Quick Prototyping Using JAM* for the Customized FT-2 Project

Man Keung Yeung
Tanjore K. Srinivas

A customized version of the Flow Through Data Manager II, called FT-2, was designed by AT&T Bell Laboratories to automate many of the operations and interfaces of different operations systems at Pacific Bell Telephone Co. A prototype system was designed using a commercial software package called JAM,* which allowed customers to evaluate the operational impact of FT-2, while the final system was still under development.

The prototype was developed in five weeks, six months before the final system was available for trial at the customer's site. The JAM software package supports both MS-DOS* and UNIX* operating systems with portable source codes, allowing the majority of the prototype codes to be reused in the final product. To make the system easy to use, we identified distinct users of the system, analyzed their job functions, used their terminology in the prototype, organized the information to fit their jobs, and made them feel comfortable using the system.

Introduction

Traditionally, systems engineers first find out from customers what they need from a system, and then discuss the requirements of the proposed system with the customer, using viewgraphs and text documents for support. Sometimes it is difficult to convey with these media the complete picture of what the system can do.

Efforts to improve this situation have included the use of structured analysis and object-oriented analysis (OOA) to make presentations more precise or more familiar to the customer. These approaches have produced only mixed results, because their conventions are not intuitive in themselves and, therefore, customers don't relate well to them.

The most direct and most powerful approach is to show what the system looks like and how it behaves in response to the customer's interaction—by developing a front-end prototype. By *front end*, we mean that part of the system that the user sees, via the *user interface*. The front end consists of the computer terminal screen, the various forms (or windows) on the screen, data fields in the forms (a data field is a point on a screen where data can be entered or viewed), help prompts, pop-up menus, and any other graphical or textual devices that help in using the system.

There are two disadvantages that have prevented prototypes from being more widely used so far:

- It is too time consuming. The prototype may not be developed in time because the front end depends heavily on the internal program and data structure of the system.
- It is too costly. The cost of developing the prototype is high, and normally it cannot be reused in the real system because of changes in the program or data structure, a consequence of the front end being driven by the internal system structure.

Both of these disadvantages can be avoided with proper planning.

This paper describes an approach to using prototyping that clarifies and augments textual instructions by having the prototype interact with the customer. To do so, and still be able to reuse the results in the final system, one must give special attention to the architecture:

- Design the front end to be independent of the data structure, or *back end* that supports it (the back end is that portion of the system which the user doesn't actually see that provides user functionalities), and
- Design a clear interface between the front-end and the back-end data structure, so that changes to one will not affect the other.

Included in this paper is a report of the experience with using this quick prototyping technique in defining some of the requirements of Flow Through Data Manager II (FTDM), a work manager system linking operating systems, when it was customized as FT-2 for Pacific Bell Telephone Co. Experience showed that as the result of this prototype, the customer's understanding and expectations of the final system were much clearer. In addition, most of the screens and menus developed for the prototype were reused in the final system.

Panel 1. Abbreviations, Acronyms, and Terms

ANS	Access Network Systems
ASCII	American Standard Code for Information Interchange
BaseWorx™	UNIX*-based software platform on which to create, run, and manage applications programs. Supports re-use of software and tools.
DKAP	Datakit® Applications Program
FS	Facilities System, an operations system
FT-2	Customized version of FTDM
FTDM	Flow Through Data Manager
GIS	Global Information Systems (formerly NCR)
HMI	Human-machine interface
INFORMIX*	Database management system
JAM*	JYACC Applications Manager
JYACC	Vendor of JAM
LMOS	Loop Maintenance Operations System
MC	Maintenance center
MS-DOS*	Operating system
ORACLE*	Database management system
OOA	Object oriented analysis
OS	Operations system
PCO	Provisioning control office
SARTS	Switched Access Remote Testing System
SQL	Structured query language
UNIX*	Operating system
VCS	Virtual circuit switch

Quick Prototyping Technology

To be used for customer communication and system definition, any prototype must:
- Be developed quickly and cost effectively;
- Mimic the final system realistically;
- Be modified quickly and easily to reflect customer input;
- Be developed independently of the final system, so that it is available before all system requirements are finalized; and
- Produce software modules that are reusable in the final system.

Based on the above criteria, the technology of quick prototyping depends on an architecture where:
- The front-end of the system can be developed and modified quickly;
- The front-end screens are independent of the internal database and program structure; and
- System functionalities can be mimicked quickly.

Each of these technologies is discussed further in the following sections.

Quick Screen Authoring Package. The basic technology package, or software tool, used to develop the prototype must be able to quickly provide a variety of menus and screens, yet make the human interaction with the system easy and convenient.

Software system front ends may be graphic-based or character-based. Figure 1 shows an example of a character-based system. Since FT-2 uses character-based terminals, the prototype is character-based. However, the approach can be applied to graphical user interface systems as well.

Character-based user interfaces have become quite mature and sophisticated in the MS-DOS* world. Any package that provides the following features can produce easy-to-use front ends:
- Horizontal menu bars;
- Automatic pull-down submenus;
- Pop-up data entry and query windows for field data;
- Cross-referencing of key field data that appear in different windows;
- Context-sensitive help messages;
- Pop-up selection lists at the field level;
- Dynamic construction of selection lists;

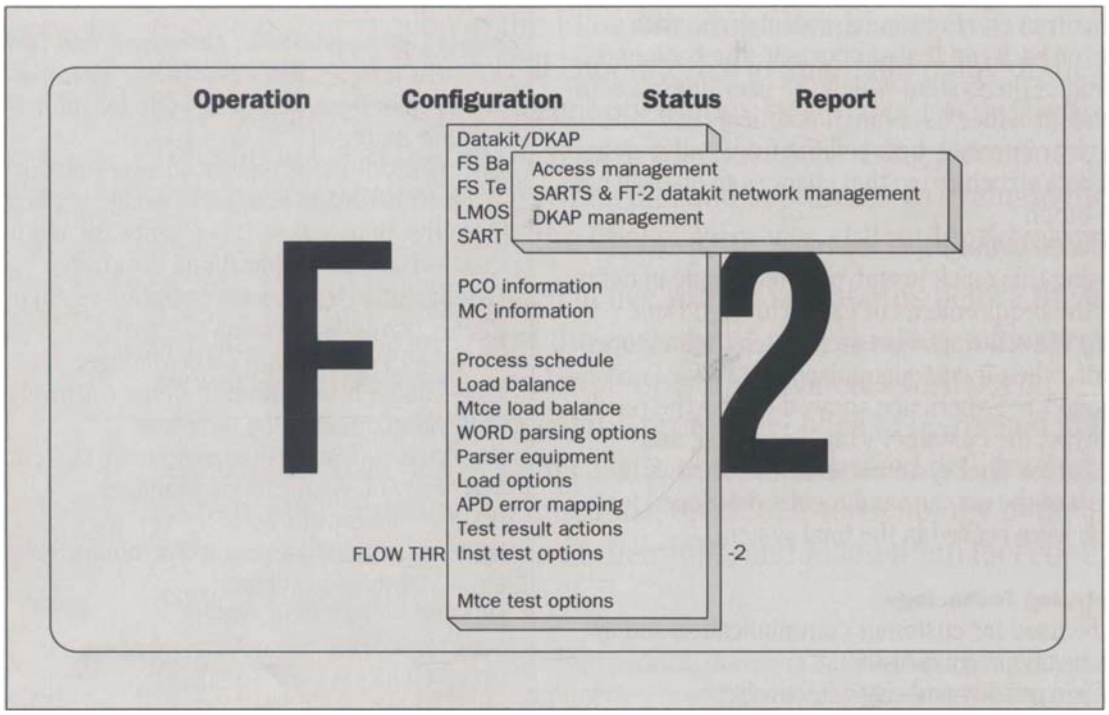


Figure 1. Software system front ends may be graphic-based or character-based. Since FT-2 uses character-based terminals, the prototype shown here is character-based. However, the approach used in prototyping FT-2 can be applied to graphical user interface systems as well.

- Action buttons;
- Easy physical-to-logical key translation, such as always using the "Alt" plus "F1" keys to access help, regardless of where the user is in the system;
- The ability to formulate and execute common operations systems (OS) commands;
- The ability to check and validate the data fields;
- Vertical scrolling to view lengthily records; and
- Horizontal scrolling to view large records, such as data tables.

In addition, it is helpful if the screen package can be run on a variety of software platforms, including MS-DOS and UNIX* operating systems, and hardware, such as 3B2 computers, and products from Sun Microsystems, AT&T Global Information Systems, Hewlett-Packard Corporation, etc.

Making the Screen and Database Independent.
Normally, the most convenient way of developing screens is to have the screen fields reflect the way the data is organized in the *back end*. An example would be the screen fields that correspond exactly to the table fields in the database. Thus, when using this approach, screens usually are defined after the data schema is defined.

This approach presents a severe handicap, however, in developing quick prototypes, as well as in developing easy-to-use systems. For a system to be easy to use, the front end should be organized according to how the user does the job, not how the system is organized.

On the other hand, the data schema is defined to facilitate program development and is, therefore, a function of the program structure. A more logical approach to front-end design is to consider the human-machine interface and the database structure to be *independent*. At first, this sounds both very inconvenient and effort intensive. An analysis shows, however, that the interface and data structures are actually two separate technology domains, addressing two separate concerns of the system. The additional effort required to develop user access that is independent of the database will pay off in the long run.

The advantage of this independence for front-end prototyping is obvious. For example, at one point, FTDM needed to be ported to co-reside with the Switched Access Remote Testing System (SARTS), an operations system that checks trunks via the local office switch. The ORACLE* data management system in FTDM had to be replaced with the INFORMIX* data management system, which is used in SARTS. Due to a well planned FTDM front end that was independent of the database structure, the modifications were localized, and the front end was not affected at all.

Another advantage of separating the human-machine interface from the database structure is that any customization required for different customers can be accommodated with changes to the screen forms and their access functions—without impacting the data structure. The key to ensuring the independence between the screens and the database structures is to define and implement an *interface* between the screen and the data structure. A properly defined and implemented interface will ensure that the system will be flexible and reliable. Either the use of structured analysis or object-oriented analysis is recommended for this effort.

Advantages of Canned Functionality. Any prototype must mimic the behavior of the final system. In response to a user's input, the prototype system has to invoke a certain function or process, such as producing a report or displaying status information.

Not all system responses are immediate, however. Such functions as data download and data lookup require a certain amount of perceptible processing time to complete. These delays can be replicated in a prototype by placing ASCII files among various built-in time delay software. These delays, and the messages advising the user of what is happening during the delay, can be implemented quickly and easily in place of the real functions or processes. These are called "canned functions."

It may sound as though canned functions are throw-away modules. Just the opposite is true. Some of these canned functions actually can be used as test cases for unit testing other parts of the system. It is very useful, for example, to demonstrate to our customers how reports are generated and what they look like. In our prototype, these reports are stored in ASCII files in the proper format. On request during the prototype demonstration, the appropriate report file is generated by a simple operations systems command to print out the file. Included in the process, then, is the canned functionality of providing a short delay as the system "processes" the command, as well as presenting the user with a message when the job is sent to the printer and also when the print job is completed.

Reusing the Prototype. An important consideration is how many of the prototype screen modules and canned functions, excluding, of course, the artificial delays, are reusable in the final system. The answer to maximizing the degree of reuse is, to repeat, dependent upon the quality of the analysis at the beginning of system design. We emphasize, again, the value of using either structured analysis or object-oriented analysis to define the user interface. Through such a careful analysis, we managed to reuse basically all of the screen modules with little or no modification in the final system.

With this simple approach, the FT-2 prototype satisfied the need to communicate the system features in a clear, cost-effective, and timely manner.

Package Selection

FT-2 runs on a 3B2/622 processor under the UNIX system, and the FT-2 front end has to support two terminals, the AT&T 4425 and 630. The screen package, therefore, not only had to support both terminals, but the key convention and screen forms had to be identical for the system to provide a consistent look and feel for the user.

Since FT-2 is an extension of FTDM, which is built on the ORACLE* system, the screen package also had to support that system. Ideally, it should support other database management systems as well.

There were two screen package candidates:
- ORACLE SQLForm (structured query language), SQLMenu, and Terminal package set, and
- The JYACC Application Manager (JAM*) package sold by JYACC, Inc.

Both packages support the ORACLE system and are available in 3B2/UNIX and MS-DOS operating systems.

JAM was selected because:
- It exceeded the criteria of the human-machine interface (HMI) requirements.
- It is easy to install. The installation process is simple and straightforward. The package comes in three flop-

py diskettes. There are only three parameters that need to be set in the "autoexec.bat" file. Only one parameter, which specifies the display screen type, may require a selection other than "default."
- The system is easy to learn. After a two-hour session, the main screen and top menu bar with pull-down menus were working with a couple of data-entry windows. Within two days, all the windows and menus for FT-2 had been designed.
- It is easy to use. The development and testing modes are so well integrated that development is extremely easy. Window and menu setup is done by drawing on the screen. What you see is what you get.
- Its architecture is well thought out. The JAM package includes a development tool for creating user menus and screens quickly and easily; a powerful fourth generation user interface language called JPL (for JYACC procedural language); an internal data block that maps into the fields of any screen, allowing cross referencing and complex data manipulations; optional database interfaces for all popular database management system products, including ORACLE and INFORMIX, and an interface with C program language functions or OS commands. The screen forms are ASCII-based and can be converted between MS-DOS and UNIX without modifications.
- The system seems robust.
- There is no run-time license fee.
- It is a recommended HMI package for the BaseWorX™ software development package.

Some of the Possible Screen Navigations. The following are the features supported by JAM software, including some examples of it's use:
- Bar menus;
- Pull-down menus;
- Screen- and field-level context-sensitive messages;
- Pop-up menus, such as the "Datakit® DKAP" menu;
- Both screen- and field-level context-sensitive help;
- Additional help buttons for special topics in a help window, as in the "Primary Dial String" field;
- Scrolling fields, as in "PCO Information" and "MC Information;'
- Scrolling help windows;
- Pop-up windows, as in the "Employee Code" field of the "MCO" Information" screen, or in the "LMOS Connection" screen;
- Pop-up selection lists, such as "Report Selection;"
- Dynamically generated, cross-referenced lists for selecting MCs, SARTS, and LMOS databases and printers;
- Coordinated information transfer, as in the "LMOS Connection" screen;
- Hidden entry for password, as in the "LMOS Connection" screen;
- Dynamic window-entry creation, as in "MC List" creation at the "MC" field of the "Mtce Test Options" screen;
- Restrictive and non-restrictive multiple selection windows, such as the "Printer Status" screen (restrictive) and the "MC" field of the "Mtce Test Options" screen (non-restrictive); and
- Quick initial setup with copying from existing entries or with defaults, such as in "MC Information."

The Menu Hierarchy

The JAM package allows the development of a menu hierarchy that is friendly to the user because it eliminates any dependence on "behind-the-scenes" technical details that are irrelevant to the user's job.

Previously, the FTDM menu was oriented toward the system architecture and used technical system terminology. For example, FTDM descriptions, such as "Request Limit Table," "amc management," and "sddmgmt," had nothing to do with the user's job and, thus, weren't part of the user's work vocabulary. These terms were carried over, in fact, from the development process.

In addition, the menu structure was very confusing. The user found that getting at the right menu entry to perform a particular job required several attempts down different menu paths. A lot of screen forms also were fragmented. Some screen forms had very few fields, so that several forms were required to perform a single function. This made the system difficult to use.

Criteria for a User-Friendly Menu Structure. A new menu hierarchy had to be designed so that the user would find it easy to use, by:
- Identifying the user;
- Organizing information according to the user's job functions;
- Employing the user's terminology; and
- Making menu entries orthogonal, that is, making every entry independent of every other entry. Thus, activities or information that fall under a particular entry will not appear under any other entry.

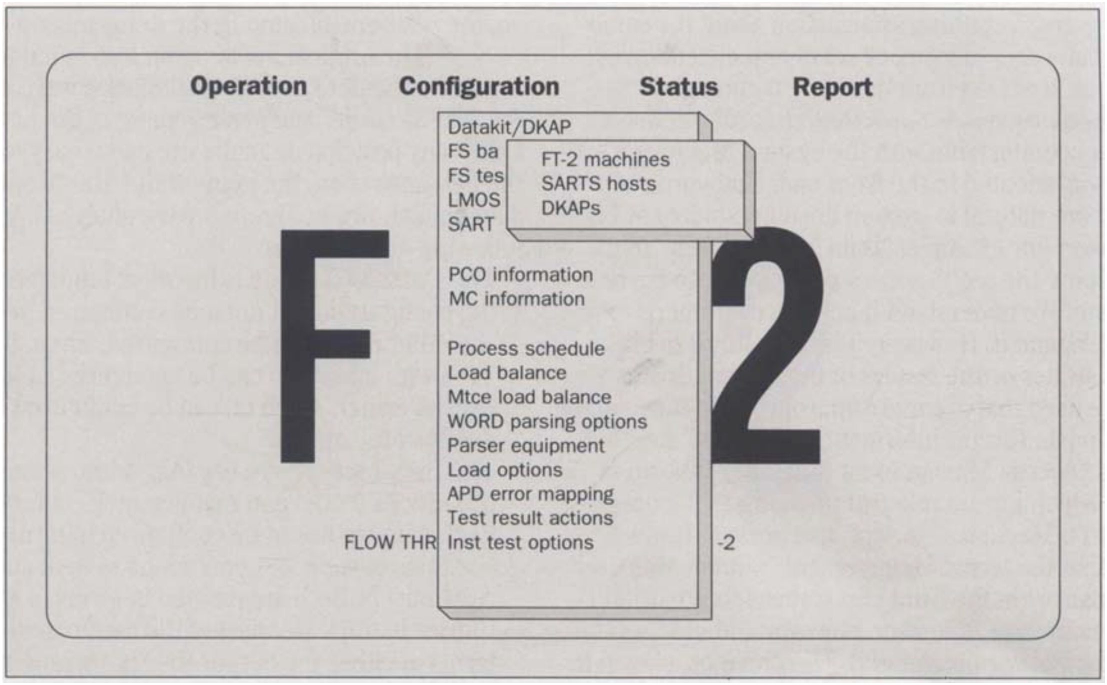


Figure 2. Words and jargon that are natural to system designers may not be so to the user. In order to make the user feel comfortable with the system, the user's terminology—such as "FT-2 Machines," "SARTS Hosts," and "DKAPs," was adopted in the front end, instead of the original language shown in Figure 1.

Using these criteria, menu structures were designed that allow the user to get at the right entry quickly and confidently. Related data also were grouped according to what the user needs to perform a single job function, and each group of data was put in the same screen form, even if the information came from multiple data tables. This way, the user is not forced to access multiple screen forms via different menu entries to fulfill the same job function.

Identify the User. For FTDM and FT-2, the user can be the administrator, the tester at the SARTS, or the manager at the SARTS special services center.

The *administrator* is responsible for the proper operation and functioning of the system, that is, for specifying all the configurations and operation parameters for the system. The *tester* needs to know about the progress of each circuit within the job stack, including the scheduled execution time for the job, or the progress status of the job once it starts. The *manager* at the SARTS center evaluates the overall activities of the organization. Therefore, the FTDM reports will be the prime concern.

The menu structure is designed so that the user can access all aspects of the above activities. However, some operations and configuration activities were restricted only to the administrator because any change in, or access to, those entries causes the system to behave unexpectedly, or to produce undesirable results. The JAM package allows some menu entries to be blanked out, depending on the login. In the prototype, the menu was designed for the administrator. The concerns of the tester and the manager could be addressed quite easily with minor effort in designing the final system. As it turns out, the customer has not requested these features.

Organize the Menu According to the Job. After the users were identified, their job functions and the information associated with each job activity were analyzed. The menu structure and screen forms were designed to facilitate the user's jobs by making it easy to locate the right menu entry, and by putting information about the same job in the same screen form, or set of associated forms, which can be accessed from the same menu entry.

Employ the User's Terminology. In order to make the user feel comfortable with the system, the user's terminology was adopted in the front end. But words and jargon that are natural to system designers may not be so to the user. For example, "Data Management" to us means keeping the SARTS access point database current and accurate. We understand it and the customer's executives understand it. However, it is not natural to either the administrator or the testers of the system. Instead, terms were used that are more appropriate to the user, such as "Update Circuit Information in SARTS." Another example is "Access Management Machine" (shown in Figure 1), which has no relevant meaning to the user, whereas "FT-2 Machines" is very appropriate. It was observed that the term "Management" initially was used quite extensively in the front-end screens, reports, and user documentation. The user, however, did not find this term comfortable or meaningful. The changes, shown in Figure 2, made the system much more understandable and easier to use.

Make Menu Entries Orthogonal. An additional consideration in designing the menu organization is the orthogonality of the entries that, as noted, make each entry independent of any other entries.

For example, the following menu entries are clearly orthogonal:
- Operation,
- Configuration, and
- Status.

However, if "Monitoring" were used instead of "Status," there could be confusion because "Monitoring" can be considered a type of "Operation."

An orthogonal menu structure provides the user with the increased certainty of getting at the right item quickly, which contributes to the user's confidence with the system and the system's ease of use.

Arrange Items Within the Same Submenu. Most often, entries within the same menu list will have some dependencies among them. Sometimes entering data into one field may require the user to reference data in another screen. It is preferable for the menu and screen arrangement to automatically handle these implicit or explicit dependencies as much as possible. In order to do so, the relationship among the items must be analyzed.

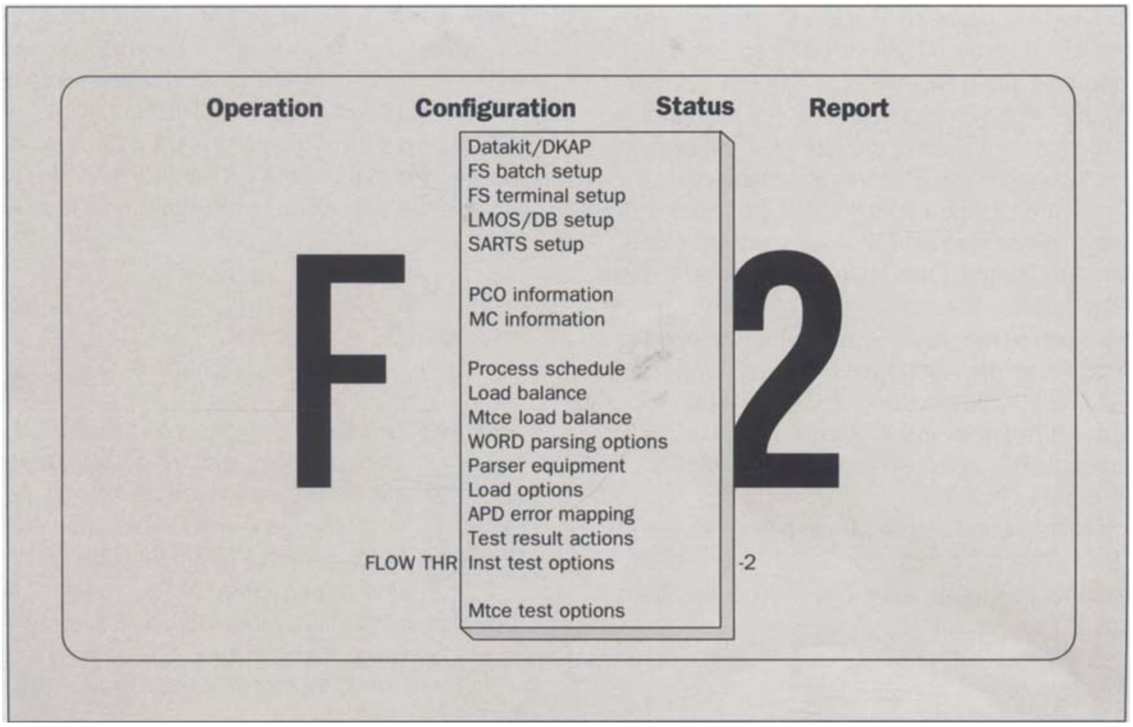
The original FTDM menu was found to be confusing and difficult to use. The categories were not orthogonal, and the order and arrangement of the items did not follow any principle to make the menu easy to use. For the demonstration, the items within the "Configuration" submenu shown in Figure 3 were analyzed, leading to the following observations:
- The Datakit VCS node is the most fundamental item in the configuration. It must be configured first, before any other items can be configured. Then, the connections with other OSs can be configured in terms of the Datakit switch. Each OS can be configured independently of the others.
- The circuit testers are organized into provisioning control offices (PCOs) and maintenance centers (MCs). Each of these has to be configured in terms of the Facilities System (FS) operations system and LMOS, respectively. Both are defined in terms of SARTS.
- Entries in the lower part of the menu are defined in terms of either the PCO or the MC. Arranging these items in the order of dependency will help clarify the configuration procedures.

These parameters are arranged in the order of dependency and grouped by their nature if they are independent. The "Mtce Test Options" entry stands alone because it is a selling point of FT-2. It could have been grouped with the other options.

Other Psychological Considerations. JAM software supports two types of pull-down menus. One menu type requires pressing the "Enter" key after moving the cursor to the proper entry in its parent menu. The other menu type automatically shows up whenever the cursor is moved to the entry.

The automatic pull-down menu is superior, especially for the main bar menu. As the user moves the cursor across the bar menu, different pull-down menus automatically appear to display the choices. Automatic pull-down menus give the user a much better chance of knowing the scope of the system, and the relative significance of an activity within the context of other system features. The users, therefore, have a much more secure feeling about the system.

Graphics also have an impact. Thin borders make the screen look less crowded, more open, and more organized. On the other hand, heavy borders draw



Datakit/DKAP
FS batch setup
FS terminal setup
LMOS/DB setup
SARTS setup

PCO information
MC information

Process schedule
Load balance
Mtce load balance
WORD parsing options
Parser equipment
Load options
APD error mapping
Test result actions
Inst test options

Mtce test options

Figure 3. The original FTDM menu was found to be confusing and difficult to use. The categories were not orthogonal, and the order and arrangement of the items did not follow any principle to make the menu easy to use. For the FT-2 prototype demonstration, the items within the "Configuration" submenu were analyzed. The conclusions are discussed in the section "The Menu Hierarchy."

the user's attention to important choices or information.

The Customer's Prototype Demonstration. A demonstration of the FT-2 human-machine interface was conducted, in December 1991, for Pacific Bell Telephone personnel in the SARTS, and in the AT&T office at San Ramon. The demonstration included the login menus, screens, error messages, and report formats.

Selecting the Audience. The demonstration was for all Pacific Bell personnel that might be impacted by FT-2, including the FT-2 administrators, LMOS administrators, special service testers, SARTS managers, and engineers from the staff organizations. The demonstration enabled the users and the staff members to understand in a more realistic and concrete manner the options and parameters detailed in Pacific Telephone's vendor design document. This document, generated by the customer, provided the system definitions.

The demonstration also provided an opportunity for the users to examine whether the procedures to administer and use FT-2 were reasonable, convenient, and easy to learn. These attributes are impossible to describe in a conventional document like the vendor design document. Demonstrating them in a prototype increased the users' confidence and involvement before the product was deployed.

As stated in the vendor design document, the FT-2 system impacts the administration organizations of LMOS, Access Network Systems (ANS), Datakit switch, and FS, as well as the SARTS and Trouble Reception Centers. Therefore, all interested parties, as determined by Pacific Bell, were invited to attend. The demonstration was designed, however, primarily to benefit the SARTS staff, the FT-2 administration, and the staff organizations. The demonstration was preceded by an overview of the FT-2 functions and how FT-2 can provide value for the Pacific Bell operations.

Choosing the Equipment. The demonstration package was developed on an AT&T 6386 WGS running MS-DOS. The demonstration was done on a Safari® laptop running MS-DOS. The screen contents were projected on to a wall screen by connecting the laptop screen to a viewplate, which was placed on a viewgraph machine.

The demonstration lasted about two hours, with a question-and-answer period following each presentation. The laptop was made available for hands-on experimentation by the attendees.

Showing the Users' Job Functions. The demonstration was designed to include the types of activities that the user is most likely to perform, such as how to:

- Set up Datakit Applications Program (DKAP), a customer-customizable software program;
- Set up FT-2 to use DKAP;
- Set up connections to LMOS high-capacity front-end databases;
- Set up connections to the Facilities System database;
- Set up maintenance centers;
- Set up maintenance options;
- Request or schedule reports;
- Use the information in the reports;
- Monitor the status of the printers;
- Monitor trouble tickets; and
- Monitor FT-2 network connections.

Audience Perceptions and Suggestions. The audiences at all three locations were enthusiastic about the demonstration and, due to the flexibility of the JAM package, most of the suggestions were incorporated into subsequent demonstrations. Indeed, some suggestions were actually implemented during the demonstration.

The most important results of the demonstration included an improved understanding by the customer of what FT-2 could do, and by the user of how it could help in his or her job. A beneficial side-effect was the good will and team spirit that was generated.

Acknowledgments

The authors would like to thank Bob Laiks of AT&T at San Ramon for supporting and arranging for this demonstration; Paul Albrecht of AT&T in Columbus for his information and comments on JAM; William Toft of Global Information Systems (formerly NCR) for the initial tutorial on the JAM package; and Scott Durstewitz and Mary Anne Carletta for their supportand assistance.

(Manuscript approved June 1994)

Man Keung Yeung *is a member of technical staff in the Traffic Data Collection and Engineering Department of AT&T Bell Laboratories in Holmdel, New Jersey. Previously, he helped develop the architecture for the Flow Through Data Manager (FTDM) and its customized version, FT-2, relying in part on his experience in designing automated steel mills before joining AT&T in 1985. Mr. Yeung has a B.S. degree in physics from the University of Winnipeg in Canada, and an M.S.E.E. degree from Queens University at Kingston, Canada.*

Tanjore K. Srinivas *is a technical manager of the Platform Software Development Group in the Services Testing Laboratory at AT&T Bell Laboratories in Red Hill, New Jersey. His group is responsible for systems engineering, architecture, and development of the new Test Data Interface Layer (TDIL) operations system and for developing graphical user interfaces for another operations system. He previously was responsible for a technical group that developed the FT-2 project. He joined the company in 1991. He has a B.S. degree in physics from Delphi University, India; an M.S. degree in experimental physics from the Indian Institute of Technology in Kampur; and an M.S. degree in computer science and a Ph. D. in theoretical solid state physics from the State University of New York at Buffalo.*