AIAA-92-1936-CP

IN-ORBIT TEST AND MONTORING SYSTEMS ARCHITECTURE*

Vasilis Riginos, Steve Teller, Pei-Hong Shen, and Walter Kelley
COMSAT Laboratories, Clarksburg, Maryland 20871-9475

Abstract

To facilitate efficient and rapid development of automated, microwave-frequency measurements for in-orbit test (IOT) and monitoring systems, an engineered platform architecture of generic software functions was designed and implemented at COMSAT Laboratories over the past several years. The Measurement Processing and Control Platform (MPCP) provides the measurement developer an operating system-like set of field-tested, reusable building blocks for such tasks as instrument and IEEE-488 bus control; graphical user interfaces (GUIs) supporting OSF/Motif-compliant X Windows; measurement scheduling; interprocess/intermachine communications; database management; and printing and plotting services. With pretested building blocks, IOT and monitoring systems can be implemented with shorter development times and reduced overall system cost. MPCP runs under the UNIX System V operating system on workstations, providing multiuser, multitasking capabilities to the IOT system, as well as extensive networking capabilities, including remote control. This platform architecture is contrasted with an alternative approach in which each measurement is a self-contained entity, which results in significant code duplication, inconsistency, and lack of software robustness, as well as difficult maintenance, upgradability, and expandability. This paper discusses the tasks performed by automated measurements, and the services provided by MPCP modules for performing these tasks.

Introduction

As satellites become more complex, the number of measurements required to test them in orbit, as well as the data volume, increase rapidly. As a result, required testing time has grown substantially with each new generation of satellite. Because testing time deprives the owner of revenue, there has been steady pressure on IOT systems to keep this time to a minimum. This pressure spurred the development of IOT systems from early manually operated systems to highly automated computer-controlled systems.[1]

The most natural transition from manual measurement systems to controlled ones is to program the computer to mimic the actions of the manual system operator. Many IOT and monitoring system developments follow this route, resulting in self-contained measurement procedures, since each measurement is developed as needed, replicating the manual system operation. A very different development approach stems from the observation that over 80 percent of these self-contained procedures perform similar tasks (e.g., instrument control, user interface, data storage, etc.), while only 20 percent is actually devoted to tasks that are unique to the measurement at hand. This insight led to the development of a fundamentally different architectural concept—an integrated platform of specialized building blocks that perform the common measurement tasks which then form the foundation for developing much simpler measurement procedures. Table 1 contrasts the major differences between these two dissimilar system architectures and measurement development methodologies.

Work started at COMSAT Laboratories about 10 years ago for developing the holistic approach of a platform architecture, after employing the self-contained measurement architecture for a number of years on earlier IOT and monitoring systems. This platform approach applies basic engineering methods and practices such as top-down design techniques, system and subsystem engineering and interface definition, data flow analysis, modularization, specialization, and layering concepts to the system's architecture and not just to the individual components. Tasks are allocated to functional modules, and each module is specialized. By analogy with equipment design, the system engineering function partitions the system into subsystems and modules. Each module is implemented by a specialist. The amplifier designer does not design the power supply. Each designer specializes and focuses on his particular task, so that each task can be done very well, improving overall system quality and robustness. System engineering integrates the separate parts into a cohesive whole.

The work at COMSAT Laboratories resulted in the Measurement Processing and Control Platform (MPCP), which provides an architectural foundation for implementing IOT and monitoring systems rapidly and efficiently. A recently deployed automated measurement system incorporating MPCP is the EUTELSAT IOT system, depicted in Figure 1.

Common Measurement Tasks

In a single-measurement system, whether manually operated or automated, one measurement at a time is performed. A device-under-test (DUT) is stimulated in a known

* This paper is based on work performed at COMSAT Laboratories under the sponsorship of the Communications Satellite Corporation (COMSAT).

manner, and its response is measured. In a multiple-measurement system, multiple measurements can be performed at the same time. In addition, some control can be exerted on the DUT such as setting parameters. For an IOT facility, the DUT is a communications spacecraft in geosynchronous orbit. In a manual system, the operator controls the instruments, and records and processes the measurement data. In an automated system, the computer controls the instruments, and the operator becomes a user who interacts with the computer rather than turning knobs and pressing buttons on instruments. The computer is much faster and more precise than a human operator, resulting in repeatable measurements. The system is easier to use and can be operated by a less skilled user because more of the intelligence resides in the computer's software.

Table 1: Comparison of Measurement Approaches

Feature	Self-Contained	Platform
Design Effort		
System	Very Light	Very Heavy
First Measurement	Heavy	Heavy
Next Measurement	Heavy	Light
Development Effort		
System	Very Light	Heavy
First Measurement	Moderate	Moderate
Next Measurement	Moderate	Light
Life Cycle		
Maintenance	Difficult	Easy
Expandability	Moderate	Easy
Portability	Difficult	Moderate
Evolution	Difficult	Easy
Quality		
Methodology	Ad Hoc	Engineering
Robustness	Low	High
Consistency	Low	High
Capability		
Remote Control	Very Hard	Moderate
Networking	Very Hard	Moderate
Concurrent Meas.	Very Hard	Moderate
Distributed Syst	Very Hard	Moderate
User Changes*	Hard	Easy

* Ease of changing plot and print formats

As satellites became more complex, later IOT systems were used by two distinct groups of users: a technician-level person skilled in standard earth station operations, and the specialized spacecraft expert.With two distinct user groups, other interface requirements were introduced: simplicity of operation for the earth station technician, and flexibility of measurement control for the spacecraft expert investigating an anomaly or conducting in-orbit tests.[1]

The computer-controlled measurement performs instrument control, user interaction, and data processing consisting of manipulation, storage and retrieval, and presentation. In addition, the measurement architecture determines the structure and characteristics of the measurements. These tasks are discussed briefly in the following paragraphs.

Instrument Control

All measurements, whether performed manually or automatically, require instrument control, which is provided by the skilled operator in a manually-operated system. The operator sets up the instruments, monitors the instruments while the measurement is being performed, checks the results for integrity and validity, and responds to abnormal and unexpected situations. In a manual system, the skilled operator can discard erroneous or flawed data or repeat a flawed measurement.

In a computer-controlled system, these tasks are performed by the computer. The computer physically interfaces to the instrument, handles exceptions, and ensures data integrity and validity of instrument-supplied data. Instrument control occurs via the IEEE-488 bus, RS-232, or some other interface mechanism. Exception handling—the ability of the computer to deal with abnormal or unexpected circumstances that can occur at unpredictable times—is one of the many nontrivial tasks that must be performed in building automated systems.

The User Interface

A second set of requirements for an automated IOT measurement facility is provision for user interaction. Measurements must provide a unified and consistent user interface, and must address three problems pertaining to the interface: validating user input, minimizing modal operation, and providing suitable feedback to the user.

In an automated single-measurement system, the user no longer directly interacts with the instruments, since these are now under computer control. Rather, the user interacts with the computer system to set up, parameterize, and control a measurement. Because of this shift in the user's role (i.e., no longer an equipment operator), the computer must now han-

dle the additional tasks of interacting with the user. In addition, there are now two user categories, as stated earlier. For both user types, the computer must check the user's input for validity. For example, the user interface should detect erroneous input data that is out of range or of the wrong type (e.g., alphabetic input when the expected input is numeric).

Second, the user interface should minimize modal behavior. The user should be minimally restricted in the actions that can be performed at any given time. The user should be in control of the measurement, not the measurement in control of the user. Maintaining user control is achieved by minimizing modal behavior of the user interface.

Since the user no longer interacts directly with the instruments as in the manually operated system, the automated system must now provide the user with feedback on the progress and status of the measurement. This feedback is especially critical when the user, by setting up a measurement that transmits a signal to the spacecraft, may accidentally interfere with traffic. The user must be able to abort the measurement if necessary, (e.g., if the test is interfering with traffic or the antenna is pointing in the wrong direction). The automated system, while relieving the user of tedious instrument control tasks, must enable the user to remain in control of the measurement situation. These user interface requirements complicate the design and implementation of automated test systems.

Data Processing Requirements

A third set of requirements for a simple, single-measurement automated system relates to data processing. In a manually operated system, the operator records and stores the data (usually in a notebook), and then processes the data into useful form by manually constructing plots and tables—a labor-intensive, tedious, and time-consuming process. Because of the tedium, only necessary plots are constructed, restricting creative analysis and presentation of the measurement data.

Data processing in the automated system involves three activities: manipulation, storage and retrieval, and presentation. Raw data may require additional manipulation, such as computations, averaging of several data sets, statistical analysis, and digital filtering. Second, the raw and processed data must be conveniently stored and retrieved from a database. The final output is information presented in a useful and comprehensible form to the people who use it to make decisions. The processed data are usually plotted and/or printed for reports. The automated system should be equipped with a hard disk, a plotter, and a high-quality printer to support these data processing requirements.

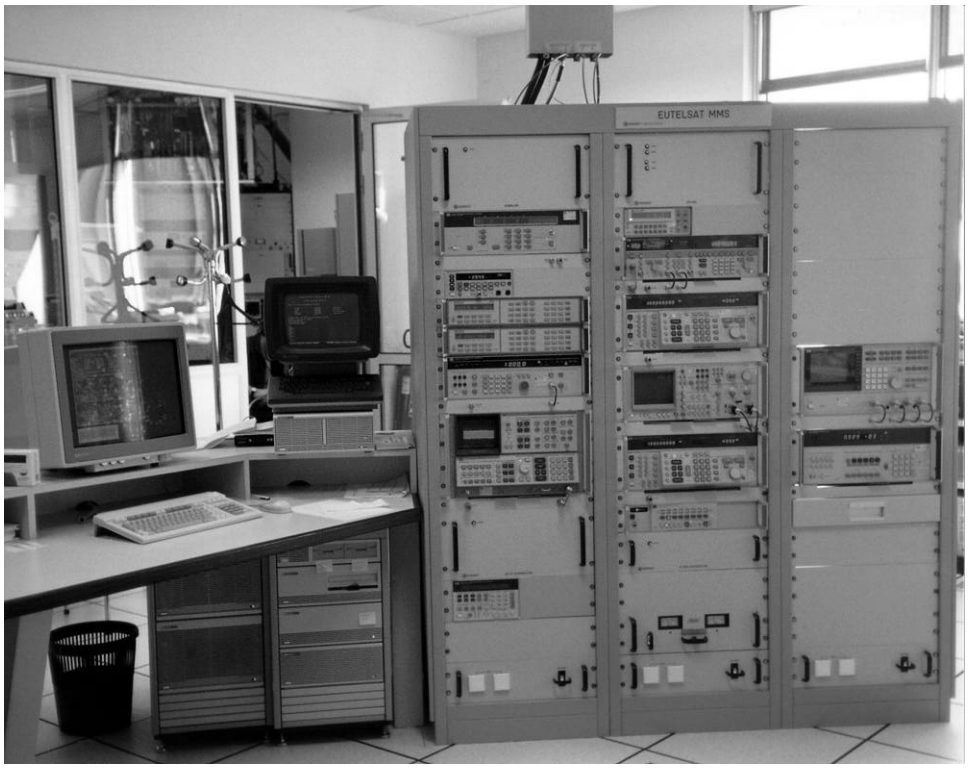

Figure 1. ***EUTELSAT IOT System Installation Uses MPCP Software Components***

Measurement Architecture

When implementing automated measurements, the architecture of the measurement itself is a critical consideration addressed during the design phase. Many years of experience in designing and implementing a large number of diverse IOT measurements at COMSAT Laboratories led to the realization that the measurements, while different in their specifics, shared many tasks in common.[2,3,4] This insight led to the concept of the desirability of separating common measurement functions from those associated with a specific measurement.All measurements require instrument control, user interaction, and data processing capabilities. These common tasks can be separated from measurement-specific elements. Each task can then be designed and implemented once as a modular unit, centralizing and specializing the function, and avoiding duplication of tasks from one measurement to the next.

Further, if this architectural separation is done properly during the design phase, the common task implementation offers the potential for *reusability*, not only from one measurement to the next, but from one system to the next. By building and perfecting a set of common building blocks, new systems can be built with a substantially reduced software development load, with its attendant high costs due to labor intensity and lengthy development time. Available software development resources (i.e., time, money, highly skilled labor, equipment, facilities) can then be focused on application-specific measurements, rather than on the supporting infrastructure. This architecture results in systems with shorter overall development time and greater reliability because they incorporate previously tested building blocks. Once conceptualized as an integrated, modular measurement architecture, each module is designed, coded, tested, debugged, integrated, and documented once. Each module can be refined independently without disturbing others.

Other tasks are common across the various measurements. All measurements must alarm the user when certain conditions are encountered. Similarly, instrument control involves many common tasks, independent of the particular instrument. For example, all instrument drivers also handle exceptions from their respective instruments and manage communications with their specific instrument across the IEEE-488 bus interface.

With a large number of common tasks to be performed at both the measurement and instrument control levels, an engineered platform architecture conceptualizes and implements common tasks as modular components separate from the measurement-specific tasks, so as to minimize task duplication and to ensure consistency across measurements. The modular concept is extended to include instrument drivers and IEEE-488 bus management and control. As many instruments may be required in an IOT system, a large number of instrument drivers are required. The instrument driver is a program that controls a specific instrument such as a frequency synthesizer, spectrum analyzer, or power meter. Each instrument driver is implemented as a separate module. With a library of such instrument drivers, a diverse set of measurements can be implemented. Low-level IEEE-488 bus management can be modularized as a set of functions which is independent of any particular instrument driver. Low-level bus functions can then be called by instrument drivers or other programs requiring IEEE-488 bus communications with a device connected to the bus, such as another computer system.

Multiple Measurement System

In response to the growing complexity of communications satellites, multiple, concurrent measurement and network capabilities, including remote access and control, became IOT system requirements or desirable features.[1] A multiple-measurement system can perform multiple measurements for any given time interval rather than just a single measurement. In a multiple-measurement system, many instruments are connected to the computer and can be selected in different combinations to perform different measurements on the DUT at the same time. With the availability of reasonably priced, powerful computers, such as engineering workstations, with better processing and storage capacities than before, a multiple-measurement system can be implemented, providing improved system utilization. However, the multiple-measurement system introduces new requirements for measurement scheduling and equipment sharing, as well as concurrent operation. Networking capabilities add other requirements.

Measurement Scheduling

In the single-measurement system, different measurements are performed sequentially with different combinations of instruments stimulating and measuring the DUT. Each measurement ties up the entire system for the duration of the measurement. If simultaneous measurements are desired, scheduling and allocation of hardware resources become important issues. For example, it may be desirable to measure the stability behavior of a spacecraft characteristic (e.g., the local oscillator frequency or e.i.r.p) by measuring it at periodic intervals over some duration of time. If the measurement system is allocated to this one measurement, no other measurements can be performed during the allocated time period. This scheme wastes much of the system's resources and utility because high-cost microwave measurement hardware is tied up for a long time period with possibly lengthy intervals between measurements, when the instruments sit idle. For example, if a particular spacecraft characteristic was to be measured once per hour every day for a week, an unscheduled system would make the instrumentation unavailable to other measurements for a whole week. If

the measurement required five minutes each time it was performed, the system would not be utilized 92 percent of the time—a considerable waste of system resources.

This undesirable situation leads to the requirement for measurement scheduling, in which the equipment resources are time-shared among multiple, scheduled measurements. The rationale for a measurement scheduler is analogous to the early days of computing, when job schedulers were developed to increase utilization of expensive computer hardware. Similarly, with a measurement scheduling algorithm, different measurements can be scheduled to have access to the limited hardware, which is typically the constraining factor for performing multiple measurements.

Networking and Remote Measurements

Remote control and networked system operation are desirable system features for a number of reasons. First, the spacecraft-under-test may be visible in one part of the world, while the spacecraft testing experts may be located in another part of the world. The time and expense to move the spacecraft experts to another part of the world is a significant obstacle. Even when it is necessary to do so, the expert's ability to perform measurements is time-limited. The quantity of data that can be obtained is restricted, and perhaps the quality of the data may suffer. Under such circumstances, the expert's creativity is restricted.

With suitable networking and software capabilities, the user does not have to be physically colocated with the measurement hardware. A networked system architecture provides the capability for remote control. Because engineering workstations offer extensive networking capabilities, remote control and distributed systems become feasible and desirable in state-of-the-art IOT and monitoring system architectures.

With a suitable network architecture the expert can control a measurement from one location, while the actual measurement is performed elsewhere. By extension, a network of measurement sites can be connected to a central site where the spacecraft experts are located. Data can be collected centrally for database storage, analysis, and presentation. If the expert can conduct measurements from one location without time and space restrictions, creative analysis is enhanced. The scheduler overcomes time constraints, while the wide-area network (WAN) capability overcomes space and geography constraints. The expert can perform tests whenever convenient. Creative insights often occur under such unstressed circumstances. The comparison with computing is between batch processing and interactive processing. When batch processing was dominant, a user submitted a job to the computing center, and a few hours later picked up the output. This slow process curtailed creativity, but it was the best that could be done to maintain high CPU utilization. Later, when interactive systems became more widely available, people were encouraged to "try things" while on-line because the feedback was immediate. Insights were more likely to occur in this less rigid interactive mode than in the batch-oriented systems.

Measurement Processing And Control Platform (MPCP)

Because of increasing satellite complexity and capacity from generation to generation, more demanding requirements are placed on computer-controlled IOT and monitoring systems, as discussed above. At COMSAT Laboratories increasing IOT system complexity led to a fundamental shift in strategy for the system and measurement architecture to accommodate these new requirements. Earlier systems had been implemented in which each measurement was a self-contained entity, with the limitations previously described. The shift in architectural strategy resulted in the concept of an engineered platform architecture for constructing new measurements in a time-efficient manner, while also providing the value-adding characteristics of modularity, functional specialization, consistency, flexibility, upgradability, and expandability. COMSAT Laboratories' IOT system and measurement architecture is one in which common-function building blocks are separated from the measurement and separately implemented.

The resulting MPCP, developed over a period of several years, is an operating system-like collection of measurement software components designed specifically to facilitate the rapid, efficient implementation and deployment of IOT, monitoring, and other measurement systems. MPCP is built on the UNIX System V operating system and executes in a UNIX-based engineering workstation hardware and networking environment. Figure 2 depicts the platform architecture as an extension of the multiuser, multitasking UNIX operating system.

MPCP Modules

MPCP consists of a number of core building blocks arranged in hiearchical layers, forming a "platform" on which automated IOT systems can be built (see Figure 2.). The building blocks communicate through well-defined interface protocols. Modules lower in the structure are closer to the hardware and provide services to upper layer modules.

The Datapool, Alarm Handler, and Scheduler modules execute as independent UNIX processes. These processes communicate with each other and with other processes, such as measurements, via mail messages packaged and delivered by the MPCP Mail Subsystem. The Mail Subsystem, a critical component of MPCP, provides interprocess communications between processes residing on the same or *different* machines connected via a local area network (LAN), and will be discussed later in this paper in the context of a distributed processing environment and system architecture.

The MPCP Datapool is a central repository of the measurement system's shared information and common data resources. It provides a pool of shared data for multiple processes residing on the same or different machines spread throughout the measurement system network.

The MPCP Alarm Handler centrally manages alarms that occur throughout the system. The MPCP Alarm Handler is notified, via mail, of alarm conditions detected by other measurement system processes, and displays the alarm or alarms to the user. Alarm messages are tailored to the individual measurement system processes, and can be easily customized to the specific requirements of a particular system implementation.

The MPCP Scheduler schedules measurements and makes available hardware and equipment resources to requesting measurements on a first-come, first-served basis. The Scheduler is discussed in more detail at a later point in this paper within the context of the multiple, concurrent measurement system.

The MPCP Instrument Drivers Library contains a set of instrument drivers for a variety of measurement equipment. Each driver performs high-level instrument command and control via the IEEE-488 bus to the respective instrument. Drivers have been implemented for spectrum analyzers; frequency, phase, and modulation measurement instruments; frequency and waveform synthesizers; power and voltage measurement instruments; RF switch control; and data acquisition units. With the existing large base of instrument drivers in the MPCP Instrument Library, new instrument drivers can be added relatively easily by copying and modifying an existing driver.

Service providers such as the instrument drivers, mail system, and database services are supported by several lower level services: an Input Handler, IEEE-488 bus control functions, standardized data formatting and handling, and user interface supporting services.

The Input Handler is an event handler dispatcher that functions as a kernel for MPCP, providing system I/O services to the other MPCP modules. When an event occurs (e.g., a mouse movement, mouse button click, or a key pressing from the keyboard), the Input Handler performs the required input and/or output operation by making the appropriate system calls to the UNIX operating system. The Input Handler allows I/O to be handled in a central module and in a consistent manner across all MPCP modules.

The functions in the linkable IEEE-488 library serve as the interface between MPCP modules, such as instrument drivers, and the I/O transactions on the IEEE-488 bus. Some services that can be called by the client program, such as an instrument driver, include writing a command string to the

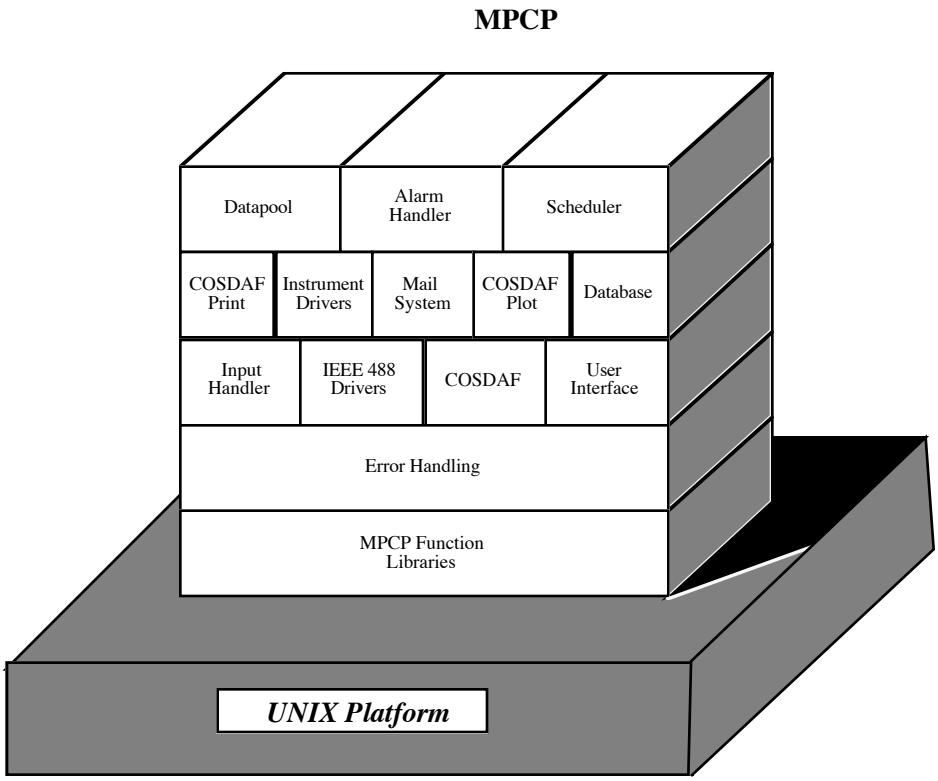

Figure 2. ***Measurement Processing and Control Platform Provides Tested Building Blocks***

instrument; reading response strings from an instrument; addressing the instrument to talk or listen; and performing a serial poll to the instrument or device.

An error handling building block implements a set of functions for handling errors and exceptions in a consistent and unified manner throughout the system. When an error is encountered, each module in the execution path, from the highest level to the lowest, writes its part of a concatenated error message onto a stack. The concatenated error message is recorded in an error log file and displayed, providing the user a detailed description of the circumstances of an error. This stacked error message concept considerably aids troubleshooting, since each error message records the chain of circumstances and events leading to an error.

The lowest level of the MPCP architecture is a collection of linkable libraries. When building automated test systems such as IOT facilities, certain tasks are encountered repeatedly. To eliminate repeated reprogramming of these basic tasks, they are gathered into libraries of object code functions that can be linked into application programs, such as measurement programs, and reused between applications. The MPCP libraries fall into three categories: measurement support utilities, mathematical and statistical functions commonly encountered in IOT systems, and general-purpose system utilities. They provide a large number of logically grouped low-level services.

The MPCP Measurement Support Library contains utility functions commonly required when implementing IOT measurements. It includes functions for such tasks as calculating spacecraft e.i.r.p., input flux, path loss, range to spacecraft, spacecraft gain, spreading factor, Doppler shift, subsatellite point, and other similar functions.

The MPCP Math Library contains mathematical functions applicable to IOT measurement systems such as trigonometric functions; linear regression; statistical parameter determination such as mean and standard deviation of a data set; minimum-maximum determination; numerical integration; decibel calculations and conversions to linear form; and interpolation.

The MPCP Utility Library includes routinely required functions such as compass calculations and transformations, calendar manipulations, list and string processing functions, and date and time manipulations.

The MPCP Measurement-User Interface (MUI) (not shown in Figure 2) supports the implementation of GUIs that are separate from the actual measurement process. An MUI is provided for each measurement, giving the user a user-friendly, graphically oriented, X Window-based interface mechanism for configuring a measurement, as well as system control functions. The user inputs measurement parameters with a mouse and keyboard while viewing windows on the display. Because the measurement process is separate from the MUI, the user can schedule a measurement to be run at a later time. The MUI communicates the measurement parameters and setup information to the Scheduler via mail messages delivered by the Mail Subsystem. When initially displayed, the MUI fills in the window with default values, which the user can edit using the mouse and keyboard. The MUI checks the user's keyboard input for range and type entry errors and performs consistency checks on the entered parameters. The MPCP Measurement-User Interface is implemented with Open Software Foundation's OSF/Motif GUI toolkit and conforms to OSF/Motif style guidelines.

The GUI uses a pointing methodology in which the user moves an on-screen pointer with a mouse to a desired action or selection on the display, and then selects it by pressing a button on the mouse. This type of interface is highly visual and intuitive, rather than character-based. It allows the user to "point" to the desired item rather than having to accurately remember and type in commands, such as required in "command-line" user interfaces typified by the earlier IBM PC and older mainframe environments. Figure 3 shows an example of an X-Window, OSF/Motif-compliant user interface window.

The graphically based user interface is easy to use, even by inexperienced operators, as well as being easy to remember because the user does not have to remember esoteric command sequences and codes. At the same time, this newer style of user interface is designed to accommodate the more experienced user's need for operational flexibility and power. The interface communicates to the user in the terminology and nomenclature of the specific application, so that the user is confronted with an immediately familiar working environment.

A particular application of GUI technology is the MPCP System Mimic Panel, a capability that schematically mimics and displays in a window the status and configuration of an earth station, and can provide station control from the window if appropriate hardware connections are installed. The System Mimic Panel is shown in Figure 4.

The System Mimic Panel displays the operational status of switches, instruments, and other station equipment. It can display the current signal path through the station. With appropriate hardware connections and by using the mouse to click on a displayed switch, the user can control the position of the switch hardware. The MPCP System Mimic Panel is a general-purpose capability that can be readily adapted to display and control the configuration of any specific station.

The GUI technology, being adopted by increasing numbers of computer manufacturers and software applications developers, is the result of over 30 years of intensive research and development activity by many individuals and organizations, and has been driven by the persistent and expanding need to create more effective, human-oriented man-machine interfaces.[5]

The MPCP Database Services subsystem (refer to Figure 2.) stores all raw measurement data taken by the measurement system and allows subsequent searching to retrieve

measurement data. File management has been standardized and centralized, and is implemented in the COMSAT Standard Data Format (COSDAF) module, shown in Figure 2. COSDAF provides standardized storage and retrieval functions for measurement data and static text files. Spacecraft configuration files, frequency plans, and earth station antenna gain and calibration information are stored as static text files. Measurement data and text files are stored with ASCII characters in a standardized format, permitting easy editing and importing into other programs.

Since all measurements require printing and plotting services, these tasks are implemented as a separate set of services provided by MPCP. Format and style are generalized and stored in "style" files, with each measurement type having its own customized style file. By using editable style files, the actual print and plot device drivers are implemented as generic modules. Flexibility is maintained by implementing data-driven drivers that read the style files for specific prints and plots. The style and appearance can be changed by simply editing the style file, not the device driver.

MPCP Plotting Services performs post-measurement plotting of measurement data, as well as concurrent plotting on the display for some measurements as they are running. Many different types of data plots to either soft (i.e., CRT displays) or hardcopy devices are supported for both real-time measurement plotting and post-measurement data analysis.

The MPCP Interactive Plotting Package (not shown in Figure 2) provides an interactive mechanism for manipulating and editing measurement data, enabling creative, flexible data analysis and presentation. The MPCP Interactive Plotting Package allows the user to perform extensive post-measurement data analysis and provides a general-purpose capability to prepare finished, report-quality plots and graphs. The data on a plot can be manipulated and edited in various ways: points can be cut and pasted; scales, axes, and labels can be changed; graphs or points can be annotated and/or marked. The plot can be zoomed in or out. New data points can be added via the keyboard, mouse, and/or from existing files. The data from one or more files can be plotted on the same graph as the data from another file. Various data

Figure 3. ***Graphically-Based User Interface Window Provides Fill-in-the-Blank Format***

transformations are supported: two or more traces can be merged or summed. Or, one can be subtracted from the other (e.g., a calibration file is subtracted from a measurement file, or one measurement trace is subtracted another). Finished plots can be stored and retrieved.

MPCP Printing Services supports printing to both soft (i.e., CRT displays) and hardcopy devices. Printouts are generated automatically at the conclusion of a measurement and as a result of a user request.

Multiple Concurrent Measurements Supported

The typical IOT and/or monitoring system installation requires a considerable investment in expensive, high-quality microwave measurement equipment and ancillary hardware. Therefore, it is highly desirable to maintain a high percentage of utilization of the equipment. This can be achieved by implementing a system that supports equipment sharing via scheduling and multiple, concurrent measurement operation.

A principal building block in the MPCP is a first-come first-served measurement Scheduler that can schedule unattended, noninteractive measurements (i.e., without a user in attendance). The MPCP Scheduler is an independent UNIX process that runs continuously to manage and queue all job requests from users and to start jobs at appropriate times. It is responsible for sharing and allocating resources such as measurement hardware; handling Remote/Local control settings of instrumentation; and providing the means for a user to determine the status of a job that is running in the background, such as an unattended stability measurement. Whenever the UNIX operating system is running, the Scheduler program executes in background mode, accepting requests for jobs and resources. Jobs are run based on the requested time and date and the availability of limited resources, usually the measurement and earth station equipment. The Scheduler resolves conflicts of measurements scheduled for the same time on a first-come first-served basis, although priority-driven scheduling or other scheduling algorithms can also be implemented. Since the Scheduler can launch jobs that perform unattended measurements, it permits measurements to be made during low-usage hours, thus increasing the system's overall utilization.

In addition to measurement scheduling, a second desirable feature in the multiple-measurement system is concurrent operation. In an IOT facility, measurement hardware is typically the limiting resource. However, if this not the case, there is no reason the system should not be able to carry out two or more measurements concurrently. For example, if one measurement requires the use of one of two up-link synthesizers and the spectrum analyzer, while a second measurement requires the use of the other up-link synthesizer and a phase noise test set, the system should be able to support both measurements concurrently. Concurrent operation increases overall utilization of the system when hardware availability is not the limiting constraint. This capability is important because it is highly desirable to minimize the

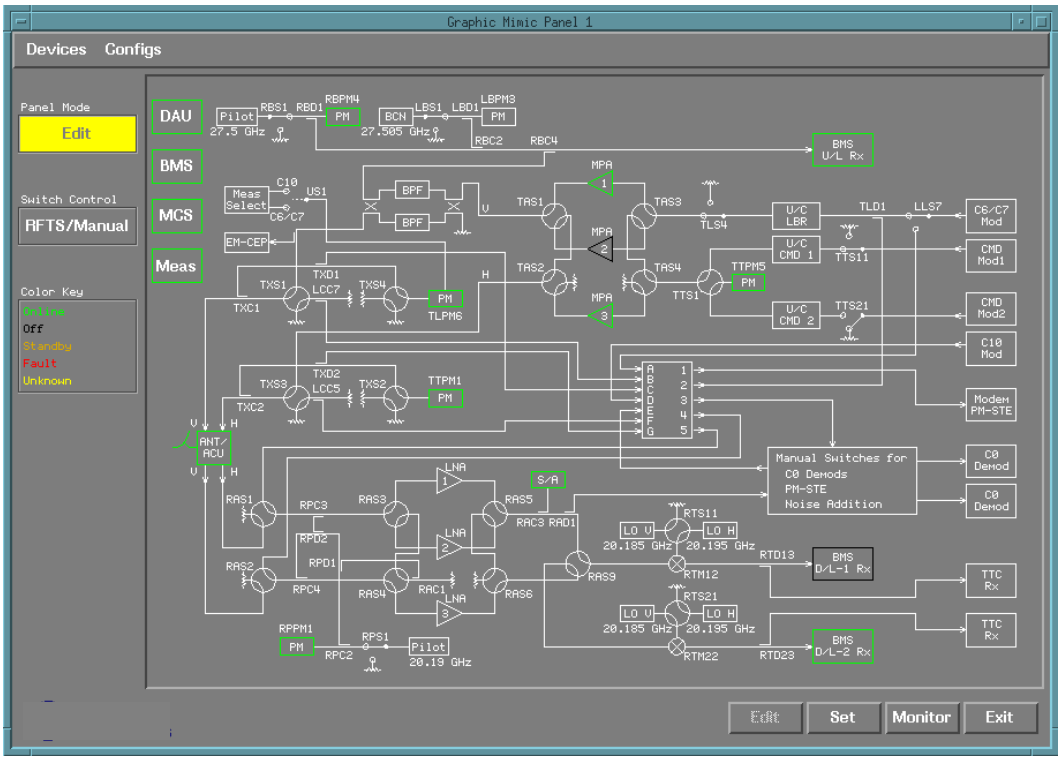

Figure 4. *MPCP System Mimic Panel Shows Station Configuration And Status*

overall testing time during an IOT acceptance test and during anomaly investigation. Testing time on a new spacecraft is expensive because the spacecraft is in non-revenue-generating mode for its owner, who desires an operational spacecraft as soon as possible after the launch.[1]

Distributed Processing and Networked Architecture

With the availability of networking technology, distributed IOT system architectures can be implemented. Figure 5 illustrates COMSAT Laboratories' network architecture for automated IOT and monitoring systems.

Multiple processors and other networks can be connected to create a distributed system architecture, providing several benefits to the user organization. The processing load can be distributed to different computers, easing the computational load on any one machine, while also providing cost-effective redundancy. System resources such as additional measurement instruments, processing capability, disk storage capacity, peripherals, and communications hardware can be added incrementally and only when needed. Expensive peripherals such as laser printers and plotters can be shared among many users. Remote control can be provided over a WAN or dedicated data link using a suitable internet router and modem combination. By adding system resources incrementally and only when required, capital investment can be better managed during the system's life cycle. Advances in technology can be accommodated as newer, better, and faster displays, central processing units, memory, disc storage, printers, plotters, and communications devices become available. The user organization is not locked in and held captive to aging technology.

A key requirement for implementing a distributed system architecture is the ability of separate processes to communicate with one another when the processes reside on the same or *different* workstations connected over one or more networks. The MPCP Mail Subsystem provides interprocess/inter*machine* mail communications capability for message exchange between such processes as the measurement-user interface, the Scheduler, the Datapool, and the measurement in order to accomplish hardware sharing, data sharing, coordinated job scheduling, and job canceling.

The interprocess/intermachine capability of MPCP Mail Subsystem also provides the measurement system with the remote access, control, and communications capabilities required to support multi-station and remote system opera-

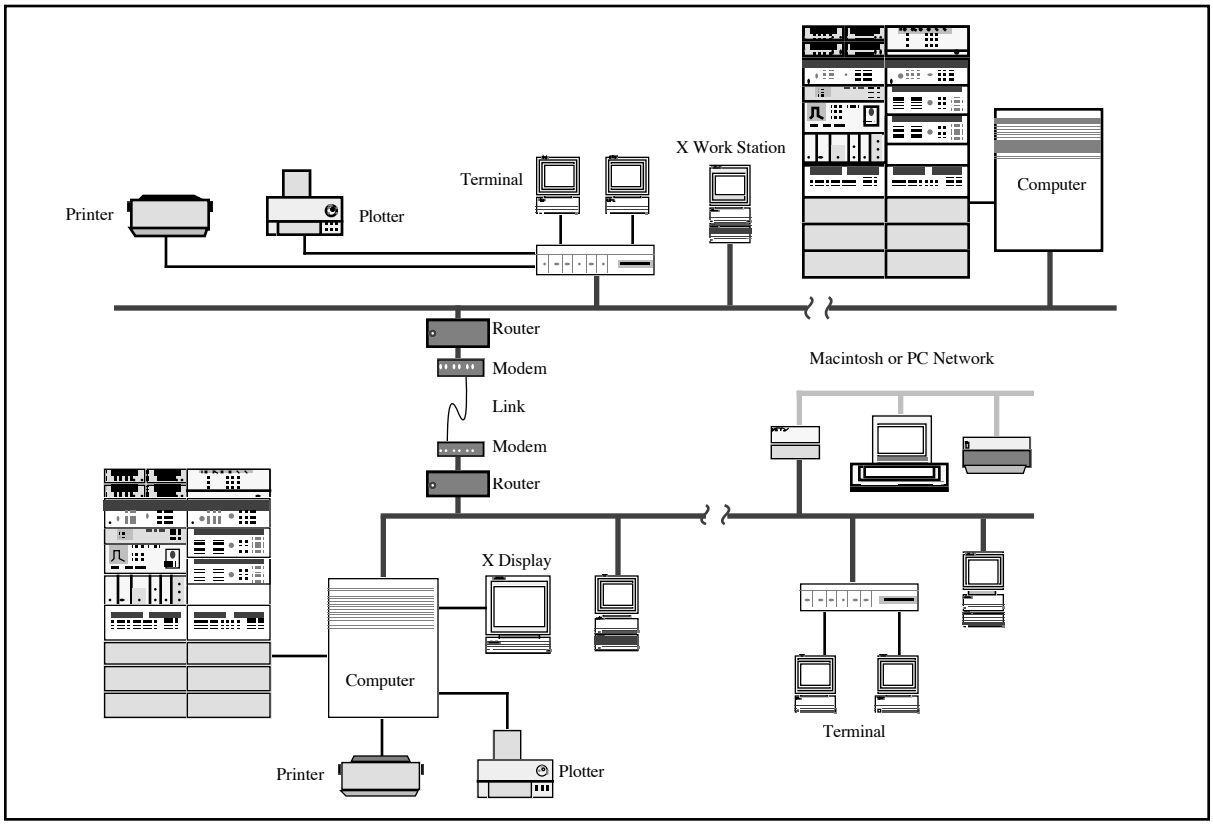

Figure 5. ***Network Architecture Permits Remote Access And Control Of Measurement Hardware***

tion across dedicated low-capacity serial data links or WANs at speeds as low as 2,400 bit/s.

A significant application of this remote communications capability is the display of the same X Window image on a remote workstation as is being displayed on the central workstation. Normally, X Window processes residing on different network-connected workstations communicate with one another through 50-Ω coaxial cables using the 10-Mbit/s channel capacity available on the LAN. Thus, it is virtually impossible to send X Window bit-mapped window images through a 2,400-bit/s serial data link with any kind of acceptable throughput.

The MPCP Mail Subsystem enables such communications through a low-capacity 2,400-bit/s link. With the addition of internet hardware routers at both ends of the modem-connected data link, as shown in Figure 5, the measurement system's LAN is extended to include a distant workstation, as if it were locally connected. Processes running on the local and remote workstations communicate with one another across the link by exchanging TCP/IP-compliant messages via the Mail Subsystem. A remote workstation, running locally its own copy of an X Window display process, can be updated via high-level mail messages sent by the central workstation across the serial link, without choking on the low-capacity data link. Similarly, the user at the remote workstation can send commands to the central workstation across the link via other high-level mail messages. Therefore, a replicated X Window image can be displayed at the remote workstation as at the central workstation, and can provide the same measurement and data processing capabilities as at the central workstation. In this system architecture, it is unnecessary to send the large number of bits associated with an X Window image across the low-capacity data link. High-level mail messages suffice, with little real-time degradation other than transmission delay through the link.

In summary then, the building-block components of the Measurement Processing and Control Platform provide common services required by IOT and monitoring system measurements, such as instrument drivers; graphical user interfaces; data processing, including storage, retrieval, plotting, and printing; interprocess and *intermachine* mail communications; IEEE-488 bus management; error handling; and support libraries of linkable object modules. In addition, MPCP supports concurrent measurement scheduling and distributed processing in a networked environment, including remote control.

Recent Systems Experience

MPCP modules are employed in the implementation of two systems recently developed by COMSAT Laboratories: the IOT system for the European Telecommunications Satellite Organization (EUTELSAT) in 1990 and the RF Terminal Supervisor (RFTS) in 1991 for NASA's Advanced Communications Technology Satellite (ACTS) program. These systems have proven the validity of the MPCP architectural concept, design, implementation, and multi-system application.

Conclusion

This paper discussed the requirements for automated, computer-controlled measurements in the areas of instrument control, user interface, and data processing. The insight that common tasks comprise up to 80 percent of a measurement's overall tasks led to the development at COMSAT Laboratories of a fundamentally different system architecture—an integrated platform of specialized building blocks, each implemented and optimized separate from the specifics of measurements. This architecture resulted in the Measurement Processing and Control Platform—a platform of reusable building blocks built on the multiuser, multitasking, network-supporting UNIX operating system.

References

[1] P.- H. Shen, V. Riginos, S. Bangara, "In-Orbit Testing of Communications Satellites: The State of the Art," Global Satellite Communications Symposium, Nanjing, China, May 1991.

[2] S. Bangara, V. Riginos, and K. Fullett, "Maritime Communications System Package of Inmarsat V," 3rd International Conference on Satellite Systems for Mobile Communications and Navigation, June 1983, London, England.

[3] K. Fullett and V. Riginos, "The Use of Desktop Computers for Measurement Systems as Applied to In-Orbit Testing of Communications Satellites," *Microwave System News*, Vol. 13, No. 2, February 1983, pp. 77–86.

[4] I. Dostis, C. Mahle, V. Riginos, and I. Atohoun, "In-Orbit Testing of Communications Satellites," *COMSAT Technical Review*, Vol.7, No. 1, Spring 1977, pp. 197–226.

[5] S. Card, T. Moran, A. Newell, *The Psychology of Human-Computer Interaction*, published by Lawrence Erlbaum Associates. 1983.