permit tracing, collect execution information, etc.)

      ____

   List any other tools of this sort in your system and briefly characterize their functions.

Tool Name	Function
____________________	________________________________
____________________	________________________________
____________________	________________________________

4. List any other problem level translation tools provided by your system. Briefly describe the main function of each tool.

Tool Name	Function
____________________	___________________________
____________________	___________________________
____________________	___________________________

7.3 Conceptual Machine Language Translators

Please answer the following questions about the tools in your system that process conceptual machine language descriptions (e.g., machine code, interpreter code, etc.)

1. List the integration tools available in your system (e.g., linker). For each, list the execution agent for the output of the tool (actual machine and operating system, etc.) and any special features of the tool (e.g., Can a linker build overlays? Can a user specify the organization of the overlay structure?).

Tool Name　　　　Execution Agent/Feature

____________________　　____________________________

____________________　　____________________________

____________________　　____________________________

____________________　　____________________________

2. List the any machine language modification tools (e.g. patching tools) in your system. List also the machine language involved.

Tool Name　　　　Machine Language

____________________　　____________________________

____________________　　____________________________

____________________　　____________________________

3. List any optimizers in your system. If the optimization process is a part of an encompassing tool, please indicate that tool. List optimizing criteria used (e.g., time, space, etc.).

Tool Name　　　　Enc. Tool　　　　Criteria

____________________　　_______________　　_____________

____________________　　_______________　　_____________

____________________　　_______________　　_____________

4. List any tools that attempt to convert a conceptual machine language input to a problem level language (e.g. disassemblers or decompilers). List the source and object languages.

Tool Name　　　　Source　　　　Object

____________________　　____________　　____________

____________________　　____________　　____________

____________________　　____________　　____________

5. List and briefly describe the function of any other conceptual machine language transformation tools.

Tool Name　　　　Function

____________________　　__________________________

____________________　　__________________________

____________________　　__________________________

8.0 MAINTENANCE, TESTING, AND DOCUMENTATION

8.1 Maintenance

The area of maintenance can be viewed as having three distinct subcatagories:

REPAIR - The correction of faulty code or procedures.
ADAPTATION - Modification to accomodate a changing system environment.
ENHANCEMENT - The addition of new capabilities.

1. Approximately what percentage of your maintenance resources is spent on each of these subcatagories ?

   REPAIR ______ %
   ADAPTATION ______ %
   ENHANCEMENT ______ %

8.1.1 Repair -

1. Number the tools used in maintenance according to the frequency with which they are used in repair:

   [ ] Page-oriented text editor
   [ ] Line-oriented text editor
   [ ] Character-oriented text editor
   [ ] Debbuger
   [ ] The File System
   [ ] Other(s) __________ __________ __________

2. Does your environment provide formatted output listings (pretty print) of source code? _____

3. Is a program debugger available in your environment? _____

4. Does your environment provide programming language syntax checking? _____ If so, is it

   [ ] part of a compiler
   [ ] part of an interpreter
   [ ] independently available
   [ ] other ________________________________

5. Does your environment provide performance statistics to facilitate the isolation of errors? _____ Which of the following kinds of information do they provide?

[ ] Time used
[ ] Number of executions
[ ] Others ________ ________ ________

6. What level of documentation is applied, following repair?

[ ] New release
[ ] Updating of current documentation
[ ] Notes on current documentation
[ ] None
[ ] Other ________

8.1.2 Adaptation -

1. Do you feel your environment is open-ended (adapted to new and changing resources and procedures)? _____ If so, please describe which of the following adaptation changes you have encountered

[ ] New or additional computer
[ ] Additional hardware resources
[ ] Change from single user to time-sharing
[ ] Other ________________________________

2. Does your environment facilitate transportation of programs to other machines? _____

8.1.3 Enhancement -

1. Does your environment provide software version controls to facilitate recovery to an earlier version? _____ If so, at which level:

[ ] System level
[ ] Program level
[ ] Module level
[ ] Other _________

2. How do you handle requests for program changes in your environment?

[ ] Telephone
[ ] Personal contact
[ ] Notes
[ ] Formal request forms

[ ] Other

3. What levels of documentation are modified due to software enhancement?

[ ] Reqirements documentation
[ ] Design documentation
[ ] User documentation
[ ] Other

4. Does your environment provide general purpose software libraries?

[ ] Mathematical/Statistical
[ ] Data management
[ ] Utility
[ ] Other(s) __________ __________ __________

5. Are new or enhanced libraries easy to incorporate into existing programs? _____

6. Do all tools which are normally used for program maintenance support version control? _____

8.2 Testing

1. Is testing accomplished after each

[ ] Repair
[ ] Adaptation
[ ] Enhancement

2. Does your environment provide test data generation? _____
If so, is the data for

[ ] System testing
[ ] Critical program path
[ ] Module testing
[ ] Other form of testing __________

3. Does your design process generate required test cases? _____ If so, can this test data be accessed at program execution time? _____

4. Does your environment contain test input/output libraries? _____

5. Can test runs with output comparisons be made easily? _____

6. Is there a testing group at your environment? _____

7. Does your environment provide operational diagnostics?
Hardware _____ Software _____

8. Does your environment include an execution failure analyzer? _____

8.3 Documentation

1. Which of the following types of documentation are produced in your environment?

[ ] Requirements Specification
[ ] Data Flow Diagrams
[ ] Program Structure Charts
[ ] Data Structure Diagrams
[ ] Data Dictionary
[ ] System Flow Charts
[ ] Program Flow Charts
[ ] Source Code Listings
[ ] Other

2. Is the required documentation sufficient? _____

3. What parts of the documentation are most useful for maintenance?

____

____

4. Does your environment provide easy access to needed documentation (e.g., source listings, design, requirements specifications, etc.)? _____ Are there version controls on this information? _____

5. Does your environment support graphical documentation (e.g., program structure charts, data flow diagrams, data structure diagrams, etc.)? _____ Can this type of documentation be easily edited? _____

6. Can portions of documentation from one life-cycle phase be easily associated with corresponding portions from another life cycle phase (e.g., module source code and corresponding functional design specifications)? _____

9.0 MANAGEMENT SUPPORT TOOLS

This section of the questionnaire collects information about those tools in your programming environment that support the management of programming projects: their organization and planning, their monitoring and control, and their evaluation and analysis. Since many of these areas are just beginning to be supported by automated tools, this section also asks about manual methods and techniques in use and, in each phase of management, asks about your need for tools, whether they currently exist or not. Identification of tools should include the name of the tool where possible and a brief description.

9.1 Organization and Planning

This subsection collects information concerning tools in your programming environment which support the organization and planning activities of management. These activities are generally distinguished by coming before the start of the implementation phase of program development. Many of the possible tools which support more detailed planning derive their inputs from the design phase of program development. The first four parts of the subsection will inventory which tools, both manual and automated, are now in use in your environment and the last part will assess your needs for these and other organization and planning tools.

9.2 Schedule Tools

1. Schedule development - Identify any tools in your environment which support the creation of overall project schedules. In general, overall schedules must be derived from the composite of the development schedules for the design, coding, unit testing, integration testing, and acceptance testing of each major configuration item (i.e., deliverable software item) such as operational computer program, system exercise software, utility software, etc. In addition, project schedules reflect tradeoffs between schedule length and risk (there are limits to how rapidly activities like coding the utility software configuration item can acquire new people effectively) and between schedule length and manpower utilization (manpower peaks and valleys may result if tasks are not scheduled so that personnel coming off one activity are used to staff up another.) Which of the tools you have identified to help in composing schedules are automated?

2. Critical path analysis - Identify any tools in your environment which support schedule creation and monitoring by analyzing networks of task dependencies. Examples of automated tools include packages like PERT and CPM.

Identify which tools are automated.

3. Implementation plans - These indicate who does what and when in detail. Identify any tools in your programming environment which support the production of implementation plans. Which, if any, are automated?

4. Test plans - These indicate the order in which modules will be unit tested and outline the specific tests which will constitute integration and acceptance testing. Unit test plans use information from the system design to identify modules which constitute primitives used by other modules and which must be tested first. The order of testing remaining modules is derived from knowledge of their functional dependencies. Identify any tools in your programming environment which support the production of test plans. Which, if ny, any, are automated?

5. Modeling and simulation - High risk areas in the design (from a performance standpoint) can be revealed by simulation of the target system. Additional resources can then be scheduled for the testing and development of those areas. Identify any modeling or simulation tools which are used for schedule development purposes. Which are automated?

9.2.1 Staffing Tools -

1. Organization planning - Identify tools which aid in planning and reporting the structure of the organization which will develop the software. For example, are there any tools which help to produce organization charts (i.e., charts which show the reporting hierarchy of individuals within the organization.) Which of these tools are automated?

2. Manpower forecasting - This involves forecasting the number of people needed at each stage in the software development project life. The forecast may, for example, show the changes in required staffing levels from month to month or from week to week. Identify any tools or methods in your environment which support this forecasting. Which are automated?

3. Staffing plans - These plans indicate where the forecasted number of people on the project will come from. How many will be hired, trained, transferred. What level of experience is required? Employee experience files are one example of a tool which could support making these plans. Identify any tools in your environment which support staffing plans. Which, if any, are automated?

4. Resource acquisition - Management must plan the acquisition of programming resources to support the number of programmers indicated by the staffing plan. For example, enough offices must be acquired and terminals must be ordered in advance to allow the programmers to be productive. Identify any tools in your environment which aid this process. Which are automated?

9.3 Accounting Tools

1. Work breakdown structure development - Management must devise a structure for recording costs during the project which will support historical cost data collection and satisfy government reporting requirements. Identify any tools in your environment which support the establishment and description of such a structure. Which are automated?

2. Budget development - Detailed budgets are created to indicate how the money will be spent which has been allocated for the project's operation. What tools in your environment support such budget development. Which are automated?

9.3.1 Tools to Support Methodologies and Project Disciplines -

1. Standards development - Each project develops standards regarding design representation and coding practices, among other things. Identify any tools which help develop these standards and disseminate them. Which tools are automated?

9.3.2 Need Assessment for Organization and Planning Tools -

Each of the organization and planning areas mentioned above are listed below. Please assign priority numbers to each area according to the importance you attach to having an automated tool to support the area in your environment. Assign a 1 to the most important area, a 2 to the next most important, and so on. Use the blank lines to write in and assign priorities to any areas not covered.

[ ] Schedule Development
[ ] Critical Path Analysis
[ ] Implementation Plans
[ ] Test Plans
[ ] Modeling and Simulation
[ ] Organization Planning

[ ] Manpower Forecasting
[ ] Staffing Plans
[ ] Resource Acquisition
[ ] Work Breakdown Structure Development
[ ] Budget Development
[ ] Standards Development
[ ] ______________________________________
[ ] ______________________________________

9.4 Monitoring and Control

This subsection collects information concerning tools in your programming environment which support the monitoring and control activities of management. Typically, these activities take place during the implementation phase of program development. The first five parts of the subsection inventory the tools, manual and automatic, now in use in your environment and the last part assesses your needs for these and other monitoring and control tools.

9.4.1 Status Reporting Tools -

1. Milestone status - Tools to support this area aid in displaying what milestones (for example, design, code, test, etc.) have been completed for each component. Typically, graphic means are employed. Identify the tools used for this purpose in your environment and whether they are automated.

2. Action item reporting - Any project has action items or problems to be resolved. These are assigned to a responsible individual for resolution by an assigned date. Reporting tools might show which items deal with a specific subject or which items are overdue for resolution. Identify any tools present in your environment for handling action items and whether the tools are automated.

9.4.2 Tools for Comparisons with Plans -

1. Performance to schedule status - Typically, graphic devices such as Gantt charts are used to display the status of a project's tasks versus their planned completion dates. Identify what tools or devices are used to compare progress against schedules in your environment. Are there automated tools for their production?

2. Rate charting - In rate charts, two lines are graphed. One shows the work units (e.g. modules) planned for completion versus time since enception of the project. The other line graphed shows the actual number of work units completed. Identify any tools used in your environment to collect and display rate information. Are they automated?

9.4.3 Accounting Tools -

1. Labor cost collection - Identify any tools which aid in the capture and accumulation of labor costs expended on the project's various tasks. Are they automated?

2. Machine cost collection - Identify any tools which aid in the capture and accumulation of computer resource costs billed to the project's various tasks. Are they automated?

3. Work breakdown structure reporting - Identify tools or techniques in your environment which report or display the costs expended to date in each element of the work breakdown structure which has been established for the project. Summaries may be available for various levels. Which such tools are automated?

4. Cost to budget reporting - These tools compare expenditures to date against budgeted expenditures for various tasks. The comparison may be reported in terms of actual figures or in other terms such as earned value versus percent expended. Identify tools used in your environment and whether they are automated.

9.4.4 Design Monitoring -

1. Resource monitoring - In embedded computer systems, budgets for program size and timing are often established and allocated to each module in a core load or to each step in a transaction's processing. Identify any tools used in your environment to compare planned versus actual portions of these budgets used up by the portion of the system implemented to date.

9.4.5 Project Resource Monitoring -

1. Hardware usage reporting - Examples of this include reporting terminal utilization to see if enough terminals are available or whether terminal or computer usage should be staggered to allow better utilization. Measures of response time or turnaround time may also be reported. Identify tools in your environment that report hardware usage. To what extent are they automated?

9.4.6 Need Assessment for Monitoring and Control Tools -

1. Each of the monitoring and control areas mentioned above are listed below. Please assign priority numbers to each according to the importance you attach to having an automated tool to support the area in your environment. Assign a 1 to the most important area, a 2 to the next most important, and so on. Use the blank lines to write in and assign priorities to any areas not covered.

[ ] Milestone Status
[ ] Action Item Reporting
[ ] Performance to Schedule Status
[ ] Rate Charting
[ ] Labor Cost Collection
[ ] Machine Cost Collection
[ ] Work Breakdown Structure Reporting
[ ] Cost to Budget Reporting
[ ] Resource Monitoring
[ ] Hardware Usage Reporting
[ ] ____
[ ] ____

9.5 Evaluation and Analysis

This subsection collects information concerning tools in your programming environment which support the evaluation and analysis activities of management. Typically, these activities take place after the completion of the implementation phase of program development. They use the results of coding and testing as their input.

1. Historical data collection - This data includes module costs, sizes, and timing. The data is collected to aid future planning and estimating for systems with similar functions. Identify what tools in your environment support this collection. Are they automated?

2. Software reliability analysis - This analysis is based on records of program bugs discovered and their nature and cause. The goal is to reduce the number of bugs and improve software reliability in released systems. Identify the tools in your environment, if any, which support this collection and analysis. Are they automated?

3. Test effectiveness - This kind of evaluation uses records of numbers of bugs discovered after testing was successfully completed to evaluate the effectiveness of the tests to which the software was subjected. Identify any tools which support this data collection and evaluation. Are they automated?

4. Standards updating - Identify tools which collect information on the performance of various design, coding, and testing standards employed and which aid in updating the standards on the basis of this evaluation for future projects' use. Are any of these tools automated?

5. Personnel performance evaluation - Identify tools which use the performance and quality of the software produced to derive statistics which aid in the performance evaluation of the people who produced it. Are these tools or techniques automated?

6. Configuration management - This takes place after initial testing during the revision and maintenance phases. It consists of two kinds of activities: configuration control and configuration reporting. Control means restricting changes to the current version or to module status to those persons authorized to make such changes. Reporting includes listing what the versions of the various modules are which make up the current system release and also includes the selection of the correct versions of each module to make up a system release. What mechanisms support these activities and to what extent are they automated?

7. Need assessment for evaluation and analysis tools Each of the evaluation and analysis areas mentioned above are listed below. Please assign priority numbers to each according to the importance you attach to having an automated tool to suport the area in your environment. Assign a 1 to the most important area, a 2 to the next most important, and so on. Use the blank lines to write in and assign priorities to any areas not covered.

   [ ] Historical Data Collection
   [ ] Software Reliability Analysis
   [ ] Test Effectiveness
   [ ] Standards Updating
   [ ] Personnel Performance Evaluation
   [ ] Configuration Management
   [ ] ____
   [ ] ____

10.0 ENVIRONMENT CONTROL AND STANDARDIZATION

10.1 Environment Change and Evolution

1. Would you characterize your environment as

   [ ] stable
   [ ] unstable
   [ ] frozen
   [ ] evolving

2. How are repairs made to the environment? (I.e., changes made to correct errors in existing components of the environment. Check all that apply.)

   [ ] Object patch
   [ ] Source code correction
   [ ] By systems programming staff

   [ ] By non-systems users (e.g., yourself or other users)
   [ ] Controlled by some form of configuration control group
   [ ] Suggested by users or users' groups
   [ ] Other means or controls ____________________

   ____

3. How are enhancements made to the environment? (I.e., changes made to add new features to the environment. Check all that apply.)

   [ ] Not made
   [ ] Object patch
   [ ] Source code change
   [ ] By systems programming staff
   [ ] By non-systems users (e.g., yourself or other users)
   [ ] Controlled by some form of configuration control group
   [ ] Suggested by users or users' groups
   [ ] Other means or controls ____________________

   ____

10.2 Environment Documentation and Education

1. Is the documentation on your environment and its facilities

   [ ] complete
   [ ] understandable
   [ ] accessible

2. Which tools or facilities, if any, lack adequate documentation and how could it be improved?

3. What percentage of environment documentation is available directly from the environment (e.g., in the form of system help files)? ______ %

4. Did you receive any formal training in the use of your environment? _____

5. If so, was it useful and/or how could it be improved?

6. Does the environment itself provide formal training facilities (e.g., in the form of user-accessible training files and/or user-executable training procedures)? _____

7. If you did not receive any formal training, would formal training have been useful to you? _____ Briefly describe how you learned to use your environment if it was not through formal training.

10.3 Environment Performance Measurement and Monitoring

1. Which of the following forms of performance monitoring facilities are available in your environment?

   [ ] Online displays
   [ ] Internal measurement facilities

      [ ] Event counting
      [ ] Trace
      [ ] Snapshots
      [ ] Other ____________________________________

   [ ] Periodic status reports

      [ ] Hourly
      [ ] Daily
      [ ] Weekly
      [ ] Monthly
      [ ] Specifiable

[ ] Other ____

2. Does your environment have a diagnostic testing facility to aid error location and repair? _____ If so, is it

   [ ] Online (i.e., environment can continue to serve users while diagnostics are being performed)

   [ ] Offline (i.e., no users can be serviced while diagnostics are being performed)

10.4 Environment Transportation

1. Is your environment specifically designed to be transportable to more than one machine? _____

11.0 RATING THE TOTAL ENVIRONMENT

In this section you are asked to rate the major divisions of your environment (i.e., those included as Sections in this questionnaire) plus your overall environment. Each is to be rated on all the goals listed in Subsection 9.2. The numbers assigned to those goals have been used to label the rows of the matrix in Subsection 9.1; the columns are labelled with the questionnaire section numbers plus one for the "overall" environment.

For each position (i,j) in the matrix you should enter a value between 1 and 5 indicating your assessment of how fully goal "i" is met by that part of your environment described in Section "j". Exceptions: the last column refers to the total environment, and the first row is explained in the next paragraph. The values have this range of meanings:

1 = completely misses the goal
3 = adequately meets the goal
5 = completely meets the goal

The first row requires some explanation. In each column you should assign a value between 1 and 5 which reflects a combination of your familiarity with the area, your expertise in the area, and the extent to which you represent the majority's view of that area. (Good luck!) The values have this range of meanings:

1 = marginally qualified
3 = adequately qualified
5 = fully qualified

We fully recognize that we have not presented a formula, or even a procedure, for calculating these values; at this point we are willing to accept subjective input. Furthermore, we do not expect you to use any (universally recognized) type of algebra to combine the individual ratings on a given row into the overall rating on that row; just "assign" an "appropriate" value.

By filling in this matrix you will be helping us to construct a relation between the tools in an environment and a most general set of high-level goals. Once such a relation is established, it is our hope that the inverse of this relation will provide a means of specifying/designing environments given a tailored set of high-level goals.

11.1 The Technical Issues Goals Matrix (TIGMAT)

TECHNICAL ISSUES GOALS

	SECTIONS									
your qual.	2	3	4	5	6	7	8	9	10	over-all
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										
20										

11.2 The Technical Issues Goals

These are the high-level desiderata for a programming environment. Each goal is represented by a correspondingly numbered row of the TIGMAT in Subsection 9.1.

1. SCOPE: provides a program development and maintenance environment which aids and supports projects of all sizes.

2. SIMPLICITY: the structure is based on simple overall concepts.

3. LOW RISK: does not go beyond the current state of the art in any avoidable respect.

4. SUPPORTIVE: provides a helpful interface to the user, and is easy to learn and use.

5. INTEGRATED: provides a well-coordinated set of (useful) tools with well-defined nesting rules/conventions.

6. OPEN-ENDED: adaptable to improvements, updates, and changes, and facilitates the development and integration of new tools.

7. PORTABILITY OF TOOLS: the tools are written in a single, high-level language and distributed in a "standard library".

8. PORTABILITY OF PROJECTS: the tools facilitate the moving of a project from one host machine to another.

9. EFFICIENCY: there is ample evidence of efforts to locally and globally minimize the use of resources by the tools.

10. UNIFORMITY OF PROTOCOL: communications between the users, the tools, and the data base conform to uniform protocol conventions.

11. BATCH SUPPORT.

12. INTERACTIVE SUPPORT.

13. DESCRIPTIVE EQUALITY: the environment itself does not forbid users from writing and using tools drawing on any capability used by other tools in the environment.

14. IMPLEMENTABILITY: the environment is easily implemented.

15. RELIABILITY.

16. MAINTAINABILITY: easily maintained by "local" personnel.

17. TRANSPARENCY: able to observe/monitor internal activity.

18. TUNEABILITY: able to predictably change the behavior of the tool(s).

19. APPEAL TO THE CURRENT USER.

20. APPEAL TO THE POTENTIAL USER.