MPCS—THE MANUFACTURING PROCESS CONTROL SYSTEM

Irwin S. Dunietz, John L.C. Hsu, Michael T. McEachern, James H. Stocking, Mark A. Swartz, and Rodney M. Trombly

*The authors, **Irwin S. Dunietz, John L.C. Hsu, Michael T. McEachern, James H. Stocking, Mark A. Swartz,** and **Rodney M. Trombly** are responsible for design and development of the Manufacturing Process Control System. Mr. Dunietz joined AT&T in 1980. He is a member of the technical staff in the Manufacturing Information Automation department at AT&T Engineering Research Center, Princeton, New Jersey. He received an A.B. in mathematics from Cornell University and an M.S.E. in computer science from Princeton University. Mr. Hsu, who joined AT&T in 1970, is a department head in the Manufacturing Information Automation department at the Engineering Research Center. He received an M.S. in electrical engineering from the University of Missouri. Mr. McEachern joined AT&T in 1962 and is a* (continued on page 45)

The central challenge of all manufacturing is making products to the right standards and delivering them at the right time. AT&T is upgrading its corporate and factory resource planning systems to improve control of day-to-day manufacturing. The Manufacturing Process Control System (MPCS), developed at the AT&T Engineering Research Center (ERC), provides this support. MPCS connects the shop floor with production scheduling, accounting, product data archive, and engineering support systems.

Development of the MPCS

Highly customized AT&T products, such as the 5ESS™ digital switch, require hundreds of different circuit packs. Coordinating the manufacture of all these circuit packs is a complex task. The absence of even one circuit pack delays completion and may trigger a factory crisis.

High-level scheduling systems provide much of the logic required to orchestrate the delivery of the material required. However, scheduling in batch mode without the knowledge of current factory conditions cannot fully solve the problem.

To guard against unplanned shortfalls, factories typically stockpile intermediate products. Carrying extensive stocks of intermediate products raises the product cost and may not prevent shortages, because suitable inventory targets are hard to define when the mix of products changes constantly. If product requirements change, the factory must rework or scrap the obsolete versions of products in the in-process inventory. This generates expenses that the final product cost must cover.

Finer control of when and how products are built can lower product cost and shorten delivery intervals. Such control over shop operations, however, depends on a timely and accurate flow of information to and from the shop floor. In the past, obtaining accurate data on the state of factory operations was difficult and expensive. This, in turn, hampered the development of better controls over the manufacturing process.

Recent advances in computer hardware and software, bar code label technology, local networking, and interactive terminals are improving the collection and processing of large quantities of factory data. These changes have generated new interest in tools that monitor events on the shop floor.

At AT&T, a high-level control system is being put in place that uses feedback from factory work centers to reset production targets. An engineering information system is being fashioned to rapidly transfer up-to-date versions of control programs and shop aids from computer-aided design (CAD) databases to shop floor computers.

To be useful, both the business planning and the engineering systems require support at the shop floor level. The *Manufacturing Process Control System (MPCS),* developed at the AT&T Engineering Research Center, provides this support.

MPCS was first installed in 1984 at the AT&T Oklahoma City Works on the CIM1 line. Producing circuit packs for the line units of the 5ESS switch, the CIM1 line contains over 100 workstations and spans some 20 process and test operations. Later, MPCS was extended to Oklahoma City's CIM3 and CIM2 lines.

MPCS software is also in use at the AT&T Merrimack Valley Works in Massachusetts and at the AT&T Teletype Corporation facility in Little Rock. In addition, it is used at the AT&T Works in Winston-Salem, Omaha, Atlanta, and Denver. Developers from these factories and from Oklahoma City have contributed to the MPCS software base.

MPCS consists of many forms, tables, and files and more than 100 programs. It is compatible with the AT&T 3B family of computers and runs under the UNIX® operating system.

MPCS Features

MPCS forms the backbone of the factory information management network, providing the link between the shop floor and production scheduling, accounting, product data archive, and engineering support systems (Figure 1). The primary purpose of MPCS, however, is to help the shop run itself more effectively.

The production monitoring software of MPCS tracks the progress of the work in process through the shop to generate inventory distribution and other measures of production status. Supervisors use these on-line throughput and inventory monitors to gauge the progress made in their areas against the production targets. When changes to a product design or a manufacturing process require that in-process products be rerouted, MPCS tables are updated with what is known as stopwork control conditions. Then, as products move through certain user selected stations, MPCS interprets the stopwork control tables and alerts the shop to exception conditions.

Through direct computer links to important shop equipment and from terminals attached to the shop floor computer, MPCS allows layout operators to monitor the state of machines under their control. These operators can also set the machines to process a different product code using up-to-date versions of machine control programs maintained in an electronic archive on the shop computer.

In circuit pack shop applications, MPCS collects test results, forwards the results to real-time trend analysis and off-line defect characterization systems, and stores the results in on-line databases for support of repair and test operations. Using graphical or form displays of suspected faults, operators at repair stations can easily determine the repairs required for individual circuit packs.

Much of the MPCS software prompts users to enter information by either displaying a form for the user to fill in or by providing a menu of valid choices. This makes MPCS programs easy to use and cuts down on user training time.

Product Tracking. MPCS supports two methods of data entry for tracking product movement. One method tracks the progress of discrete product units from one reporting station to another. The other method tracks groups of products of a single type (typically called *manufacturing lots*) as each lot completes processing at one workstation and moves on to the next. Because the data-

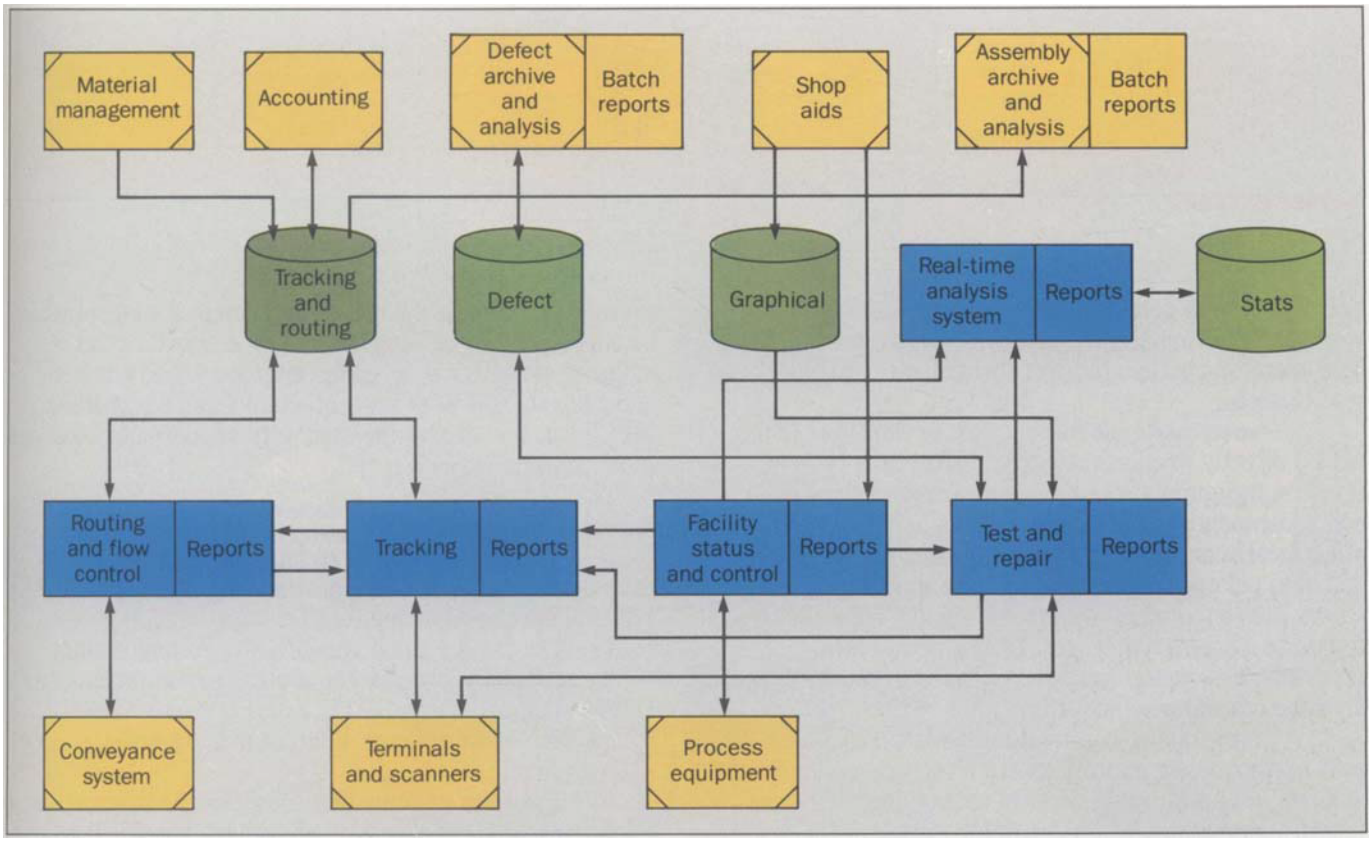

Figure 1. A high-level view of MPCS.

entry software must fit within normal shop operations and must also work efficiently, it has proven to be the least portable part of the MPCS software.

In both methods of data entry, the serial number associated with the individual product units or lots is the basis of product tracking. Usually encoded in a bar code format on labels attached to the product, the serial number is also printed as alphanumeric text, making the identifier readable by the operators as well as by bar code devices.

Automatic scanners, when combined with product transport systems, reduce the number of operators needed to run a shop. Using laser light, these automatic bar code scanners can read the serial number without making physical contact with the label.

Where automatic scanners cannot be used, hand-held scanners, based on the same laser technology, allow operators to scan bar codes with ease. In order to justify manual data entry, however, the value of the information must exceed the value of the operator's time. At workstations that serve as product control points, this cost is offset by the operator's ability to act on exception messages as well as to enter data. In such cases, operators serve as a flexible shop-control interface without the expense and effort required to create a hardware/software solution.

Flow Control. Building on product tracking information, MPCS can also provide tools for controlling product flow. At present, MPCS supports a limited form of process sequence checking to verify that the product showing up at selected workstations has been previously scanned at mandatory antecedent operations. Work that violates such sequence restrictions is flagged for manual rerouting to the earlier process area.

Another type of flow control is the imposition of stopwork restrictions that arise when design changes affect material currently in production. At operations selected as stopwork control points, MPCS determines the product type corresponding to each serial number and notes any stopwork restrictions imposed on that product type. If the material is under stopwork control at that process opera-

tion, the system generates exception messages on the workstation terminal associated with that stopwork point. The operator can then redirect the material into holding or rework areas.

Facility Status and Control. Most of the effort to link MPCS directly to equipment on the shop floor has centered on the automatic machines that insert electronic components into printed wiring boards. These machines are critical in circuit pack assembly. Using device interface software tailored to insertion machine applications, MPCS allows shop layout operators to use remote consoles connected to a 3B computer to download new control programs, change the run state of an insertion machine, or view the current machine status.

In the circuit-pack test area, MPCS can automatically extract defect information for a class of in-circuit test stations. As better tools for local networking of 3B computers and shop equipment become available, MPCS will be able to support the downloading of test set programs and the uploading of test set status.

Defect Data Management. MPCS defect data management addresses two important needs:

- Identifying defects in individual product units so that these defects can be repaired.
- Providing feedback on the state of the manufacturing and test operations.

To assist prompt diagnosis, the *Real-Time Analysis System* (RTAS)[1] also developed at ERC, can use the data collected by MPCS to alert shop supervisors and engineers to aberrant process behavior when defects first appear.

MPCS supports defect data collection through interface with certain types of test sets and through a terminal forms entry procedure at test and inspection stations. The data collected include serial number, workstation name, the operator identifier, location and nature of the defect, reason for the test or inspection, and the product type if it is not available from the tracking database.

MPCS can display current defects logged for each serialized product unit. If supporting graphics can be derived from design databases, the defect data display can be supplemented by an annotated assembly diagram, a mapping of electrical nodes to component lead numbers, and a highlighted display of defects. Within a test area, MPCS can also display the summary of previous defect and repair reports.

System Architecture

The MPCS system architecture has three objectives that separate MPCS from earlier shop information systems developed within AT&T:

- The system runs under the UNIX operating system.
- The system allows users to access *current* production information.
- The system operates in different manufacturing environments.

Previous systems were constructed by groups at a specific AT&T Works location or within a specific product division to address data collection requirements for a specific shop context. Because such systems were intended to work within a limited domain, they were not portable nor were they easy to integrate with other manufacturing information systems.

From information entered in batch mode during the working day, such systems computed shop results as a batch process overnight. Users then saw such information in printed reports the next day. Few of these systems had system monitoring or on-line reporting features. Almost none ran on a UNIX operating system.

Intended as a general approach to shop floor control, MPCS is constructed from a collection of information management tools (Figure 2). Sometimes, parallel MPCS software modules exist to handle similar feature requirements in different shop contexts. The MPCS architecture is flexible enough to allow the core features, software options, and custom software to operate together as an integrated system.

Database and Database Interface. It should be clear from the description of MPCS features that large quantities of data must be managed for MPCS to do its job.

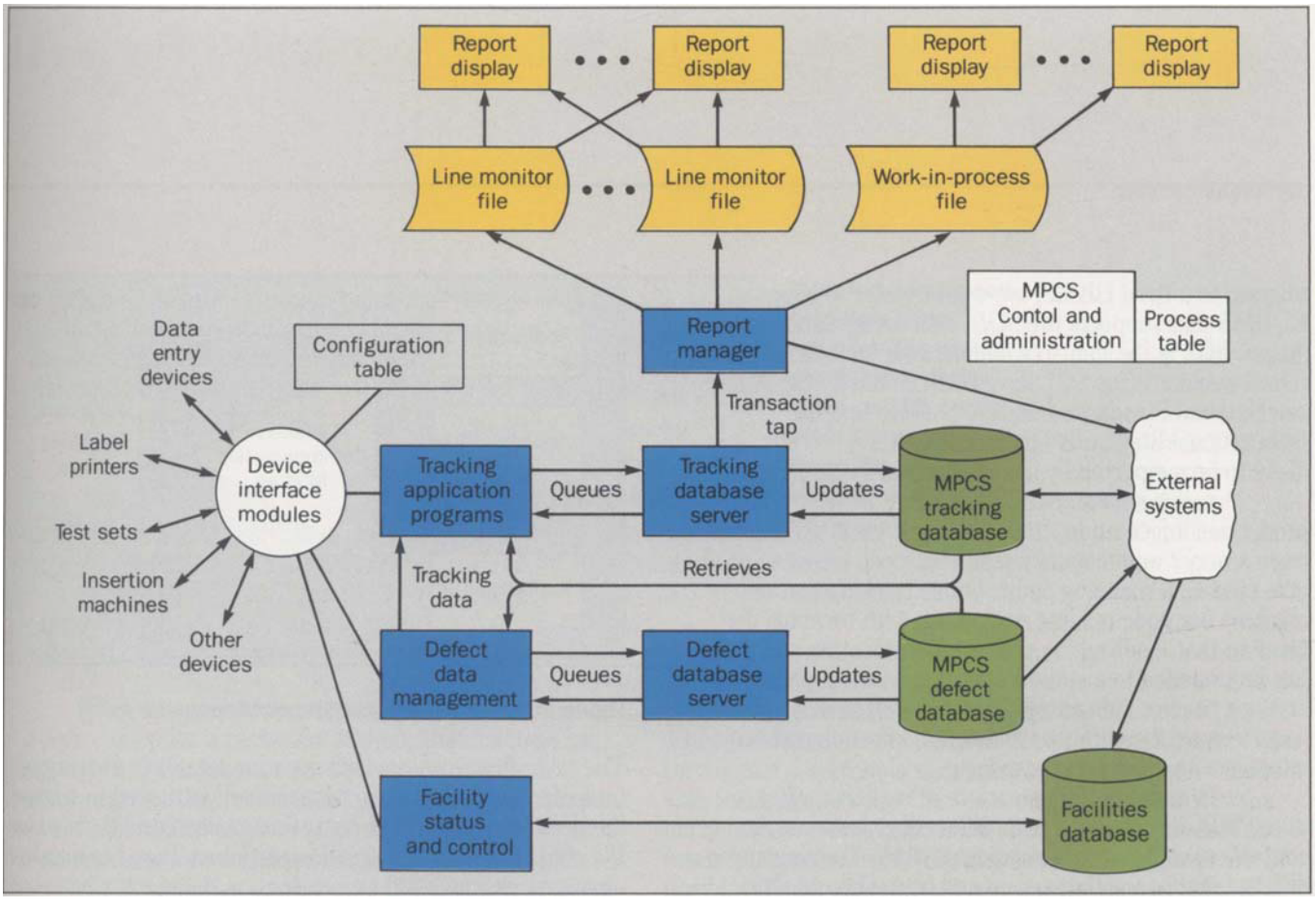

Figure 2. A simplified system architecture diagram.

MPCS must record, with a minimum of overhead, the events that take place on the shop floor. It must also be able to access data efficiently in response to queries. There are two primary factors that determine how well MPCS meets these goals: the logical organization of the database and the database management system (DBMS) used. The latter is discussed below.

We looked for a DBMS that would support the relational data model[2] to take advantage of its simple interface and the flexibility it supports. When we began developing MPCS, the only officially supported relational database management system (RDBMS) on AT&T UNIX was the *URSA*™ system.[3]

Unlike many systems, URSA allows only one process to explicitly open a database for update. The update process must either close the database or terminate for another process to gain update access. Because many MPCS application processes need to update the database, this limitation is one we needed to address.

We chose to have a single process, *dbServer,* perform all database updates. Each application program spawns an URSA process to handle queries. However, to handle updates, applications invoke database update modules. Each such module sets up a particular database update request and then invokes a communication module, *dbComm,* to transfer the request to *dbServer* over a message queue (Figure 3). The application programmer determines whether the process should wait for the results of the update, or whether it should resume as soon as the request is queued.

Because URSA is no longer a supported product, we plan to migrate to a supported RDBMS that features superior performance, reliability, and concurrency control. To minimize conversion costs should we later decide to

migrate to a third DBMS, we would prefer to adopt an RDBMS that supports the *SQL*[4] data manipulation language. SQL is the industry standard for such languages. (Draft standards for SQL have been issued by the American National Standards Institute [ANSI][5] and the International Standards Organization [ISO].[6]) Thus, it is likely to be supported by most relational systems.

Data Collection. MPCS has many ways to collect production information. In the simplest case, an unmanned laser scanner automatically reads bar code labels as products pass by a scanning point. Other tracking stations combine bar code reading equipment with terminal displays so that feedback on product identification and status can be provided to a station operator. More sophisticated stations require data entry through forms and menus or require special commands and communication protocols to interact with machine controllers.

In addition to this variety of station configurations, MPCS must cope with different types of facility controllers and bar code reading devices. Terminals also differ at stations within a shop and from shop to shop. Reusability of the application programs that collect and manage production data clearly requires insulating the application code from the data collection interface and from the details of station configurations.

Device interfaces. The MPCS device interface model provides a consistent communications interface between application programs and the manufacturing equipment or tracking devices. By using standard functions to access devices, the application program code can ignore the details of device communications. The MPCS device interface model consists minimally of three levels of software modules. Each level encapsulates all knowledge of some group of related functions, while providing a consistent interface to modules at other levels. Specifically, the levels are the device class level, the device type level, and the device input/output (I/O) level.

Each module in the device interface model depends on a system configuration table that lists all the devices with which MPCS must communicate (Table I).

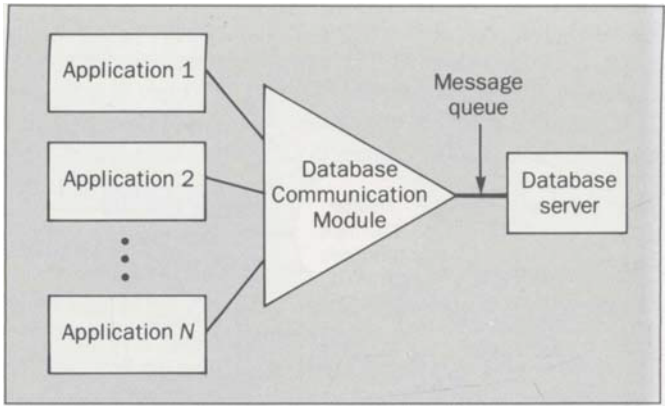

Figure 3. Serving database transactions.

The table lists each device's location (station), class type (manufacturer or model), capabilities, path used to connect the device to the MPCS control computer, and interface I/O control modes (baud rate, parity handling, canonical input processing, etc).

Application programs usually get access to devices through functions supplied by modules at the device class level. This level contains modules that group devices together by common capabilities or functions. Examples of device classes are bar code readers, insertion machines, label printers, and terminals.

Because devices are grouped functionally, each device class module consists of a set of software functions needed to take advantage of device capabilities. The insertion machine class has twenty functions including program download and machine status inquiry, while the bar code reader module is much simpler, providing only read and reply functions. In developing an application program, it is usually clear what device class modules are needed. Thus, a tracking data collection program needs the bar code reader module, and an insertion area facility controller needs the insertion module. To communicate with a device, then, the only additional information needed is a station identifier that associates a device with a location in the shop.

3b port	Conc. port	Station name	Device capab.	Device class	Device type	Input/output control modes
tty25	0	CONC1	4	9	9160	10:0:5ad:0
tty25	A	MM1	2	33	CIlaser	0:0:0:0
tty25	B	MM2	2	33	CIlaser	0:0:0:0
tty30	0	LBLPRT1	76	3	5420	526:1805:dad:1c
tty31	0	LBLPRT1	0	4	scanmark	10:0:5ab:0
tty40	0	MM1	2	16	universal	0:0:4bd:0
tty41	0	MM2	2	16	universal	0:0:4bd:0
tty42	0	RAD1	1	16	panasert	0:0:4bd:0

Table I. A sample configuration table segment. The device capabilities, device class, and I/O control fields are encoded bit fields. For example, class 16 is the insertion machine class.

When an application program invokes a function for a device class for a particular station, that function has to be mapped to the function for the appropriate device type. For many of the devices, software at the device type level is partitioned into two distinct modules: a module that maps device functions to properly formatted device commands and a module that implements the data link protocol required to communicate with a device.

Modules at the device type level are implemented using various read and write functions provided by the device I/O level. The device I/O level shields higher device interface levels from the operating system functions used to transmit and receive data. This level is useful if MPCS programs need to run on non-UNIX system computers that have C-language compilers, such as the AT&T PC 6300. Functions provided at the I/O level include communication line management (opening and closing lines, line setup, line status, and bookkeeping), reading data either in fixed numbers of bytes or in packets, and writing data. The device I/O level also ensures that multiple processes cannot concurrently access the same device line.

User interfaces. Programs that acquire input based on users' responses depend on form and menu management functions. Like the device interface modules, form and menu packages insulate application programs from screen management details while providing consistent displays to program users.

Maintaining consistency in user interfaces throughout a system is necessary to minimize user confusion and training time. This is especially important in the many instances in which users must deal with several different programs. In MPCS, the same terminal keys are used in all interactive programs to exit, request help, and edit data. Command prompts and error messages generally appear in the same relative position at the bottom of the screen. The overall consistency of MPCS screens, however, needs improvement. We are working on a more consistent user interface.

Flow Control. Within a manufacturing process, product flow requirements vary from station to station or operation to operation. Some tracking stations simply record operations performed on each product, while other stations additionally check for product routing errors and other exception conditions. MPCS treats checking and control functions as attributes of locally defined stations and operations. This affords a flexible, table-driven approach to flow control. It also limits the number of tracking programs to two: one for product unit tracking and another for lot tracking.

Reporting. A natural source of information for various production reports should be the information collected and stored in the MPCS database. DBMS performance

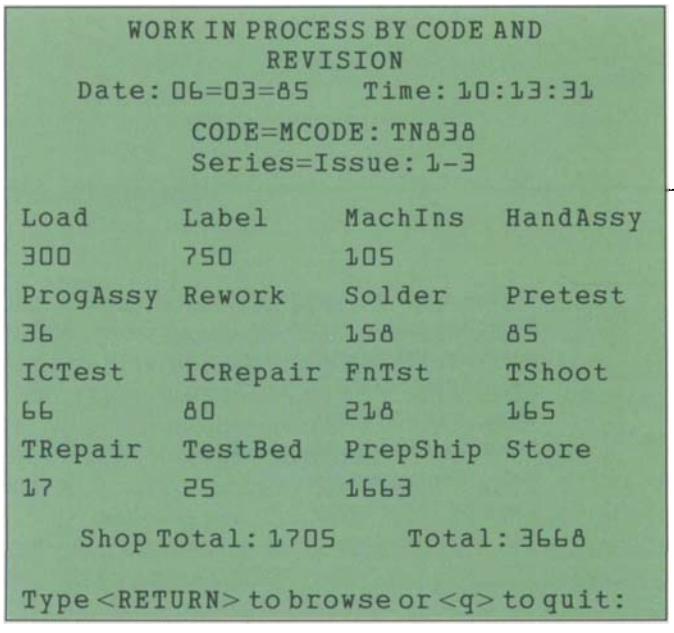

Figure 4. A work-in-process report by code and revision.

problems, however, have led to report programs that collect input from a tap into the transaction processing module of the database server. A report preprocessor reads transactions from the tap and stores them in files in a way that minimizes the amount of extra processing needed by display programs. Report programs display particular sets of these files based on commands from report users and periodically check to see if the files have been updated. Updates are automatically displayed on the report screens, so that shop operators and supervisors continuously have access to the most recent production statistics.

One drawback of this approach is that extra disk space is needed to store data already stored in a less accessible form in the MPCS database. In addition, there is a loss of flexibility in designing new reports and changing existing reports. These drawbacks have been more than offset by the ability to access current production information and the ability of MPCS to support many reporting and monitoring programs concurrently.

A sample work-in-process report is shown in Figure 4.

Administration. MPCS is intended to be a general, broadly applicable shop control system. Because of this, some degree of customization is necessary to tailor the system to meet local manufacturing needs. Customization includes selecting MPCS features, providing data about local operations, defining the system hardware architecture, and specifying some report and screen contents.

MPCS customization. Customization of MPCS may be carried out using a combination of four different techniques:

1. *Table definition.* The hardware configuration table (discussed earlier) or the shop operation table are examples of tables that must be defined locally. Recent MPCS implementations have depended on as many as 20 different tables to specify locally defined information.
2. *Run-time environment definition.* System or user profiles generally define environment variables that MPCS uses to determine file and directory organization. Environment variables give system administrators control over where MPCS is to reside on their computer system.
3. *File and directory editing.* MPCS supplies a menu package that creates menus based on the items listed in a file or on the files listed in a directory. By adding and deleting lines or files, the contents of menus can be affected. Also, monitor and report programs often use a definition file to determine what to monitor and report. Reporting schemes such as this give the shop some control over how production data are viewed.
4. *Software development.* The well-defined module interfaces used in MPCS make local addition of modules and programs possible. Module interfaces are explained in manual pages for programmer reference. Software that could be developed locally would include device interface modules for new equipment and reporting programs to provide data views other than those supplied with "standard" MPCS.

System management. Most of the frequently used administrative functions have been centralized under the control of a menu-based management program. It provides commands for maintaining database tables, controlling MPCS processes, adding and deleting devices at stations, and altering device setups. Several useful monitors are available through the same program. Monitor commands are provided for viewing the current status of MPCS processes, inspecting the system error log, monitoring

throughput of message queues, and checking current status of device interfaces.

Considerations for the Future

Many current and potential users of MPCS would like to extend MPCS to make it more useful. Some of the most common concerns are discussed in this section.

Future User Features. The basic features of MPCS can support a broader range of shop control features. MPCS product tracking programs can also collect quality data and display information to guide the operators through their work. Additional workstation control and data collection features can be added within the existing MPCS framework to handle new types of process equipment.

Product tracking data can support better control of product movement, even allowing shop floor computers to manage the conveyance of products on the shop floor. As new work is assigned to stations, displays of the jobs available to a class of workstations, with the jobs ranked by priority, can guide the selection.

Soon MPCS will be linked more closely with business resource planning systems. MPCS will also tie into systems that manage engineering information, unit and frame assembly, and customer order and warranty data. Data from MPCS will also flow into the systems that handle machine maintenance, quality analysis, and long-term reliability studies.

The real-time analysis system (RTAS) software described earlier is able to alert shop floor personnel of a decrease in product quality, which often suggests a problem in the manufacturing process. It does not, however, pinpoint what that problem might be. To address this more general problem, the *Intelligent Analysis System (IAS)*[7] is being created. This system will use trends noted by RTAS along with product design information and knowledge of the manufacturing process to try to diagnose the problem. Our hope is that IAS will successfully diagnose problems with components, shop floor equipment, and test set programs.

Better Encapsulation. As we have worked in different factory settings, we have learned that a feature such as production monitoring, which works well in one context, may not work as well in another. The underlying methods of data entry must differ because products are different or because methods of manufacture vary. Shops making a few high-volume products place different demands on shop floor computers than shops with many product types.

By identifying MPCS features that are product-specific, most of the software logic in MPCS can be generalized to apply to a broad class of factory problems. Common modules can be defined to monitor product throughput and inventory whether the product is tracked in single units or in bulk. Such modules would invoke functions that completely encapsulate core system data, concealing details of the data's physical representation. The use of these functions shields higher-level modules from changes in the details of the implementation.

Distributed Architecture. Allowing MPCS software to be distributed across multiple processors has several advantages, including:

- Performance improvements based on increased parallelism.
- Reliability improvements based on decreased risk of total system failure.
- Cost reductions based on decreased incentive to buy more hardware than needed to meet short-range projections of transaction volume.

Partitioning the MPCS architecture will also allow users greater freedom to choose only the features they require.

The fundamental concern in transforming MPCS software to support a networked configuration is determining where the database should be maintained. Choices range from reproducing the database on every processor to maintaining a centralized database on a single processor. Duplication does improve system reliability, but it encourages database inconsistency. A detailed analysis of the tradeoffs for different distributed database organizations will be necessary.

Research into a distributed version of MPCS also involves choosing a physical medium, selecting networking software, and removing from the MPCS software the

implicit assumption that a single processor is used. Decisions on the hardware and communication software should be based on suitability for the factory environment, reliability, effective throughput, and ease of interface.

Conclusions

By circumstance more than by choice, we started the work on MPCS to address specific problems with a limited set of features. With some success, we tried to write software that could be re-applied in other places. Having developed a working system, we were able to learn from users what important new features were needed to help them do their jobs. Then by working in new settings to solve similar problems, we began to better understand how to separate general problems from specific ones and to use this understanding to create new software and to reorganize existing software.

The UNIX system is well-suited for shop floor applications other than critical real-time process control. MPCS programs port easily from development to application, because the programs run in the same environment in which they were written.

The AT&T 3B computers meet the data management needs of shop floor control. Unlike remote batch computing, the 3B family of computers under the UNIX system allows MPCS software to sustain a dialogue with users. The rapid feedback possible with a time-sharing system helps ensure the accuracy of the data that operators enter. In addition, the on-line monitors present information to managers when they are ready to use it. Both factors account for much of our success with MPCS.

Factory applications are a taxing environment for computer systems, affording little time for computer system maintenance or back-up. The computer operators who run the systems need simple tools for their day-to-day tasks that do not allow unrestricted access to all parts of the system. The system administrators need more support in resolving suspected hardware and operating system faults. The UNIX system commands used by the operators and administrators of MPCS then, require strengthening.

The newer AT&T product offerings, such as the 3B2, provide commands for hardware support and UNIX administration that are easier to use. However, the older systems also require such capabilities. Apart from these problems, which should be resolved as AT&T matures as a commercial supplier of computer products, the combination of 3B computers and the UNIX system functions smoothly in shop-floor systems.

Acknowledgments

Many at ERC have contributed to creating MPCS. In addition to the authors, the original development group included Andy Baily, Dave Drake, Dave Gross, Ethel Hopkins, Bill Jensen, Joe Kelly, Dennis Mancl, Paul Murphy, Bob Parry, Mary Kay Podlecki, Randy Potter, Rao Ponamgi, George Van Ausdall, and Gordon Whitney.

Others, including Carol Feltz, Rod Gardner, Ken Gehner, Ken Larson, and Mona Yousry, developed test data collection and analysis software that link with MPCS. Some communication software was provided by Kurt Horton.

Winston Samaroo of ERC, along with Dave Hearon, Paul Baker, and Warren Bailey from Network Systems laid out the high-level plan while John Basgall, Wendy Clark, Danny Gallant, Larry Deputy, Bill Dickerson, Charles Emerson, Jim Kennedy, Barbara Rogers, Benny Sharp, Truman Hefner, Meg Joshi, John Wall, Roy Waller, and Joan Wellman helped us bring the plan to life.

We gratefully recognize the contribution of these individuals and the many others who have since become part of the overall MPCS effort.

References

1. M. A. Yousry, G. W. Sturm, and C. J. Feltz, "A Strategy for Analyzing Manufacturing Data in Real-Time," to be published, *Journal of Quality Technology.*
2. E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *CACM,* Vol. 13, No. 6, June 1970, pp. 377-387.
3. *URSA System User's Manual, Release 4.0,* Western Electric, August 1983.
4. D. D. Chamberlin et al., "SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control," *IBM Journal of Research*

and Development, Vol. 20, No. 6, November 1976, pp. 560-575.
5. "American National Standard Database Language SQL," (draft proposed) ANSI X3.135-198x, February 1985.
6. "Database Language SQL," Draft International Standard, ISO DIS 9075, May 1986.
7. B. E. Parry and M. A. Yousry, "Merging Statistical and Knowledge-Based Techniques for Process Diagnosis," *IEEE Software,* Vol. 3, No. 2, March 1986, pp. 53-54.

Biographies (continued)

supervisor in the 5ESS™ Line Unit Manufacturing department at AT&T Technologies in Oklahoma City, Oklahoma. He is responsible for the manufacturing process control center in Oklahoma City, which provides computerized support for all circuit pack manufacturing. Mr. Stocking, who joined AT&T in 1975, is a supervisor in the Manufacturing Information Automation department at the Engineering Research Center. He received a B.S. in chemical engineering from Rensselaer Polytechnic Institute and a Ph.D. in chemical engineering from the University of California, Berkeley. Mr. Swartz joined AT&T in 1980 and is a member of the technical staff in the Manufacturing Information Automation department at the Engineering Research Center. He received an A.B. in computer science from Cornell University and an M.S. in computer science from Rutgers—The State University. Mr. Trombly, who joined AT&T in 1978, is an assistant manager at the AT&T Merrimack Valley Works in Massachusetts. Previously, he was a supervisor at the Engineering Research Center. He holds a B.S. in computers and systems engineering and an M.S.E.E. from Rensselaer Polytechnic Institute.

(Manuscript received February 19, 1986)