PDM: A requirements methodology for software system enhancements

by R. G. Mays
L. S. Orzech
W. A. Ciarfella
R. W. Phillips

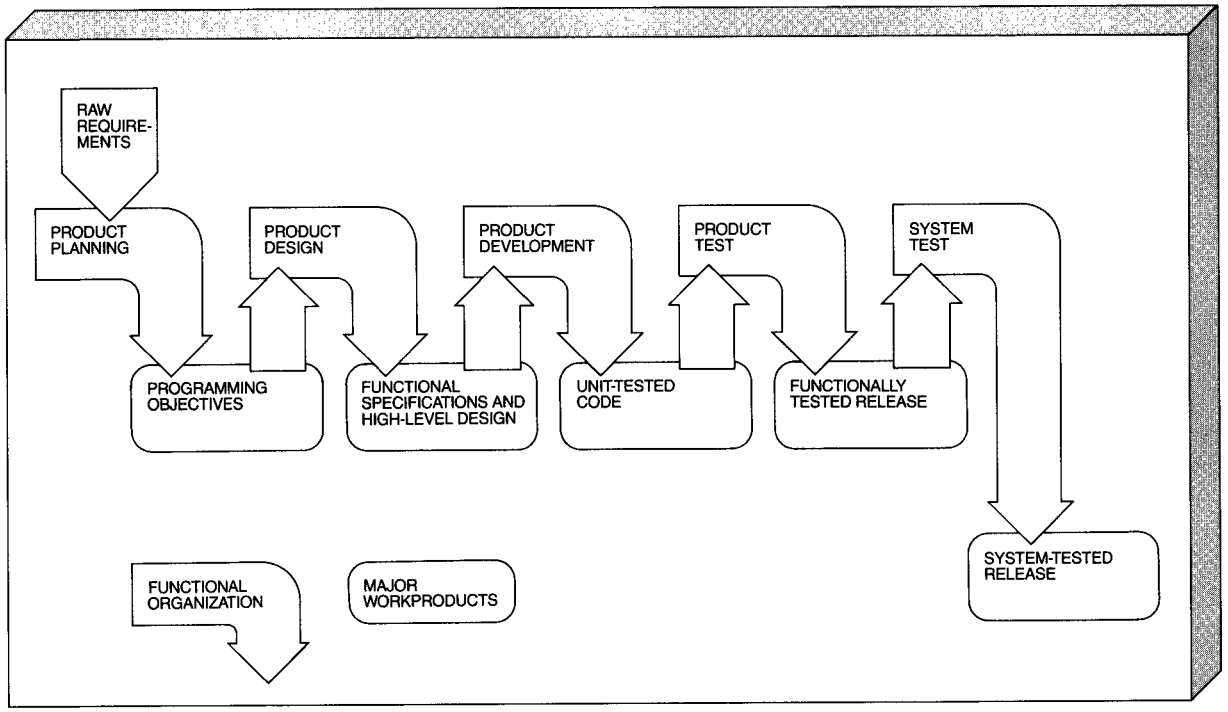
Traditional requirements processes often do not address the many problems encountered in the development of software products. Conventional processes begin with the structural definition of the proposed system, under the assumption that the raw requirements are understood. How this understanding is developed is not formally addressed. The IBM software development process requires a methodology to develop the rationale of the requirement, both in terms of its underlying problem and its business justification, prior to the development of the functional specification. Conventional requirements processes address a single software application intended for use by a uniform set of end users. The resulting system is usually a one-time replacement of some existing system. Many IBM software products, however, address requirements received from a large, diverse set of customers who use the products in a wide array of computing environments. Product releases are typically developed as incremental enhancements to an existing base product. This paper describes the Planning and Design Methodology (PDM), a requirements planning process that supports the collection, analysis, documentation, and tracking of software requirements. The process includes requirements collection, definition of the underlying problems, development of an external functional description that addresses the problems, and development of system and product designs from the external functional descriptions. PDM has been applied in three development areas with positive results.

Releases of IBM software products are usually developed as incremental enhancements to an existing base product. The responsibility for product development rests with a single development organization, which typically is divided into several subordinate organizations by function:

- *Product Planning* is responsible for product requirement analysis, business justification analysis, and development resource planning.
- *Product Design* is responsible for the functional specification and the high-level design of the release.
- *Product Development* is responsible for component and module-level design, coding, and unit testing.
- *Product Test* is responsible for functional testing of the release.
- *System Test* is responsible for system-level testing before the product release is shipped. The System Test organization is generally independent of the Product Development organization.

Figure 1 illustrates a typical process flow in the development organization and identifies the major workproducts of the transformation of raw requirements into a product release. The Product Planning organization receives requirements from a number of sources including customers, customer representative organizations, IBM marketing and service divisions, and internal IBM users of products. IBM customer representative organizations include SHARE Inc., Guide Inc., SHARE International, SEAS, G.U.I.D.E.

Figure 1 Typical development workproduct flow from raw requirements through product release



Europe, COMMON, and ASG. For each product released, Product Planning develops a *programming objectives* document that outlines the functional objectives of the release, operating environments supported, and characteristics such as performance, reliability, availability, serviceability, and usability. From the programming objectives document, the Product Design organization develops a prose *functional specification* document and a *high-level design* document. Product Development, Product Test, and System Test then carry the release through the software development stages of component and module-level design, coding, unit testing, functional testing, and system testing.

Requirements may affect one or more products and thus may be viewed as either product-level or system-level in scope. *Product-level requirements* are product-specific in that they are directed to a single product. For example, consider the requirement "Add a block move capability to the full-screen editor." This can be implemented in the editor without affecting the functions of other products. *System-level requirements* are cross-product in that they require a function to be implemented in more than one product. Consider the requirement "Increase the number of devices that can be attached to a node of the network beyond the current maximum." This requires changes to the basic communications architecture, which in turn requires changes to practically all of the communications software products supporting that architecture.

In general, product-level requirements are forwarded to and managed by the Product Development organization. System-level requirements, however, are handled by the System Planning and System Design organizations, which perform functions equivalent to those of their product-level counterparts. A *system-level design* document is produced that specifies the cross-product design direction to be taken when the function is implemented. When the system-level design document has been approved, the prescribed functions are included in the individual product programming objectives of the affected products. The normal development process is then followed.

Many product planning organizations use automated or semiautomated tools for recording and maintaining product requirements. However, a for-

mal, documented process and methodology are needed to transform raw requirements into a programming objectives document and to transform the objectives into functional specifications and high-level design. Informal requirements-analysis methods expose the product to errors. For example, requirements that are not fully understood may result in the development of functions that do not address the real customer need. If an incorrect level of priority is assigned to a requirement, customer needs may not be addressed on time. If an end-user profile and end-user tasks are not given sufficient consideration, the function delivered may be difficult to use.

The increasing size, complexity, and number of IBM software products and the growing sophistication of customer use have caused the number of product requirements to grow rapidly. The growth of software requirements makes the use of a formal process imperative. A requirements process should be guided by the following principles.

Understand the rationale of the requirement. Ideally, a requirement defines the need or problem to be addressed and the properties or characteristics a solution must have to be acceptable. The requirement should be augmented by descriptions of the environment and the externals of the software. To the extent possible, requirements should not impose constraints on design or implementation by describing software internals. The specification of internals should be deferred for the high-level design phase.

A properly defined requirement should include the following important points:

- The end-user problem that is to be resolved or the need that is to be filled
- The end-user operational characteristics of the proposed solution
- Constraints upon the design that are to be observed during the development of the function

Unfortunately, requirements rarely address or identify these points. Frequently, a requirement is expressed not as a problem but as a solution, as in the following example: "Add a session-limit parameter to the ACTIVATE command." A requirement may even be expressed as a design, as in the following example: "Set the TASKEXT bit in the task control block to indicate that the task has been restarted."

Before a requirement is accepted for further analysis, the rationale, that is, the underlying end-user problem, should be understood. This applies particularly to proposed solutions and designs. A proposed solution may place the function in the wrong component of a product or even in the wrong product. A proposed design may induce problems in other customer environments, whereas a better-conceived design approach may address the underlying end-user problem more effectively.

Verify the rationale and the proposed solution with the customer. Once the underlying end-user problem has been defined, it should be verified with the customer who originally submitted the requirement, as well as with other customers. The customer should agree with the definition of the problem. If other aspects of the problem come to the surface, they also should be incorporated in the rationale.

Customer verification should continue once a proposed solution has been developed. The proposed solution should correctly answer the requirement from the end-user viewpoint and be usable in the end-user operational environment. Customer verification should be completed before significant effort is expended in development.

Define the operational environment. Many IBM software products operate in multiple hardware and software environments. For example, a product may run on CPU models ranging from an IBM 4331 to an IBM 3090 and in networks ranging from ten terminals to 10 000 terminals. A product may use a range of operating systems such as Multiple Virtual System (MVS), Virtual Machine (VM), and Disk Operating System/Virtual System Extended (DOS/VSE). The operational environment also involves the customer use of the product: the types of customer applications that are supported, the size of data bases that are supported, and the expected response time per transaction.

In many cases, a given requirement applies only to some of the product operational environments. Which environments are applicable may affect how the requirement should be addressed. For example, a bulk data transfer application may require intermediate data storage at each communication node, whereas interactive transaction applications do not have such a requirement. This difference will affect the placement of functions in existing products as well as new products in the support of bulk data transfer.

Prioritize and establish business justification. Since there are always more open requirements than there

are available development resources to respond to them, requirements should be selected on the basis of their relative importance on a priority basis. Requirements need to be given priority on a basis that considers customer benefit as well as development cost. The prioritization should be verified with customers.

Emphasize usability as well as function. The analysis of an end-user problem should include the usability and human factors aspects of the problem. An end user may have difficulty with a product because it requires tasks to be performed that are difficult, error prone, or time consuming, or that require a high skill level. These factors may have a greater impact on the end user than any functional deficiencies of the product.

Earlier approaches to the requirements process

Requirements methodologies typically begin at the structural level, with the specification of the functional, data, and behavioral characteristics of the proposed system. These elements are expressed as data flow diagrams, hierarchical functional structures, or data abstractions, which are further refined through functional decomposition and data decomposition.

The following are some general observations on requirements methodologies. The focus of traditional approaches is on functional specification (i.e., what the system is to do) and on design (i.e., how the system is to do it). The question of the rationale behind the requirement—that is, the *why* both in terms of the underlying end-user problem and in terms of business justification—is not usually addressed. These methodologies assume that the raw requirements are already understood and that a proposed system can be modeled. How the raw requirements are filtered for acceptance and subsequent transformation into the proposed system is often not considered. Conventional requirements methodologies generally assume a single, uniform user base and a uniform operational environment. How requirements are to be handled for diverse customers and operational environments is generally not addressed. Conventional approaches also frequently develop a single replacement system from a single set of relatively static requirements. These methodologies are not geared to handling the large volume of small, incremental enhancements that are received continuously for integration into an existing product base. Aspects such as value assessment, prioritization, and business case are generally not considered. In summary, the traditional requirements methodologies are not applicable to the *requirements planning stage*, which precedes the identification of structural requirements.

The Planning and Design Methodology (PDM) was developed to address the need for a formal requirements process. PDM was designed to handle both system-level and product-level requirements, with the following major objectives:

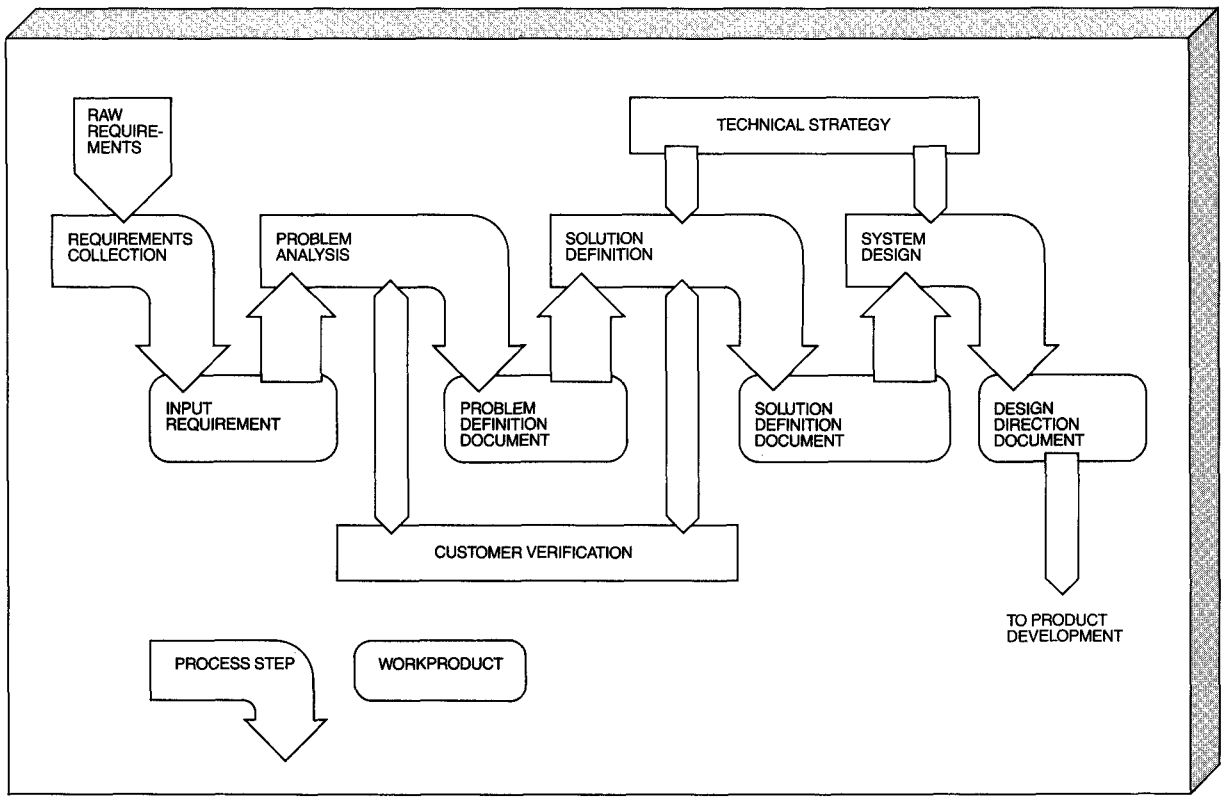
- The ability to categorize, prioritize, coordinate, and manage a large number of diverse requirements from the time they are received until they have been successfully integrated into a product design
- Support for the systematic analysis of requirements from the customer viewpoint by first identifying the underlying end-user problem
- Support for the verification of the problem definition and its proposed solution through customer reviews
- Support for the assessment of a requirement's relative priority, value, and business justification or business case
- Support for traceability of the requirements from initial receipt through implementation
- Support for verification of the transformation of requirements at each step of the process, particularly at the critical transition from external specification to internal design
- Usability and human factors emphasis during requirements definition

In this paper, we describe the Planning and Design Methodology and discuss some of its major features. We then give a brief summary of experiences in applying the methodology in three areas. The discussion concentrates on the requirements planning aspects of PDM and does not primarily address the integration of the methodology with the business justification analysis and resource planning aspects of product planning.

The Planning and Design Methodology (PDM)

In the Planning and Design Methodology, shown in Figure 2, the process begins with *requirements collection*. Here, raw requirements are received, recorded, categorized, and prioritized. *Problem analysis* is performed on each *input requirement* that has a sufficiently high priority. Problem analysis identi-

Figure 2 Overview of the Planning and Design Methodology



fies the underlying end-user problems. A *problem definition* document is written that describes the current customer system and its problem conditions.

The problem definition documents are reviewed and the requirements are reprioritized. Those with a sufficiently high priority are selected for further analysis, and a *solution definition* is developed. The solution definition document describes a solution in terms of end-user functions that resolve the problems identified in the problem analysis. Finally, a number of solution functions are selected and synthesized into a *system design.* A *design direction* document is written that identifies the product(s) and product components that will be affected by the design. Upon approval of the design direction, the usual product development process takes over.

During each phase of the PDM process, the requirements are reprioritized and reviewed, and necessary corrections are made to the workproduct. Both the problem definition and solution definition documents are verified by selected customers and appropriate customer representative organizations. A functional or product *technical strategy,* described in a later section, is used to guide the definition of external functional specifications and the system design.

The Planning and Design Methodology can be applied to both product-level and system-level requirements to assist the requirements-planning activity. For system-level requirements, the final PDM workproduct (i.e., the design direction document) is combined with both the programming objectives and the functional specifications of the affected products. For product-level requirements, the design direction document is integrated into the programming objectives and functional specifications of the product release.

Requirements collection. Raw requirements are collected both formally, through standard submission channels such as customer Product Application and

Support Requirements (PASR), and informally from other sources. Sources for requirements include IBM customers, IBM marketing and service divisions, customer representative organizations, IBM internal-use and development needs, the IBM Research Division, customer surveys, competitive awareness, and public literature. Any information that represents a customer or IBM need, or an opportunity for IBM, may be an input requirement.

The contents of the input requirement are listed in Table 1. The input requirements are maintained in a requirements repository. The highest-priority requirements are promoted to problem analysis. Shown in Table 2 is a typical input requirement.

Problem analysis. In the problem analysis phase each prioritized requirement is examined to identify and describe the associated problem(s) from the end user's perspective. The resulting workproduct, the problem definition document, is formatted for readability and processability and includes backward traceability to the relevant input requirements. Input requirements do not always exist in a one-to-one relationship with problems because one requirement may relate to more than one problem and vice versa. The basic elements of a problem definition document are the following:

- *Control information*, such as author, status, and requirement category. The control information includes a backward reference to the input requirements each document is addressing, and allows for the insertion of a future forward reference to the solution definition that further defines this problem.
- *Current system function*, which describes the system function the end user is currently using. The current system description focuses on those functions of the product(s) that are related to the problems being described. For example, for a problem involving the ease of use of an interactive editor, the specific editor functions involved are described.
- *User tasks*, which describe the tasks the end user performs that are related to problems being described. For example, for a problem with an interactive editor, the specific tasks that the user performs in using the editor are described. The description of user tasks is included to focus attention on usability problems.
- *Problem conditions*, which describe what is wrong with the current function and user tasks, along with the impacts or ramifications of the problem condition on both the customer and IBM. The problem conditions describe the specific problem areas in both the system function and the user

Table 1 Input requirement contents

Control information:

- Input requirement control number
- Source reference information, such as document number and location within document
- Submittor information, such as name, address, date received, and submittor's assessment of the requirement's priority
- Name, address of person recording the requirement and the date recorded
- Assigned priority
- Status and disposition information
- Response to the submittor and response date
- Forward reference to the problem definition(s) that address this requirement
- Category and subcategory of the requirement, and associated search keywords. The categories and keywords permit the grouping of similar requirements. The categories for communications management requirements include, for example, problem management, configuration management, change management, operations management, and ease of use.

Abstract of the requirement

The requirement text as originally received

Table 2 Example of an input requirement for a hypothetical communications trace function

Control information:

• Input requirement number	PR0102
• Source reference	PASR N18723
• Submittor information	J. Duncan, Branch Office 245, Chicago, IL
• Submittor's priority	Medium
• Author (person recording)	R. G. Mays, Raleigh
• Assigned priority	High
• Status	Promoted to problem analysis
• Response to submittor	Accepted, 2/18/85
• Category/subcategory	Problem management/ problem determination
• Search keywords	Trace, operator commands
• Problem definition document	PR1071

Abstract: Allow keyword abbreviations in the trace command.

Requirement text: It is very difficult to key in the long trace commands to start up a communications trace. Customer desires the ability to use one- or two-character abbreviations for the command keywords. For example, L= for LINE=; N= for NODE=; EV= for EVENT=.

Table 3 Problem definition document content

Control information:

- Problem definition control number
- Author of the problem definition and date written
- Assigned priority
- Status and disposition information
- Category and subcategory of the problem and associated search keywords
- Backward reference to the input requirements which this problem definition addresses and forward reference to the solution definition document(s) which address this problem

Abstract of the problem, with a description of specific cases of the problem, if applicable

Current system function description, including a description of input and output data associated with the system function

User task description and end-user profile

Problem conditions associated with the system function and user tasks, including the impact of the problem condition on the customer and IBM

Environment description

Value assessment

tasks. The impacts of the problem conditions are one basis for establishing the value of the requirement.

- *Environment description* of the hardware and software configurations in which the problems exist. An environment description might include the hardware and network configurations on which the problem occurs, and the operating system and other supporting software that relate to the problem.
- *End-user profile*, which includes the user class (e.g., operator, system programmer, network planner) and user characteristics, such as abilities, skills, training, and knowledge. Some system functions do not involve people directly, but are used by customer-written applications. From the viewpoint of a software product, customer applications as well as people may be end users.
- *Value assessment*, which estimates the value of solving the problem(s) from both the customer's and IBM's standpoints. The value of a problem solution can be estimated in a number of ways, depending on the nature of the problem. Examples are hardware or labor cost savings, increased productivity, lower required skill level, and shorter training time.

The problem definition document is organized and formatted to give a complete picture of the end-user problem from both a functional and user-task viewpoint. In practice, the actual descriptions of current function and user tasks can be brief, because the focal point of the problem definition is the problem conditions and their impacts. Table 3 summarizes the major items contained in a problem definition document. The problem definition documents are maintained in a requirements repository. The ex-

Table 4 Example of a partial problem definition for a hypothetical communications trace function

Control information:

• Problem definition number	PR1071
• Category/subcategory	Problem management/ problem determination
• Author	R. G. Mays
• Status	Customer review completed
• Input requirements addressed	PR0085, PR0102
• Solution definition document	PR2045

Abstract: The communications trace function is too complex to use and error prone.

Current function: The trace function consists of entering trace commands at the operator console to control the starting and stopping of line traces and formatting and printing of the traces for output.

User tasks: The trace commands are generally entered by the system programmer. The operator does not enter them because the commands are complex, and a significant level of expertise is required to enter them properly. The system programmer generally looks up the command, writes it out, enters the command at the console, and waits for an indication that the desired communications event has occurred. The system programmer then enters commands to format and print the trace.

Problem conditions:

- The trace facility requires too high a skill level.
  Only experienced system programmers can reliably do communications traces. The impact of this is that customers who do not have an experienced staff are unable to do adequate problem determination.
- Trace commands are frequently keyed in erroneously.
  Trace commands are long strings of parameters that are subject to keystroke error. If an error is made, the only recourse is to re-key the command, a condition that can be very frustrating.
- There is no on-line help facility.
  The user must frequently reference the trace manuals and often guesses at actions and command formats.
- The system programmer must guess whether the desired communications event has occurred.
  The impact of this is that large quantities of excess trace data are often collected, formatted, and printed needlessly.

ample in Table 4 is typical of the content of a problem definition.

This method of describing requirements by defining the current end-user environment and its problem conditions can be extended to cases in which a system function is simply missing. For example, the "current system" may be a manual procedure and the "problem condition" is simply that the function is not provided by current products. This situation represents a market opportunity that has not yet been addressed. The other aspects of the problem definition, such as problem impacts and environment description, can assist in the analysis of customer value and priority of the market opportunity.

Solution definition. The solution definition phase develops a cost-effective external solution for the problems described in one or more problem definitions. The resulting solution definition document describes the new or enhanced functions necessary to resolve the problems. Only the external function description is given at this point. (The internal design is begun in the next phase of PDM.) The solution definition document resembles the problem definition document in structure and content. The basic elements of the solution definition document are the following:

- *Control information*, such as author, status, and requirement category. The control information includes a backward reference to the problem definitions that this document is addressing and allows for the insertion of a future forward reference to the design direction that implements the solution function.
- *The proposed system function*, which describes the new or enhanced system functions required to resolve the problem conditions. The focus in this phase is on an external function description, that is, "what" the proposed function will do.
- *User tasks*, which describe the tasks the end user will perform in using the proposed functions. Only those tasks related to the resolution of problem conditions are generally described.
- *Resolved problem conditions*, which describe the problem conditions that are resolved by the proposed system function and user tasks and the rationale for their resolution. Reference back to specific problem conditions in the problem definitions permits a verification of the completeness of the solution in addressing the original requirements.
- *An environment description*, describing the hardware and software configurations in which the solution functions are intended to operate.

Table 5 Solution definition document content

Control information: • Solution definition control number • Author of the solution definition and date written • Assigned priority • Status and disposition information • Category and subcategory of the solution and associated search keywords • Backward reference to the problem definition document(s) this solution definition addresses, and forward reference to the design direction document that implements this function
Abstract of the solution
Proposed system function description including a description of input, output, and internal data associated with the proposed function
User task description and end-user profile
Problem conditions that have been resolved by this solution definition, including the rationale for their resolution
Environment description
Value assessment
Alternate solutions considered and design decision rationales

- *An end-user profile*, which describes the types of end users or end-user applications that are affected by the solution functions. The end users of the solution function may not be the same as the end users described in the problem definitions, for example, if the skill level has changed or if the system function now interfaces with a different class of user.
- *A value assessment*, which estimates the value of providing this particular solution to the customer. This assessment differs from the assessments made in the problem definitions, because it is specific to a particular solution function.

Table 5 summarizes the major items included in a solution definition document. The solution definition documents are maintained in the requirements repository. The example in Table 6 is typical of the content of a solution definition.

System design. The system design phase refines solution definitions into a coordinated set of system-level designs that may be allocated to specific products or product components for implementation. The resulting design direction document describes how the solution functions are to be implemented within the affected product or product areas. Typically, a number of solution functions are combined in a system design. Table 7 shows the major sections

Table 6 Example of a partial solution definition for the hypothetical communications trace function

Control information:

• Solution definition number	PR2045
• Category/subcategory	Problem management/ problem determination
• Author	R. G. Mays
• Status	In internal review
• Problem definition addressed	PR1071
• Design direction	Not started

Abstract: Improve the communications trace function's ease of use.

Proposed function: The proposed trace function will consist of a full-screen facility for entering trace commands at the operator console. All of the existing trace functions will be supported. An additional trace function will be provided to specify communications events to be associated with the trace. The trace will then monitor communications events and, when the specified event is detected, give an indication to the user and/or terminate the trace. The trace command and all of the corresponding specifications can be saved in a disk data set and called up and modified prior to running. A help facility will be provided to explain specific fields on the screens and their possible values.

User tasks: The trace commands will be entered by filling the appropriate fields on the screen. The user may be either a system programmer or an operator. The user will need to refer to the trace manual only if a detailed explanation of the trace command, with examples, is needed.

Problem conditions addressed:

- The trace facility requires too high a skill level.
  With the on-line interface and help facility, less experienced system programmers will be able to enter the commands. With canned trace commands, operators will be able to run preliminary traces for problem determination without having to call in a system programmer.
- The trace commands are frequently keyed in erroneously.
  This is addressed by the full-screen prompting.
- There is no on-line help facility.
  This will be provided.
- The system programmer must guess whether a desired communications event has occurred.
  The event monitoring function addresses this.

of the design direction document. The design direction document undergoes a review and approval process which includes review by the product development organization(s) that must implement the design. Upon approval of the design direction, the design is incorporated into the product programming objectives and functional specifications for an upcoming product release. If more than one product is involved in the implementation, the product releases are coordinated.

Methodology features

We now describe the following concepts of PDM that underlie each phase of the methodology:

- The definition of formal process activities and verification criteria
- The use of viewpoints and aspects for verification
- The availability of a technical strategy to guide product design directions
- The involvement of work groups whose members may include such concerned persons as requirements planners and analysts, designers, developers, marketing organization representatives, and field support engineers.

Table 7 Design direction document content

Control information:

- Design direction control number
- Author(s) and date written
- Assigned priority
- Status and disposition information
- Backward reference to the solution definition document(s) this design direction addresses

Abstract of the design direction

The functional detail that will be implemented, including usability and performance criteria. The mapping of the design functions back to the solution definition functions, including implementation rationales, permits a verification of the completeness of the design direction.

Detailed user-task and end-user descriptions

Data descriptions of input, output, and internal data associated with the design

External, interproduct, and component-level interfaces

Affected product components and the mapping of functions to these components, including function sizings in terms of lines of new and changed code

Environment description, including configurations supported, customer migration and coexistence considerations, function dependencies, and architectural specification requirements such as changes to the Systems Network Architecture (SNA).

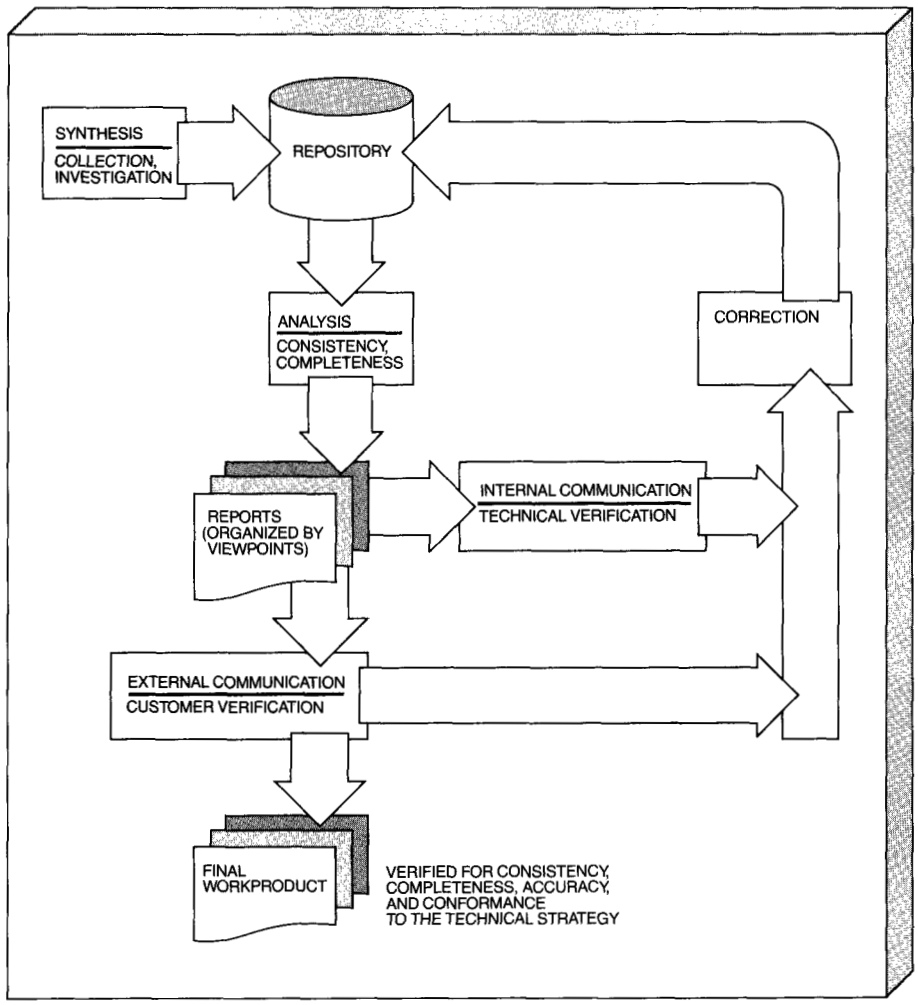
Additional design constraints, such as reliability, availability, serviceability

Alternate designs considered and the design-decision rationales

Implementation considerations, such as the staging of the function across releases, the packaging of the function, and the sequencing of the implementation

Value assessment

Figure 3 PDM formal process activities



Formal process activities and verification criteria. The requirements process can be viewed as a series of steps that transform initial requirements—often amorphous, such as "provide the customer with a single view of the system"—into a clear, concise, and structured specification document. To facilitate the transformation, PDM identifies three principal activities to be applied within each phase of the methodology: *synthesis, analysis*, and *communication.* Figure 3 illustrates the relationships among these three activities.

In PDM, *synthesis* is the construction of the workproduct(s) of a phase from a number of initially unrelated sources of information. The workproducts thus constructed might, for example, be a consolidated statement of a particular set of user problems, a corresponding statement of an overall system solution, or a design direction that defines how solution functions are to be integrated into a product set. Synthesis involves an investigation of the problem area and the collection, refinement, and extension of supporting information. The investigation is guided by the predefined document structures for the problem definition, solution definition, and design direction documents.

Analysis is the examination and verification of a given workproduct for *consistency* and *completeness. Consistency* is a check to determine whether the

elements of a workproduct relate to and interface with one another without contradiction, and whether the workproduct conforms to its prescribed standards and document structure. This verification includes checking for consistent use of names, terms, descriptions, and relationships among elements. An example of an inconsistency intrinsic to the workproduct itself might be the occasional use of the term

Completeness is a check to ensure that all necessary sections have been fully addressed in the workproduct.

"network setup" as an ambiguous substitution for the term "network initialization" in the document. For the solution definition and design direction documents, an additional consistency check is performed to verify that the functions described conform to the technical strategy. *Completeness* is a check to ensure that all necessary sections have been fully addressed in the workproduct. The completeness verification also includes a check that all items from the previous phase have been satisfactorily addressed in this phase. For example, the completeness check determines that all problem conditions identified in the problem definition have been resolved in the solution definition. This check also determines that all external functions specified in the solution definition have been included in the design direction.

In practice, the synthesis and analysis activities are an iterative process of collection, extension, verification, and correction that is aided by re-presenting and examining the workproducts in various *viewpoints*. The use of viewpoints is described in the following section.

Communication is the third of the PDM transformation activities. Vital to each phase of PDM is the communication of the resulting documents to technical and nontechnical personnel for review and approval. This communication relies on the concept of *viewpoints* to present a structure whose content is relevant to various types of reviewers. *Internal reviews* are conducted to verify conformance to the accepted technical strategy and to verify overall technical correctness. *External reviews* are conducted with customers and marketing and service organizations to finally verify correctness from the end user's viewpoint. Here, *correctness* means verification with the originator of a requirement that the workproduct does satisfy the expressed requirement. In PDM this involves two phases. During problem analysis, the descriptions of the end-user problems and their effects must be complete and accurate. During solution definition, the proposed system function and proposed user tasks must resolve the end-user problems.

Viewpoints and aspects. The workproducts developed during an application of PDM are complex and involve many relationships among the document elements, not all of which are relevant at all times to all organizations. To permit different reviewers to develop different perceptions of the material from different points of view, the workproducts in each phase of PDM can be partitioned into different sets of viewpoints, constructed from the major elements of a workproduct. The same entity or element of a workproduct can be presented in multiple viewpoints. Presented this way, the entity may appear correct in one context and at the same time incorrect or incomplete in another context. The major viewpoints defined in PDM and their primary focus are the following:

- *System function*—system functions, with secondary focus on data elements and user tasks
- *User task*—user tasks, with secondary focus on user descriptions, system functions used, and input/output data elements
- *Data definition*—data elements, with secondary focus on the system function
- *Problem condition*—problem conditions and their impacts, with secondary focus on the system functions and user tasks involved
- *Environment*—customer environment description, with secondary focus on user descriptions, system functions, and input/output data elements
- *Target product*—target products and their components, with secondary focus on the system functions to be implemented and their data elements

Viewpoints, in turn, contain various *aspects* or subordinate sections. For example, aspects associated with the environment viewpoint are defined as

- Customer enterprises—the segment of customers expected to utilize the target product
- Systems interfaces—hardware devices, components, and features that support or are supported by the products under consideration, and software within the existing operating environments that support or are supported by the products under consideration
- Human interfaces—end-user profiles, system functions with which the end user interacts, information the end user provides to the products under consideration, information the end user receives from the products under consideration

The technical strategy. Requirements are received piecemeal and must, initially at least, be considered individually or in small sets. A broad perspective of the product or product area is needed so that functions are not developed at cross-purposes to one another. The *technical strategy* provides this perspective, serving to guide the development of proposed solution functions and system designs. It provides a framework within which functions can be devised, particularly in the context of multiple products that must function together consistently. Defined for a single software product or set of products—for example, communications products—the technical strategy provides a strategic direction for the product or product set, and is revised, maintained, and reviewed frequently. The technical strategy describes

- The basic system structure and the system-level functional evolution for its product set
- The relationship of its products to any pertinent architectural standards, e.g., the IBM Systems Network Architecture (SNA)
- User and internal interface standards
- Planned staging of function implementation across the product set

Work groups. The definition and analysis of customer requirements must be complete and accurate if the resulting functions derived in system design are to fulfill customers' needs. It is frequently necessary, therefore, to convene a work group consisting of individuals outside the development organization to supply additional expertise and perceptions regarding the problem and proposed solutions. Participants in a work group are drawn from wherever expertise in a particular problem area exists, such as IBM marketing and service organizations, customer special-interest organizations, and consultants to IBM. Work groups are normally convened for two primary objectives: (1) *Requirements work groups* are formed to create the problem definition and solution definition documents. This type of work group is typically made up of system planners, product planners, and consultants. (2) *System-design work groups* are formed to create the design direction document from a set of solution definitions. This type of work group is typically made up of system designers, product designers, and consultants.

Through broader participation in the requirements process, work groups provide better specification of problems and solutions than exclusively technical approaches. Work groups also foster increased understanding of customer needs and system requirements within the development organization. The work-group approach reduces last-minute changes caused by unexpected new requirements. Work groups see the development and understanding of customer satisfaction and business goals as common objectives.

Experiences with the Planning and Design Methodology

Early experiences. Two pilot projects were undertaken in 1981 and 1982 to evaluate the effectiveness of the PDM concepts of formal activities (synthesis, analysis, communication), verification criteria, and the viewpoint/aspect representation. The first project involved the development of a new version of a compiler, and the second was concerned with the development of a major new function for the IBM Multiple Virtual System (MVS) control program. In both projects, the PDM methodology was introduced after some initial requirements and design work had commenced. Although a formal programming objectives document had not yet been completed, some of the requirements specification had been written. The PDM requirements specification was developed as a combined solution definition and design direction document, including both the external function description and details of the internal design. In both projects, a formal requirements specification was needed to clarify the design work that had already begun.

The Problem Statement Language (PSL) and its associated software, the Problem Statement Analyzer (PSA),[1,2] were used as the specification language and the repository. PSL/PSA was chosen because its language structure supported a network model of the specification, and its automated analysis functions facilitated the display of the various viewpoints and

aspects called for in the methodology. PSA analysis programs also helped to ensure consistency within the requirements specification as expressed in PSL; PSA also produced several additional reports that communicate the requirements specification to reviewers and approvers.

Meetings for reviewing the requirements document were held. Participants included the requirements planner and one or more of the key designers. At the initial review meeting, a short tutorial was given on reading and interpreting the formal specification. At each review meeting, the material was presented and reviewed in a structured agenda, according to the viewpoints previously described. Many misunderstandings and unresolved issues surfaced and were discussed, and the requirements specification was updated on the basis of the meeting results. A subsequent review was scheduled. After several refinements of the specification, a consensus was reached on the requirements, and unresolved issues were documented and incorporated in action plans for resolution.

The reviewers agreed that the structure and formalism of the PDM requirements specification helped to raise important issues that normally would not have come to the surface until later in the process. The participants noted several benefits of PDM. For example, the reviews of the requirements specification improved communication between the Product Planning and Design organizations and speeded the recognition of problems. The formalism provided by PSL/PSA and PDM forced a crisper statement of requirements as compared to traditional prose. The requirements specification document made it more difficult for issues to be ignored or glossed over.

These early PDM experiences demonstrated that structure and formalism help to clarify the requirements specification. Key issues can be brought up and resolved earlier in the development process than with traditional methods. PDM improved communication between requirements planners and designers and improved the ability of the developers to prepare a precise response to the requirements.

Experiences in a communications management application. In late 1982, the complete PDM methodology involving all its phases (requirements collection, problem analysis, solution definition, and system design) was applied within a complex communications management product development environment. The objective of the PDM application was to produce a system-level design for the next incremental enhancement of certain communications management products. The responsibility for the development and maintenance of these interacting products was shared among several organizations.

This application of PDM differed from the two experiences just cited in that it was decided not to use PSL/PSA for automated support. This decision was based upon several factors. PSL/PSA in its delivered form was found difficult to use by requirements planning personnel whose backgrounds were not in software development.

The need for end-user flexibility with PSL/PSA led to the initiation of a concurrent development project to provide this capability. A front end was produced that greatly reduced the need for the end user to know the syntax and semantics of the PSL language and PSA commands and provided appropriate reports on request. Although the resulting support tool did not totally resolve this key human factors requirement and was used only on a limited basis, the experience gained provided important insights into requirements for such tools in the future. We expand upon this idea in the following section.

This project involved ten planners and system-level designers. Several work groups were formed to perform the necessary prioritization, reviews, and verification. Members of these work groups were drawn from system planning, system design, individual product organizations, and various IBM marketing and field support divisions. Selected customers who were very knowledgeable about the affected communications management products and the communications networking interest group of SHARE Inc. were utilized for customer verification. The technical strategy, which continues to guide subsequent product directions, was developed by an internal IBM strategy and planning working group.

Slightly fewer than 300 raw requirements were gathered during the requirements collection phase. Of these, 35 were rejected, 23 were combined with others because of equivalence or similarity, 57 were to be studied further, 38 were classified as future objectives, and 144 were migrated to the problem analysis phase. During the problem analysis phase, problem definition, value assessment, combination of similar problem solutions, and prioritization resulted in 31 of the original 300 requirements being forwarded to the solution definition phase. As shown in the solution definition example in Table 6, a

requirement solution may, in effect, be composed of many subsolutions.

The definition of a solution for a particular requirement does not necessarily imply that the requirement is forwarded to system design. Prioritization based upon evaluations of technical and business feasibility and resource availability can result in additional attrition. Consolidation may also occur, because a solution may either be equivalent to the solution of other requirements or subsumed by other solutions. Customer and internal review resulted in a consensus prioritization of the 31 solution definitions. Of these, the 21 highest solution definitions were selected for system design work and were consolidated into six design directions.

Thus the system design phase for this application resulted in six design direction documents, each of which provides a solution addressing a number of the original requirements. These documents were referred to respective product development organizations for final review and inclusion in product development plans. To date, final testing and release have not been completed for these communications management products.

Although the long-term effects of the application of PDM upon quality and productivity in the total development life cycle have yet to be measured, its application during the planning and system-level design phases was termed very successful by the participants and management of the project. The following results and effects were noted:

- Discipline was introduced into a previously loosely defined and controlled requirements planning process. The identification of specific steps within the process and their anticipated goals and workproducts enabled increased communication, coordination of tasks, definition of work objectives, and enhanced control.
- The PDM process served as a problem-solving technique that quickly put forward all the relevant and salient issues and facts during analysis and review.
- PDM helped remove the parochialism and subjectivity of organizational and functional interests.
- The evaluation of potential solutions was facilitated by the clear identification of the underlying problems to be solved and the user environment into which the solution must be integrated.
- A fixed set of discrete product-level requirements was produced and expressed in a design direction document. These product requirements resulted from the PDM process, which continuously exposed, tracked, filtered, and clarified needs and objectives. Significant to this success was the direct involvement of customers, planning, and product development personnel in the review and verification process.
- The precision, consistency, and depth of the design direction documents greatly facilitated the generation of product development plans detailing the necessary resource requirements, schedules, and test procedures to be employed.
- Automated support is crucial to further gains in quality and productivity in the PDM process. Automation is particularly valuable in the management and analysis of data and in the production of documentation. Additional value could be gained if the PDM process and quality metrics were also automatically controlled and extracted.[3]

Future directions

Most of us will agree that advances in the technology of requirements planning are vital to the continued reduction of software development and maintenance costs.[4–6] Although the need for formalisms to express requirements has emerged as a long-term goal, progress is slow in defining rigorous format, content, and quality criteria by which requirements specifications can be evaluated.[7]

PDM is an embryonic stage in the production of software requirements that are concise, well-defined, understandable, and verifiable against the original source requirements as specified by a customer/end user. The complete realization of these goals by PDM is currently limited by the amount of information captured by PDM in the source requirements and value assessments, which are received and maintained in narrative form. This is especially true in the earlier phases of PDM, in which information is often fuzzy, subjective, or expressive of the "why" as opposed to the more rigorous "what" or "how." Although initial studies[8] have demonstrated the flexibility of the relational model in capturing semantically vague requirements expressions, further research is necessary to realize more flexible semantic modeling techniques to bridge the gap between the customer/end user and the requirements planner.[9-11]

The need for automated support of PDM is strongly evident. The maintenance of control over and the insight into data collected during the phases of PDM is essential, not only because of the volume of infor-

mation, but also because of its complexity and the length of the product life cycle. Nevertheless, any automated support provided for PDM must be easily modified and adapted for two reasons: (1) The technology and the redefinition and refinement of the PDM process are anticipated to evolve in new directions. Information elements within PDM and their interrelationships and dependencies are expected to become formal notations for expressing program requirements structures. (2) The PDM process will not be identical across differing customer/end-user bases, product structures, development organizations, and recipients of PDM workproducts. It is essential to have the ability to generate dialects of PDM rapidly for different environments, where these environments are themselves fluid.

The automation of PDM in support of many organizational and product structures will require new capabilities to strengthen it and increase its flexibility. Regardless of the implementation, future PDM directions will require additional capabilities:

- Textual searches to locate keywords or character strings of interest, as provided, for example, by the IBM product STAIRS.[12] Much of the source information required by PDM is and will continue to be provided in narrative form; isolating particular requirements and their contexts across voluminous narrative documents is necessary.
- Expression, storage, and retrieval of information elements in terms of open-ended relationships as afforded by relational data bases, such as SQL/DS[13] and DB2.[14] Flexible semantic models enable the iterative expression and refinement of usually implicit or incomplete raw requirements into discrete and precise statements.
- Rules-driven verification mechanisms based on universal and locally established criteria.
- Interactive requirements gathering, problem and solution definition. The current PDM accepts prewritten source requirements and involves the customer/end user in a noninteractive manual process of iterative verification.
- Adaptability of the PDM process to local product and organizational needs. Anticipated variations of the PDM process and any automated support will require the flexibility of application-generator systems, such as the IBM Application System[15] and The Information Facility.[16] These systems not only exhibit relational capabilities, but also give the user the ability to rapidly customize the application.

These possibilities for future PDM developments have, from a pragmatic view, automation solutions that are within the grasp of the practitioner today. However, to extend the automated support beyond these capabilities will require advances in two areas of current research—expert systems and very high level language processing. We anticipate that developments in these fields will result in automated capabilities to derive missing requirements by inference and validate problem and solution definitions. This can be done by using an accumulated main body of knowledge about customers and end users, their operational environments, and existing systems. The other technical advance needed is a means of algorithmic translation from high level solution statements to product-specific designs.

Concluding remarks

The Planning and Design Methodology has proved to be valuable in systematically analyzing, documenting, and managing a large set of requirements from initial input through design. PDM is particularly applicable to IBM software development environments that must provide continuously and incrementally enhanced products. Experiences to date with PDM have been positive on the development projects in which it has been used, especially on a large system-level communications management application. Nevertheless, experiences have also pointed out the need for future directions in which additional gains in quality and productivity can be attained. Much work remains to be done in the area of planning and requirements engineering. Nevertheless, PDM has initiated the transfer of technology into requirements planning, resulting in a more manageable, disciplined, and tractable software development phase.

Acknowledgments

The authors wish to acknowledge the significant contribution made by Edward A. Altieri, Dorene H. Palermo, and William E. Warner, Jr. to the development of the concepts of requirements collection, problem analysis, solution definition, and system design. Richard A. Tidwell and Richard L. Fredrick also contributed to the basic definition and refinement of these concepts, which form the heart of the methodology. We also wish to acknowledge the contributions of Dr. A. Lip Lim, under whose direction PDM was initially developed, and Gerard M. Cocco and Joseph M. Gdaniec, who helped improve PDM under the direction of Dr. Lim. We also acknowledge William R. Brown and Richard A. Tidwell, who contributed to the refinement of the PDM process and the development of a prototype tool to support it.

Cited references

1. D. Teichroew and E. A. Hershey, "PSL/PSA: A computer aided technique for structured documentation and analysis of information processing systems," *IEEE Transactions on Software Engineering* **SE-3**, No. 1, 41–48 (January 1977).
2. L. S. Orzech, "PSL/PSA: A computer-aided tool and technique for specification and analysis of high-level designs," *IBM/FSD Software Engineering Exchange* **2**, No. 1, 2–9 (October 1979).
3. G. F. Hoffnagle and W. E. Beregi, "Automating the software development process," *IBM Systems Journal* **24**, No. 2, 102–120 (1985, this issue).
4. R. T. Yeh and P. Zave, "Specifying software requirements," *Proceedings of the IEEE* **68**, No. 9, 1077–1085 (September 1980).
5. M. W. Alford, "Software requirements in the 80's: From alchemy to science," *Proceedings of the Annual Conference, ACM 80,* Nashville, TN, October 27–29, 1980; ACM, Baltimore, MD (1980), pp. 342–349.
6. T. E. Bell and T. A. Thayer, "Software requirements: Are they really a problem?", *IEEE, Proceedings, 2nd International Conference on Software Engineering*, San Francisco, CA (October 13–15, 1976), pp. 61–68.
7. *IEEE Guide to Software Requirements Specifications*, ANSI/IEEE Standard No. 830-1984, available from the IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854.
8. M. P. Perriens, "Using QBE to process software requirements and specification data," *IBM/FSD Software Engineering Exchange* **2**, No. 1, 10–20 (October 1979).
9. R. Balzer, N. Goldman, and D. Wile, "Informality in program specification," *IEEE Transactions on Software Engineering* **SE-4**, No. 2, 94–103 (March 1978).
10. M. L. Wilson, *Requirements Modeling Using the Requirements and Design Aid*, Technical Report TR-03.199, IBM Santa Teresa Laboratory, P.O. Box 50020, San Jose, CA 95150, 1982.
11. S. J. Greenspan, *Requirements Modeling: A Knowledge Representation Approach To Software Requirements Definition*, Technical Report CSRG-155, Computer Systems Research Group, University of Toronto, Toronto, Canada (March 1984).
12. *STAIRS/Conversational Monitor System, Terminal User's Guide*, SB21-2783, IBM Corporation; available through IBM branch offices.
13. *SQL/Data System, General Information*, GH24-5012, IBM Corporation; available through IBM branch offices.
14. *Database 2, General Information*, GC26-4073, IBM Corporation; available through IBM branch offices.
15. *Application System Introduction*, SC34-2209, IBM Corporation; available through IBM branch offices.
16. *TIF: The Information Facility, Reference Manual*, SC26-4479, IBM Corporation; available through IBM branch offices.

Robert G. Mays *IBM Communication Products Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709.* Mr. Mays joined IBM in 1981 and is currently working in Communications Programming Process Planning and Control on projects related to software process technology and development. His prior assignments include advanced communications architecture, systems planning for communications and systems management projects, and requirements process development. Mr. Mays received his B.S. degree in chemistry in 1968 from the Massachusetts Institute of Technology. Prior to joining IBM, he worked for 12 years in management systems development at the Eastman Kodak Company. He is a member of the Association for Computing Machinery and the IEEE Computer Society.

Leonard S. Orzech *IBM Communication Products Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709.* Mr. Orzech joined IBM in 1962. He is currently working in the area of product planning and design methodology development. Prior to his current assignment, he has held a variety of programming, design, analysis, education, and research positions within the Federal Systems Division. Applications of these technologies have included design automation, formatted file systems, product assurance, integrated logistics, on-board training, and intelligence systems. Mr. Orzech received his B.A. and M.S. degrees from the University of Connecticut in 1962 and 1976, respectively.

William A. Ciarfella *IBM Data Systems Division, Neighborhood Road, Kingston, New York 12401.* Mr. Ciarfella joined IBM in 1980. He is currently working in the Software Engineering function at the Kingston Programming Center. Since joining IBM, Mr. Ciarfella has worked on software requirements and design systems. Most recently, he was one of the authors of the IS&SG Programming Process Architecture. He received his B.S. degree in mathematics and computer science from the SUNY College at Brockport.

Richard W. Phillips *IBM Information Systems and Storage Group, P.O. Box 390, Poughkeepsie, New York 12601.* Mr. Phillips attended the University of Wisconsin, and joined IBM in 1954. He has held several technical and management posts, both in the field and at IBM development laboratories for OS and MVS software development. His technical accomplishments have included the development of programming quality measurement systems and predictive models for analyzing program quality. From 1980 to 1983, Mr. Phillips was team leader in the advanced technology effort and early pilot projects that preceded and helped to found the methodology described in this paper. He is currently active in a software engineering function in IBM. Mr. Phillips is also an Adjunct Associate Professor at Rensselaer Polytechnic Institute, where he teaches software engineering courses in the graduate study program. He is a member of the Association for Computing Machinery and the IEEE Computer Society.

Reprint Order No. G321-5244.