REPORT:

Authors:
***David C. Opferman** is director of the Large Computer Development Laboratory at AT&T Information Systems in Naperville, Illinois. He is responsible for the development of super minicomputers. He joined AT&T in 1967. He received the degrees of B.S. in Electrical Engineering from Pennsylvania State University, and M.S. and Ph.D. in Electrical Engineering from the University of Pittsburgh. **Robert H. Yacobellis** is supervisor of the Software Engineering Group at AT&T Information Systems in Naperville, Illinois. He is responsible for the development methods and quality metrics used throughout the 3B computer design/development division. He joined AT&T in 1967. He received a B.S. in mathematics from Carnegie-Mellon University, and M.S. and Ph.D. degrees in*
(continued on page 72)

A DESIGN METHODOLOGY FOR SYSTEM QUALITY

Introduction

A new approach to building complex systems by targeting specified quality levels for both hardware and software components has been developed within AT&T. This design procedure (Figure 1) is useful because it:

- Uses up-front customer analysis to minimize product defects and increase customer satisfaction. (Protection of the customer's investment is critical in this analysis.)
- Provides both system-level and end-to-end perspectives on quality, including critical elements of a quality design/development process.
- Deals with systems as total concepts, rather than separating the hardware and software approaches.
- Reduces system development costs by early detection and removal of defects.
- Provides ways to quantify and measure both progress and quality.
- Designs specified quality into the developed system.

The approach is based on current development practices as followed in the information industry as well as on firsthand experience within AT&T projects. Key sources include the AT&T 5ESS™ digital switch, the AT&T Bell Laboratories Quality Assurance Center and Software Engineering Technology Transfer Program, and the AT&T 3B Computer Systems of the AT&T Information Systems Line of Business. The approach used within these and many other areas of AT&T is being refined on a continuous basis.

System Quality

Quality acceptance in developed products is in the eye of the beholder—the customer or user of the products. A design methodology should provide techniques, tools, and guidelines to aid in developing products that consistently meet customer-required quality levels. This is especially critical for the complex collections of hardware and software that make up today's computer-based information management systems.

Users may be either external to the developing company or downstream developers within the company. Quality may be judged in terms of factors such as timeliness of product delivery, level of defects discovered by the customer, or usability of the product for solving customer needs.

The experiences of AT&T and many other companies indicate that building quality into a product (for example, building the "right" product) saves time and money and increases profitability. Products that are targeted to meet the needs of their users save on costs of both redesign and maintenance. In fact, maintenance costs have often been reduced by a factor of three.

Furthermore, product realization can be managed by building quality into it, by carefully planning the development process, and by obtaining customer input. This report focuses on both the developing product and its internal structure. More details of this approach may be found in the references, including information on how the approach is employed in the AT&T 3B computer line.

System Methodology Approach

Several basic concepts are essential to the design methodology used in the AT&T

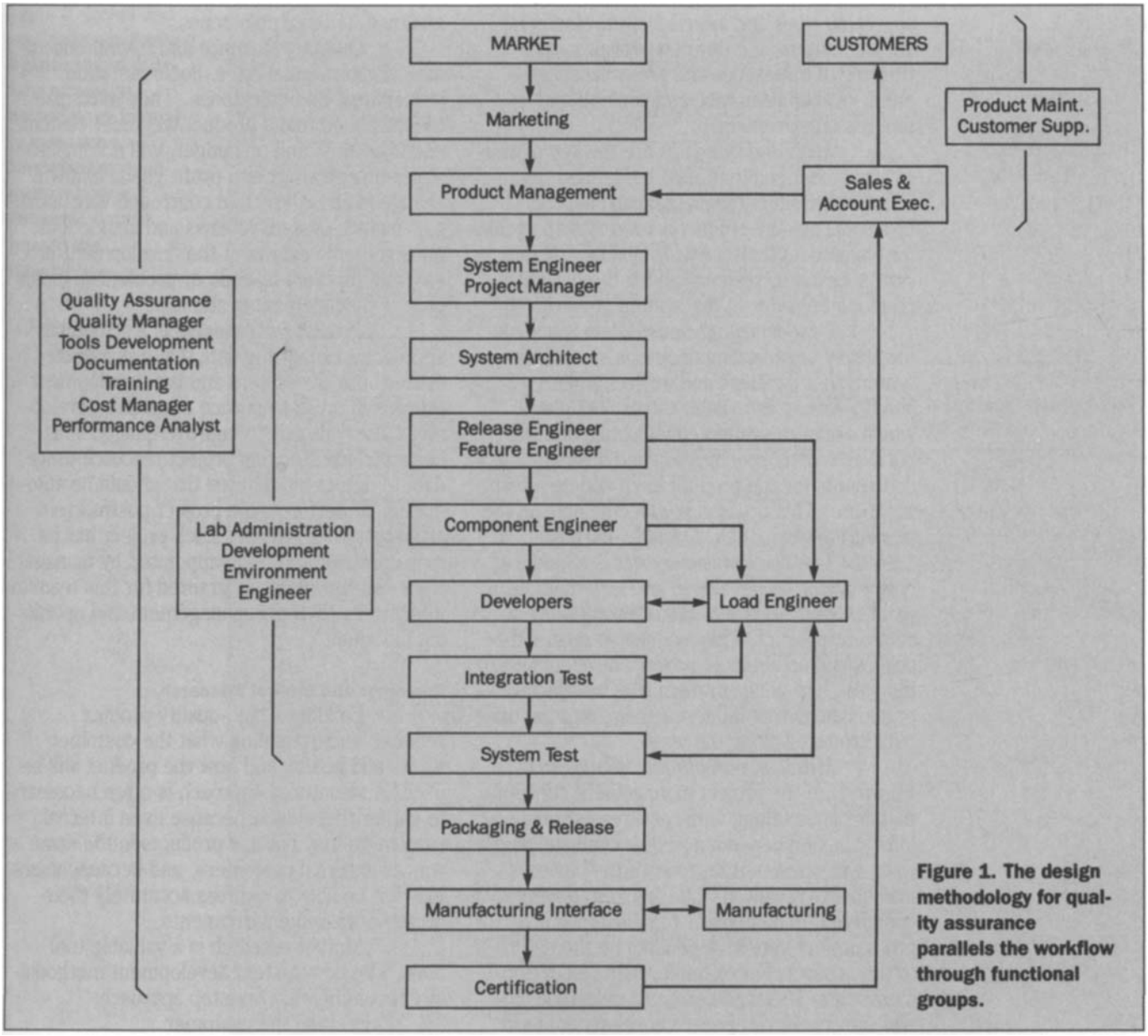

Figure 1. The design methodology for quality assurance parallels the workflow through functional groups.

approach: roles and interfaces; development documentation; the development process; tracking of milestones and project management; quality assurance and control; and process improvement.

Roles and interfaces are the key organizational functions that must be defined (Figure 2) so that product realization can succeed. Examples are the *product manager*, responsible for ongoing bottom-line profitability, and the *system architect*, responsible for the design and resulting behavior of the system.

Development documentation is a structured way of recording decisions about a given system as it develops and evolves (Figures 2 and 3). Examples include the project plans, which organize system construction and support, as well as specification and design documents for the product itself and its internal structure. This process is based heavily on the pioneering work of Dr. David L. Parnas[4].

The *development process* is a model of how a system is developed and how multiple parallel activities are organized to produce a coherent product. This model also relates the phases and activities of product development to the roles that perform them, the interfacing organizations that influence them, and the outputs produced along the way.

Milestones and project management tracking are the project management concepts that are used (along with resource estimates, intervals, and dependencies) to evaluate progress. Examples of milestones are: "business plan quality reviewed," and "system testing completed." The number of milestones in a given project varies, depending on the nature of the product. For example, feature developments have 18 scheduled basic milestones, while software component developments are assigned 14 basic milestones.

Quality assurance and control encompass organizational roles, documentation, procedures, and milestones. They maximize the likelihood that a product will meet customer needs on time and on budget, will accomplish marketing strategy and profit goals, and will change in an orderly and controlled way. Examples include system reviews and inspections, management reviews of the development process, and problem-to-code or problem-to-circuit change management systems.

Process improvement is a structured approach to analyzing data that are collected to assess both the system and the development process in order to reduce costs and development intervals and to improve quality. This could include studying project resource usage data to detect procedures that should be automated, or performing a project postmortem after customer release. Each project has its own improvement team appointed by management. Permissions are granted for this team to interface with upper management and operating functions.

Customer and Market Research

Building a high-quality product requires understanding what the customer wants and needs, and how the product will be used. A structured approach is often necessary to gather these data, because even internal customers may not use products in the same way as external customers, and because users may not be able to express accurately their preferences and environments.

Market research is a valuable tool here. The new system development methodology recommends a six-step approach:

1. Understand the customer.

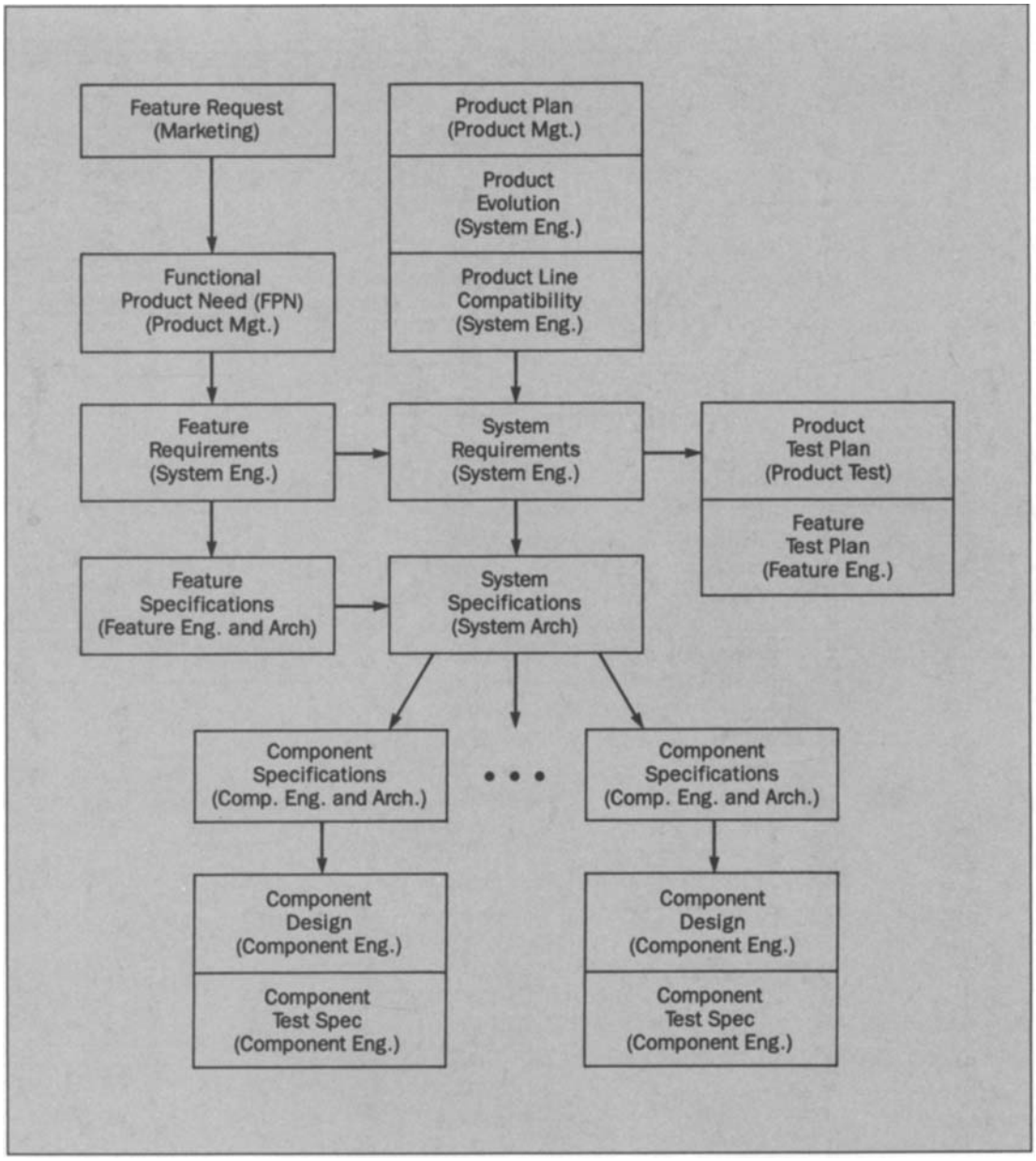

Figure 2. Roles and interfaces are the key organizational functions of design methodology in quality assurance. Development documents follow a hierarchical path through the organization.

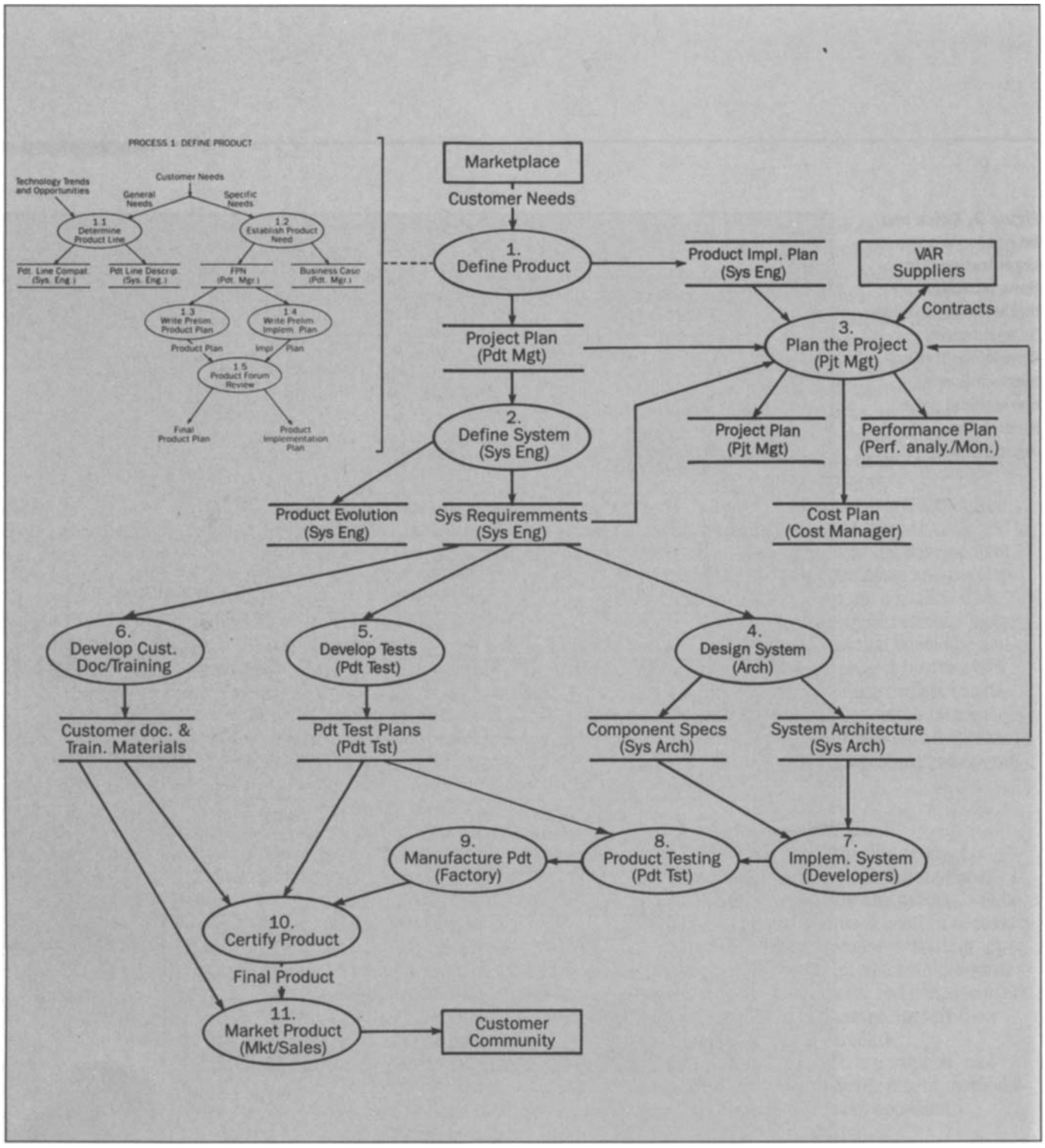
PROCESS 1: DEFINE PRODUCT
Technology Trends and Opportunities
Customer Needs
General Needs
Specific Needs
1.1 Determine Product Line
1.2 Establish Product Need
Pdt. Line Compat. (Sys. Eng.)
Pdt Line Descrip. (Sys. Eng.)
FPN (Pdt. Mgr.)
Business Case (Pdt. Mgr.)
1.3 Write Prelim. Product Plan
1.4 Write Prelim. Implem. Plan
Product Plan
Impl. Plan
1.5 Product Forum Review
Final Product Plan
Product Implementation Plan
Marketplace
Customer Needs
1. Define Product
Product Impl. Plan (Sys Eng)
VAR Suppliers
Contracts
Project Plan (Pdt Mgt)
3. Plan the Project (Pjt Mgt)
2. Define System (Sys Eng)
Project Plan (Pjt Mgt)
Performance Plan (Perf. analy./Mon.)
Product Evolution (Sys Eng)
Sys Requiremments (Sys Eng)
Cost Plan (Cost Manager)
6. Develop Cust. Doc/Training
5. Develop Tests (Pdt Test)
4. Design System (Arch)
Customer doc. & Train. Materials
Pdt Test Plans (Pdt Tst)
Component Specs (Sys Arch)
System Architecture (Sys Arch)
9. Manufacture Pdt (Factory)
8. Product Testing (Pdt Tst)
7. Implem. System (Developers)
10. Certify Product
Final Product
11. Market Product (Mkt/Sales)
Customer Community

Figure 3. System development in the quality assurance methodology depends on data from the marketplace describing customer needs to drive the product realization process. The inset details the process of defining the initial release of a system.

Table I Computer Survey

		Not Important	Somewhat Important	Important	Quite Important	Most Important	DK
18.	Hardware Features	1	2	3	4	5	6
19.	Software Applications	1	2	3	4	5	6
20.	Service	1	2	3	4	5	6
21.	Training/ Documentation	1	2	3	4	5	6
22.	Company Reputation	1	2	3	4	5	6
23.	Price	1	2	3	4	5	6
24.	Compatibility with existing system	1	2	3	4	5	6
25.	Availability of software migration tools and techniques	1					6
26.	Interconnectability with existing hardware	1					6
27.	Availability of networking and communications capabilities	1					6
28.	Ability to customize software	1	2	3	4	5	6
29.	Ability of system to grow modularly	1	2	3	4	5	6
30.	Existence of verifiable customer base	1	2	3	4	5	6
31.	Ability to benchmark vendor's machine with your applications	1	2	3	4	5	6

When adding a new computer to your system to expand your application capabilities, there are several factors that would influence your purchase decision. Using the 1-6 scale with 1 being "Not Important," 2 "Somewhat Important," 3 "Important," 4 "Quite Important," 5 "Most Important,"and 6 "Don't Know"(DK). Please READ EACH QUESTION AND CIRCLE ONE NUMBER.

The table is an excerpt from a data processing computer survey administered to specific market segments of the online transaction processing marketplace. Persons completing the survey were given the instructions shown in the inset.

2. Understand the customer's environment and tasks.
3. Understand our competition.
4. Assess how new technology or product offerings would benefit the customer.
5. Analyze our unique strengths and opportunities as well as key areas in which quality improvements would improve customer satisfaction.
6. Use this knowledge to create or modify systems and procedures to deliver quality products.

Each of these steps involves work, but each is a necessary part of defining products and enhancements.

Understanding customers means knowing what they like and dislike, what they know and are used to, and how various product characteristics contribute to their view of a product's quality. Useful tools for this step include surveys, focus groups, concept tests, and factor analysis (Table I). The work of Dr. David Garvin[1] of the Harvard Business School is particularly relevant for understanding the concept of quality.

Understanding the customer's environment and tasks may involve surveys, in-depth interviews, or on-the-job observation and interaction. Marketing research often stops short of this because industry-segment data are judged to provide enough detail. However, we believe that understanding how customers do their work is crucial in predicting how a new product or enhancement will affect that work, as well as in being important in planning customer documentation and training.

An equivalent level of detail is necessary in order to assess competitors' strengths and weaknesses and to determine how important it is for AT&T's products to be compatible with theirs. It should be apparent that the first four steps listed above may be interactive, because all of them contribute data useful to the others.

Of course, the designers may be either hypothesizing or actually working on a new system while performing Steps 1 to 4. The decisions they make while performing these steps are based on prior interactions with customers, knowledge of changing technology, etc. Grouping Steps 1 to 4 together with Step 5 enables the designers to find the small number of product characteristics that are vitally important to customers, so organizational resources can be focused on them.

Based on market input, potential product features might be eliminated, or key features might be delivered early in a product's life. In addition, a useful product or feature might be discovered. This is especially relevant in the computer and information management industries, in which some companies have been put out of business because they treated computers as commodity products. Customers are willing to pay for solutions, but not for the tools with which they are expected to construct solutions.

Finally, by understanding which aspects of products and processes are most important to the customer, AT&T can form quality improvement goals and use them to drive product development. (A useful reference for this approach is the work of Victor Basili[2,3] at the University of Maryland on goal-based measurement and data analysis, described later.)

System Structuring

The task of discovering what the customer will buy is not a trivial one, as indicated

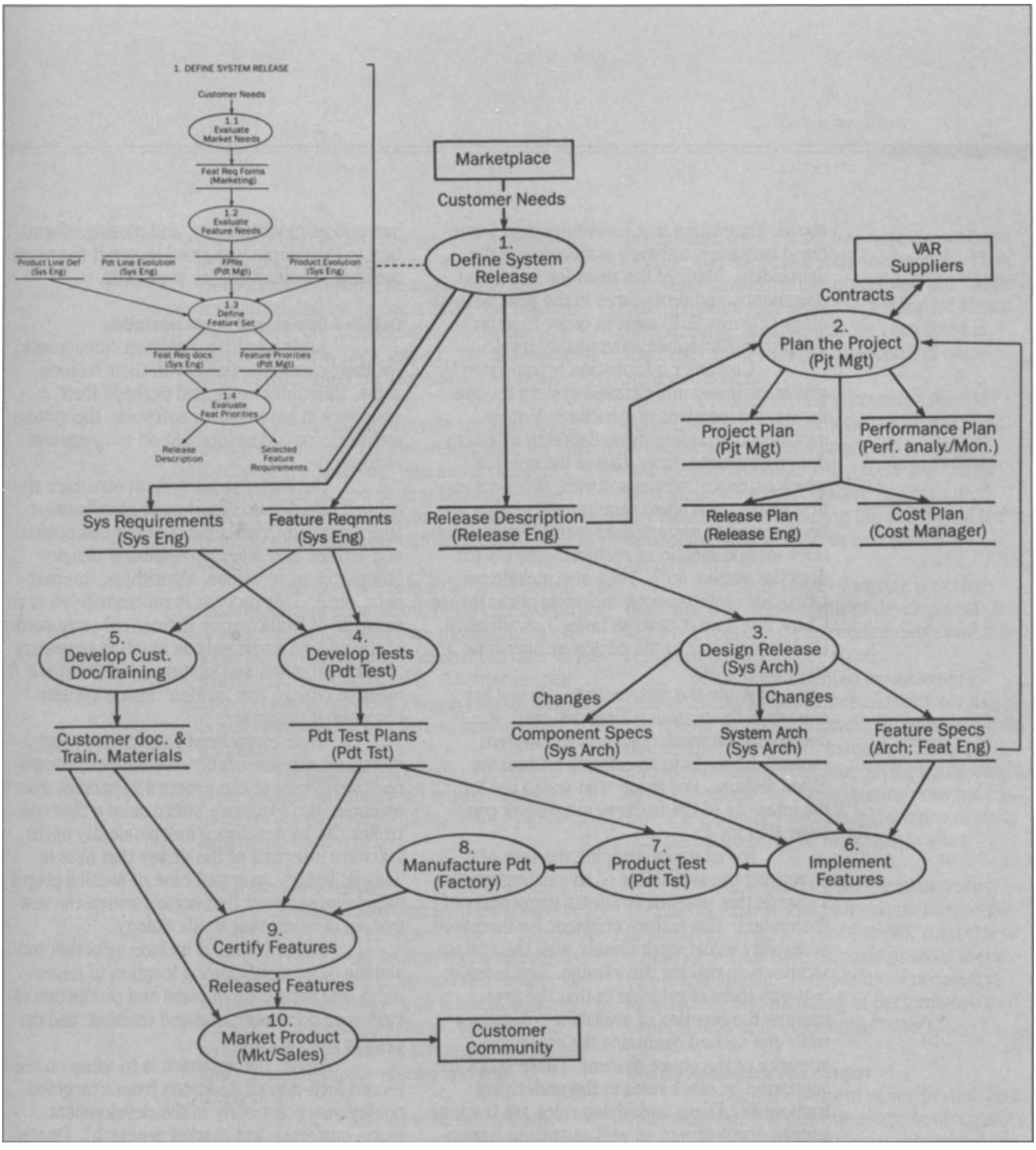

Figure 4. Feature development procedures are, like system development, market-driven by data generated from customer needs, but within the constraints of the existing system. Although the roles are basically similar, the interfaces are different for features than for initial releases. The inset shows details of how the system release is defined by customer needs.

above. Translating that knowledge into a combined hardware-software system is equally demanding. Many of the roles (or functions), documents, and procedures in the new methodology (Figures 2, 3) exist in order to improve the program's visibility and quality level.

Customer information is translated by *system engineers* into detailed system requirements, independent of structure. *System architects* then refine these data into a system design or architecture. This is the point at which hardware versus software decisions are first made—and when requirements begin to be allocated to system *components*. Such allocations include functional requirements ("what does the system do"). They also include nonfunctional requirements, including performance ("how fast does it process tasks"), availability ("how useful is it to the customer over time"), cost, and quality.

While the architecture role is a key one when the system is first designed, it becomes even more crucial as the system evolves. Changes to an existing system are called *features*, and those who watch out for the integrity of the features are *feature engineers* (Figure 4).

For example, imagine the task of increasing the availability of an existing system, a change that potentially affects many parts of the system. The feature engineer for increased availability would work closely with the system architects in making this change. The tension between them is valuable in that the first ensures the meeting of availability objectives while the second maintains the structural integrity of the entire system. These tasks are supported by other roles in the underlying framework. These underlying roles are tracking feature development as well as system component changes, documenting the feature as a unit and as modifications, and routing information to the people who need to see it to ensure meeting specified quality standards.

Detailed Design and Implementation

After identifying system components (or their changes), along with their relationships, their interfaces, and perhaps their residency in hardware or software, the system architects pass development on to *component engineers*.

Decisions about system structure are refined into detailed *component specifications* (the externally visible parts of the component) and further into *internal component designs* (subpieces or functions, algorithms, internal data, etc.). This process is recommended as an example of "information hiding," wherein parts of the system know only as much as necessary about each other, and system details that are likely to change are "hidden" inside system component designs.

Once component relationships and interfaces are reasonably well defined, component development can proceed in parallel. For example, the hardware internals of a disk controller can be developed independently of the software internals of the driver that uses it. This is, in fact, an actual case of parallel coordinated development that occurs within the new system development methodology.

Other examples include selection and startup of a manufacturing location (if necessary), the formation of plans and production of customer documentation and training, and the system testing effort.

Again, the approach is to focus on and record high-impact decisions from a targeted quality viewpoint early in the development (e.g., customer and market research). Dependent activities (e.g., implementation and

testing preparations) can proceed independently once these decisions are made. Decisions that cannot be made early are accommodated by saving a space for them in higher-level documents, structuring the system to minimize their impact, and assessing their impact when the decisions have been made.

By and large, the new methodology uses the same procedures, roles, etc., for both hardware and software. But some design, implementation, and testing techniques are specific to whichever medium is chosen, in order to allow intelligent architectural decisions.

One example is in automated design description and generation, in which hardware development leads software development via tools for computer-aided design/computer-aided manufacturing (CAD/CAM). Another is code inspections, in which the inspection technique is just beginning to be used on hardware information.

These differences are expected to grow smaller as integrated system development allows hardware and software systems to learn from each other.

Integration and Testing

As mentioned earlier, testing follows a parallel path to system development. This path runs from the lowest level of the system (formation of "unit" test plans and test scripts for software and hardware with internal designs and code) to the highest level, in which customer needs and market strategies are turned into customer profile testing and test marketing.

In each case, as with the system development, the AT&T approach distinguishes between the planning for the testing activity, the process used during test development and testing, and the actual tests themselves. Thus, unit tests have coverage objectives, are "white box" in nature, are stored in a standard format under change control, and are correlated in their execution with the development of the system internals.

Product testing recognizes several stages of system construction. The unit test is followed by system integration testing, which often combines subsystem and component integration, plus feature integration. A key factor here is ensuring the stability of the developing system base as new pieces are added to it.

The next stage of testing is system test, which is independent of development. It tests both adherence to requirements and system performance under stress.

Customer-oriented or customer-profile testing simulates actual customer use of the system in order to assess its reliability from the perspective of the customer. This is an important concept, because giving equal weight to all requirements during testing does not capture the frequency and job importance of tasks that customers impose on products after delivery.

Finally, product certification testing checks the final product for completeness (documentation, ordering procedures), correctness (behavior versus user manuals or sales literature), and usability (availability of necessary information, and matching of performance to expectations of a "reasonable" user).

Quality Assurance and Control

Testing is one form of verification and validation, but it is just one example of a quality control technique. Others exist throughout the process.

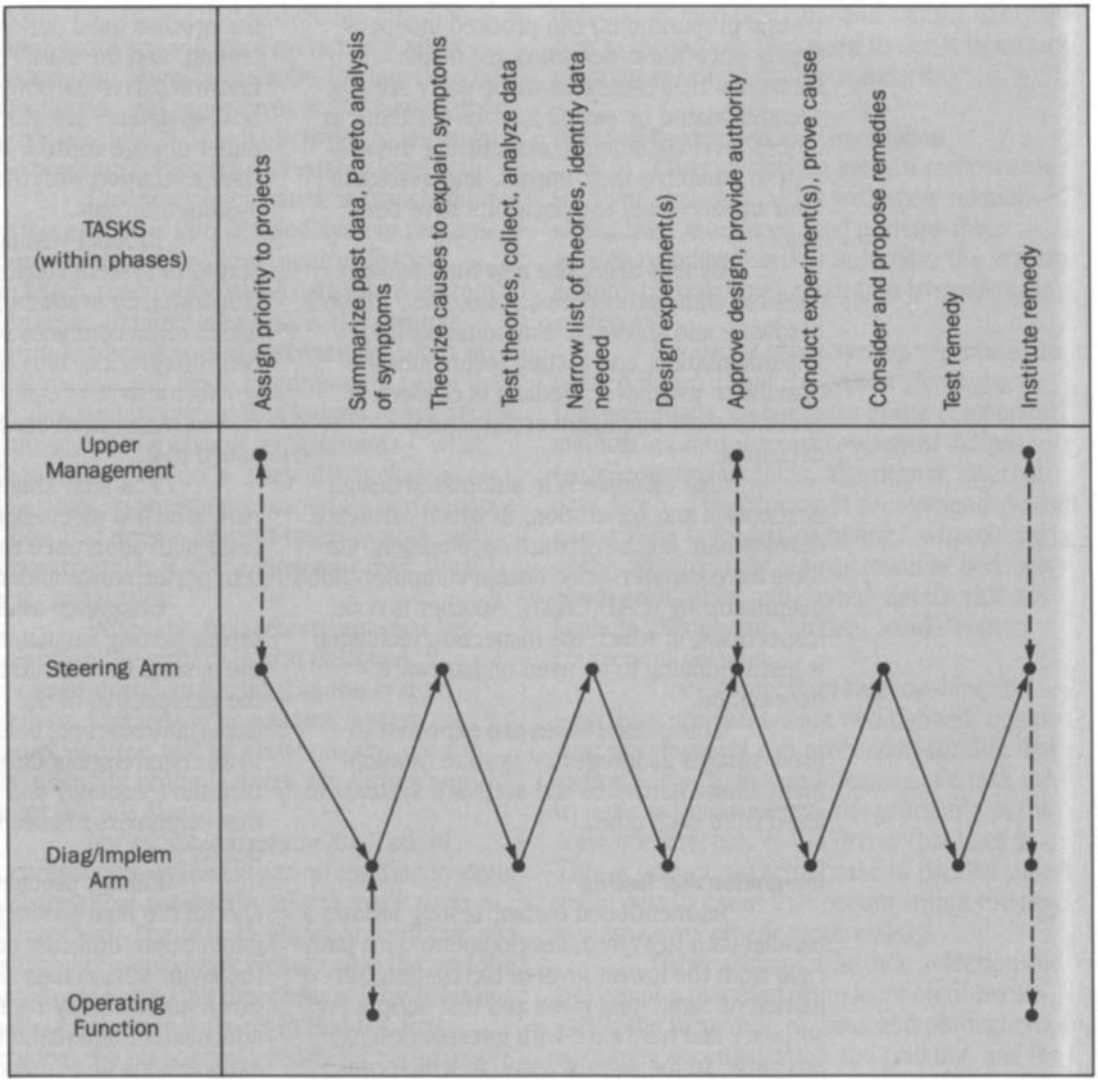

Figure 5. The responsibilities of the improvement team's steering and diagnostic/implementation arms are mapped in this diagram. The tasks listed at the top encompass one phase or activity, and are repeated in each activity involved in the process. Solid lines show the flow of information and delegation of tasks. Dashed lines show interaction with upper management to obtain approval or to implement solutions.

Another important technique is the *development control team,* a group of development managers who review both design and development documents, as well as the processes and issues involved with the review of those documents by development peers (*peer reviews*). This team often reviews document outlines and designates document reviewers in advance, thus playing a quality assurance role similar to that of the peer reviews.

Quality assurance makes use of quality control to detect and avoid adverse change, and quality improvement to implement beneficial change— in effect, to meet *new* quality standards. The improvement team is appointed by management for each project, and consists of a steering arm (to direct) and a diagnostic/implementation arm (to work) through the quality improvement process (Figure 5). If an improvement project is interdepartmental, the team must include representatives from those departments.

Quality assurance and control activities within a product's development cycle usually have entry and exit criteria that are based on general notions of quality (e.g., 100-percent code or branch coverage during unit test). Criteria can also be based on specific product quality objectives based on statistical models (e.g., "no more than N faults remaining and greater than Y reliability at system delivery," or "performance better than X percent of a competitor's product").

Such objectives are examples of measurements of progress, or *metrics,* taken during the product's development to alert management to problems and to quantify progress toward a customer-ready system. The Basili paradigm is relevant here: Beginning with product goals, define subgoals, formulate questions (whose answers would demonstrate goal fulfillment), and identify measurements (that supply data to answer those questions). Then periodically measure, analyze, and compare the resulting data against goal-derived objectives to assess progress.[2,3]

This concept, combined with tie points (called Q-values) in the development process, enables AT&T to build targeted quality into complex systems and track the quality level over time.[6]

Summary

This report describes a design methodology for use of targeted quality control when building complex systems that can include many hardware and software components. The design methodology is comprised of many components, but all of them are based on two fundamental beliefs: that focused quality control can be managed in AT&T products and that the customer is the ultimate definer of quality.

All of the quality program components are already in use on AT&T projects, and some projects are using all components plus additional techniques. There are documented cases of AT&T projects that, though using only a few quality program components, have reduced maintenance costs by a factor of two-thirds in no additional time. A systematic use of the overall approach should result in dramatic savings of both time and effort.

References

1. David Garvin, "Product Quality: An Important Strategic Weapon," *Business Horizons,* March-April 1984.
2. Victor R. Basili, "Quantitative Evaluation of Software Methodology," *Proceedings of the Pan Pacific Computer Conference,* Melbourne, Australia, 1985.

3. Victor R. Basili, "A Methodology for Collecting Valid Software Engineering Data," *IEEE Transactions on Software Engineering*, Vol. SE10, No. 6., 1985.
4. David L. Parnas, "A Rational Design Process: How and Why to Fake It," *IEEE Transactions on Software Engineering*, Vol. SE12, No. 2, February 1986.
5. J. M. Juran, *Quality Control Handbook, Third Edition*, Section 16, McGraw-Hill, New York, 1974.
6. R. H. Yacobellis, "Software and Development Process Quality Metrics," *Proceedings IEEE International Conference on Data Engineering*, 1984.

Biographies (continued)

information sciences from the University of Chicago, as well as an M.B.A. degree from the University of Chicago Executive Program.

(Manuscript received October 25, 1985)