The Development of a large Application System for the HP3000 Computer

Nancy Federman
Robert Steiner
Manufacturing Systems Operation

INTRODUCTION

Design, development, and market analysis are only the initial steps in producing a profitable product. Manufacture of the product is the next step. Do you really know what is involved? The first step is to specify the engineering data. What is the structure of the product? You need to determine the parts and subassemblies that comprise the product and the quantities of each required per unit. Which of these components will you manufacture? Which will you subcontract out to other manufacturing companies, and which will you purchase?

Now that you have some of the engineering specifications the next step is to determine the manufacturing procedure - how is the product to be made? You will need to decide the manufacturing operations necessary to construct the product and then specify the location in your shop where the work will be done (workstations). The sequence of these production steps, called the routing, plus the labor and material required at each operation must be detailed.

The product is specified and the manufacturing plant and process are established. You are now ready to begin production. How many products should you build? You need to consider customer demand, current and forecast, as well as the capacity constraints of the factory. Once your production plan is established you need to determine your material requirements. How many component parts do you need to meet your production schedule? You also need to determine the schedule of production for the manufactured components. When do the purchased parts need to be ordered? You need to balance the desire to have components always available with the economic necessity of keeping inventory levels as low as possible.

Controls are needed to monitor the flow of materials in the stockroom and on the factory floor. When do the component parts need to be issued to the production lines and what quantities are required? You will want to monitor the shop floor and track the work in process. How much material is wasted? How efficient is your work force? When the products are finally completed you need to keep them in a finished goods inventory and track their sales.

Finally, you will be concerned with calculating the costs - labor, material and overhead. The production costs will help you decide your pricing policies and determine your profitability.

From this brief and simplistic overview of a manufacturing operation it should be clear that there are many intricate relationships to deal with. Effective and efficient management of a manufacturing operation is difficult to achieve and many neighborhood businesses as well as sophisticated manufacturing companies are turning to computers for help.

MATERIALS MANAGEMENT/3000

Hewlett-Packard is among the vendors providing application systems for manufacturing management - a manufacturing company offering a solution to the problems of manufacturing companies. Materials Management/3000 is an interactive material planning and control system designed to make it easier to deal with the complexities of operating a manufacturing company. It is primarily designed for manufacturers who build standard products to stock in discrete manufacturing steps (fabricators and assemblers) These companies have a significant investment in inventory. Materials Management/3000 can help them balance their inventory levels with customer demand for timely shipments to optimize their dollar investment.

The complexity of the manufacturing environment can also be seen in the internal complexity of the software product which provides the solution. Materials Management/3000 consists of over 400,000 lines of SPL code which make up 161 transactions referencing 9 data bases. There are 291 screens, both transaction screens and menu screens. There are also 168 batch programs. The application data bases and screens can be customized by the user. Materials Management/3000 operates on any of the HP3000 family of computers and uses the 264X family of terminals. The system uses HP's IMAGE data base management system and HP's V/3000 screen handler.

Materials Management/3000 provides a solution to the previously outlined problems of manufacturing companies by supplying 10 major modules. Their inter-relation is diagrammed in Figure 1.

Master Production Scheduling: MPS is an on-line management planning and production scheduling tool. It is used by the master scheduler to generate a production schedule for the plant's marketable products and to set the mix for the product options. Input to the module includes the current customer orders, forecast customer orders, the current production schedule and the current level of product inventory. The output of the MPS calculations is called the master production schedule. This schedule contains suggested manufacturing orders including quantities and starting dates. The schedule is then input to the Material Requirements Planning (MRP) module to plan the manufacture and purchase of the required component parts. MPS also includes a "WHAT IF" simulation capability which allows the user to generate tentative schedules, and view the impact of modifications on the current schedule.

Rough Cut Resource Planning: Rouch cut resource planning (RPP) is a management tool used by the master scheduler to compare the resources needed to implement the master schedule with the available critical resources to help produce a realistic master schedule. The user specifies the critical resource requirements for each master schedule part, and the maximum capacity of these resources. Examples of critical resources are labor hours, floor space, dollar investment in work in process inventory, material supplies, etc. The RPP reports highlight the capacity constraints and help the user to resolve competing demands for the critical resources. An on-line RPP report can also be used to help evaluate simulated master schedules by comparing their resource requirements to the requirements of the current schedule.

Material Requirements Planning: MRP simulates the complex flow of materials required to manufacture products and generates a material plan. MRP planning starts with up-to-date information about the current inventory levels and the planned production requirements. Using part and bill of material information the material requirments for each part are calculated. The plan is started with the highest level assemblies and then proceeds through the lowest level parts. MRP will reschedule current work and purchase orders and suggest new orders as necessary to meet the demand. MRP is a regenerative system - a complete material plan is generated every time MRP is run.

Parts and Bills of Material: This module provides on-line maintenance of engineering, accounting, and planning information about each part and product, and information on how the parts relate to one another to form the product structure (bill of material). Responsibility for maintaining this data will normally be shared among several departments - accounting, engineering specifications, and planning. Part and structure data can be reviewed on-line or through printed reports. The part and structure data is used by many of the other modules in Materials Management/3000, including MPS and MRP.

Routings and Workcenters: The Bill of Material defines the parts and subassemblies that comprise a product but doesn't document how the various components are actually assembled. The routings and workcenter module maintains information that describes the locations where the products are made (workcenters) and the proper sequence of manufacturing (routings) This information is used to generate cost information for the Standard Product Cost module, and to help develop detailed production schedules. Responsibility for this data usually resides with manufacturing specifications.

Standard Product Cost: SPC provides manufacturers with the capability to accurately calculate the standard costs associated with the manufacture of each product. All current cost information may be edited and reviewed on-line. The standard cost of a product is determined by accumulating all relevant material, labor and overhead costs for the components of the product as well as the costs associated with the actual construction of the finished product. These standards can be used to determine product pricing and profitability. Marketing as well as the material manager would use this data.

Material Issues and Receipts: This module helps to control stockroom inventory by maintain timely and accurate records of all actions that affect inventory balances. The data includes receipts of work orders or purchase orders, material issues from stock to a particular work order, filling of a backorder, or an unplanned issue. All record keeping and updating is done on-line and a record of all inventory activity is kept on-line for a user-specified period of time as an audit trail. Stock room personnel are the primary users of this system.

Inventory Balance Management: Inventory balance management is a module to maintain information about inventory balances and the warehouse locations where the inventory is stored. The current inventory status can be affected by three types of transactions - material movement, inventory counts, and stock adjustments. All three types of transactions will trigger an immediate update of the inventory counts as well as create an audit trail record. All udpates are done automatically in an on-line mode. Current inventory balance data from this module is used by MPS and MRP to determine the next master schedule and the next material plan. All activity that affects inventory status can be reviewed on-line. An inventory value report is also available. Materials Management/3000 also allows for multiple stock locations - a separate on-hand balance can be maintained for each stock location in each warehouse. This system also helps with the actual counting of inventory which is periodically used to verify the inventory totals.

Work Order Control: A work order is an internal factory authorization to build a specified quantity of a particular subassembly by a specified date. All work orders require the issue of on-hand inventory for their completion. Prior reservation of on-hand inventory is the best method of preventing shortages at the time of issue. Allocation, or logical reservation, of on-hand inventory will help predict and prevent these shortages. The timely notification of exceptions to the material plan can allow corrective action before the results become disastrous. The output of this tracking system is the reports noting exception conditions. The materials manager can then act on these reports. The actual issuing of parts and work orders, and the actual receipt of finished products is accomplished by using the material issues and receipts module. MRP is a prime user of information from this system.

Purchase Order Tracking: A purchase order represents a scheduled receipt for purchased items. Entering a purchase order requires the entry of more information than that required on a work order - e.g. vendor information, shipping information, price information. It is also possible to group multiple delivery dates and/or multiple parts on the same purchase order. Purchase Order Tracking system monitors these scheduled receipts and also maintains vendor information. Users can get a report on the current orders for a particular vendor, or the value of outstanding purshases by scheduled receipt date. The purchasing department and the materials manager would normally use this module.

What distinquishes Materials Management/3000 from other materials management systems in the marketplace is not just the product features but the design and implementation philosophy behind it. This philosophy evolved from previous experience with application systems, a knowledge of the competitive marketplace, and first-hand experience with manufacturing company operations. The design philosophy can be summarized as providing a functionally complete solution which fits the business practices of the user, is friendly and easy to use, and is supportable by HP.

APPLICATION STRUCTURE

Materials Management/3000 evolved from a previous product, MFG/3000, which was released in December of 1977. MFG/3000 began in the HP3000 manufacturing area as a computerized solution to HP's internal materials management problems. As the system was completed and put into use it became clear that other medium to large manufacturing companies were having the same sorts of problems. It was decided to to turn the home-grown system into a product.

MFG/3000 was sold both as an object code product and a source code product. A source code product is one in which the actual computer programs are sold to the user. The user then can modify the code directly if they wish to implement any modifications. A source code product is difficult for the factory to enhance and puts a support burden on the customer. The bug fixes and factory enhancements must be sent to the user in source code format. The customer must then implement the changes manually. If the customer has made modifications to the source code the changes from HP may not be compatible. It is also very difficult for the factory to support a source code product. If the user reports a bug the source of the error is difficult and time-consuming to detect since the fault could be in the original code or in the user-modified code.

An object code product is one in which only the executable code is sent to the customer. The user cannot make any modifications to this product and so has gained factory support at the price of flexibility and local user control. The idea of an object code product is a difficult one to "sell" to the customer. The application cannot be tailored to fit the individual needs of the customer. On the other hand, it is not possible for the factory to anticipate all the detailed and distinctive capabilities peculiar to any particular customer.

HP's solution was to develop an object code product that was user customizable. The big advantage is that HP can fully support and enhance the product while the user can tailor the application to suit their individual needs.

Most of MFG/3000 customers had purchased the object code version. The majority of those customers that did purchase the source code were interested in changing the field edits, the data item characteristics, and the data items in the reports, not the program logic. The implementation strategy was to include standard and accepted functions in the program code and provide software tools to allow the customers to tailor, or "customize," the system to fit their own individual requirements. So the program code was separated from the data item characteristics, the screen formats, and the other parameters that characterize each particular installation of the product. This would allow the factory to maintain the logic of the programs while the user could tailor the edits and data item characteristics, as well as modify the appearance of the application.

The next problem to tackle was to develop an efficient mechanism to "interpret" the user-supplied execution-time parameters. The first design decision was to put the customizable information into tables. A table-driven application was chosen over a compiled application because implementation of the latter design meant the development of a new language, a challenging task in itself. And control over the customer use of the language would be difficult. Data item customization via re-compilation of all the source code programs would be time-consuming and prone to error. The factory would also lose some control over the integrity of the application once the source code was distributed to the customers. A table-driven applciation seemed the wise choice.

Experience with MFG/3000, with its rudimentary edit table, led to the decision to expand this table and add to it a data dictionary which would contain tha structure of the data base, the specification of the data items itself, and the format of the screens and reports. Now that the contents of the tables had been agreed upon the next problem was to provide efficient execution-time access to the data in the tables.

Tables implemented in files or data bases would be too slow to access at execution time. If the tables were places on the stack too much memory would be used. Extra Data Segments could provide efficient execution-time access with good memory utilization. The design agreed upon put the "source" version of the application parameters into a data base, so the user could edit them. This data was them compiled into extra data segments for execution-time access.

The only problem with extra data segments is that they are not sharable across sessions. An important underlying assumption that the applciation would have many users. So the application had to be designed to allow for multiple users. The solution was to develop a control program to manage all the Materials Management/3000 user terminals so that they would appear to the MPE operating system as just one session. This solution fit in nicely with the design goal to have a dedicated system - the user would interface with a program that was optimized for the non-computer professional instead of with MPE. This control program would automate some of the standard control functions, such as scheduling terminals and initiating batch jobs.

Some of the tools necessary to implement an object code user-customizable applciation were already available. IMAGE/3000, the data base subsystem, eliminates data redundancy and resulting maintenance problems. V/3000, the forms data entry subsystem, makes it easy to design and implement a friendly, consistent user interface. The MPE message system provides a facility for creating customizable report headings and user error messages.

To meet our objectives it was necessary to develop two more tools, the Application Customizer and the Application Monitor. The Customizer provides a method for the customer to tailor Materials Management/3000 to fit an individual environment, and the Monitor automates many of the day-to-day administrative functions ususally performed by an operations staff. The Monitor accomplishes its function by starting and stopping terminals at predetermined times and scheduling background jobs such as MRP to be run on a regular basis. System security is controllable because users may not use the application (i.e., Materials Management/3000) unless system administrator has instructed the Monitor to start the application on a specific terminal. The Monitor also includes review capability of the application-generated error messages and other system activity, such as the background job schedule or current terminal activity. To the application program, the Monitor provides many services normally associated with operating systems. The application programs may request services such as process initiation, interprocess communcation, and resource allocation for on-line terminals and printers. The application designer can concenterate on solving application-oriented problems and call on the monitor to provide other functions that are necessary but not directly involved with materials management functions.

The key componenet of a customizable application is the application data dictionary, which serves as a repository for application-dependent information such as data item characteristics, data base schemas, V/3000 form descriptions, security passwords, terminal configuration, and background job schedules. The Application Customizer was designed to maintain the data dictionary, and it performs two major functions. The first is a facility for customers to alter or customize the application system using a simple menu-driven fill-in-the-blanks sequence of forms. Since this is the part of the Customizer most visible to the customer, the bulk of the design effort went into making customization functions simple and easy to understand by nonprogrammers. The second function performed by the Customizer is to transform the information present in the data dictionary from data structures suitable for run-time access by the application programs. These transformed data structures, collectively known as the run-time application ta dictionary, are used by the application programs to determine the values of all customizable parameters in the system.

Fig. 2 shows how the Customizer, the Monitor, IMAGE/3000 V/3000 and the application software interact.

CUSTOMIZATION TECHNIQUES

The rest of this article describes some of the methods used by the designers of Materials Management/3000 to design programs that can operate efficiently in a customizable environment. Because Materials Management/3000 is a customizable application, the customer has the ability to change many of the characteristics of the system by modifying items in the application data dictionary, rather than using the traditional time-consuming and error-prone method of modifying source code and compiling programs. Designing customizable applications is therefore complicated by the fact that many assumptions traditionally made by application programmers are not true. Customers may modify data item characteristics, add and delete items, modify the on-line user interface, and define additional processing.

Changing Data Item Characteristics

An assumption traditionally made is that once a data item is defined, its characteristics will not change. In a customizable environment that assumption is no longer valid. Because it is possible for the customer to alter the length, type and precision of any field, the application program has no idea what the characteristics of fields will be until the program is executing. For example, there are three broad categories of data type used by Materials Management/3000: alphanumeric strings, numeric fields, and date fields. An application designer may assume a data item is one of these three general types, but cannot know the specific format of the field. Numeric fields may be any of five numeric data types: display numeric (with explicit sign and decimal point), zoned numer (with implicit decimal point and sign overpunch), packed decimal, 16-bit integer, and 32-bit integer. Any numeric field may be changed to any other numeric type and the length and the precision (number of deicmal places) of display numeric, zoned numeric, and packed deccimal numbers may also be altered by the customer.

The solution is to place field definitions in tables that are accessed by the application program at execution time. These tables form the run-time application data dicitonary generated by the Application Customizer and are accessed only by a set of Application Customizer routines called intrinsics. This enables the designer to code the application without specific knowledge of the structure of the tables. As the Application Customizer is enhanced, the tables may change, but the application programs will not have to be modified because the intrinsics insulate the application from the Application Customizer.

A field may have several occurrences in an application, each having slightly different characteristics. For example, a numeric field may be present on an IMAGE data set, and also on a data entry screen defined for a transaction that updates the data set. The item on the screen will be defined as being display numeric type, with a length of ten digits including two decimal places, the same item on the data set will be defined as being packed decimal, with a length of 15 digits including four decimal places. The designer can develop customizable programs without concentrating on these differences because of the intrinsics provided by the Application Customizer to handle all arithmetic and data movement operations.

A table lookup is required every time a data item is maniputlated by the application. Materials Management/3000 is structuared to provide the the best response time for users who perform the same transaction many times, using few or no other transactions. An example is loading dock personnel who perform"receive stock" transactions almost exclusively. When a transaction is entered, only that portion of the customizer tables that contains data item defiinitions are used by the trasnaction is moved to the program data area. The data item definitions remain in memory until the user branches to another transaction. With the needed data item definitions in program data memory, Customizer intrinsics may access data defininitions with a minimum of overheaad. This conserves memory and provides fast response time for subsequent executions.

Since there are Customizer intrinsics that perform data movement and arithmetic operations, instead of coding SPL statements to manipulate data, the application designer codes calls to intrinsics that add, subtract, multiply, or divide numeric data items, and move numeric or alphanumeric items. These intrinsics reference the data item definition tables, performing data validation, decimal point normalization, data type conversion, and security checking. If an error prevents proper processing, the intrinsic returns an appropriate error code, and the user can be informed.

Modifying Fields

In addition to changing data item characteristics, it is possible for the customer to add and delete some fields appearing on screens and data sets. Materials Management/3000 is designed to perform specific inventory control functions, so a working set of data items must be present for the application to perform its function properly. These data items are defined as critical to the application and may not be deleted by the customer. Other data items in the released product are included for optional processing and may be deleted by the customer for reasons of efficiency or to prevent user confusion. On the other hand, a customer may want to adapt the application to perform additional functions not anticipated by the application designers. This will require the addition of data items to data entry screens and data sets. A method must be used to represent the association of data items with screens and data sets to the application programs.

Fortunately, much of the processing in Materials Management/3000 and many other data processing applications involves the movement of complete records from place to place. For example, "add" transactions simply construct a record from the data items entered on a data entry screen, and after appropriate validation edits, move the record to an IMAGE data set. "Change" transactions retrieve a record from a data set, update it with fields entered from the screen, and then move the record back to the data set. When adding or deleting a data item on a data set or screen both the designer and the customer must associate the item with a specific record format. Record formats are nothing more than collections of data item definitions that correspond to the fields on a data set or data entry screen record. A data entry screen record and the corresponding data set record it will update will contain many of the same data items, although they may have different characterisitics. Since it is unknown until execution time exactly what items will be present on a given record, the Application Customizer provides an intrinsic that moves corresponding data items from one record to another.

The operation of the MOVE CORRESPONDING intrinsic is very simple. The intrinsic is passed the record format definitions present in the source format, the instrinsic searches the target format for a corresponding item definition. If a match occurs, the data is moved from the source to the target record, changing the data type, length, and precision if necessary. This process continues until all corresponding fields have been moved from the source to the target record. The MOVE CORRESPONDING intrinsic allows the designer to think on a record level, not being conerned with individual data items. This makes it possible for the customer to add and delete noncritical data itmes at will.

Fig. 3 shows an example of MOVE CORRESPONDIN operation. Each record is described by a format maintained by the Application Customizer. Every item is assigned a unique item number by the Customizer. This item number is used to identify all occurrences of an item. Each format consists of a format header, which contains pointers and information concerning other control structures, and a collection of item definitions, organized in ascending item-number order. The MOVE CORRESPONDING intrinsic performs its function for each item in the source format (in this case the screen format) which has a matching item definition in the target format (the data set format). The intrinsic locates the field and determines its length, type, and precision, using information stored in the item definition. In this example, the source field is located at byte 0 and is ten bytes long. An item type code of 3 indicates that the field is in display numeric format and the precision is two decimal places.

The target field is located at byte 54 of the data set record and is eigth bytes in length. An item type code of 5 indicates that the field is in packed decimal format, and the precision is four decimal places. MOVE CORRESPONDING copies the field from the screen record to the data set recod, changing the type, length, and precision of the data according to the item definition.

Changing Screens

In addition to changing field and record characteristics, the customer has the ability to modify the appearance of the application itself. Data entry screen appearance, and even the sequence of screens may be altered by the customer.

V/3000 provides a relatively simple method for altering screen appearance. Screens may be redesigned by repainting them using a few control character sequences on HP's 26xx series terminals. This gives the customer the power to alter screens so they look like forms that are presently in use, lessening the technology shock thay many users experience. Screen alterations are then entered into the run-time application dictionary via the customizer and translated into updated record format definitions. The application program is thereby insulated from cosmetic changes to screens. The MOVE CORRESPONDING and other Customizer intrinsics handle changes in data field order as easily as additions and deletions.

In Materials Management/3000, screens corresponding to transactions are at the bottom of a large tree of menus. The 26xx terminal series has eight dynamically definable softkeys. These keys are used by the application as the primary method of moving from screen to screen. The top of each screen in Materials Management/3000 contains eight labels, each corresponding to a data entry screen or a menu. The user may navigate through the menu tree by pressing a softkey that will cause the application to transfer to the desired transaction, or to a menu that will list seven other choices. The eighth function key is always labeled EXIT and takes the user to the screen's parent.

The customer has the ability to modify these labels through the Customizer, creating subtrees for different users. For example, security reasons may require that a customer prevent stock room personnel from altering any engineering data. By removing any labels that identify transactions dealing with engineering data, it is possible to restrict the stockroom personnel to a closed set of transactions.

The application determines softkey definitions by looking up values in a screen sequence table, which is part of the run-time application dictionary and is accessed by Customizer intrinsics. An entry in the sequence table is associated with every screen. Before displaying a screen, the corresponding entry is moved to the program data area. If the user presses a softkey, the application looks up the value that corresponds to the key pressed and transfers control to the appropriate screen or menu. This allows the customer to be very flexible in tailoring the system and relieves the designer of the burden of determining the screen structure while coding.

An additional feature becomes very powerful for experiendced users of Materials Management/3000. A 16-character input field called the command window is present on all menu screens. If the function desired by the user is not directly accessible from a menu, the desired function name may be entered into the command window and the corresponding screen will be accessed directly, eliminating the need to navigate through the menu tree. Whenever the application detects an entry in the command window, a Customizer intrinsic retrieves the appropriate value, effectively providing a ninth softkey. The command window may be altered via the customizer and V/3000 to accept only selected labels. This provides an additional measure of security, while providing the means for the experienced user to travel rapidly from screen to screen.

Processing Logic Customization

It is impossible for the designers of a general-purpose application to anticipate the needs of every customer. Customers will almost always want the application to do some additional processing, beyond the capabilities of the standard product. With noncustomizable applications, the customer would either have to purchase source code and modify it, or live with the standard product. Materials Management/3000 provides two methods of modification. The first involves V/3000 and the second involves the Application Customizer. V/3000 provides a set of powerful functions, including: checking for minimum length, data type checks, range checks, pattern checks, and data formatting. However these functions apply only to data entered on the screen records. To allow customer-defined manipulation and movement of data between screens and data sets, a set of functions called processing specifications may be entered using the Customizer.

Processing specifications are defined by the customer for each transaction where additional processing is desired. Simple commmands allow the user to add, subtract, multiply, divide, and move data items. These commands are compiled and placed in tables that are accessed by Customizer intrinsics at execution time. In most of the product, each transaction is structured so that after all normal processing occurs but before any data sets are updated, the processing specification interpreter is called. This is a Customizer intrinsic that performs the operations indicated by the customer-entered statements. It is possible to alter almost any data item on any data set that is to be updated by a transaction. This tool allows the customer to extend the usefulness of the appplication program to areas that were not originally anticipated by the designer.

Fig. 4 shows how customer processing specifications are implemented. The format header of the screen format contains a pointer to any processing specifications the customer may have defined for the transaction. All processing specifications are generated by the Customizer and placed in a processing specification table, which resides in an extra data segment. The processing specification interpreter uses the pointer and length fields in the format header to locate and move the processing specifications defined for this transaction to the stack. The Customizer generates an intermediate language in the form of triples, which consist of an operation code and two operands. Each operand field is either a constant, a register, or a format/field number combination. In this example, the customer wishes to convert the value entered in field 135 from pounds to grams and accumulate the result in field 222, which is described in format 14. This might occur in the situation where the customer wishes to record the year-to-date quantity ordered for management reporting. Field 135, described in format 22, corresponds to the quantity-ordered, which is accumulated on some other record for use in preparing periodic management reports. The normal unit of measure for ordering is pounds, but for some reason, management has decided to accumulate the total quantity in grams. The first triple moves the constant 454 to register 1. The second triple multiplies the contents of field 135 by the contents of register 1, and places the result back in the register. This converts the value of the field from pounds to grams. The contents of the register are then added to the contents of field 222 in the third triple. Upon returning from the processing specification interpreter, the transaction will update all of the effected data sets. This method of implementation allows the customer to add to or override the processing specified by the application designers.

Local Languages

HP's market for manufacturing applications is worldwide. The application designer cannot assume that the users of an application understand the English language. Materials Management/3000 is designed to be completely localized to any language supported by the 26xx series terminal without reprogramming. Localization may be accomplished by translating the screens using V/3000, by modifying report headings and error messages stored in message catalogs, and modifying other literals maintained in the application data dictionary. Materials Management/3000 uses many single-word literals to control processing. For example, a user may enter engineering information about a part, such as whether it is normally purchased or fabricated. The English version of Materials Management/3000 codes this information as P or F on the data base. The literals P and F will have different interpretations in other languages. Therefore the customizer maintains another table containing all literals defined by the application designer. When manipulating literals entered by users, the application must first look up the current value of the literal. The table is loaded into the program data area and accessed by Application Customizer intrinsics. Because the table is located in the program data area and accessed directly, there is very little additional overhead. NonEnglish-speaking customers have an application product that is easily understandable by their users, and the support burden is minimized for HP because only one version of an application system needs to be supported instead of one for each language.

Security Checking

An advantage of manipulating data in an interpretive mode is that other functions may be added with a minimum of effort by the designer. One example is security checking. Many auditors demand that security access be carried down to the data item level. In Materials Management/3000, each user is assigned a password that will grant that user access to only the data items that he or she is expected to review or update. The password is entered only once on a special security screen. The user may view only screens that contain data items for which that individual has access, and on screens that may be accessed, not all data items may be reviewed or updated. This allows the user to see and manipulate only the authorized items.

This type of security would require a lot of design and coding effort in a conventional system. In Materials Management/3000 the Customizer intrinsics that manipulate data also perform a security check. The Customizer maintains a table containing all valid passwords along with a list of data items to which that password grants access. Each time a Customizer intrinsic accesses a data item, a table lookup is performed. If the user does not have access to the item, an error message is displayed on the terminal. This powerful feature is implemented with a minimum of overhead and design effort.

CONCLUSIONS

Materials Management/3000 is HP's first user-customizable, factory-supportable application system. The team that designed and implemented Materials Management/3000 has verified that an object code user-customizable product is a good idea and that it works. The performance of such an application can be acceptable. As the installed base expands we are gaining a better understanding of our customer needs. In general, the customers really like the product. They just want HP to expand its capabilities. Two examples are discussed next.

We are looking into providing more customization features. A frequent request is to allow the user programs greater access to the application data base. This could be accomplished by allowing the user access to our customizer instrinsics. The data base would only be accessed through these intrinsics and therefore the integrity of the data base could be insured. Users are also requesting the ability to add data sets, and gain more flexibility in associating data items. Even with expanded capabilites, though, some customers will still want more flexibility. For example, they want to write their own programs to perform expanded data validation and they want the application to call these programs. Another feature request is program logic customization. The user would be able to select among pre-coded algorithms.

In another area, the design team is exploring the implications of a distributed application. Currently, Materials Management/3000 is a single application system that runs on one HP3000 machine. A distributed application system would involve functions and data spread out among several machines. We need to understand how to distribute the data and how to handle the customization of data distributed throughout an application network. These are only two areas of research. There is a lot to do.

Built on the technology of an application monitor and an application customizer, Materials Management/3000 is HP's first step toward providing a total solution to the problems of manufacturing companies.

ACKNOWLEDGMENTS

We would like to recognize the lab and marketing teams who spent two and one-half years designing, implementing, documenting, and marketing the product. Special thanks go to Steve Baker, Dick Dolan, Beth Eikenbary, Mike Kalashian, and Bob Poulos for their contributions to this article.

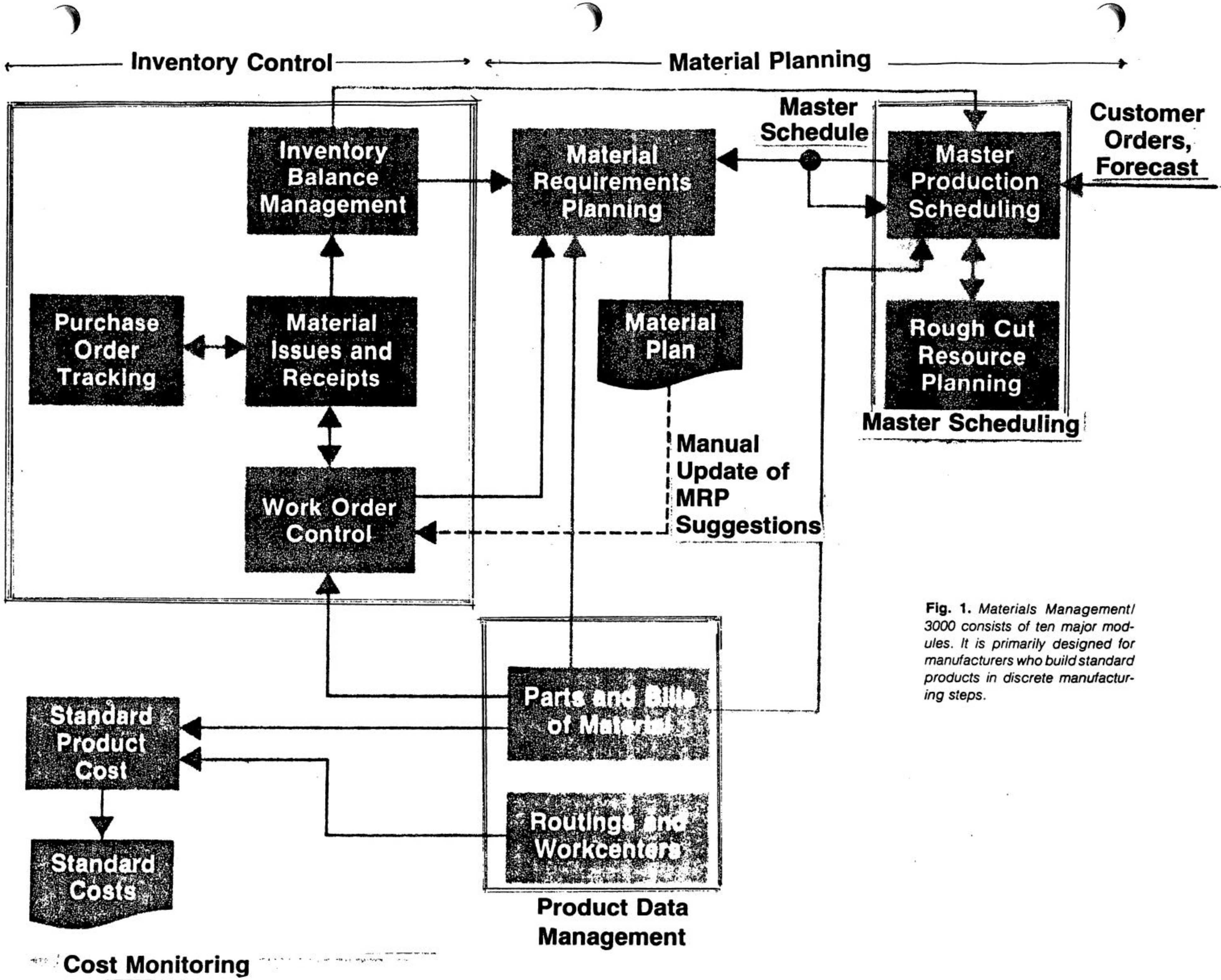

Fig. 1. *Materials Management/3000 consists of ten major modules. It is primarily designed for manufacturers who build standard products in discrete manufacturing steps.*

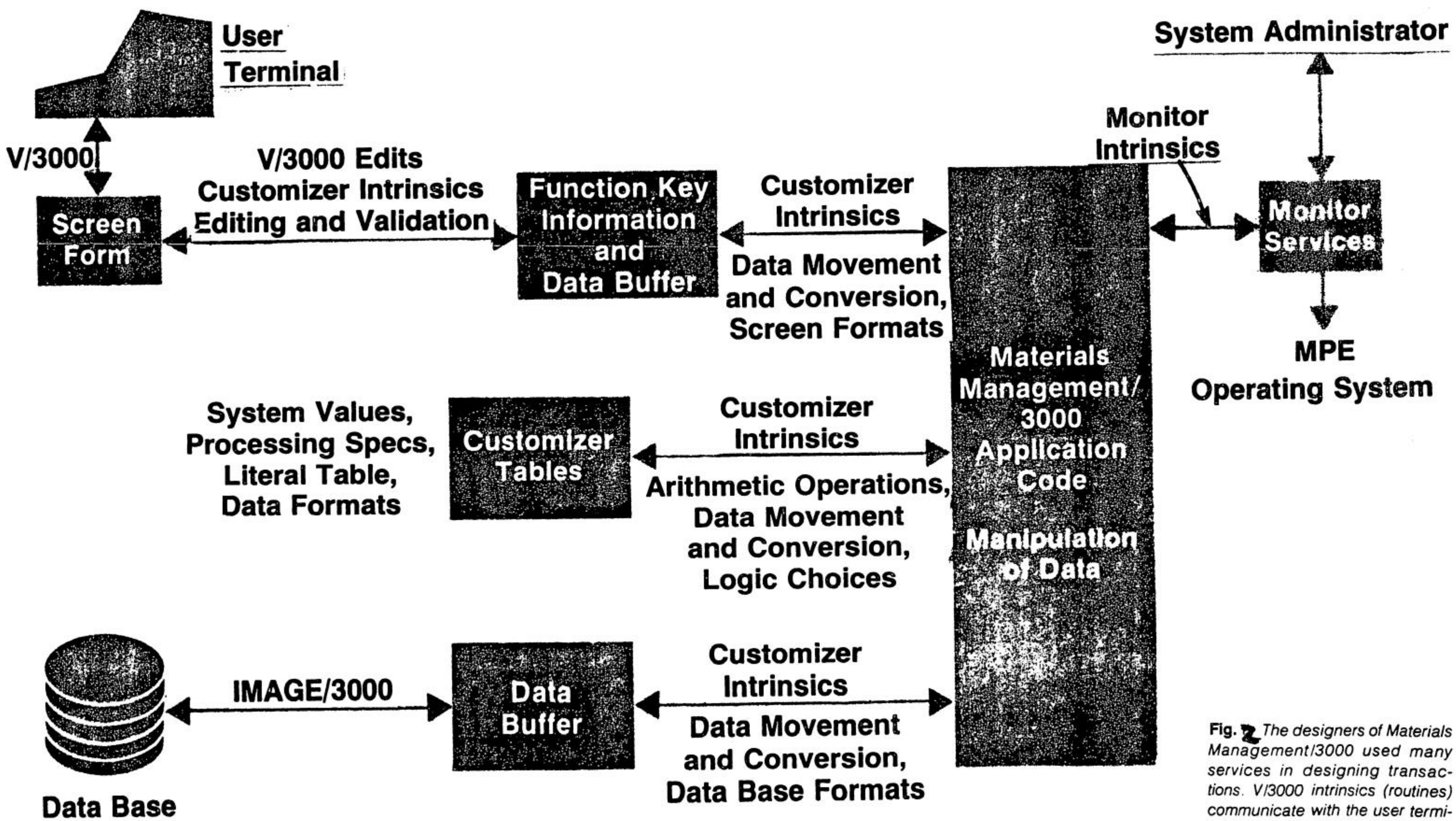

Fig. 2 *The designers of Materials Management/3000 used many services in designing transactions. V/3000 intrinsics (routines) communicate with the user terminal. IMAGE/3000 intrinsics store and retrieve data. Application Customizer intrinsics retrieve data item definitions, screen formats, data set formats, and customer-added processing specifications. Customizer intrinsics also manipulate any data items whose characteristics are unknown to the application designer and must be looked up [in th]e Customizer tables.*

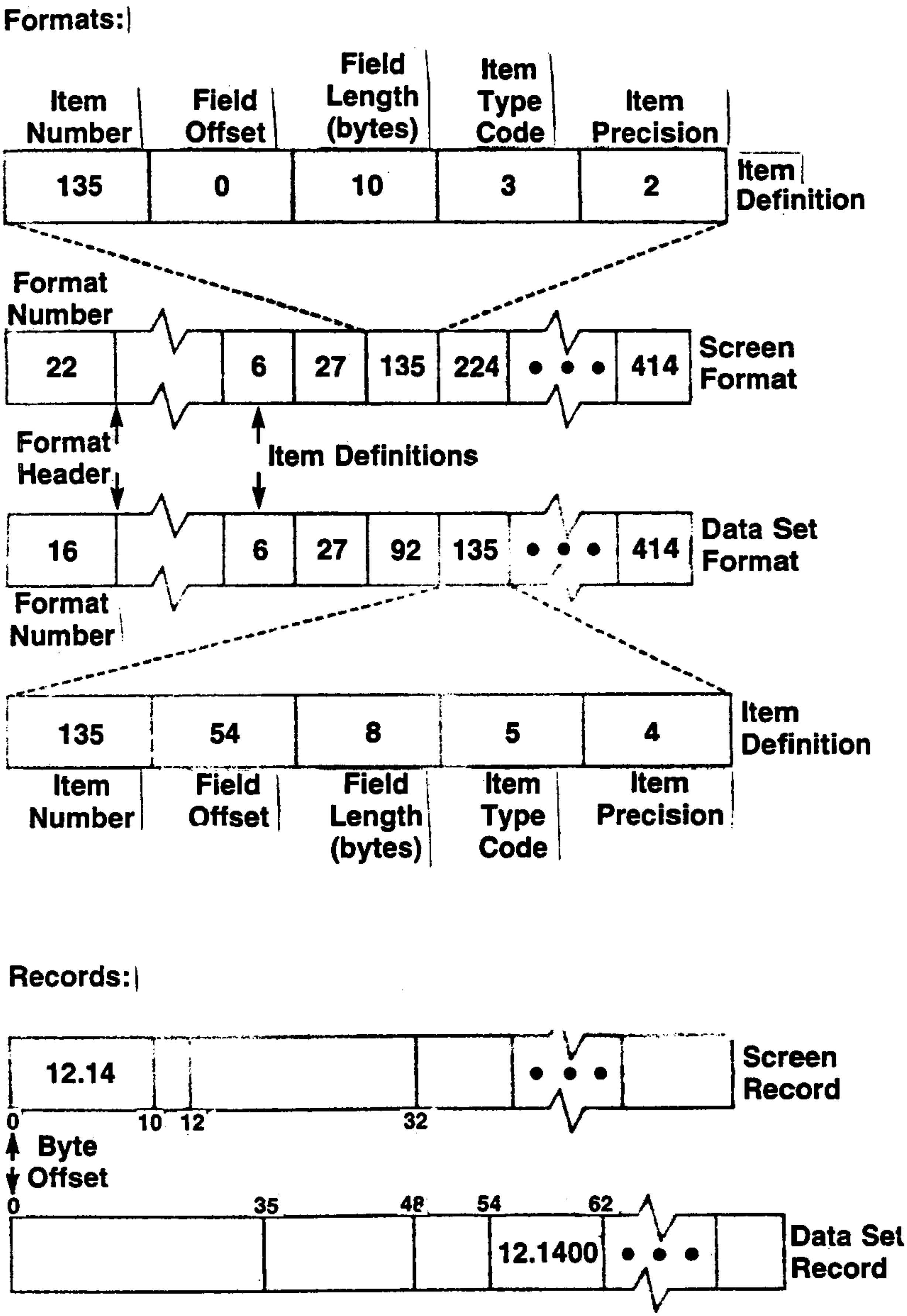

Fig. 3 An example of the operation of the MOVE CORRESPONDING Customizer intrinsic. See text for details.

Format Header

Format Number		Processing Specification Pointer	Processing Specification Length	
22		•	5 * 3	

3 = number of statements

Operation Code	Operand 1 Format Number	Operand 1 Field Number	Operand 2 Format Number	Operand 2 Field Number
0	−2	454	−1	1
3	22	135	−1	1
1	−1	1	14	222

Processing Specification Table

Operation Codes

- **0** Move
- **1** Add
- **2** Subtract
- **3** Multiply
- **4** Divide

Fig. 4 *An example showing how customer-specified processing is implemented. See text for details.*