AS/400 software quality management

by S. H. Kan
S. D. Dull
D. N. Amundson
R. J. Lindner
R. J. Hedger

This paper describes the software quality management system for the Application System/400® (AS/400®) computer system. Key elements of the quality management system such as customer satisfaction, product quality, continuous process improvement, and people are discussed. Based on empirical data, recent progress in several quality parameters of the AS/400 software system are examined. The quality action road map that describes the various quality actions that were deployed is presented, as are the other elements that enabled the implementation of the quality management system.

IBM develops and manufactures the Application System/400* (AS/400*) computer system at its Rochester, Minnesota, site. Generally available to customers since August 1988, the initial release of the AS/400 had 7.1 million lines of source code in its software system—the base operating system and licensed program products. Since then, many new functions and enhancements have been added with at least one new release each year. The customer base has been expanding, with more than 225 000 licenses at mid-year 1993. The typical release usually has about two million lines of new and changed source code. With such a large development effort and so many customers, continuous quality improvement is a necessity. Indeed, focusing on quality has always been one of the top priorities at IBM Rochester. The concern of the site about quality can best be indicated by its winning of the Malcolm Baldrige National Quality Award in 1990, and its obtaining ISO 9000 registration for the entire site at the end of 1992.

In early 1990, IBM began deploying the corporate strategy of market-driven quality (MDQ) to its divisions and business units. Capitalizing on the new MDQ momentum and the ongoing effort, the software development laboratory at IBM Rochester quickly undertook several important activities: benchmarking studies of quality leaders such as Motorola, Inc., and IBM Houston (which develops the NASA Onboard Shuttle flight software system and has achieved defect-free quality[1]); assessment of the AS/400 development process; analysis of in-process and field defect data to guide improvement efforts; and development of a long-term quality improvement strategy. A quality action road map was soon established, and deployment of key action items followed immediately. These key items included: the laboratory-wide implementation of the defect prevention process (DPP),[2,3] a strong focus on the design review and code inspection (DR/CI) process (referred to as the "back to the basics" focus in the laboratory), component test improvement, departmental 5-UP measurements, quality recognition based on peer nomination, and others. Strong management commitment, the entire team's passion for quality, and the anticipation of what the MDQ vision could bring about formed the best climate for improvement actions.

Since the initial MDQ deployment, the AS/400 software quality management system of IBM Rochester has evolved significantly; it now encompasses all aspects of software quality. This paper describes the key elements of the system: people, product quality management (both in-process and post-general availability [GA]), continuous process improvement, and customer satisfaction management. Presented is the AS/400 quality road map that describes the goals and key action items that drive these elements. Included are examples of the results achieved as described through empirical data. Where appropriate, the climate for quality improvement (such as management commitment and the mind-set change of the entire team) is also discussed.

There are five elements of the AS/400 software quality management system.

It should be noted that discussions in the following sections are about the AS/400 software quality management system, and there are no "silver bullets." Many quality improvement techniques and recommended solutions exist in the literature. Putting these techniques into practice systematically and persistently makes the difference. It is important to continue to look for new technology for quality breakthroughs; it is equally important to focus on implementation and to bridge the gap between state of the art and state of practice.

Furthermore, we confine our discussions to the two key ingredients of quality: reducing product defects and improving customer satisfaction. The lack of functional defects, or reliability, is the most basic measure of quality. Customer satisfaction, in contrast, represents the final evaluation of the product and service by the customer, based on all variables. Other quality attributes such as performance, installability, usability, and so forth are discussed only in the context of customer satisfaction. Improving those quality attributes is important. In IBM Rochester software development, specialized groups address each of those key attributes. However, detailed discussions of them are beyond the scope of this paper.

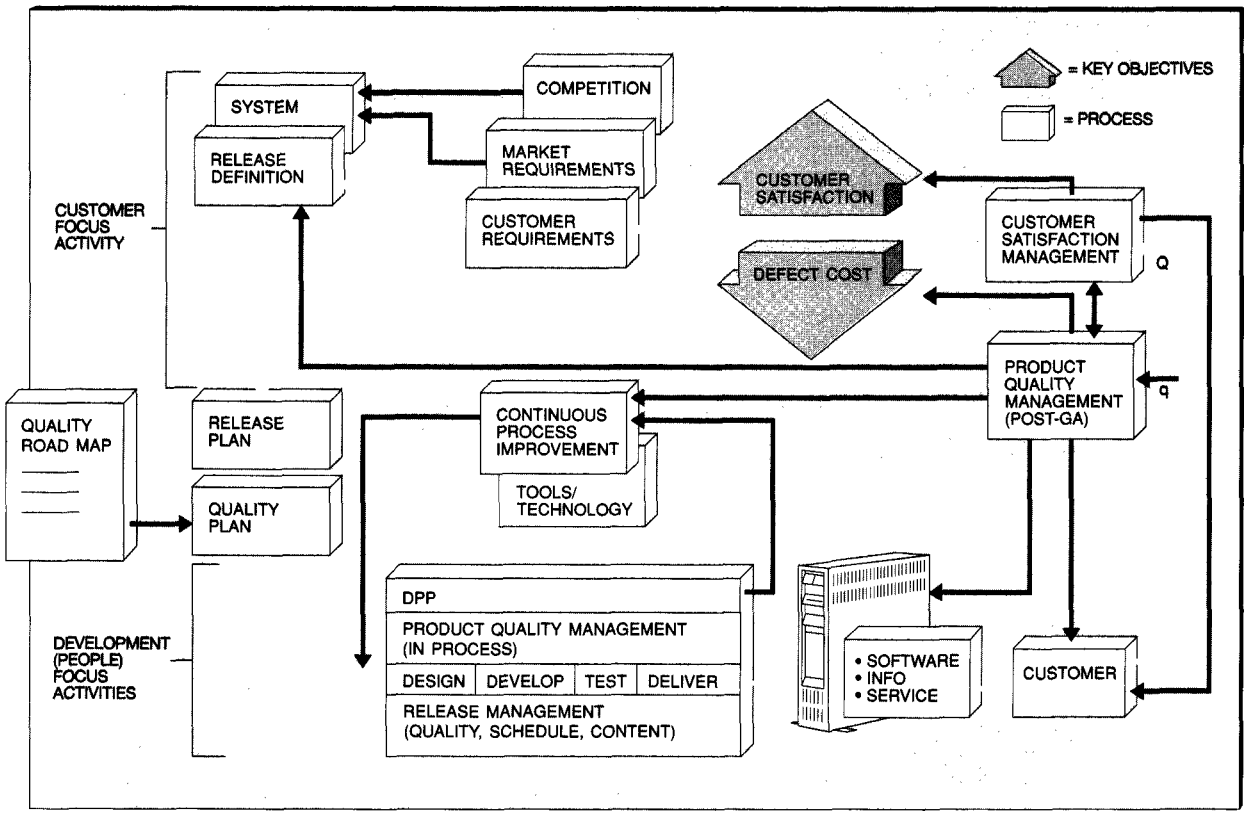
AS/400 software quality management system

After customers receive and use a product they have chosen, they grade the product based on their personal usage experience. If they experience many problems or frustrations in using the product, their opinion of the product will be negative. If the product is essentially defect-free or solves a key problem for their business, their opinion of the product may be very positive. A quality management system must focus on reducing the number of defects that a customer experiences with a product, and if a customer does experience a problem, the quality management system must ensure that the customer receives a quality "fix" in an acceptable amount of time. In other words, the end result of a quality management system is to reduce defects and increase customer satisfaction. These results are the key objectives of the various elements of the AS/400 software quality management system (SQMS), depicted in Figure 1.

There are five elements of the AS/400 SQMS: (1) people, (2) in-process product quality management, (3) continuous process improvement, (4) post-GA product quality management, and (5) customer satisfaction management.

As shown in Figure 1, before development begins, competition, marketing, and customer requirements are analyzed, prioritized, and selected for implementation on the basis of resources. These analyses and activities provide direction for the system and form the release definition for development. A release plan is developed that defines the detailed contents and schedules of a release. The AS/400 SQMS also requires a quality plan. This plan documents the specific quality actions from the quality road map that will be leveraged in this release. The quality road map describes the quality technologies that can be deployed across many releases of AS/400 that will help achieve the quality goals. During the development of a release, the people factor, the in-process quality management element, the continuous process improvement element (which encompasses tools and technology used in the development process), and the overall release management are

Figure 1 AS/400 software quality management system



the significant pieces affecting the deliverables. When the release development is complete and the product is delivered to customers, the post-GA product quality management element and the customer satisfaction management element are in full operation. Feedback from customers is then incorporated into the customer requirements analysis, which influences the definition for future releases.

The most important element of the quality management system is people. People must be highly motivated and talented to execute and improve the development processes to deliver a high-quality product to customers release after release, with each release quality goal more challenging than the previous one.

In-process product quality management is another important element of the AS/400 SQMS. The objective of this element is to measure the results of the various process steps used in developing a product. These in-process measurements help to determine if the product is on target for achieving product quality goals. Action plans are created to improve the quality of functions that are not meeting the quality plan goals. The release management process uses the in-process quality measurements at checkpoints during the release to assess the overall quality of the release. Content and schedule are also critical parts of this assessment.

Another very important element of the AS/400 SQMS is continuous process improvement. This element provides a foundation for improving the processes used in the development cycle. Process improvement is triggered by defects found by customers using the product; by the defect prevention process (DPP), which is a mechanism for preventing defects from recurring that are found while developing the product; and by use of tools

and technology that enable process automation, increase defect discovery, or reduce defect injection. Process owners continually look for ways to improve their processes and communicate improvements to their users.

The post-GA product quality management element addresses the problems that AS/400 customers may experience with software products, both defect-oriented (requiring a fix) and non-defect-oriented (user errors, usability problems, etc.). The key objectives of this element are to fix customer problems quickly, with quality, and to learn from the errors that were made. The number of problems customers report, defects found in the product, defective fixes that impacted customers, and the response time to problems are measured. Some problems reported by customers may become new requirements that in turn are fed into the software planning process. Others may result in suggested improvements to the software development processes.

The objective of the customer satisfaction management element is to improve customer satisfaction via a closed loop process. To understand overall customer satisfaction and customer satisfaction in different parameters of the product, large-scale surveys are conducted. Analysis of survey data identifies areas of the product that are deficient. As a result, a set of new product requirements is defined that can lead to improving the product and hopefully to improving customer satisfaction. Customer satisfaction management also addresses critical problems that customers are experiencing, problems that need immediate and high-priority attention by developers, marketing, and service teams. A team of people monitors critical situations and reacts to quickly resolve problems before they seriously impact the customer's business. Such quick responsive action can turn a dissatisfied customer into a satisfied one.

In the following subsections, each of these elements is described in more detail. Real examples of results achieved from the AS/400 SQMS are included. For a better flow of the information, we realign the order of discussion as follows:

- Customer satisfaction
- Product quality
  - In-process
  - Post-GA
- Continuous process improvement
- People

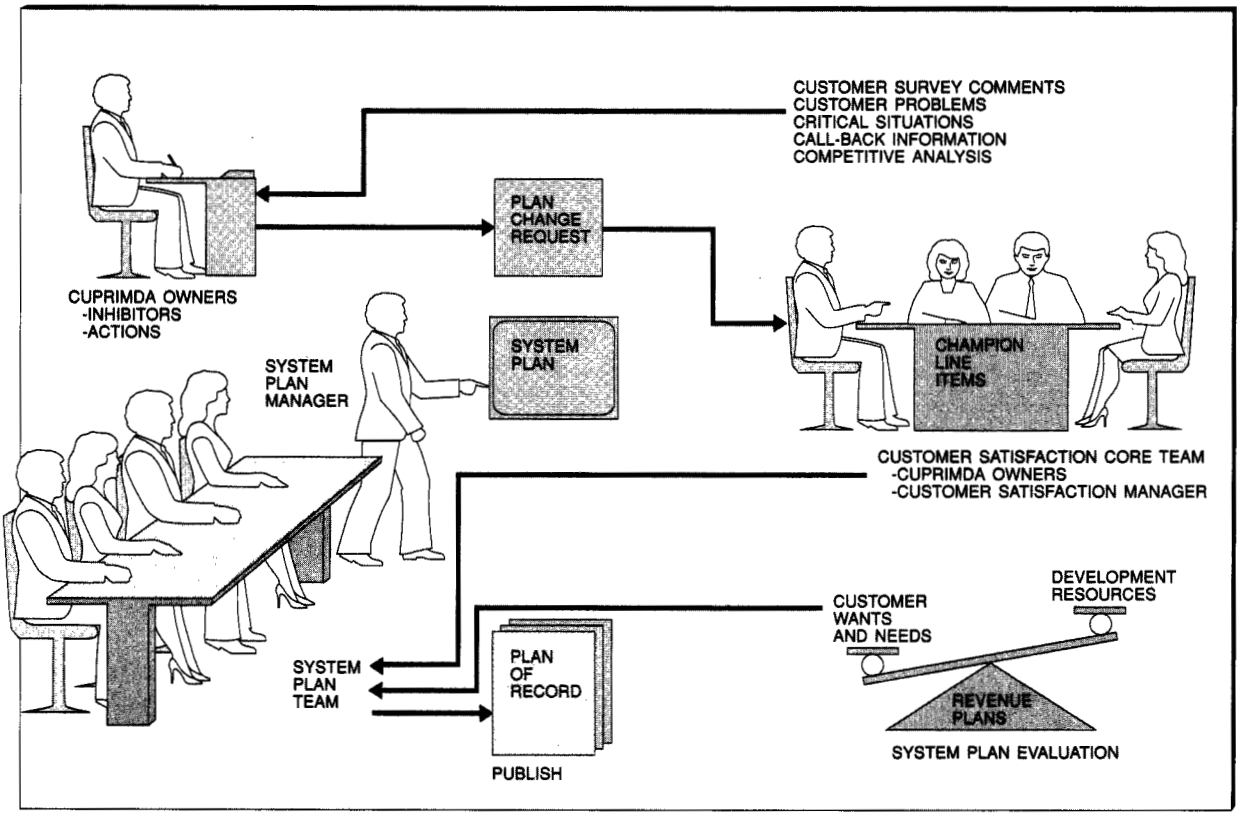
Customer satisfaction management. The ultimate goal of IBM's MDQ is to satisfy customers totally with the products and services provided. The goal for the AS/400 in customer satisfaction is to be the undisputed leader worldwide. To work toward this goal, a customer satisfaction management process is in place (Figure 2), and, in addition to overall satisfaction, specific categories called CUPRIMDA are monitored and measured for improvement:

The goal for the AS/400 in customer satisfaction is to be the undisputed leader worldwide.

- Capability (function)
- Usability (ease of use)
- Performance (response time and throughput)
- Reliability (defect-free)
- Installability (ease of upgrade)
- Maintainability (ease of maintenance)
- Documentation (information)
- Availability (nonoutage time)

Each of these categories has been assigned to an owner and is measured at the system and product level by continuous surveys. As Figure 2 indicates, each CUPRIMDA owner is responsible for gathering and analyzing customer satisfaction information from various sources (such as surveys, problems reported, critical situations, competitive analysis, and customer calls), identifying inhibitors for improvement, and recommending solutions to improve customer satisfaction. When the recommended solutions call for specific items to be added or changed in the development plan, formal plan change requests (PCRs) are created. These solutions (PCRs) are reviewed by a core team of category owners and chaired by a development director. The director is also a member of the system plan team and provides the necessary

Figure 2 Customer satisfaction management process



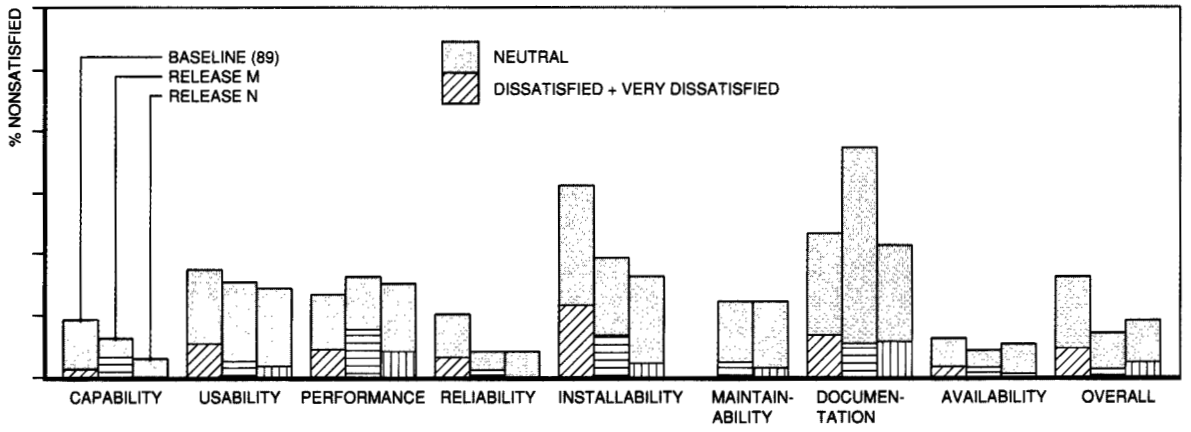
focus in the plan change process. The core team becomes the champion of customer satisfaction line items for inclusion in the software planning process. These line items are presented to the system plan team for consideration in the official plan. The plan team evaluates the line items against other customer wants and needs and the resources available to develop them.

In addition to initiating line items that address improvement in their parameters, the CUPRIMDA owners are responsible for (a) coordinating and driving other actions to improve their parameters, and (b) monitoring and projecting customer satisfaction levels for their parameters.

On rare occasions, a customer may run into a critical AS/400 problem. The causes of these problems vary and are often unique situations. When they occur, the critical situation management process goes to work. The primary focus of the team assigned to use this process is to resolve the critical situation as fast as possible, and to turn a dissatisfied customer into a satisfied customer. This process enables people from development, manufacturing, marketing, and service to immediately address the critical problems. On-site assistance may be required to understand the problem or provide the fix. Causal analysis is performed on these problems to prevent future recurrences.

Figure 3 shows an example of customer satisfaction in each of the CUPRIMDA categories based on the IBM marketing and services (M&S) surveys. It compares customer nonsatisfaction in each of the CUPRIMDA categories as well as for the overall operating system for the baseline (year-end 1989) and two recent releases. Customer nonsatisfaction is the percentage of customers that are either neutral, dissatisfied, or very dissatisfied. Analyzing customer satisfaction in terms of the nonsat-

Figure 3 AS/400 M&S customer satisfaction survey results



isfied percentage enables attention to be focused on areas that need improvement. The M&S surveys are conducted in the United States by an independent consulting company. They are blind surveys: the interviewee does not know what company the consulting firm represents. Customer responses are obtained through telephone interviews based on structured questionnaires.

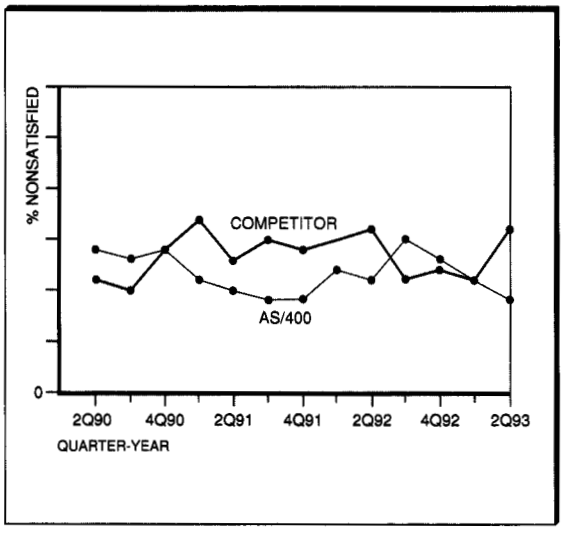
In Figure 3, the first bar of each group represents the percentage of nonsatisfied responses of the parameter at baseline; the second and third bar represent the percentage of nonsatisfied responses for two recent releases. For the maintainability parameter there are only two bars because data were not available for the baseline. For each bar the upper empty segment represents the percentage of neutral responses, and the lower shaded segment represents the percentage of dissatisfied and very dissatisfied in a five-point Likert scale. The total length of the bar thus represents the percent of nonsatisfied customers.

Compared to the baseline, capability, reliability, installability, and overall satisfaction have made significant improvement in recent releases. For capability and reliability, the nonsatisfied level is quite low; in fact, there were zero percent dissatisfied responses for the latest release in the survey. For installability, the percentage of dissatisfied and very dissatisfied responses had dropped to a very low level; however, additional improvement is needed as the percentage of neutral responses was still relatively high. For usability, documentation, and availability, some improvement was observed. For performance, no clear perceived improvement was observed; this parameter is clearly a continuing challenge to our improvement effort. (It should be noted that the data pertain to customers' satisfaction with software based on surveys. We have been continually focusing on performance improvement of the software system. Furthermore, compared to the original hardware models in 1988, at this writing [mid-1993], AS/400s are 2 to 10 times more powerful. Performance of the high-end AS/400 models has increased an average of 60 to 70 percent a year.)

In Figure 4, the quarterly trend of percentage of nonsatisfied responses with the overall operating system quality is shown for the AS/400 system and a major U.S. competitor. The data are also from the IBM M&S surveys. Data are shown starting from the second quarter of 1990; earlier data on the competitor are not available. For AS/400, the data are based on the latest release in the field.

The M&S data show that AS/400 lagged behind the competitor in overall satisfaction with the oper-

Figure 4 Trend of overall customer satisfaction with operating system: AS/400 and competitor



ating system before 1991. Since then, AS/400 has gained substantially and had a lower percentage of nonsatisfied responses than the competitor. However, AS/400 lost this advantage in the second half of 1992, only to be back on the improvement trend again in 1993.

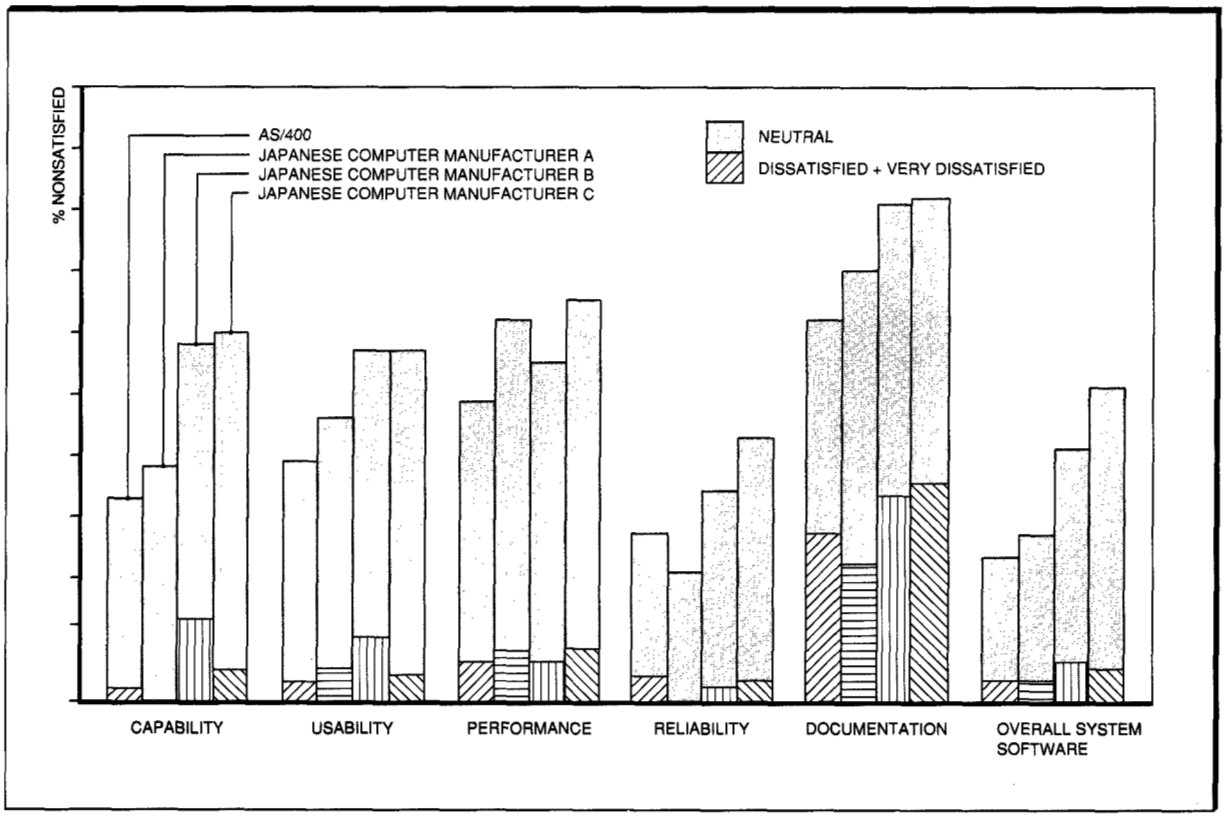
The data in Figure 4 show the volatile nature of customer satisfaction, which needs constant attention. Our analysis also indicated that overall customer satisfaction is affected not only by product quality, but also by factors such as marketing, distribution, support, and very importantly, how the company is perceived by the customers. Recently, the AS/400 Division Director of Development commissioned a special task force on customer satisfaction, which is now evaluating all possible factors affecting satisfaction from the customer's view and studying the mechanisms for improvement.

Figure 5 shows the customer satisfaction data for the AS/400 software system and the products of three Japanese computer manufacturers (JCM). The data are based on a survey conducted by IBM Japan in 1992. In the figure, the first bar in each group represents the percentage of nonsatisfied AS/400 customers. The other three bars represent data for three JCM products: A, B, and C, respectively. AS/400 had the best customer satisfaction overall and for specific categories except reliability, which was ranked second. It should be noted that the scale is the same for Figure 5 and Figure 3. The difference between the data in the two figures is intriguing. However, we cannot validly compare data across the two figures because the sources of data are two entirely different surveys.

Product quality management. To reduce defects and improve reliability in all aspects, the dynamics of software reliability must be understood. Figure 6 shows a schematic representation of the software reliability dynamics. When a software product is developed and becomes available to the marketplace, there is a certain level of latent defects (defects that have not manifested themselves earlier). Customers use the product, detect the defects, and report them. As defects are discovered, reported, and fixed, the latent defect rate in the system decreases. Over time, the system becomes more and more stable, and its reliability grows. This process is known as aging or reliability growth. Fix quality and old code improvement are pertinent factors for a smooth aging process. If fix quality is not perfect (for instance, incorrect or with unintended consequences that may or may not be found immediately), new errors are injected into the system. Perhaps even more significant is that defective fixes are detrimental to customer satisfaction. From the customer's perspective, encountering defects while using the product is bad enough; if a fix turns out to be defective, frustration will only multiply.

Theoretically, with aging, the reliability of the software system can only get better, and the curve will be a monotonic decreasing function. This pattern of software reliability is different from the hardware "bathtub" reliability pattern in which reliability deteriorates at the end of the product life when hardware components begin to wear out (hence, the defect rate curves up again at the tail). If fix quality is good, through normal aging a software product could reach the six sigma quality level[4] within a few years. In reality, after the first release of a software product, enhancements and new functions are made, and new releases of the product become available at regular intervals. The latent defect rate of the new and changed source instructions (CSI) is usually higher than that of the base system that has been undergoing aging. Therefore, when a new release is available, the overall system latent defect rate

Figure 5 Customer satisfaction: AS/400 and three Japanese computer manufacturers



will increase, and the reliability worsens slightly. This phenomenon is represented by the spikes in Figure 6. The more CSI a new release contains, the higher a spike will become. If the base system has been aging for many years and CSI quality has not improved, the gap between the two (base code quality and CSI quality) will continue to widen. As a result, the overall latent-defect-rate function may curve up again at the tail, resembling the hardware "bathtub curve."

Therefore, continuous improvement in CSI quality is necessary to minimize the impact of new releases on system quality. CSI quality is the genuine indicator of the quality of the development process. It must be tracked separately using in-process measurements and dealt with specifically in quality road maps and plans. The following subsection on in-process product quality management describes how we manage CSI quality indicators.

Figure 6 Software reliability dynamics

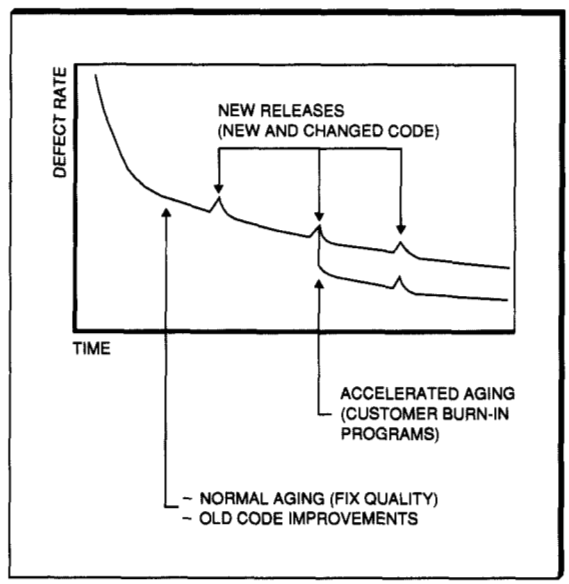
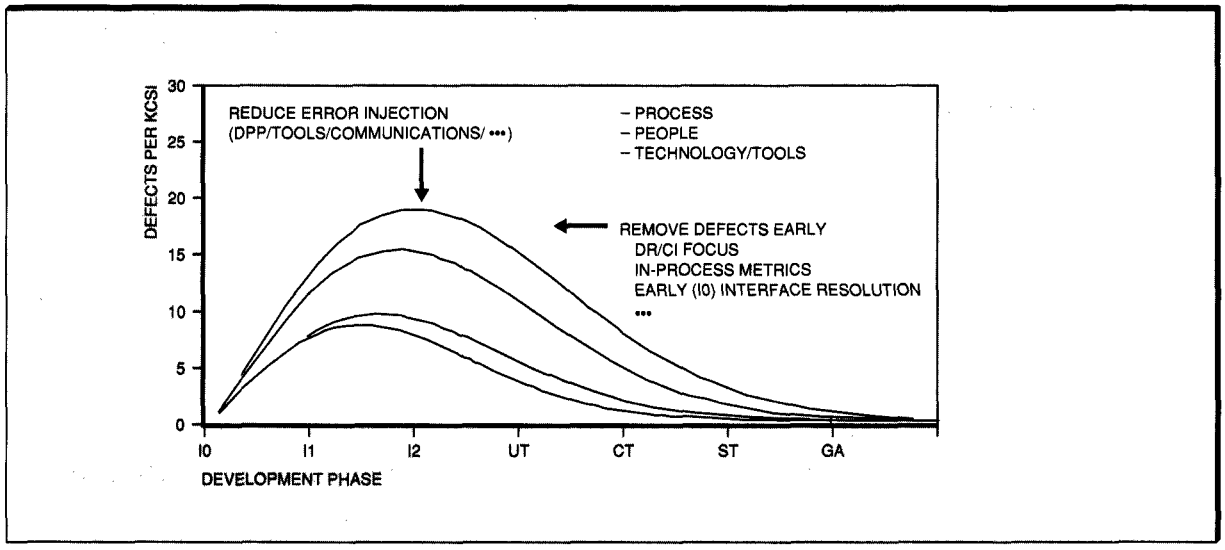


Figure 7 Development quality improvement directions

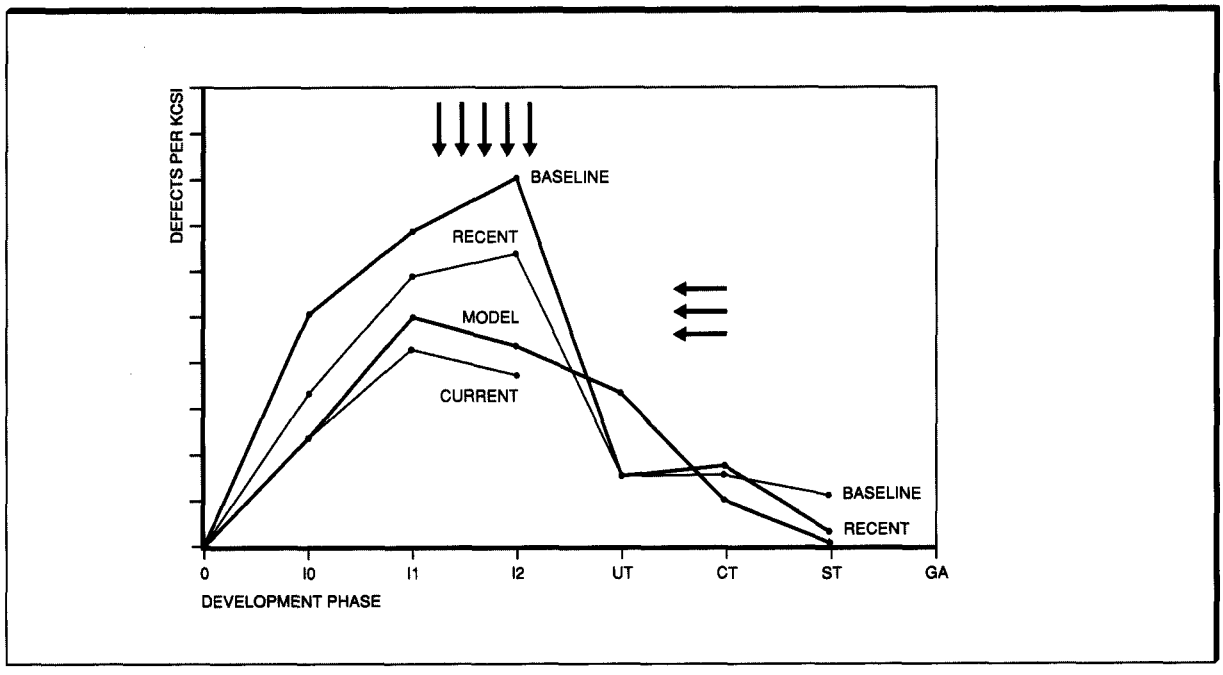


Another way to improve reliability (reduce defects) relies on the concept of accelerated aging (similar to accelerated testing of hardware). Many customer burn-in programs are based on this concept. In such programs, high-defect-finding customers are invited to participate in special programs to accelerate the defect discovery process. Customer burn-in programs are conducted when programming development is complete and before the general availability of the product. Customers are encouraged to move their production applications over to the new release so that defect discovery is effective. At the same time, the development organization provides special technical support to these customers so that potential risks to their businesses are minimized. The defects found are fixed as quickly as possible with excellent quality. Therefore, when the majority of customers receives the new release after the burn-in program(s), they will benefit from better quality. The customer burn-in program for the AS/400 is called the Customer Quality Partnership (CQP) program (discussed later in this paper).

By understanding the software reliability dynamics, we can address all possible aspects for better reliability from the customers' perspectives.

In-process product quality management. The in-process product quality management element focuses on improving code quality, particularly CSI quality. Two key directions must be followed to improve CSI quality: (1) reduce the number of errors injected during the development process, and (2) remove defects as early as possible. Figure 7 illustrates these directions in relation to the defect removal model. The upper curve is the defect removal model derived from an actual defect removal pattern by phase of development: high-level design review (I0), low-level design review (I1), code inspections (I2), unit test (UT), component test (CT), system test (ST), and general availability (GA). The lower curves are the intermediate and final goals that are to be achieved. They represent more efficiency and effectiveness in the development process (much lower error injection and very early defect removal). The purpose is to shift the peak of the curves in two directions simultaneously: move to the left as much as possible and at the same time push the curve downward. A key intermediate goal is to minimize the defect rate during formal machine testing, therefore leading to low field-defect rates. In formulating the AS/400 quality road map (see the subsection about the quality road map within the next section), improvement actions for both directions were addressed. For example, the pre-I0 inspection process was developed to specifically reduce error injection. Minibuilds, which provide the developer with a preintegration test environment,

Figure 8 AS/400 defect removal patterns



are a good example of an early-defect-removal-type action item that was implemented.

The two key directions expressed by the above model are also closely related to the axiom: do it right the first time. In the context of the model, this axiom has a threefold interpretation:

- The best scenario is to prevent errors in the first place (i.e., reducing the error injection in the process).
- When errors are introduced, improve the front end of the development process to remove as many of them as soon as possible. Specifically, in the context of the waterfall development process, rigorous design reviews and code inspections are needed.
- If the project is beyond the design and code phases, before the code is integrated into the system library, unit test or any additional tests by the developers serve as the gatekeeper for defects that escaped the front-end process. In other words, the phase of unit test or preformal test (the development phase prior to system integration) is the last chance for the do-it-right-the-first-time axiom. If high defect rates are found after integration (during formal machine testing phases), clearly the axiom is not achieved by the project. Failure to reduce the errors injected or to remove defects early results in higher maintenance costs and negatively impacts customer satisfaction.

We have made significant progress in AS/400 software quality since year-end 1989. Because of the systematic implementation of improvement actions within the context of the quality road map, improvements in development effectiveness have been made. For instance, overall error injection during the development process was reduced by more than one-fourth. The front-end phases (before system integration and build) were executed better. Because of the front-end improvement, a 30 percent reduction in the overall formal machine testing defect rate and a reduction in the system test defect rate by more than 60 percent have been achieved. Our overall strategy of pushing for earlier defect removal and reducing error injection is being realized. Figure 8 shows a concise summary of the shifting of the defect removal patterns for the AS/400 development. The patterns compared are the 1989 baseline measurement, the

Figure 9 AS/400 defect arrivals during system test time frame

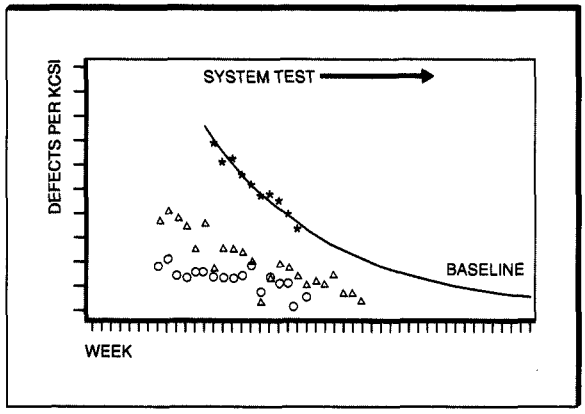
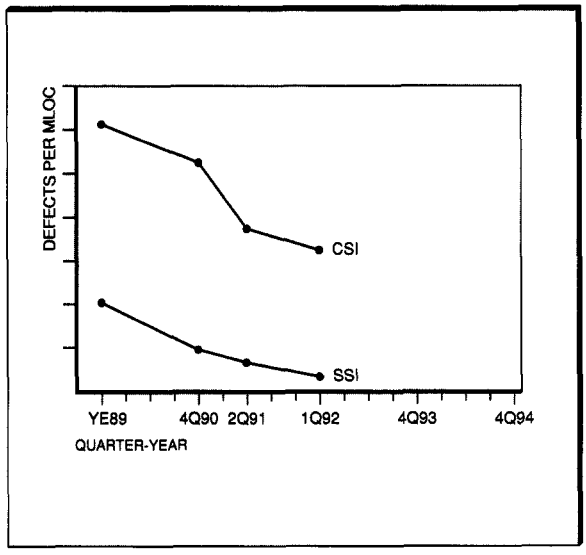


Figure 10 Reductions in AS/400 field defect rate



most recent release shipped, the projected model for the current release (under development), and the current release results through the I2 (code) inspection phase. Compared to the baseline pattern, substantial reduction in error injection has occurred at both the front-end inspection phases (I0, I1, I2) and the back-end formal testing phases (CT, ST). Also, the peak of defect removal has been shifting from the I2 phase toward the I1 phase. The defect removal pattern of the current release (under development) seems to be quite close to the model curve.

Figure 9 shows a more detailed picture of the defect reduction during the last phase of the development cycle: system test. The baseline model was derived with actual 1989 data as the basis. The data are expressed as weekly defect arrival rates per thousand new and changed source instructions (KCSI). Data points denoted by triangles and circles represent two recent releases; significant release-to-release decreases are observed. Before testing is concluded for each new release, the defect arrival rates must stabilize at an extremely low level. And, as mentioned before, the CQP customer burn-in program follows system test to further reduce latent defects.

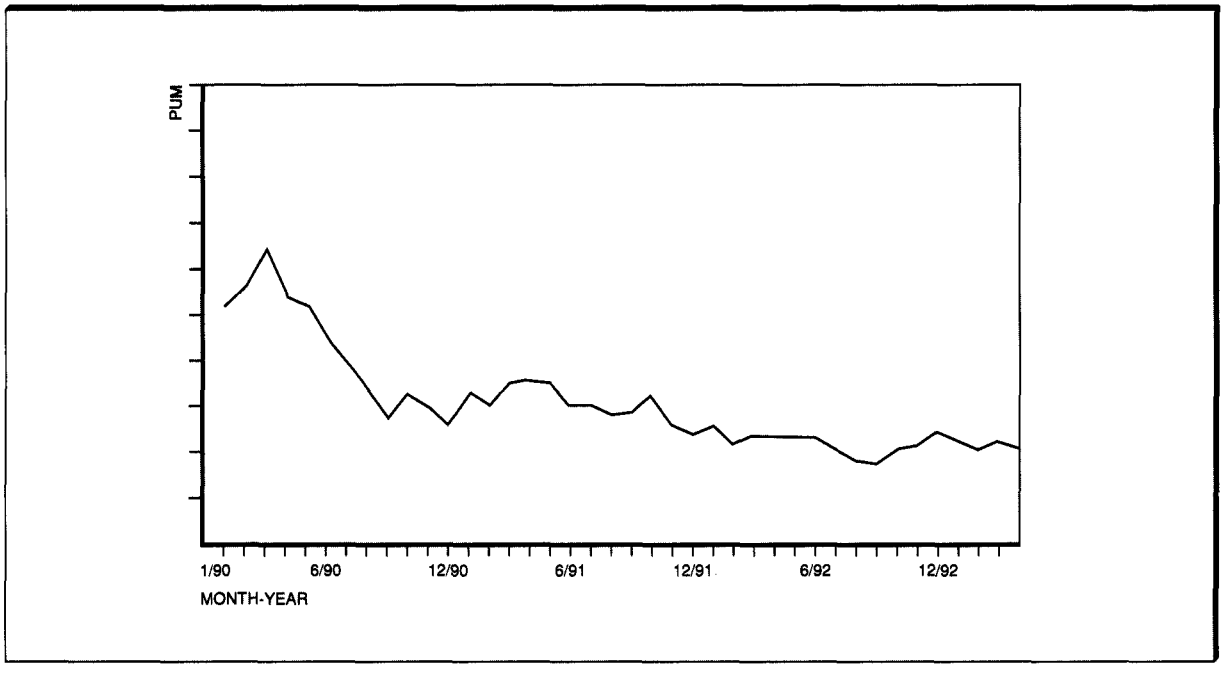
Not surprisingly, the field defect rates of the AS/400 software system have decreased significantly over the past several years. As Figure 10 shows, the CSI defect rate improved by about two times, and the defect rate for the entire system (SSI: total shipped source instructions) improved much more. In the figure, the Y-axis is expressed in terms of defects per million lines of code (MLOC), either CSI or SSI; the X-axis shows the year and quarter in which new AS/400 releases became available to the general market. The defect rates are based on actual defect arrivals after general availability (GA) of the releases and have been normalized to a four-year life-of-product (LOP). Therefore, valid release-to-release comparisons can be made. The SSI defect rates are much lower than the CSI defect rates because SSI defect rates are a function of several variables:

- The size of CSI for each release
- A reduction in CSI defect rates
- Defect reduction due to aging
- Defect reduction through customer burn-in programs

It should be noted that not only the defect rates have declined, the absolute number of product defects has also been declining. In summary, through the attention applied to improving CSI quality and because of accelerated aging, the number of product defects customers experience has decreased substantially from release to release.

Post-GA product quality management. The post-GA product quality management element tracks the quality of the product after it has been shipped. Tracking is done to ensure that AS/400 customers are receiving timely answers and fixes to their product problems.

Figure 11 AS/400 non-defect-oriented problems per user month (PUM)



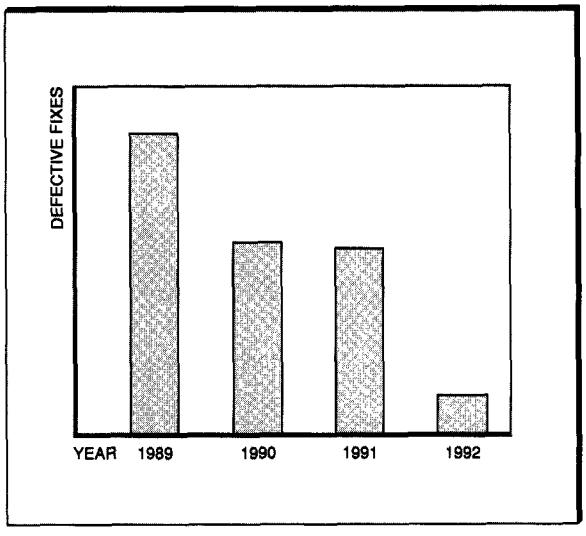
The problems customers encounter when using the software product can be classified into three categories. The first are the functional defects in the product. At IBM, when a customer problem appears to be a defect in the product, an APAR (authorized program analysis report) is written. In the previous subsection (as well as in Figure 10), we have described the reduction in the product defect rates (and defects) for the AS/400 software system for the past several years. Because the defect rate is low, the overwhelming majority of AS/400 customers have never written an APAR. The second category of customer problems, and by far the most common, is the case where the product does not do what was expected, but the solution does not require a change to the product. Those in this category are called non-defect-oriented problems. When these problems occur, IBM provides information to the customers to resolve them. The third category of problems is a defective fix to a customer problem. Defective fixes have a significant negative impact on customer satisfaction. That is why the goal is always zero defective fixes.

Each of these categories of problems is measured weekly and reviewed monthly in order to understand progress toward achieving specific goals for each release, and to be able to take immediate and appropriate action if any of these measurements get out of control.

Figure 11 shows the results of tracking non-defect-oriented problems over a three-year period. Presented is the trend for problems per user month (PUM): the average number of problems that a customer encountered monthly. The PUM rate peaked in early 1990, and soon after started to decline through September 1990. It then fluctuated at a more or less consistent level. From October 1991 to February 1992, another drop was observed but was less significant than the first one.

Although significant improvement in the PUM rate was observed compared to the baseline (early 1990), the bigger challenge is to curb or even reduce the absolute number of problems. We have begun to address this challenge. In cases where many customers call service for the same non-defect-oriented problem, an analysis is done by the development and service teams to determine if a change can be made to the software product or to documentation that would eliminate repet-

Figure 12 AS/400 fix quality improvement – number of defective fixes by year



itive service calls. It is too early to see the results of this effort, but the goal is to reduce the PUM rate further and to curb the growth of the raw number of non-defect-oriented problems while continuing to expand the customer base.

Significant improvement in fix quality has been observed. It is summarized in Figure 12, which shows the year-to-year number of defective fixes. In 1992, the percent of correct fixes exceeded 99.99 percent; the absolute number of defective fixes is close to zero (in single digit). Of the 32 AS/400 licensed program products, 30 had no defective fixes for a year or more. Again, the goal is to achieve zero defective fixes.

Key factors that contributed to the improvement of fix quality include the rigorous practice of DPP, use of the formal inspection process on fixes, and an expert system that was developed internally and used to enhance the fix process. For instance, stage kickoff sessions are conducted weekly so that developers who have an APAR to fix are kept up to date with the fix process and the various ways to prevent defects. When a defective fix is discovered, a proactive approach is taken to determine which customers ordered the fix so that they may be alerted in advance. Experience shows that this customer call-back process with regard to a defective fix has enhanced customer satisfaction.

Continuous process improvement. The continuous process improvement element of the AS/400 software quality management system provides the basic foundation for improving any software development process. The foundation consists of process ownership, definition, documentation, measurements (including baseline measurements), yearly goals, maturity assessments, and corrective actions for improvement. Having all development processes laid across this foundation enables effective and efficient process improvement. The team for development quality and process technology keeps the process foundation strong through ongoing process owner education, establishing documentation consistency, performing yearly maturity assessments, establishing measurement and tracking guidelines, enabling cross-process synergy, and enabling compliance with standards such as ISO 9000.

Process optimization through data and analysis, strong process discipline, process benchmarking, and the introduction of object-oriented technology are the major process improvement activities that have been deployed.

The main AS/400 software development process (modified waterfall process) was analyzed based on objective data, resulting in recommendations for improvement. For instance, we found significantly high defect rates associated with interfaces and small changes. Therefore, the high-level design and review subprocess (I0) and the design change request (DCR) subprocess were modified so that interface issues are resolved early. We recommended that development managers use top-notch developers to fix defects reported from the field (small changes are highly error-prone activities). Based on defect origin data, the highest error injection was found in the code development phase (not surprising since the transition from low-level design to actual code involves an explosion of details that provides numerous chances for injecting defects). Thus, a renewed focus on front-end inspection was initiated.

For significant quality improvement in a large-scale development environment, and for complex system software with numerous interdependencies such as the AS/400, strong process discipline is required. A back-to-the-basics program (de-

scribed later) was initiated to emphasize strong process discipline. The benefits of process discipline have been demonstrated by Japanese computer companies. The formula for success in advancements in development productivity and quality by some Japanese computer manufacturers is a combination of continuous refinement of the classical development methodology, wide-scale use of quality control techniques, and development and use of CASE (computer-aided software engineering) tools.

People are the most important element of the AS/400 software quality management system.

Benchmarking other companies in the industry and other IBM locations was also used to assess and implement various approaches for process improvement. For instance, a benchmark exchange took place in 1991 between IBM Rochester and Hewlett-Packard Commercial Systems (the HP 3000** computer system). By examining the "IBM Rochester way" and the "HP way" in areas such as product development process, release cycle, process and quality management, and human resources, both sites were able to institute improvements. Another example of wide-scale implementation is the defect prevention process (DPP) from IBM Networking Systems in Research Triangle Park, North Carolina.[3] DPP adds intelligence to the process and makes it more iterative. After a successful pilot in 1989, DPP was implemented throughout the entire Rochester software development laboratory. Other approaches evaluated included the Cleanroom methodology,[5] the experience factory,[6] the small team approach, the object-oriented design/object-oriented programming (OOD/OOP) process and technology, and various modeling and metrics approaches.

Continuous process improvement also requires equipping people with the proper tools and technology that lead to quality improvements. Long-term investment for technology transfer (such as powerful workstations and the support for the OOD/OOP process) was necessary. However, the learning curve for new processes and technology is long; continuous incremental tools improvements were needed for the immediate needs when optimizing the current process. DPP and the network of technical process owners were the key identifiers of requirements for tools improvement. Through causal analysis and the assessment by action teams and technical process owners, the cost and benefits of any tools improvement were well understood and prioritized.

To date, powerful workstations have been installed according to plan, several large-scale OOD/OOP projects are in progress, and numerous tools-related DPP items have been implemented to improve the current process. Significant progress was made in developing an environment and a library system for distributed development.

Our continuous process improvement can be summarized as: continually optimize the current process for effectiveness and efficiency; selectively use the small-team approach for flexibility and innovations; use benchmarking to apply both IBM and non-IBM knowledge and experience for process improvement; and continue to move toward the OOD/OOP process with related technology.

People. People are the most important element of the AS/400 software quality management system. The human side of quality and productivity, although less tangible, is more important than process and technology. Given a certain process and technology, it is people who make the difference. A "conscientious programmer" is our key assumption in terms of the people aspect of quality improvement. According to Maslow,[7] it is human nature to strive for self-actualization when basic needs are fulfilled.

This assumption has significant implications. It means that to formulate specific plans for quality and process improvement, the programmer's perspective must be taken into account, especially in terms of feasibility. In fact, many actions in the quality road map originated from the development teams. The assumption also means that process improvement does not always consist of adding more requirements, checklists, and process steps. The optimization effort must continuously

look for ways to simplify the process. A good example is the simplification made to the inspection defect gathering process. Two old inspection defect databases were merged into a single new database that was tightly integrated with the current project management process. At the same time, inspection defect reports were improved.

We must ask what motivates individuals and teams to perform the necessary tasks that will result in excellent product quality. Although the answers may vary, a well-accepted one is that people are motivated by incentives, awards, public recognition, peer respect, and so forth. In fact, having incentives together with public recognition is a key principle in quality management. Public recognition takes many forms and does not always involve monetary awards. Incentive programs could be difficult to formulate and implement. However, it would be remiss if quality road maps and plans fail to include this positive and powerful approach. A good program should be based on measurable product quality outcome and targeted to the teams and groups that directly influence product quality.

Several quality incentive programs, in addition to the regular IBM award programs, have been implemented at IBM Rochester. The AS/400 Division General Manager's MDQ Award is a team award that is designed to promote MDQ initiatives and foster teamwork. It runs three cycles per year, and its scoring guidelines include assessments on approach, deployment, and results. It includes quality contributions in all areas in the AS/400 Division such as planning, engineering, programming, production, market support, site support, and so forth.

In software development, there is a monthly quality award in all functional areas. The award, first presented in 1990, is based on peer nomination. It is well received and widely used by team members to recognize their peers for quality contributions in terms of process, product, and support of the AS/400 software development. It is an individual award. Compared to the MDQ award, this award is less formal and is given out monthly within each third-level area of management in the software community.

The third award, recently announced in a large development area, focuses on development quality. It is specifically targeted for innovative successes that enable the product to achieve excellent quality. It focuses on good development characteristics such as good design, less rework, meeting key dates, low amounts of testing defects, and minimum amounts of field defects. A key feature of this award is that improvements must be measurable. There are two tiers to the award: the in-process award relies on indicators such as driver stability, low numbers of testing defects, and client satisfaction (the other teams that use the candidate's code); the long-term award relies on the actual field defect rate after GA and customer feedback. In addition to the objective criteria, a peer review board also considers factors such as code complexity, usage, previous history of code quality, and relative improvement. The award program is funded from the savings derived from having less maintenance costs due to reductions in product defect rates. Submissions can be self-nominated or nominated by management and can be for teams or individuals.

A quality management system cannot exist without some structure that enables its implementation.

AS/400 SQMS structure, deployment, and measurement

A quality management system cannot exist without some structure that enables its implementation. At IBM Rochester we devised a key structural element called a quality road map that defines the goals the system is to achieve for each release of AS/400 and the key actions that must be deployed to achieve those results. Details of what actions are deployed are documented in a quality plan for each release. Metrics, measurements, and analysis are used to track implementation progress and to guide the improvement effort. The following subsections describe the AS/400 software quality road map, give examples of key actions that were deployed, describe how a detailed quality plan for each release was devel-

Figure 13 AS/400 software quality road map

YEAR	GOAL MSSI	GOAL MCSI	
1991 RELEASE	QI-91A	QI-91B	• DEVELOPMENT LINE ITEM CUT-OFF DATE • IMPROVED DR/CI EFFECTIVENESS – INCREASED COVERAGE – BETTER REVIEWS • PHASE-BASED DEFECT REMOVAL MODEL • MINIBUILDS • INTEGRATION/BUILD DEFECT FEEDBACK LOOP • DOUBLE BYTE CHARACTERS SET TESTING FOCUS • IMPROVED CT – NETWORK OF COMPONENT TEAM LEADERS • EXPERIMENTAL PRE-ST CODE FREEZE • ENFORCED ST ENTRY CRITERIA • 5* COMPONENT ACTIONS • PROTOTYPING PROJECTS – USER-CENTERED DESIGN (UCD) • CQP (CUSTOMER BURN-IN) • CUPRIMDA ACTIONS • DPP LAB-WIDE ROLL-OUT • 3RD-LINE AREAS MONTHLY QUALITY AWARDS – PEER NOMINATIONS • DEPARTMENTAL 5-UP
1992 RELEASE	QI-92A	QI-92B	• RELEASE KICK-OFF SESSIONS • IMPROVED FRONT-END CHANGE CONTROL PROCESS • SIMPLIFIED DR/CI AND UT DEFECT DATA COLLECTION • IN-PROCESS MEASUREMENTS • ST – RAISE TESTING • PRE-ST CODE FREEZE (DEVELOPMENT COMPLETE) • COMPILER IMPROVEMENTS • DPP LAB-WIDE IMPLEMENTATION • NON-DEFECT-ORIENTED CUSTOMERS' PROBLEMS CLOSED LOOP PROCESS • SPECIAL QUALITY LINE ITEMS IN DEVELOPMENT PLAN
1993 RELEASE	QI-93A	QI-93B	• INTEGRATION ARCHITECT • BUILD/INTEGRATION QUALITY IMPROVEMENT • TEST COVERAGE FOCUS • OPTIMIZATION OF FIELD TEST PARTNERSHIP (FTP) AND CQP • WORKSTATIONS AND DISTRIBUTED ENVIRONMENT • OBJECT-BASED DEVELOPMENT PROJECTS
1994 RELEASE	QI-94A	QI-94B	• CUPRIMDA TEST DRIVE REQUIREMENTS SPECIFICATIONS VALIDATION • PRE-I0 PROCESS – SYSTEM ARCHITECTURE INSPECTION (SAI) – COMPONENT ARCHITECTURE INSPECTION (CAI) • OOD/OOP DEVELOPMENT PROJECTS • C++ • IMPLEMENT REUSE STRATEGY • FORMAL TOOL-BASED DEPENDENCY MANAGEMENT • DEVELOPMENT QUALITY INCENTIVES PROGRAM

oped, discuss the deployment of key quality practices in the software development community, and describe the tracking, measurement, and analysis in the AS/400 SQMS.

Quality road map. A quality road map describes the various quality technologies and actions that will be used over a long period of time to leverage improvements to overall software product quality. In the road map the quality goals are specified by each year and each release of AS/400, along with the actions to achieve the goals. Detailed descriptions of these actions are documented in the quality plan for each release. A simplified version of the AS/400 software quality road map is shown in Figure 13. Symbols such as QI-91a and QI-91b represent the specific system quality goals for CSI and SSI (shipped source instructions) for each release. The road map has been revised several times since the initial MDQ deployment in early

1990. It is updated continuously on the basis of accumulated experiences and knowledge, making it a living document. The action items in the road map cover all elements of the AS/400 software quality management system. Its purpose is to achieve defect elimination and customer satisfaction goals. For each product release, the action list is cumulative. For instance, actions implemented for the 1992 release include those listed under both the first and second panels. For the 1993 and 1994 releases, the items are subject to change since the releases are under development. We continuously assess the effectiveness and feasibility of each action.

It should be noted that a number of actions on the quality road map were originated by development teams. An example is the use of minibuilds by the database development team. A minibuild is a process that creates an isolated test environment for testing new function. This test environment gives developers an opportunity to remove defects prior to integrating function with the remaining parts of the system. This improvement is significant because integration and build are on the critical path of the development process and driver problems can cause schedule delays and significant quality and productivity loss (especially drivers for external development locations). From this positive experience, many areas are now taking advantage of this approach. The build and integration team took this approach one step further and established the build/integration defect feedback (to development teams) closed-loop process, so as to reduce the chance of similar problems recurring in the future.

Another instance is the adoption of test coverage measurement and the development of a new test coverage measurement tool by a developer. These examples attest to the importance of the people aspect of software development and to a transformation taking place in a quality-oriented culture.

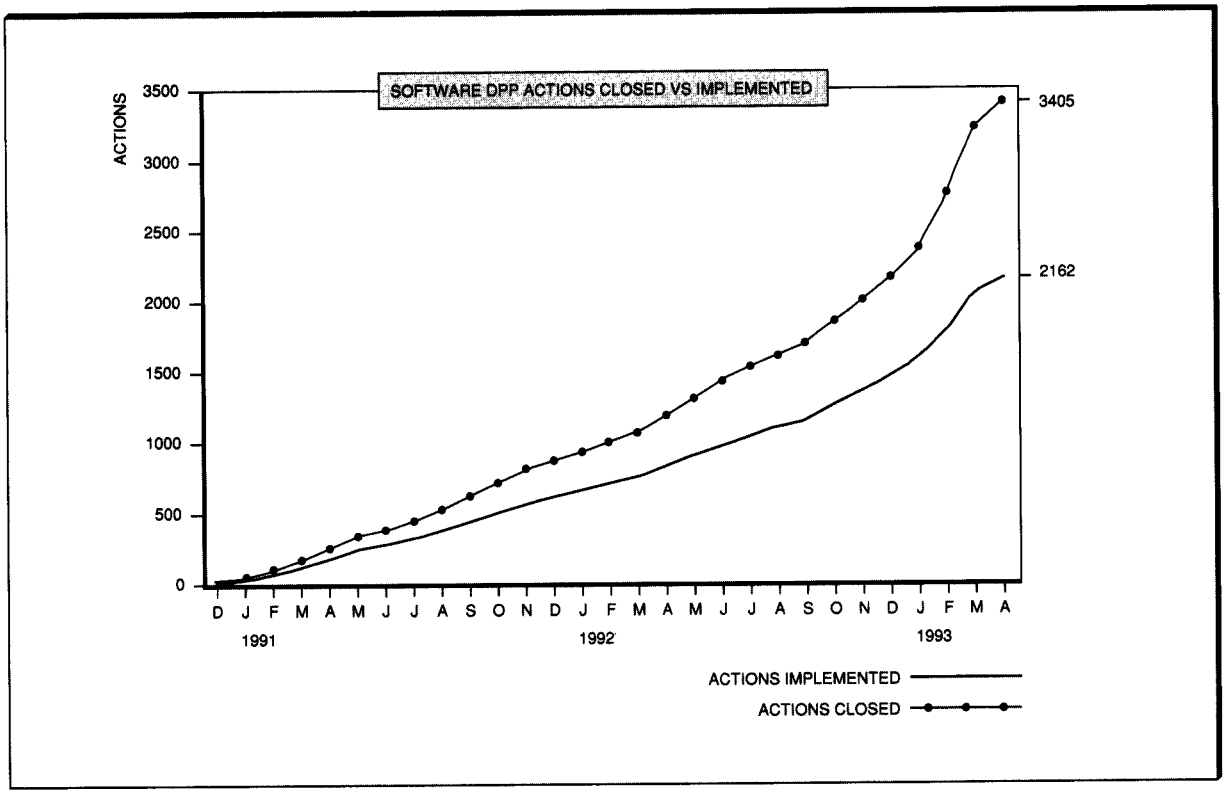
It is worth noting that the position of release integration architect was established recently as a result of a benchmarking exchange with the HP 3000 system and a continuous focus on integration quality. The integration architect oversees the interdependent nature of the release line items and serves as the gatekeeper for integration and build quality. This architect develops detailed code integration plans and schedules; defines and manages code integration procedures; and ensures that cross-product and cross-component dependencies are met.

Examples of key quality road map actions. Described below in more detail are some examples of quality actions that were deployed.

Defect prevention process. The merits of DPP were first recognized at an IBM Software Engineering Interdivisional Technical Liaison (ITL) conference. A pilot program was started shortly afterwards in mid-1989. With positive experience from the pilot and with the developers' enthusiasm, DPP was positioned as a strategic action in the MDQ deployment in early 1990. Robert Mays of IBM Research Triangle Park, co-founder of DPP, was invited to Rochester to give two special seminars, to which the responses were overwhelming. One session was videotaped, and eventually the entire programming community saw it. A mind-set change to a prevention-oriented focus was under way. With strong management commitment, a deployment team was formed that developed and delivered DPP seminars and formulated a long-term strategy of self-sufficiency of DPP education. Rochester developers were trained to deliver the formal education. An action team structure and a network of technical process owners were established. An action tracking tool was also developed. To date, most programmers and managers have been formally trained in the DPP process. Figure 14 shows the number of DPP actions that have been implemented and closed through the first quarter of 1993. The graph indicates that more than 3400 suggested actions were closed, of which more than 2000 have been implemented since the creation of the DPP program. Many of the proposed actions are extensive in the amount of effort required to implement them; all cannot be implemented because of resource limitations.

The scope of the DPP actions that were implemented varies. Many are small items, such as adding entries to the common error lists and to stage kickoff meetings. Other items, such as the post-compiler module checker and the release kickoff sessions, are fairly significant. Regardless of the scope, all items are pertinent to the improvement of the development process and to quality. The following example perhaps is better in giving a flavor of the DPP actions. From statistical analysis at the system level as well as from the causal analysis sessions of development

Figure 14 DPP actions implemented



teams, defects related to interface problems needed to be dealt with. Interface defects constitute a large percentage of the total defects during all phases of development and from the field (APARs). Interface problems are to a large extent human communications problems that are preventable. One action proposed and implemented was to add a new command in the AS/400 development support environment that would allow developers to subscribe to the modules their code depends upon. If the modules change, the tool automatically notifies the subscribers. Therefore, developers are better able to manage dependencies and reduce interface defects.

There are several key characteristics of Rochester's DPP deployment: strong buy-in from both management and nonmanagement, developers as DPP instructors (after training), and the integration of DPP and process improvement (action teams and the network of technical process owners).

"Back to the basics" design review/code inspection focus. The "back to the basics" focus refers to executing the design review/code inspection (DR/CI) process in a much better way. Benchmarking analysis of the IBM Houston Space Shuttle Onboard Software system project indicated that rigorous implementation of the process can make an order of magnitude difference in development quality. Furthermore, causal analysis of APAR data indicated that often the bugs could have been caught early in the development cycle. Analysis of defect origin data indicated that the code development phase had the highest defect injection. Hence, a renewed focus on the front-end reviews and inspections was initiated, and a series of actions related to DR/CI was then undertaken.

This strong DR/CI focus led to significantly higher inspection coverage and better execution. The result is that the front-end execution becomes much

better and fewer defects escape to the testing phases, as described in earlier sections.

It is worth noting that software reviews and inspections are distinctly different from manufacturing inspections, which are at the back end of the production process and are known to be a poor method for quality assurance. Quality improvement teachings often call for the abandonment of manufacturing inspections in favor of acceptance sampling (with the front-end focus on design quality). Software reviews and inspections, on the contrary, are the vital techniques at the front end of the software development process.

Furthermore, software design reviews and code inspections are more efficient in defect removal, as Fagan's study[8] showed. In our analysis, the ratio of the cost of finding and fixing a defect during design, test, and field use is: 1 to 13 to 92. This ratio, interestingly, is very similar to the ratio reported by the IBM Santa Teresa laboratory some years ago: 1 to 20 to 82.[9]

Customer Quality Partnership Program. The customer quality partnership (CQP) program was designed to accelerate the normal field aging process to achieve further reductions in defects before GA. It was based on the concept of accelerated aging in quality engineering and the results of the analysis of the AS/400 customers in terms of their defect discovery patterns. For AS/400, the dominant majority of APARs are reported by a very small percentage of customers. These customers and those who will exercise the new release functions are, hence, good candidates for customer burn-in programs. The CQP program works as follows:

- Motivate a group of high APAR-writing customers and customers who use specific new release functions to move their production to the new release as soon as possible. Defect discovery is thus accelerated. The program starts at the end of the development cycle after code freeze (when code quality has reached a satisfactory level) but before general availability.
- Provide a streamlined process and additional technical support to facilitate the installation of the new release and for problem-solving. The bottom line is to minimize customers' business risk and to enhance their satisfaction.
- Fix the defects as quickly as possible with unsurpassed quality focus.
- When sufficient defects are detected and fixed, integrate the fixes into the system library so that the defects are eliminated before GA to AS/400 customers.

The AS/400 CQP program was conceived and proposed at the end of 1990. With strong management support and intensive planning and preparation, the program moved into implementation in early 1991. Now it is being implemented for each new AS/400 release. Defects found by the program are subject to DPP causal analysis, and findings are used to improve the development process. For each implementation, a postmortem analysis is done to improve the next implementation. At the end of each implementation, customers are surveyed. Their responses have been very positive.

5☆ component quality analysis. This analysis was done to identify error-prone components in the system. A component in the AS/400 software system is a group of program modules that perform specific functions, for example, workstation manager, print function, spooling, storage management, message-handling, and so forth. The analysis has been done for each release of the AS/400 software system since 1989. A composite index is formed based on CSI defect rate, SSI defect rate, and the raw number of APARs (customers do not care about the normalized rate). The composite index gives a more balanced view of component quality than any of the individual indicators. Components are ranked based on the composite index from 0☆ to 9☆ (9☆ being the most error-prone). Components obtaining 5☆ or more are required to take special improvement actions. Over the past several years, the 5☆ component analysis has become a key driving force for code quality improvement. Components have taken various approaches for improvement: involve customers in early requirements and design phases, place a rigorous focus on front-end design and reviews, conduct causal analysis on defects, invite external experts for early testing during development, test coverage measurement, initiate selective module break-up for error-prone modules, conduct special customer testing via the Field Test Partnership program, recommend their high-APAR customers to join the CQP program, and so forth.

Quality plan. At the beginning of each release cycle, a system quality plan is established in which the committed quality goals are defined and improvement actions to achieve the goals are described. To make the quality plan effective, a bottom-up commitment approach is used. The development quality and process technology team first develops a proposal. After a series of brainstorming sessions with groups of developers, the proposal is revised. The proposal is then reviewed with upper management in the software laboratory. The agreed-to proposal becomes the basis of the bottom-up commitment process to be conducted in each development area from the component level upward. The committed quality goals and actions from all areas, together with system-level actions, form the system quality plan. The final written document, after review and approval by management, is then made available to the entire software development community.

The system quality plan represents the overall system approach to quality improvement.

Note that the system quality plan represents the overall system approach to quality improvement. Quality improvement is everyone's responsibility. It is made clear that each component, department, area, and product manager owns its quality activities, including planning, implementation, and outcome. Indeed, each department and area in the software development community has established its quality improvement plan. For each department, the committed quality goals and actions are described in the quality control book (QCB) of the department. This on-line QCB, together with other on-line repositories in which the process documents and system quality plans reside, are available to all members at IBM Rochester via IBM Rochester's Quality Management System (RQMS) tool. This tool basically provides a single user interface to important quality management system information, which is also essential for meeting ISO 9000 certification requirements.

Moreover, to make efforts in quality improvement a part of the overall development effort, quality investment is also included in the development plan process. Several plan line items have been used to address the resources needed for the deployment of several major quality practices. Good examples of quality line items include the funding of the formal action teams of DPP, 5☆ component improvement, and the work and resources needed to improve the products so as to reduce the problems customers encounter, even though such problems may not be defect-related (non-defect-oriented problems).

Deployment. Thus far, quality road maps and quality plans have been presented. We now discuss the approach to implementing key quality practices in the software development community. Note that for quality improvement to happen, bottom-up commitment is the most important factor. Indeed, the improvement of software quality for AS/400, discussed in earlier sections, is due to the efforts by everyone on the entire software team. Many improvement actions were also self-initiated by development teams. In the following discussion on deployment of pervasive quality practices, the importance of individual commitment should not be forgotten.

Management commitment is equally important. According to Townsend and Gebhardt,[10] upper management participation is a prerequisite to practicing total quality management (TQM). Without management participation, the organization practices quality by proclamation instead of TQM. In Rochester, active management participation is one of the key factors that makes the quality management system work and the deployment of actions successful. Each month the AS/400 Division General Manager reviews the overall quality of the AS/400. Separately, the Director of Development reviews the progress and status of development quality. For software development, there is also a monthly quality meeting in which the directors and third-line managers review status and make operational decisions.

Perhaps the strong top-down management commitment to quality in Rochester can best be illustrated by the establishment of the Rochester De-

velopment Quality Council by the Director of Development. Its focus is on the use of the market-driven quality techniques and practices in development programs. Several times a year high-leverage development programs are reviewed by Rochester senior management, technical professionals, and development managers. The review meetings provide a communications link between the various program teams and facilitate the exchange of good quality practices across development. The scope of influence of this program covers the entire AS/400 development laboratory, not just the software community.

Bottom-up and top-down commitment together ensure success. In deploying key quality practices to the entire software community, this two-way approach is used as much as possible. For instance, the successful laboratory-wide implementation of DPP clearly is a result of strong commitment and effort from both management and all members of the programming community along with the mind-set change of all. On some occasions, either the bottom-up or the top-down approach has been observed to play a more significant role than the other. For instance, the minibuild approach is clearly a developer's own initiative. The customer quality partnership program (CQP) represents a well-planned approach with strong management support and resource commitment.

Deployment model. Deployment of quality practices differs between large organizations and small groups. In small groups new practices can be experimented with anytime. Large organizations, however, cannot afford broad experiments. A regular deployment pattern has been observed from Rochester's experience with deploying DPP, DR/CI focus, CQP, and other quality practices. From this experience the following deployment model was developed:

1. Given a new practice or innovative process, and after study and evaluation, the first step is to develop an informal proposal for small-scale implementation and to obtain management's initial approval. This step can be done by anyone.
2. The most important step is to obtain the commitment of developers and to start a small-scale pilot project. If the pilot is voluntary and does not require management funding, the chance for success is better. It is vital to have the developers' commitment because without it, the chance for future success is minimal.
3. Upon completion of the pilot project(s), analysis is conducted in terms of empirical data as well as causal analysis sessions.
   (a) If the results are overwhelmingly positive, proceed to the next step.
   (b) If the results are moderate, or the buy-in is not enthusiastic, refine the process or practice and its implementation procedures and loop back to step 2.
   (c) If the results are neutral or negative and the pilot members' buy-in is lukewarm, pursue another approach.
4. A formal proposal is prepared. It is presented to management whose formal commitment is sought, especially if resources are required.
5. Upon formal management approval, a detailed implementation plan is prepared for the entire organization, and the implementation is begun. Aggressive schedules with highly intensive effort have the best chance of success. A comfortable deployment schedule may run the risk of losing momentum.

Supplier quality requirements. Because the development of AS/400 involves many other IBM locations and suppliers, the quality of products developed by other sites and non-IBM suppliers is reviewed. Their product quality must not adversely affect the AS/400 system quality. For each release, specific goals are set. All suppliers are required to meet these goals. In terms of outcome, their product quality must be as good or better than the quality of Rochester-developed products. In this respect, the Rochester team has the challenge to demonstrate its ability to achieve the aggressive quality goals.

Supplier quality requirements are documented in the product management plan (PMP) for each supplier or external-site product. The PMP is part of the contractual agreement between IBM Rochester and its suppliers.

For supplier products, quality plans are required by the release commit checkpoint. Quality assessment and certification must be conducted for each product. To be consistent with Rochester development, in-process measurements are required. Suppliers that use the IBM Rochester development process must provide the same indi-

cators that IBM Rochester development uses. For those that have their own development process, whatever data that are indicative of in-process quality must be provided to Rochester. This requirement reduces the risk of project failure.

IBM Rochester provides the AS/400 system quality plan to all suppliers and IBM sites that develop AS/400 licensed products and provides input to their product quality plans when needed. To the extent possible, Rochester also provides consultation to other IBM sites and suppliers in the areas of quality improvement actions, metrics, defect models, and quality projections.

The product quality goal of defect rate, expressed in terms of numbers of defects per thousand source lines of instructions, is a life-of-product requirement. Therefore, before a supplier product is accepted by Rochester, the quality level must meet a certain defect rate criterion that indicates that the life-of-product quality goal is likely to be met. A defect rate criterion is established for an acceptance test (just prior to system test) that is based on the system defect removal model. This methodology is consistent with the literature and actual AS/400 experience. By acceptance test time, product development is complete. The defect rate detected during this period is a good indicator of the projected field quality.

Myers[11] has a counter-intuitive principle in software testing that basically states that the more defects found during formal testing, the more will be found later. For AS/400, actual data confirmed the positive relationship between formal machine test defect rate and field defect rate. The reason for the relationship is that at the late stage of formal testing, error injection in the product is already formed. Greater amounts of testing defects are an indicator of high error injection in the product. An analogy can be drawn between the total defect rate since formal test (formal test defects and field defects) and an iceberg: the tip of the iceberg being the formal test defects and the submerged part the field defects. Therefore, high defect rate during acceptance test means more defects will escape to the field unless extraordinary actions (extra testing, customer burn-in, and so forth) are implemented.

Tracking, measurement, and analysis. Tracking, measurement, and analysis together form an important element of the structure of the AS/400 SQMS.[12] We believe that software development must move in the direction of quantitative approach to become a true engineering discipline and to be more efficient and effective. Such an approach is especially important for complex large development projects with numerous interdependencies such as the AS/400 software system. It is important to note that the data collected, and the metrics used and analyzed, need not be overwhelming; it is more important to have the information from these activities focused, accurate, and useful. Such information enables data-based decision-making for project and quality management.

Quality management measurements and tracking are an integral part of IBM Rochester's business checkpoint process.

Quality management measurements and tracking are an integral part of IBM Rochester's business checkpoint process when developing a new product or enhancing an existing product. The four major business checkpoints that our products must achieve prior to shipping to customers are: commit, announce readiness, announcement, and general availability. Each checkpoint requires a specific quality activity to be completed. For example, a quality plan is required for the release commit checkpoint. An interim quality assessment is required at the midrelease announcement checkpoint. At the general availability checkpoint we conduct the final quality assessment and projection. By merging quality management with the business checkpoint process, quality becomes a key factor in business decision-making.

In addition to the in-process metrics that are used for quality management during the development process, post-GA metrics and customer satisfaction survey results are examined to assess product quality in the field. In-process and post-GA

Figure 15 Inspection effort and defect report

SUMMARY INSPECTION REPORT – DEFECT RATE AND INSPECTION EFFORT

*** AREA/DEPT A — RELEASE N

Insp Type	# Insps	Insp LOCs	DCR LOCs	Defs	Prep Hours	Insp Hours	Total Hours	Rwrk Hours	# AT
I0	57	32190	38150	363	492.9	416.5	909.4	487.3	412
I1	52	25045	38150	430	328.9	407.5	736.4	357.7	271
I2	117	35099	38150	597	647.8	611.7	1259.5	410.8	490

*** AREA/DEPT A — RELEASE N

Insp Type	%Insp CVG	Defs /Kloc	Sys Model	PrepHr /Kloc	InspHr /Kloc	TotHrs /Kloc	Sys Std	RwrkHr /Kloc	#AT /Insp
I0	84.4	11.3	7.0	15.3	12.9	28.3	19.6	15.1	7.2
I1	65.6	17.2	15.0	13.1	16.3	29.4	22.8	14.3	5.2
I2	92.0	17.0	13.0	18.5	17.4	35.9	26.0	11.7	4.2

measurements and customer satisfaction survey results are reviewed monthly by executives.

In-process measurements. In-process quality measurements enable one to implement real-time quality management. To achieve this task we use a defect tracking system for the entire development cycle. It is integrated with the change control process of the AS/400 software development process. Metrics, such as the phase-based defect removal pattern, phase effectiveness,[13] inspection effort and coverage, in-process escape rate, percentage of interface defects, integration/build defect arrivals, number of unit test defects found before and after code integration, testing defect arrivals and defect rates by phase, severity distribution of defects, late performance changes, and so forth are used and interpreted in the context of an overall defect removal model.[14,15] These metrics, combined with the standard project status metrics (for example, the cumulative curves of actual versus plan for various phase activities such as the completion of design reviews and the execution of test cases), provide a sound basis for informed decision-making with regard to project and quality management.

We have been using in-process metrics at the system and product levels for AS/400 since the development of its first release. For each release we continue to refine our metrics and improve our analysis. Currently we are deploying the use of in-process metrics to more granular levels: components and first-line departments. The major vehicle being used is a set of standard reports under a common report interface.

Figure 15 and Figure 16 show two examples of in-process metrics reports. In Figure 15 the number of inspections completed so far by stage (I0—high-level design inspection, I1—low-level design inspection, I2—code inspection) are shown. Information on actual lines of code inspected (inspection LOCs), total lines of code in the current plan for the department (DCR LOC), number of defects found (Defs), inspection effort in terms of preparation hours (Prep Hours) and actual inspection hours (Insp Hours), rework hours (Rwrk Hours), and the number of attendees at inspection meetings (# AT) are presented in the upper panel. Shown in the lower panel are the normalized metrics such as the percentage of inspection coverage (%Insp CVG), defects per KLOC (1000 lines of code), total inspection hours per KLOC (TotHrs/Kloc), etc. The system model in terms of inspection defect rates (Sys Model) and inspection effort (Sys Std) is also presented for comparison.

In Figure 16, the defect rate is classified in terms of defect origin (RQ—requirements, SD—system defects, I0—high-level design, I1—low-level design, I2—code development) and defect type (LO—logic, IF—interface, DO—documentation). Classification by defect origin allows us to calculate and monitor in-process escape rate and the percentage of interface defects. Since specific targets are set for escape rates and interface defect reduction, the reports also indicate target excep-

Figure 16 Inspection defect origin report

SUMMARY INSPECTION REPORT– DEFECT ORIGIN AND DEFECT TYPE

*** AREA/DEPT A RELEASE N

Insp Type	# Insp	Defs	Defect Origin RQ	SD	I0	I1	I2	% Distribution RQ	SD	I0	I1	I2	Total
I0	57	363	24	13	326	0	0	6.6	3.6	89.8			100%
I1	52	430	3	9	56	362	0	0.7	2.1	13.0*	84.2		100%
I2	117	597	3	0	19	34	541	0.5	0.0	3.2	5.7	90.6	100%

Insp Type	# Insp	Insp LOCs	Defs	Defect Type LO	IF	DO	Defect Type (%) %LO	%IF	%DO	Total
I0	57	32190	363	59	76	228	16.3	20.9	62.8	100%
I1	52	25045	430	263	61	106	61.2	14.2 @	24.7	100%
I2	117	35099	597	361	57	179	60.5	9.5	30.0	100%

*** IN-PROCESS ESCAPE RATE SYSTEM TARGET
I1 DEFECTS : <= 5% ARE ESCAPES FROM HLD (I0)
I2 DEFECTS : <= 2% ARE ESCAPES FROM HLD (I0)
I2 DEFECTS : <= 6% ARE ESCAPES FROM LLD (I1)

*** INTERFACE DEFECT REDUCTION
I0 : INTERFACE ISSUES FINALIZED AT I0 EXIT
I1 & I2 : GOAL IS TO REDUCE INTERFACE DEFECTS TO <= 5% OF TOTAL DEFECTS

(*) : EXCEEDS SYSTEM TARGET SIGNIFICANTLY (2x+). CONSIDER RE-INSPECTION AND OTHER ACTIONS.

(@) : EXCEEDS SYSTEM TARGET SIGNIFICANTLY (2x+). CONSIDER RE-INSPECTION AND OTHER ACTIONS. MAKE SURE INTERFACE ISSUES ARE FINALIZED AT I0 EXIT.

NOTE: WHEN INTERPRETING * AND @, BE CAREFUL WITH SMALL NUMBERS.

tions, such as those shown in Figure 16 (indicated by the * and @ symbols).

These in-process metric reports are available at the component and product level, and for various organizational units. It should be noted that in-process metrics cannot be used in a piecemeal fashion. An integrated approach must be used in order to yield useful information; the metrics must be interpreted within the context of the defect models used (described earlier and in Kan[14]), and compared with one's history. For example, the single metric of inspection defect rate does not tell much about the quality of the front-end process. However, when combined with the inspection-effort metric and compared with the targets of a defect model (or one's inspection effort/defect rate in the previous release), we can form a 2 × 2 inspection effort/defect rate matrix. If inspection effort increased and defect rate decreased, that is the best case scenario (indicating low error injection and sufficient effort on the front end). If effort decreased but defect rate increased, that is the worst case scenario. If there is both low effort and low defect rate, that is an unsure situation. The scenario of significantly higher effort and higher defect rate, from our experience, is a good, not bad, scenario.

As another example, one of the exit criteria of high-level design inspection (I0) is that all interface issues are finalized. Therefore, it is not surprising to see a high percentage of interface problems among I0 defects. However, if the percentage of interface defects remains high (compared with the model targets or with one's history) at subsequent inspection phases, it indicates

that interface issues can be a problem and should receive greater attention.

Post-GA measurements. Post-GA measurements are used to examine the quality of software products (that have been shipped to customers) as compared to product quality goals. An analysis of the post-GA measurements results in actions to improve the overall software development process so that the next software release will be better than the previous release. As discussed earlier, the 5☆ component analysis has been a key driving force for the improvement of component quality during the past several years. Another form of analysis that is performed regularly is the root cause analysis of APARs.

The key post-GA measurements that are tracked monthly are: product defects (APARs) and defect rates by release, number of defective fixes, total number of problems reported by customers, and average problem fix time. Other indicators that are tracked include number of APARs by severity, number of valid versus invalid APARs, number of APARs that are delinquent in providing a fix, and the number of non-defect-oriented problems. Each metric serves specific purposes. The number of defects and problems, and defect or problem rates for that matter, indicate the quality of the product that is being used by customers. Average problem fix time, number of delinquent APARs, and number of defective fixes measure the efficiency and quality of the fix process. The ratio of valid versus invalid APARs reflects the effectiveness and efficiency of problem determination and the skill level of the service team.

Customer satisfaction measurements. The following data are used to gauge customer satisfaction of the AS/400 and provide input to the product improvement plans:

- IBM Marketing and Service (M&S) survey data and questions, as discussed earlier
- AS/400 customer feedback survey data, questions, and comments sorted by CUPRIMDA category
- Customer critical situation data sorted by category
- Customer partnership call comments sorted by category (a 90-day-after-install customer contact call)
- Customer problems sorted by category

This information is presented regularly to software development executives. Like the in-process and post-GA metrics, analysis is an inseparable part of the tracking and reporting system. Our experience indicated that good analysis is paramount in transforming data into useful information and knowledge. For example, as shown earlier, survey data indicated that among the CUPRIMDA categories, documentation (D) has the highest level of customer nonsatisfaction. However, these data do not mean that improving documentation is the first priority in order to improve overall customer satisfaction. To answer such questions, one needs to examine the correlation between CUPRIMDA categories and overall satisfaction, and customers' purchase decisions. Interestingly, from in-depth analysis based on advanced statistical techniques, we found that reliability is the most significant factor affecting overall customer satisfaction with regard to software quality, and reliability is one of the parameters with the highest customer satisfaction. After further analysis, this finding is not surprising, given the mission-critical applications that are run on the AS/400 system by customers. We certainly need to continue to strive for higher levels of quality to achieve total satisfaction in all CUPRIMDA categories.

Summary

We have described the various elements of the AS/400 software quality management system: customer satisfaction management, product quality management (in-process and post-general availability), continuous process improvement, and people. The structure of the quality management system: quality road maps, quality plans, deployment, supplier quality requirements, tracking, measurement, and analysis were explained. Examination of real data indicates we have achieved substantial quality improvement.

It is absolutely essential to establish goals and to use those goals to drive continuous improvement. Establishing a quality plan makes sure that goals are documented and are used to drive change and innovation in the development of each new release.

Continuous improvement of process and tools requires an ongoing focus and a closed feedback loop so that changes are made as a result of past mistakes. It is important that the focus is placed

both on defect prevention and on defect detection. The advantage of prevention and early defect removal is very clear.

Once a quality plan has been established by the development team, it becomes very important that quality be measured as the release is being developed. In-process quality management is the key to being able to recognize quality problems early, in enough time to take actions before the product is made available to customers.

After a product is made generally available, it remains essential that measurements and analyses continue so that problem areas are identified early and that a closed-loop system be used to feed back those problems in order to prevent them in the future.

All of the above requires a dedicated team—a team that is committed to making improvements happen and delighting every customer. A dedicated team makes the quality management system function.

With a systematic approach, we continue to refine the AS/400 software quality management system based on feedback and learning through measurement and analysis. With the total participation of the entire team, and based on process, technology, and measurements and analyses, we continually strive for further improvement in the quality of AS/400 and in customer satisfaction.

Acknowledgment

The authors wish to thank the entire AS/400 software team. The authors are merely catalysts. It is the dedication and overwhelming commitment by the AS/400 software team that is responsible for AS/400 software quality improvement.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Hewlett-Packard Corporation.

Cited references

1. C. Billings, J. Clifton, B. Kolkhorst, E. Lee, and W. B. Wingert, "Journey to a Mature Software Process," *IBM Systems Journal* **33**, No. 1, 46–61 (1994, this issue).
2. C. L. Jones, "A Process-Integrated Approach to Defect Prevention," *IBM Systems Journal* **24**, No. 2, 150–167 (1985).
3. R. G. Mays, C. L. Jones, G. J. Holloway, and D. P. Studinski, "Experiences with Defect Prevention," *IBM Systems Journal* **29**, No. 1, 4–32 (1990).
4. M. J. Harry and J. R. Lawson, *Six Sigma Producibility Analysis and Process Characterization*, Addison-Wesley Publishing Co., Reading, MA (1992).
5. H. D. Mills, M. Dyer, and R. C. Linger, "Cleanroom Software Engineering," *IEEE Software* **4**, No. 5, 19–25 (September 1987).
6. V. R. Basili, G. Caldiera, F. McCarry, R. Pajersky, G. Page, and S. Waligora, "The Software Engineering Laboratory: An Operational Software Experience Factory," *International Conference on Software Engineering* (May 1992), pp. 370–381.
7. A. H. Maslow, *Motivation and Personality*, 2nd edition, Harper & Row Publishers, New York (1970).
8. M. E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal* **15**, No. 3, 182–211 (1976).
9. H. Remus, "Integrated Software Validation in the View of Inspections Review," *Proceedings of the Symposium on Software Validation*, Darmstadt, Germany, North Holland, Amsterdam, The Netherlands (1983), pp. 341–350.
10. P. L. Townsend and J. E. Gebhardt, *Commit to Quality*, John Wiley & Sons, Inc., New York (1990).
11. G. J. Myers, *The Art of Software Testing*, John Wiley & Sons, Inc., New York (1979).
12. S. H. Kan, R. J. Lindner, and R. J. Hedger, "In-Process Metrics of the AS/400 Software Development Process," presented at the *38th Annual ASQC Minnesota Quality Workshops and Conference*, Minneapolis, MN (March 5–6, 1991).
13. S. H. Kan, "Determining the Phase Effectiveness of the AS/400 Software Development Process," presented at the *First International Conference on Applications on Software Measurement,* sponsored by the American Society for Quality Control and Software Quality Engineering, Inc., San Diego (November 1990).
14. S. H. Kan, "Modeling and Software Development Quality," *IBM Systems Journal* **30**, No. 3, 351–362 (1991).
15. S. H. Kan, "Software Quality Engineering Models," *Encyclopedia of Computer Science and Technology*, A. Kent and J. G. Williams, Editors (forthcoming, Marcel Dekker, Inc., 1994).

Accepted for publication September 16, 1993.

Stephen H. Kan *IBM AS/400 Division, Highway 52 and NW 37th Street, Rochester, Minnesota 55901.* Dr. Kan is an advisory programmer in IBM Rochester's Development Quality and Process Technology department. He received B.S. degrees in sociology and computer science, M.S. degrees in statistics and sociology, and a Ph.D. in demography from Utah State University. He joined IBM in 1987. Prior to that, he was a programmer, statistician, and research scientist for eight years in academia and industry. He is a Certified Quality Engineer by the American Society for Quality Control. In his current assignment, his interests focus on software quality strategy and software quality plans, supplier quality requirements, defect removal models, and software quality statistical analysis.

Samuel D. Dull *IBM AS/400 Division, Highway 52 and NW 37th Street, Rochester, Minnesota 55901.* Mr. Dull is a senior programmer in the Development Quality and Process Technology department, IBM Rochester, responsible for the AS/400 software quality management system. He received a B.S. in electrical engineering from the Pennsylvania State University. He joined IBM in 1970 in Poughkeepsie, New York, and was assigned responsibilities related to System/370 Models 158 and 168 hardware performance measurements on different operating system platforms. In 1976, he transferred to Rochester and has held various technical and managerial assignments on System/34, System/36, and AS/400 software development and support. He has also managed internal development tool projects and, prior to his current assignment, was the release manager for AS/400 Version 2 Release 1. He is currently concentrating on total quality management for software development.

David N. Amundson *IBM AS/400 Division, Highway 52 and NW 37th Street, Rochester, Minnesota 55901.* Mr. Amundson is a manager in IBM Rochester development software. He has an M.S. degree in computer science from the University of Minnesota and a bachelor's degree from the University of Wisconsin–Eau Claire. He began his IBM career as a development programmer and has held first- and second-line management positions in systems assurance. During 1983 and 1984, he was on the staff of the Director of Quality and Assurance in the System Products Division headquarters. Since returning to Rochester, he has been a first- and second-line manager in systems support, development quality, and programming development.

Richard J. Lindner *IBM AS/400 Division, Highway 52 and NW 37th Street, Rochester, Minnesota 55901.* Mr. Lindner is a senior programmer in software quality consulting at IBM Rochester. He joined IBM in 1966 and has held various technical and managerial assignments in product, tools, and process development. He became a professional engineer through training in the IBM Undergraduate Engineering Education program in conjunction with the University of Minnesota. He held both engineering and programming assignments for System/3 development, and held both programming development and managerial assignments for the development of System/38. Since then he has focused on development of support tools and processes for software development.

Richard J. Hedger *2453 Northern Hills Court, N.E., Rochester, Minnesota 55906.* Mr. Hedger has retired from IBM. He was a program manager responsible for the quality technology area in the IBM Rochester laboratory. He received a B.E.E. degree in 1962 and an M.S.E.E. degree in 1968, both from the University of Minnesota. He joined IBM in June 1962 and has held various technical and management assignments in product and tools development, process and support for System/3, System/38, and AS/400 products in the Rochester laboratory. He was on assignment in White Plains, New York, in the System Products Division and in the Information Systems and Components Group headquarters, where he was responsible for software process, measurements, and tools.

Reprint Order No. G321-5533.