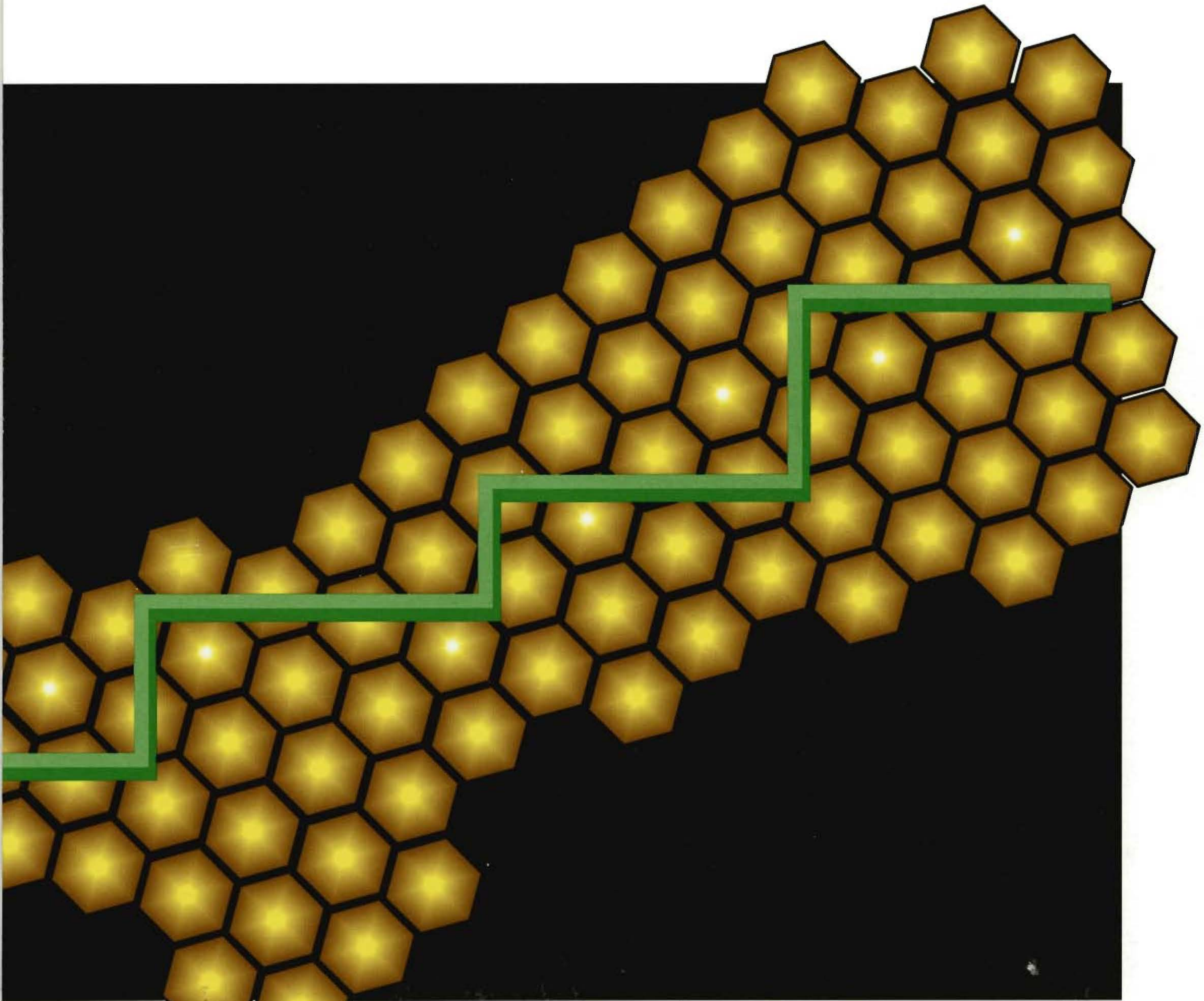
Software Process and Quality

Digital Technical Journal

Digital Equipment Corporation



Volume 5 Number 4

Fall 1993

The *Digital Technical Journal* is a refereed journal published quarterly by Digital Equipment Corporation, 30 Porter Road LJO2/D10, Littleton, Massachusetts 01460. Subscriptions to the *Journal* are $40.00 (non-U.S. $60) for four issues and $75.00 (non-U.S. $115) for eight issues and must be prepaid in U.S. funds. University and college professors and Ph.D. students in the electrical engineering and computer science fields receive complimentary subscriptions upon request. Orders, inquiries, and address changes should be sent to the *Digital Technical Journal* at the published-by address. Inquiries can also be sent electronically to DTJ@CRL.DEC.COM. Single copies and back issues are available for $16.00 each by calling DECdirect at 1-800-DIGITAL (1-800-344-4825). Recent back issues of the *Journal* are also available on the Internet at gatekeeper.dec.com in the directory /pub/DEC/DECinfo/DTJ.

Digital employees may order subscriptions through Readers Choice by entering VTX PROFILE at the system prompt.

Comments on the content of any paper are welcomed and may be sent to the managing editor at the published-by or network address.

ISSN 0898-901X

Documentation Number EY-F920E-DP

Book production was done by Quantic Communications, Inc.

Cover Design
The honeycomb structure has long inspired mathematicians and architects. On our cover, the honeycomb represents the structured discipline and dynamic development that are characteristic of software processes as described in this issue. Overlaid on the honeycomb is a stepped line representing the five levels of process maturity. These levels are defined by the Software Engineering Institute, a federally funded organization, and are discussed in the paper by Davies and Dumont.

The cover was designed by Susanna B. Ries of Digital's UNIX Software Publications Group.

Contents

6 ***Foreword***
Anthony F. Hutchings

Software Process and Quality

9 ***Modeling the Cost of Software Quality***
Stephen T. Knox

18 ***Changing the Rules: A Pragmatic Approach to Product Development***
Paul K. Huntwork, Douglas W. Muzzey, Christine M. Pietras, and Dennis R. Wixon

36 ***Defining Global Requirements with Distributed QFD***
John A. Hrones, Jr., Benjamin C. Jedrey, Jr., and Driss Zaaf

47 ***DEC TP WORKcenter: A Software Process Case Study***
Ernesto Guerrieri and Bruce J. Taylor

59 ***SEI-based Process Improvement Efforts at Digital***
Neil L. M. Davies and Margaret M. Dumont

69 ***Assessing the Quality of OpenVMS AXP: Software Measurement Using Subjective Data***
Robert G. Thomson

Editor's Introduction

Jane C. Blake
Managing Editor

Digital is continually seeking to adopt, improve, or devise processes that will deliver the highest quality products to our customers. In this issue of the *Digital Technical Journal,* software engineers from several of Digital's organizations present their experiences with modern software process methods, such as Voice of the Customer techniques and the Software Engineering Institute's (SEI) framework, that direct the development focus on the needs of customers.

One of the first hurdles for software process advocates is making a clear case for the value of implementing software product development processes. Steve Knox's paper offers a Software Cost of Quality Model that addresses the cost and schedule concerns of many software managers. The model demonstrates that among the incentives for improving software process is a two-thirds decrease in the cost of quality, as a percentage of development, as process maturity grows.

Digital's software processes are still in the early stages of maturity as defined by the SEI (described in a later paper). Nevertheless, software engineers who are using process techniques are already seeing significant benefits in the form of products that meet customer needs. Paul Huntwork, Doug Muzzey, Chris Pietras, and Dennis Wixon describe the techniques they used to gather customer requirements for the TeamLinks for Macintosh groupware application. TeamLinks designers utilized Contextual Inquiry and artifact walkthroughs, and a Vector Comparative Analysis tool to quantify the data obtained. The authors review the key requirements—and surprises—uncovered and the impact these had on design.

Quality Function Deployment is another process for obtaining an accurate, prioritized set of customer requirements, specifically through well-planned, structured meetings. John Hrones, Ben Jedrey, and Driss Zaaf present an enhanced approach to QFDs, i.e., a Distributed QFD for gathering customer requirements from around the globe. They reference a Digital-internal QFD conducted by Corporate Telecommunications Software Engineering.

The motto of the team that built DEC TP WORKcenter was "Use the process, but don't let the process use you." The team was in fact able to successfully adapt several processes—Contextual Inquiry, QFD, conceptual modeling, and rapid prototyping—to serve quality and schedule goals. Ernesto Guerrieri and Bruce Taylor analyze the effectiveness of these and other design-phase processes vis-a-vis the WORKcenter project and make recommendations for their general application in future software projects.

Many of the software methods described in this issue originated at the Software Engineering Institute, a federally funded organization which promotes software process infrastructure to achieve productivity and quality. Meg Dumont and Neil Davies provide a brief overview of the five levels of the SEI's Capability Maturity Model and discuss two case studies of their organizations' experiences with the CMM. Included are their evaluations of the challenges presented by the model and future directions for Digital's process-improvement efforts.

In the papers above, engineers stress the importance of learning customer requirements as early as possible in the project. For engineers porting the OpenVMS operating system to the Alpha AXP platform, customer requirements/expectations for this mature and complex system were well known. As Robert Thomson explains, ensuring that these expectations were met for the AXP product and at the same time meeting the aggressive Alpha AXP program schedule would require a new quality-assessment process. Robert describes how subjective data, obtained by means of a questionnaire for developers, can be used to assess the quality of a software release.

The editors thank Tony Hutchings, Technical Director of Digital's Software Engineering Technology Center, for selecting the subjects and writing the Foreword for this issue.

Jane Blake

Biographies

Neil L. M. Davies Neil Davies is the OpenVMS quality manager and is responsible for creating metrics, goals, and programs to achieve engineering excellence in the OpenVMS organization. In this role, Neil works with Digital's engineering management to introduce the techniques established by the Software Engineering Institute. Neil is also involved with the task force aimed at defining metrics for cycle time, applied time, and product quality and reducing the cost to qualify new systems. Prior to joining Digital in 1992, Neil was the software quality manager for all of Hewlett-Packard's computer systems.

Margaret M. Dumont Meg Dumont is the engineering manager for process development in the UNIX Software Group (USG). She helped organize a Software Engineering Institute Assessment of USG and created and staffed a process improvement group that is focused on the recommendations from the assessment. Meg is involved with the future direction and ongoing improvements within the quality function. Prior to this work, she created a standards engineering function within USG and was responsible for standards compliance in the DEC OSF/1 AXP operating system V1.0 to V1.2. Meg joined Digital in 1980.

Ernesto Guerrieri Ernesto Guerrieri, a senior software engineer in the Production Systems Group, is the DEC TP WORKcenter project leader. He is an adjunct professor at Boston University. Prior to joining Digital in 1990, he was employed by SofTech, Inc., where he was the chief designer and developer of reusable Ada products for information systems development (RAPID) center library. He holds a Ph.D. in software engineering from Rensselaer Polytechnic Institute and an M.S.C.S. from the University of Pisa. Ernesto has published various papers in software engineering. He is a member of ACM, IEEE Computer Society, and Sigma Xi.

John A. Hrones, Jr. As a consultant in the Software Engineering Technology Center (SETC), John Hrones works with clients to develop and implement TQM programs and is program manager for the Corporate Formal Inspection effort and for Six Sigma for Software. John joined Digital in 1973. He led software development efforts in MUMPS, BLISS, DSR, and OPS5 and originated the ELF facility and the Software Tools Clearinghouse. He managed the Corporate Forms, Base Graphics, and Engineering Quality Technology groups before joining SETC. John received a B.S. from MIT and an M.S. from the University of Michigan.

Paul K. Huntwork Paul Huntwork is a consultant engineer in Digital's Software Engineering Technology Center, an organization that collaborates with development groups to adapt or invent world-class methods for use in their product development activities. He joined Digital in 1987 after leading reengineering projects in software development, verification, manufacturing, and distribution at Computervision. Paul also led proto-SEI process assessment and maturation drives at Control Data Corporation, using techniques drawn from IBM's Federal Systems Division.

Benjamin C. Jedrey, Jr. Ben Jedrey is an information systems consultant within Corporate Telecommunication Software Engineering. He has program/project management responsibilities for developing telecommunications software and hardware solutions. He also manages next-generation routing-bridge filtering and network cost-reduction projects. Since joining Digital in 1966, Ben has contributed to information systems and programming and system development, and has provided support for financial applications at the corporate and field operations levels. Ben was previously employed by the brokerage firm Estabrook & Co.

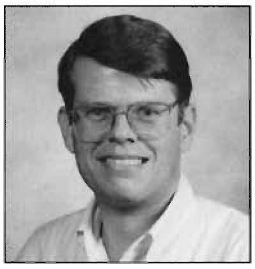
Stephen T. Knox Steve Knox is a principal software engineer with the Software Engineering Technology Center. Currently, he is assigned to the Networked Systems Management organization to improve software and development processes. Steve came to Digital in 1989 from Tektronix, Inc., to further develop the Contextual Inquiry process. A Quality Engineer certified by the American Society of Quality Control, Steve received the 1991 High Performance Systems Technical Leader Award. He holds an M.S. (1986) in psychology from Portland State University.

Douglas W. Muzzey Doug Muzzey is a software engineering manager in Workgroup Systems. He is the development manager for the TeamLinks for Macintosh and TeamRoute workflow products, and he sponsored the usability and customer partnering for the TeamLinks product family. In prior work, Doug contributed to communications and systems products and managed programs in Software Manufacturing, License Management, and Corporate Programs. Doug joined Digital in 1979. He holds a B.S.C.S. (1978) from Florida Technological University and an M.B.A. (1991) from Rivier College.

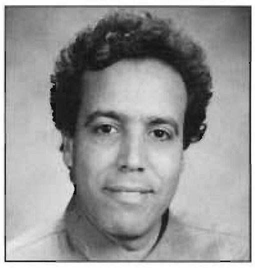
Christine M. Pietras A senior engineer in WorkGroup Systems, Chris collaborates in designing effective business solutions, incorporating field research data about customer work throughout the software development process. Her concentration is in user interface design and usability evaluation for TeamLinks products on the Macintosh and Microsoft Windows platforms. Chris joined Digital in 1985, after receiving an A.B. in mathematics from Smith College. In 1991, she earned an M.S. in industrial engineering and operations research from the University of Massachusetts—Amherst.

Bruce J. Taylor A principal software engineer, Bruce Taylor is the software architect of the DEC TP WORKcenter project. Prior to joining Digital in 1991, Bruce worked in CASE tool development at Intermetrics, Inc. He designed the repository database for the SLCSE software development environment and has published many papers on the use of database technology in the software development environment. He has a B.A. in English and an M.A. in computer science from Duke University. His current research interests include repository support for complete software life-cycle environments and software quality strategies.

Robert G. Thomson A senior software engineer in the OpenVMS AXP Group, Robert leads the validation of Volume Shadowing's port to the Alpha AXP platform. During the OpenVMS port, he measured quality and contributed to functional verification, for which he was co-recipient of an Alpha AXP Achievement Award. Since joining Digital in 1986, Robert has also contributed to improvements in system availability measurement and in symmetric multiprocessing and Backup performance. He has a patent pending and two published papers based on this work. Robert holds an M.S. in computer engineering from Boston University.



Dennis R. Wixon Dennis Wixon has worked in the area of user interface design for 20 years. He helped design the VT200 series keyboards and the DECwindows, Motif, and most recent windows interfaces. Currently a principal engineer in the Usability Expertise Center User, Dennis manages the Contextual Inquiry Program and conducts user needs analysis training. Before coming to Digital in 1981, he designed and programmed statistical analysis tools at Clark University. Dennis holds B.A., M.A., and Ph.D. degrees in psychology and has published more than 25 papers in psychology, statistics, and interface design.



Driss Zaaf Principal engineer Driss Zaaf of Corporate Telecommunications is a project leader for several network applications and products. In earlier work, he led software projects in the Publishing Technologies Group in Galway, where he helped develop a distributed library system. Before joining Digital in 1985, he was employed by CPT Corporation and Bull Groupe. Driss received an M.S. (1980) in science from the Faculty of Science, Rabat University, Morocco, and a Degree in Telecommunications Engineering (1983) from the Ecole Nationale Supérieure des Télécommunications de Paris, France.

Foreword

Tony F. Hutchings
Technical Director of Software Process and the Software Engineering Technology Center

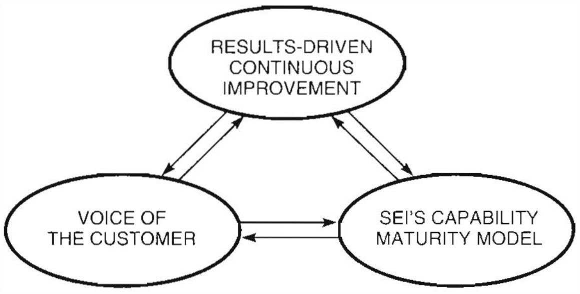
In the early 1980s, when the semiconductor and microprocessor industry was still relatively young, a few wise people recognized that *the* distinguishing factor for the winners in the race would be process, i.e., base technology, design methods, and CAD tools. They were right. Great processes are among the key reasons why Intel is today "top of the pile" and why our Alpha AXP chips achieve exceptionally high performance.

The formula works as follows: Brilliant, innovative people plus outstanding process produce consistently great results, repeatedly. This is in fact true of all product development efforts and is also therefore the case with software in the 1990s. We have thus devoted an entire issue of the *Digital Technical Journal* to software process and quality.

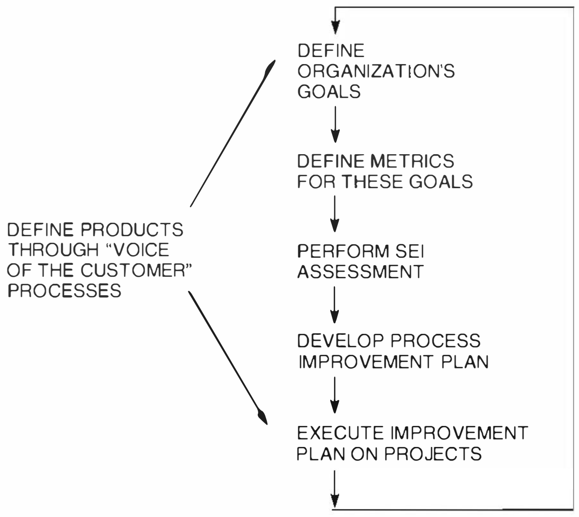
The most popular and effective models and methods for quality and process improvement hold several characteristics in common:

- All put the customer first, including knowing when customers and their requirements are being satisfied and when we and they are achieving desired results in the marketplace.
- All have a basis in applied measurement, using data from the application of the processes to help determine what changes to make.
- All are closed loop; that is, there is a clear path for feeding back observations to improve the current state of the process.

We are increasingly being asked: What is Digital's overall vision for software quality and process improvement? From a completely mature organization, the answer to that question would be something like the following: Every project sets its own clearly measurable, customer-driven quality goals; puts appropriate learning and improvement practices in place; continually monitors its progress toward its goals; and makes adjustments to process as needed to ensure it meets its goals. Fine words, but in reality we are not yet at that state in our corporate life. We have, however, developed a process-improvement strategy, or vision, which we hope will encourage all projects and groups to move toward the kind of state described above. That vision is best illustrated by the following diagram.


RESULTS-DRIVEN CONTINUOUS IMPROVEMENT
VOICE OF THE CUSTOMER
SEI'S CAPABILITY MATURITY MODEL

We imagine this vision would map to an implementation model as follows:


DEFINE ORGANIZATION'S GOALS
DEFINE METRICS FOR THESE GOALS
PERFORM SEI ASSESSMENT
DEVELOP PROCESS IMPROVEMENT PLAN
EXECUTE IMPROVEMENT PLAN ON PROJECTS
DEFINE PRODUCTS THROUGH "VOICE OF THE CUSTOMER" PROCESSES

The strategy comprises three important concepts: Using Voice of the Customer techniques to implement the intention of being a customer-driven company; basing our process application on assessing our current levels of performance and therefore the opportunities for introducing new "best practices" to overcome our weaknesses; and continuously using quantitative and qualitative analysis to determine how we might achieve better and better results. Our Voice of the Customer concept embraces such powerful techniques as Contextual Inquiry (for understanding customers' work and what might delight them in the future) and Quality Function Deployment (for rigorously prioritizing customers' requirements and how to satisfy them with world-class product concepts). Our application of the Software Engineering Institute's (SEI) approach to improving processes relies on performing organization-wide assessments of process capability and on developing long-lasting improvement plans, drawing on the rich pool of best practices described in their Capability Maturity Model. Our notion of Continuous Improvements rests on empowering engineering teams to study the results of their work with measurable data, analyzing the root causes of any process problems, and systematically implementing improvements to their processes such that they achieve better results.

The relationship between these concepts is subtle yet vital: All our process work needs to be customer-driven, and yet these Voice of the Customer techniques themselves need to be open to improvement as we learn from real data coming from their application; the advice in the SEI's Capability Maturity Model is sound and we need to choose judiciously the most appropriate best practices according to the state of maturity of each organization; nevertheless, as these practices are applied, we need to learn what is working and what is not and adjust their definition and application accordingly; these practices should also be chosen, at all times, to maximize the benefit for our customers as well as for ourselves.

None of the three mutually reinforcing elements of the composite strategy is sufficient individually to drive the massive and sustainable changes we want to see in software engineering process at Digital. The SEI's Capability Maturity Model framework alone under-emphasizes the extraordinarily important and powerful Voice of the Customer and Market; the Voice of the Customer and Market alone provides insufficient structure on which to hang an entire process improvement strategy; Continuous Improvement alone, as likely to be practiced at Digital, is at a level of intervention too low to move entire organizations sufficiently quickly toward orders of magnitude improvement in productivity and quality.

How are others in the industry tackling the problem of improving their quality and productivity? Many of the techniques and processes which we are now mastering or planning to are also in use by other leaders in our industry. For instance, Voice of the Customer processes (as typified by Quality Function Deployment) are in regular use at Hewlett-Packard; Formal Inspection (called Peer Reviews by the SEI) is practiced at Hewlett-Packard, IBM, and a host of other industry leaders; the use in software metrics is commonplace at Hewlett-Packard and Motorola; Continuous Improvement teams abound at Motorola, IBM, etc.

We have made great strides in the past two years in the application of better and more modern quality processes in Digital's software engineering community. No longer is the notion of using Voice of the Customer techniques really contested; few doubt the cost-effectiveness of Formal Inspections as a defect-detection technique; there is a ground swell of support for the SEI's organizational assessment model and a belief that its associated Capability Maturity Model offers a rich source of really good advice on the steps to take to improve one's process capability; and so on. We are even beginning to compile case studies from within Digital that demonstrate the positive impact of these processes, techniques, and concepts on project quality and schedule. Of course, we need many more such experiences before we can say that we are truly "best in class" in these areas.

Readers may well ask how the various papers in this issue relate to the strategy described here. Different aspects of the application of our Voice of the Customer techniques are emphasized in two papers: Contextual Inquiry and Rapid Prototyping are discussed in the paper "Changing the Rules: A Pragmatic Approach to Product Development"; an approach to using Quality Function Deployment across different geographies is covered in "Defining Global Requirements with Distributed QFD." Examples of how we are applying the SEI's assessment and Capability Maturity Model approaches are covered in "SEI-based Process Improvement Efforts at Digital." Another form of quality assessment is shown in the paper "Assessing the Quality of OpenVMS AXP: Software Measurement Using Subjective Data"; the business case for implementing SEI-like programs is covered in the paper "Modeling the Cost of Software Quality." Finally, in the paper "DEC TP WORKcenter: A Software

Process Case Study," many of these separate concepts are shown in practice: the use of a requirements analysis process, of defects metrics, and of overall continuous improvement.

Digital's software engineering processes are improving quite quickly and radically. To be completely successful will require a high degree of commitment and significant effort by management and engineers alike. The opportunity is, however, clearly there.

Stephen T. Knox

Modeling the Cost of Software Quality

This paper offers an extrapolation of the manufacturing and service industries' Cost of Quality Model to the business of software development. The intent is to provide a theoretical account of the changing quality cost structure as a function of a maturing software development process. Thus, the trends in expenditures due to the four major quality cost categories—appraisal, prevention, internal failures, and external failures—are presented over the five levels of software process maturity, according to the Software Engineering Institute's (SEI's) Capability Maturity Model for Software (CMM). The Software Cost of Quality Model conservatively proposes that the total cost of quality, expressed as a percentage of the cost of development, can be decreased by approximately two-thirds as process maturity grows from Level 1 to Level 5 of the SEI's CMM.

Introduction

Two questions often asked of quality function professionals by a software project manager are, How much will working on these quality processes cost me? and What can I expect in return for my investment? The manager recognizes that to implement a quality improvement project, resources must be allocated toward processes not currently being undertaken, and prior management experience has proven that usually the resources available are barely adequate to meet aggressive project and schedule deliverables. Also implicit in the manager's questions is the expectation of some point of diminishing returns: Even if there is benefit from an investment in quality-related work, help me understand the point at which the investment will be more costly than what I can get in return.

Background—The Traditional Cost of Quality Model

The concerns expressed by our present-day hypothetical software manager are the same concerns expressed by industrial management during the 1950s. At that time, the quality function professionals saw the need to extend quality attainment efforts beyond the traditional inspection and test activities to the processes further upstream in the manufacturing and product development groups. Quality function managers, hoping to increase the scope of the quality effort, were faced with the task of convincing upper management of the necessity to allocate additional resources to quality attainment. Management demanded that the quality function quantitatively demonstrate the amount of resource investment that was necessary and the expected return on that investment.

The quality function professionals responded by developing an investment model that expressed quality in terms of costs—the cost of attaining quality (the investment) and the cost of not attaining quality (the return). Their argument was that moderate increases in the former (typically, appraisal processes, such as inspection and test, and some defect prevention processes) would result in significant decreases in the latter (e.g., defects, scrap, repair and warranty costs), up to some point of diminishing returns. The traditional Cost of Quality Model shown in Figure 1 graphically represents their investment model.[1] The three curves portray moderate increases in prevention and appraisal costs resulting in dramatic decreases in failure costs. The point of inflection in the total cost of quality quadratic curve represents the point of diminishing returns on quality investment.

Figure 1 reflects the belief of the 1950s' quality function professionals that attaining 100 percent conformance to specification would be prohibitively expensive. The rationale was that zero-defects production would require extensive testing and inspection at every point in the design, manufacture,

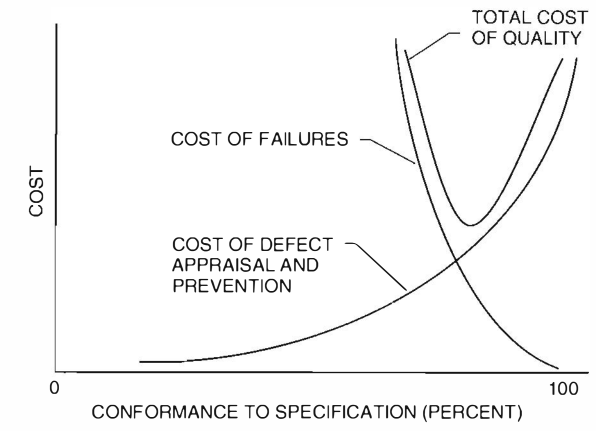


Figure 1 Traditional Cost of Quality Model

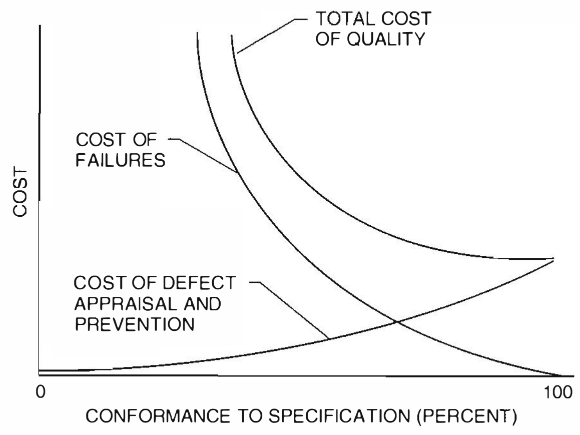


Figure 2 Revised Cost of Quality Model

and delivery process. Consequently, they conceived of a point of diminishing returns on quality-related investments. This point of maximum quality attainment for the minimum amount of investment is exactly the point of interest to our hypothetical software manager.

The modeled point of diminishing returns, however, was not verified by empirical cost of quality data.[2,3,4] In actual practice, investment in quality attainment shifted from appraisal to prevention processes as the quality function moved upstream into the manufacturing process and product design groups. Defect prevention processes, such as statistical process control and robust product designs, actually reduced the overall cost of attaining quality, contrary to the expectation of the quality function of the 1950s. Designing durable products to delight customers and manufacturing these products in a well-controlled environment resulted in fewer defects at the point of final inspection. Thus, appraisal costs were reduced significantly. (The author has participated in cases where successful application of defect prevention processes led to the complete elimination of expensive inspection and test.[5])

The Revised Cost of Quality Model

The quality function managers of the 1950s could not conceive of a quality investment model that did not rely heavily on inspection and test. Actual experience, however, uncovered that an increased emphasis on defect prevention processes led to significant reductions in appraisal costs and, in some cases, eliminated final inspection. The empirical cost of quality data resulted in a revised model, published in 1988.[2] As shown in Figure 2, the Revised Cost of Quality Model extracts the point of diminishing returns.

The three curves express the changing quality cost structure as quality attainment efforts shift from appraisal processes to the processes designed to achieve higher-quality output *before* final product test. In the revised model, the costs due to defect appraisal and defect prevention rise moderately as investments are made to improve product quality. The moderate increases in the costs of appraisal and prevention result in dramatic decreases in the failure costs. Unlike the corresponding curve in Figure 1, appraisal and prevention costs do not increase exponentially, since the means of quality attainment shifts from defect appraisal to defect prevention. The total cost of quality curve in Figure 2 consistently decreases as quality improves; therefore, the curve does not have a point of diminishing returns.

The Software Cost of Quality Model

The Revised Cost of Quality Model has been used extensively in the manufacturing and service industries as a benchmark against which actual quality costs are compared. The model has thus helped organizations identify opportunities for continuous improvement.[4] Also, a leading government research corporation, MITRE Economic Analysis Center, recently advocated using this method for reducing the cost of quality in software development.[6] What is lacking, however, is a model of quality costs in the domain of software development.

Important differences exist between the domains of the industrial environment and the software development environment. While an extrapolation of the Revised Cost of Quality Model can be made to monitor software quality costs (as suggested by MITRE), the author believes greater detail on and adjustments to the cost trends are required to account for differences between the domains. This paper presents a model that incorporates these differences. The Software Cost of Quality Model offers a rationale that addresses the reasonable concerns expressed by our hypothetical software manager.

Modeling the Cost of Software Quality

As background for a discussion of the Software Cost of Quality Model, this section deals with the subject of attaining software quality cost data and lists the software quality cost categories.

Software Quality Cost Data

Whereas the literature has sufficient data to support estimates of the costs related to not attaining software quality (e.g., defect and software maintenance costs), the author has been unable to locate rigorous accounting of costs related to attaining quality (e.g., testing and defect prevention). This is not surprising, given the relative lack of cost metrics tracked in software development. Capers Jones asserts that full quality costs have been tracked in some projects; in a personal conversation with the author, Jones cited his own work at International Telephone and Telegraph (ITT).[7] Other consulting firms (e.g., Computer Power Group) reported to the author that some clients kept limited metrics of defect costs. In follow-up investigation, however, the author has not found any rigorous accounting of defect appraisal and defect prevention costs in software development.

Consequently, the Software Cost of Quality Model offered in this paper extrapolates two key concepts from Gryna's Revised Cost of Quality Model (shown in Figure 2): (1) moderate investments in quality attainment result in a significant decrease in the cost of not attaining quality, and (2) an emphasis on attaining quality through defect prevention processes results in an overall decrease in the cost of traditional testing activities.

Software Quality Cost Categories

Following the modern trend in the industrial and service industries, the Software Cost of Quality Model subdivides the driving cost elements into four categories: appraisal and prevention (the costs of attaining quality, i.e., the investment), and internal failures and external failures (the costs of not attaining quality, i.e., the return).[2,3,4] Table 1 provides some examples of these elements in software development. The list of elements within each cost category is meant to be exemplary, not exhaustive.

Appraisal Costs Traditionally, the costs associated with appraisal activities are those incurred by product inspection, measurement, and test to assure the conformance to standards and performance requirements. In software development, these costs are usually related to the various levels of testing and to audits and assessments of the software development process. Appraisal costs also include costs (e.g., quality assurance) incurred by organizations that provide test support and/or monitor compliance to process standards.

Prevention Costs While appraisal costs are those used to find defects, prevention costs are those incurred by process improvements aimed at preventing defects. The examples of prevention costs listed in Table 1 are the costs that worried our hypothetical software manager, because for the most part, defect prevention processes in software are not traditional. Such processes are perceived as "front-loaded" processes, which lengthen the initial development schedule and threaten the probability that a project will deliver on the scheduled

Table 1 Software Quality Cost Categories

Appraisal	Prevention	Internal Failures	External Failures
Unit/Integration Testing	Contextual Inquiry/ Quality Function Deployment (QFD)	Defect Management	Problem Report Management
Quality Assurance	Project Management	Test Failure Rework	Warranty Rework
Field/Acceptance Tests	Requirements Management	Design Change Rework	Customer Support
Audits/Assessments	Formal Inspections	Requirement Change Rework	Lost Market Share

target date. Ironically, field testing (an appraisal cost) and the subsequent rework of found defects (internal failure costs) are traditionally accepted by software managers as legitimate yet frustrating tasks in the development cycle. One goal of software defect prevention processes is to reduce (and possibly eliminate) the need for expensive field testing.

Internal/External Failure Costs Failure costs are primarily due to the rework, maintenance, and management of software defects. Internal failures are software defects caught prior to customer release, whereas external failures are detected after release. Consistent with the initial cost of quality findings in the manufacturing industry data, the majority of quality costs in software are incurred by internal and external failures. The literature indicts the rework from software defects as the most significant driver of all development costs. Independent studies show costs associated with correcting software defects that range from 75 percent of the development effort at General Motors, to an average of 60 percent for U.S. Department of Defense projects, to an average of 49 percent, as reported in a survey by 487 respondents from academia and industry.[8,9,10]

The Model

Figure 3 depicts the Software Cost of Quality Model. The curves represent how the quality cost structure changes as a software development environment improves its capability to deliver a high-quality, bug-free product. Whereas the x-axes in Figures 1 and 2 reflect improving process capability in an industrial environment, the x-axis in Figure 3 is based on the Software Engineering Institute's (SEI's) Capability Maturity Model for Software (CMM).[11] The Software Cost of Quality Model incorporates the CMM, which offers a descriptive road map for improving software development processes. The details of this road map provide a rationale for theorizing the changing quality cost structure within the domain of software development.

The Maturing Software Development Process

The CMM is too extensive to describe fully in this paper. (Humphrey presents a detailed accounting.[12]) The central concept of the CMM is that a software development environment has a measurable process capability analogous to industrial process

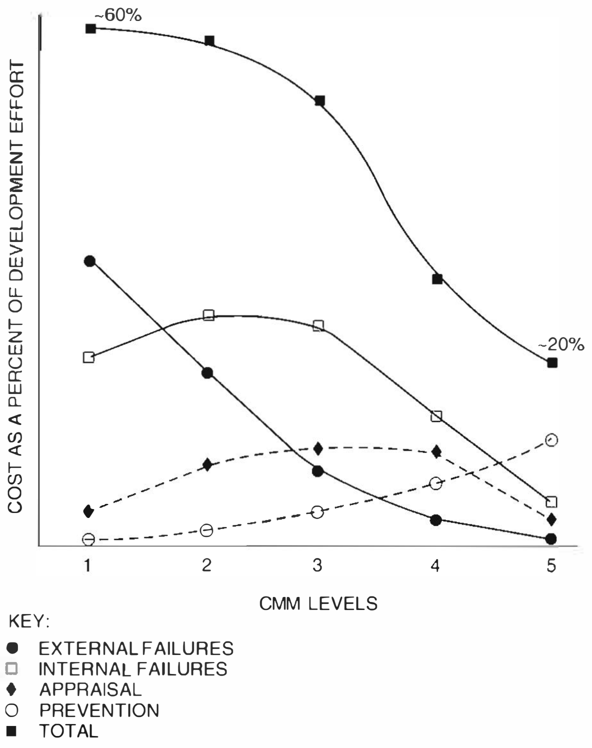


Figure 3 Software Cost of Quality Model

capability. In the software domain, process capability can be measured through assessment. The CMM proposes five levels of capability, ranging from the chaotic, ad hoc development environment to the fully matured and continually optimizing, production-line environment.

The SEI estimates through their assessment data that most software development environments are at the initial, chaotic level of capability. The SEI has also declared that although some individual projects show the attributes of the highest level of capability, no organization measured has demonstrated full maturation. Since no organization has made the journey to full maturation, and since scant data exists on the appraisal and prevention costs as they apply to software development, the Software Cost of Quality Model uses CMM Levels 1 to 5 as the discrete milestones at which the appraisal, prevention, and internal and external failure cost trends can be theorized.

Software Cost of Quality Model Assumptions

Before the cost trends in Figure 3 are examined in detail, two data-driven assumptions need to be declared. First, the total cost of quality (the sum of the costs associated with appraisal, prevention, internal failures, and external failures) at CMM Level 1 is equal to approximately 60 percent of the total cost of development. This assumption is based primarily on internal failure cost data taken from the literature and external failure cost data tracked at Digital. The estimate of internal failure costs comes from recent data collected by Capers Jones. The data indicates that software rework due to internal failures consumes 30 to 35 percent of the development effort for projects the size of those typical at Digital.[13] The lower range of this figure has been added to the cost of the Customer Support Center (CSC) management of external failures, which an unpublished study by the Atlanta CSC estimates to be 33 percent of the development costs (available internally only, on TPSYS::Formal_Inspection, *Cost of a Software Bug*, Note 31.0). Thus, the estimate of a total cost of quality equal to 60 percent of the development cost is based on the sum of the estimates of just two of the many cost elements, namely, rework due to internal failures and CSC management of external failures.

The second assumption is that the total cost of quality will decrease by approximately two-thirds as the development process reaches full maturity, i.e., CMM Level 5. This assumption is based on normative case-study industrial data cited by Gryna.[2] The data details the recorded change in the total cost of quality at the Allison-Chalmers plant during seven years of its quality improvement program.[14] Table 2 summarizes the reduction in the total cost of quality at Allison-Chalmers and relates this reduction to a similar change theorized in the Software Cost of Quality Model.

Although it may be unwise to assume that a normative trend for the manufacturing industry can be applied to software development, note that the assumed two-thirds decrease in the total cost of quality is more conservative than the estimates of SEI's Dr. Bill Curtis. He claimed return on investments (ROIs) in the range of 5:1 to 8:1, as an organization progresses in process maturity.[15] (Note: These claims have received empirical support from Quantitative Software Management [QSM] Associates, who report measured decreases in required effort and overall development cost on the order of 5:1.[16])

The Changing Cost Structure

Given the two grounding assumptions just discussed, the paper now presents a theoretical view of the changing cost trends between Level 1 and Level 5. The theory is based on the expected returns on investing in process maturity as outlined by the CMM. This section examines the details of Figure 3.

CMM Level 1

The SEI estimates that 90 percent of the software organizations today are at Level 1, which is characterized by an ad hoc, undefined, and sometimes chaotic development environment, highly dependent on heroic individual effort to meet delivery dates. Little attention is given to fundamental process management in this highly reactive atmosphere, and rework to correct internal and external failures is often perceived as necessary "fire fighting" to avoid disaster. At this level, the major costs of software quality are due to rework and maintenance. Testing is sporadic, so appraisal costs are minimal and most defects are experienced by the customers, resulting in expensive warranty costs and loss of market share. The costs associated with defect prevention approach zero.

CMM Level 2

A software organization at Level 2 has instituted the fundamental processes to manage resources, artifacts, and change. Project management, configuration management, and requirements management are the key processes that characterize a CMM Level 2 development environment that is, at the least, repeatable. In Figure 3, appraisal and internal failure costs increase at this level, primarily due to the

Table 2 Reduction in Total Cost of Quality (TCQ)

	Allison-Chalmers (% of Cost of Sales)	Software Cost of Quality Model (% of Cost of Development)
Initial TCQ	4.5	60.0
Improved TCQ	1.5	18.0
TCQ Decrease	67.0%	67.0%

formation of a quality assurance organization that monitors compliance to proscribed testing standards. Since, at Level 2, the organization applies testing activities more rigorously, more defects are found and reworked internally.

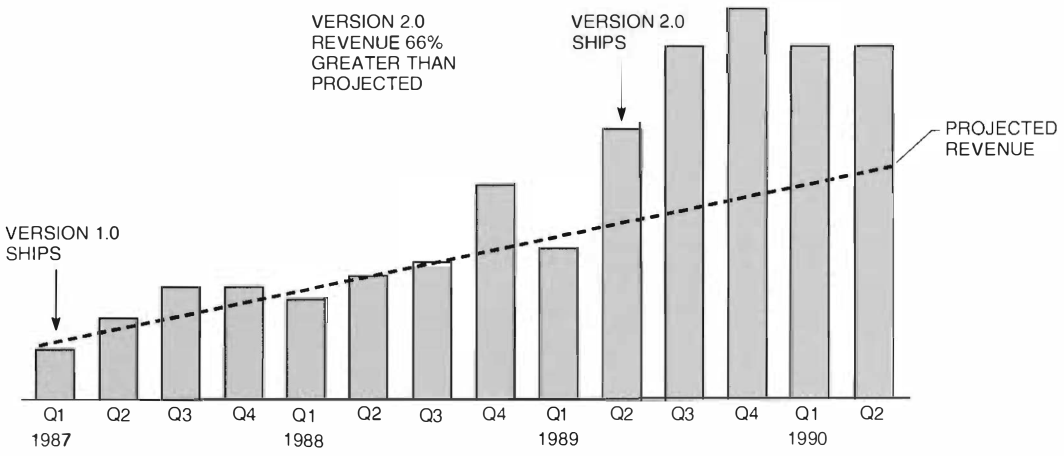
The increased testing activity and additional resources allocated to fix defects cause the apprehension that our hypothetical software manager expressed earlier. The manager experiences fear and uncertainty about being able to fix all the found defects and deliver the product on the scheduled date. Although our hypothetical software manager is probably aware that adherence to rigorous testing results in fewer defects shipped to the customer, a manager's success is often measured on the ability to deliver a product on time. The reduction in external failure costs at Level 2 occurs too late in the process to mitigate the career risk of seriously missing the delivery date.

CMM Level 3

According to the CMM literature, the major gains at Level 2 are the creation of repeatable processes that provide the base underpinning of a maturing development environment. Figure 3 illustrates that the investments to improve quality have been primarily in the appraisal category. But at CMM Level 3, the development environment has achieved a point of stability. A defined, documented framework exists within which the creative act of software design can be executed in a controlled manner. Quality attainment now emphasizes investing in the prevention activities, such as Contextual Inquiry into customer problems and Formal Inspections of specification and design documents. Such prevention processes are intended to ensure a more accurate understanding of and a greater conformance to customer requirements. Investing in prevention results in a steep decline in the external failure costs and gaining back lost market share.

Our hypothetical software manager is entitled to be more than skeptical about such claims; however, empirical data substantiates them. For example, Figure 4 details the 66 percent increase over projected revenue for VAX RALLY version 2.0, a direct result of improvements made to earlier versions—improvements suggested by the Contextual Inquiries conducted with VAX RALLY version 1.0 customers.[17] Figure 5 clearly demonstrates that Contextual Inquiry leads not only to increased revenue but to the higher productivity and lower defect density experienced by POLYCENTER System Census version 1.0, when compared to four other system management applications.[18] These applications, represented in Figure 5 as A, B, C, and D, were developed without the use of this critical defect prevention process.

While generally considered to be part of the appraisal process, Formal Inspections, when applied



NOTE: The bars represent the relative revenue per quarter; the dotted line represents the linear fit of projected revenue, based on version 1.0 performance. Version 2.0 includes the improvements suggested by Contextual Inquiry.

Figure 4 Effects of Contextual Inquiry on VAX RALLY Revenue

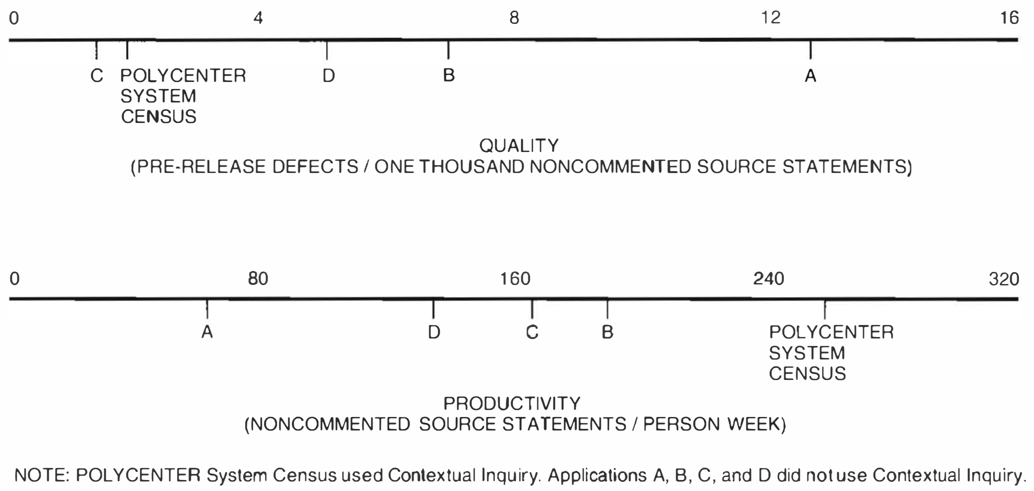


Figure 5 Effects of Contextual Inquiry on POLYCENTER System Census Quality and Productivity

to source documentation such as specifications and design, are similar to process control monitors. These inspections ensure that critical functionality is not omitted as the development process proceeds from the stated requirement for a solution to the specification and design of that solution. The effectiveness of the Formal Inspection process in preventing potential inconsistencies and omissions accounts for its rating as the most efficient defect removal method, as shown in Table 3.[19] Thus, applying Formal Inspections as a defect prevention process means fewer defects to test and fix internally and a more satisfied customer using the product.

The data in Table 3 is not intended to fully account for the magnitude of the trends at Level 3. Rather, the data offers a rationale for the overall direction of these trends. If a disparity exists between the data and the acceleration of decreasing failure costs in Figure 3, bear in mind that the model is the more conservative estimator.

Table 3 Defect Removal Efficiencies

Method	Efficiency (Percent)
Formal Inspections	65
Informal Reviews	45
Unit Testing	25–50
System Testing	25–50
Regression Testing	20–50
Field Testing	30
Beta Testing	25

CMM Levels 4 and 5

Although it has seen evidence of CMM Levels 4 and 5 in a few discrete projects (e.g., one Japanese project reported to be at Level 5), the SEI reports that it has not yet measured a Level 4 or a Level 5 organization. At these higher levels of maturity, the dominant cost of quality is due to the prevention elements, primarily from the cost elements of metric-driven continuous improvement and process control. The software process at these levels has become so well characterized by metrics that it has achieved a state where development schedules are predictable. Requirements are now understood quantitatively. The costs attributable to traditional appraisal activities, especially field testing, are dramatically decreasing, since product quality can now be appraised by monitoring the development process as opposed to expensive testing of the product. By Level 5, appraisal and failure costs have dropped to the level expected of a Six Sigma organization. The model proposes that the total cost of quality has decreased by approximately two-thirds, which is consistent with the normative industrial data.

Conclusion

This paper is not an endorsement of the SEI's Capability Maturity Model for Software, which is used here to describe discrete states within a maturing software development process. Although the CMM offers a rational, staged approach to achieving a predictable and highly productive development environment, the CMM is not the only road map to improving Digital's software process. For example, the variety of customers served in commercial software development offers special challenges to ensure that these customers' work experiences are brought into the design and development process. The CMM does not detail Voice of the Customer processes, which are practiced increasingly at Digital. In addition, some key processes specified for CMM Levels 3, 4, and 5 (e.g., Formal Inspections and metric-driven Continuous Improvement) are effective in reducing defects. These processes are already used in many of Digital's organizations, even though it is doubtful that any of the software development groups at Digital would be assessed as being beyond CMM Level 2.

The author believes that CMM Level 5 is the goal, regardless of the road map for attainment. The Software Cost of Quality Model explored in this paper offers the same argument for improving process capability that was offered in the manufacturing industries: the major costs of quality are the waste and the resource loss due to rework, scrap, and the lost market share when products do not possess the quality to address the problems faced by customers. The key to reducing quality costs is to invest in defect prevention processes, many of which are detailed by the CMM.

So, the response to the initial concern expressed by our hypothetical software manager is the following: You will not experience a point of diminishing returns from investing in quality-attaining processes. Certainly, there is a steep learning curve, and the short-term gains are not apparent. Given the software life cycle, most of the short-term gains will be experienced after the development is complete and the product has been shipped.

Since investments in quality, however, are not meant to realize quick, dramatic returns, the defect prevention processes probably offer the most immediate visible evidence that the overall cost of quality has been reduced. Yet, regardless of whether the investment is made according to the CMM road map or using some other quality attainment plan, software managers must keep in mind that quality attainment processes require a great deal of hard work. Also, the investment must be constant to achieve the significant, long-term payback, as reflected in the Software Cost of Quality Model.

References

1. J. Juran and F. Gryna, *Quality Planning and Analysis* (New York: McGraw-Hill, 1980).

2. F. Gryna, "Quality Costs," *Juran's Quality Control Handbook,* 4th ed. (New York: McGraw-Hill, 1988).

3. J. Campanella, *Principles of Quality Costs,* 2d ed. (Milwaukee, WI: ASQC Quality Press, 1990).

4. J. Atkinson et al., *Current Trends in Cost of Quality: Linking the Cost of Quality and Continuous Improvement* (Montvale, NJ: National Association of Accountants, 1991).

5. S. Knox, "Combining Taguchi Signal-to-Noise Metrics with Classical Regression for Robust Process Optimization," *Technical Report ISB Quality* (Marlboro, MA: Digital Equipment Corporation, 1990).

6. E. deGuzman and T. Roesch, "Cost Analysis Provides Clues to Spot Quality Problems," *The MITRE Washington Economic Analysis Center Newsletter* (McLean, VA: April 1993): 2.

7. C. Jones, *Applied Software Measurement* (New York: McGraw-Hill, 1991).

8. B. Boehm, *Software Engineering* (Redondo Beach, CA: TRW, 1976).

9. J. Hager, "Software Cost Reduction Methods in Practice," *IEEE Transactions on Software Engineering,* vol. 15, no. 12 (December 1989).

10. W. Goeller, "The Cost of Software Quality Assurance," *1981 ASQC Quality Congress Transactions,* San Francisco, CA (1981): 684-689.

11. M. Paulk, B. Curtis, and M. Chrissis, "Capability Maturity Model for Software," Technical Report CMU/SEI-91-TR-24 (Pittsburgh, PA: Software Engineering Institute, Carnegie-Mellon University, 1991).

12. W. Humphrey, *Managing the Software Process* (Reading, MA: Addison-Wesley, 1989/1990).

13. E. Yourdon, *The Decline and Fall of the American Programmer* (Englewood Cliffs, NJ: Yourdon Press, Prentice-Hall, 1992).

14. O. Kolacek, "Quality Cost—A Place for Financial Impact," *Transactions of the 1976 Annual Conference of ASQC*, Milwaukee, WI (1976): 131.

15. B. Curtis, "The Superior Software Organization," *Software Process Improvement Network Meeting,* Boston, MA (January 1993).

16. L. Putnam, "The Economic Value of Moving up the SEI Scale," Technical Report QSMTR93-01 (McLean, VA: Quantitative Software Management, Inc., 1993).

17. J. Neilsen, *Usability Engineering* (New York: Academic Press, 1993): 4.

18. S. Knox, *Newsletter of Continuous Improvement for Networked Systems Management,* vol. 2 (Maynard, MA: Digital Equipment Corporation, April 1993).

19. C. Jones, "Software Metrics and Total Quality Management," *Case Outlook,* vol. 6, no. 4 (1992): 1-11.

Paul K. Huntwork
Douglas W. Muzzey
Christine M. Pietras
Dennis R. Wixon

Changing the Rules: A Pragmatic Approach to Product Development

Developing quality software rapidly and at low cost has been an elusive goal. Nevertheless, meeting this goal is essential in today's competitive environment where more and better products appear at accelerating rates and customers demand systems that support "what users need to do" in a natural and cost-effective manner. This paper discusses the processes used by the TeamLinks for Macintosh project team to achieve customer focus throughout the development of a groupware office product. Listening to customers radically reshaped the product and led to more rapid decisions, shorter development cycles, higher quality, and greater customer satisfaction.

Where We Started

Product Overview

TeamLinks software allows Windows PCs and Macintosh computers to be integrated into enterprise-wide networks. The product utilizes Digital's extensive line of network applications and services, such as electronic mail, file sharing, workflow procedures, and work group applications.

The TeamLinks product also makes use of the latest personal productivity and client-server technology as a platform for comprehensive office solutions. Just as Digital's ALL-IN-1 Integrated Office System (IOS) allows organizations to rapidly develop organization-wide network applications in a time-shared environment, TeamLinks software provides capabilities that allow the creation of company-wide client-server office applications tailored to meet the needs of any operation.

TeamLinks software provides customers with an intuitive graphical user interface that integrates their powerful personal productivity tools, such as word processing and spreadsheet applications, into local and wide area networks. This feature is independent of whether the user's desktop system is a Windows PC or a Macintosh computer.

Product Goals

For enterprise-wide work group computing strategies to have customer appeal, they must address both PC and Macintosh desktop computers. The introduction of TeamLinks for Windows during the spring/summer of 1992 further highlighted the need to immediately introduce similar functions on a Macintosh platform. The use of inside-outside strategic planning identified three primary factors that required consideration during the development of admissible product delivery strategies.[1]

First, we must satisfy the wants of the potentially available market. Customers require both Windows and Macintosh desktop solutions for their enterprise work group computing. Both the TeamLinks Program Office and customers requested a Macintosh platform that supported the core TeamLinks services of mail, ad hoc workflow, and filing, with product availability within six to nine months.

Second, we must deliver an acceptable solution with the available resources. Macintosh users are frequently recognized as demanding consumers of software applications. Although the breadth of experience in developing Macintosh products within the group was limited, the development team consciously planned objectives aimed at satisfying demanding consumers. The team's goals consisted of satisfying customers' basic office needs and having the product recognized as a quality TeamLinks implementation on the Macintosh platform.

Third, we must develop a product within the opportunities and constraints of today's environment. In many development environments, the reality of budgets with minimal and ever-decreasing resources is rapidly becoming today's normal mode of operation. Changing strategies, requirements, and management infrastructure are also particularly characteristic of current development environments.

Product Strategy

After resolving our initial project goals, we developed strategies to satisfy the goals. We chose to establish design partnerships with customers to iteratively obtain comments to use as a basis for refining the project's specific deliverables.

Most problem-solving strategies are simple variations of (1) define the problem, (2) develop solutions, (3) test, and (4) refine the solutions. The TeamLinks project team chose an iterative and concurrent adaptation of this strategy.

First, we identified our implicit working assumptions. Initially, the project assumed that all components present in the TeamLinks for Windows product would simply be ported to the Macintosh platform and retrofitted with a Macintosh user interface.

Second, we developed product plans based on our initial goals and implicit working assumptions. Iterative design techniques require prototypes that customers may evaluate and comment on. The project's initial product plans were utilized as the first product prototypes for collecting customer responses.

Third, we verified and refined our plans based on validated information. As product prototyping got under way, the team analyzed information from competitive products, industry consultants, and customers. A key consideration for the development team was that throughout the life cycle of the project, specific product deliverables would be changed as customer opinions became clear. As incoming data evolved into information, the cost and benefits of each change would be carefully weighed against the project's goals.

Product development thus proceeded on two fronts: one formulated in advance, the other created in response to new developments, customer comments, and experience with successes and failures of the plan.

Select the Best Work Model

Since the emergence of the software industry and continuing through the present, the ability of software groups to produce high-quality software has fallen far short of customer needs and demands. In response to this condition, government and academic specialists proclaimed a "software crisis" in 1969 and endorsed a concept of software engineering based on authoritative, hierarchical organizations and sequential application of specialized functions.[2] This model of software engineering is still prevalent in textbooks. Ironically, the model was created at a time when the competitive advantage of total worker participation in cross-functional teams, an outgrowth of Deming's approach to management, was being demonstrated in other industries.[3] The cross-functional approach is now widely recognized as a superior method of new product development. Figure 1 shows how cross-functional teams speed up work. Twenty-four years of the sequential model have not diminished the software crisis. We feel privileged to have been able to apply the cross-functional model to the development of the TeamLinks for Macintosh product. Descriptions of other best practices used by the TeamLinks team follow.

Find Out What Your Customer Needs

Determining the needs of our customers involved field research, quantitative research, and design justification through grounding.

Field Research One of the most powerful rationales for field research is the realization that effective design begins with the discovery of exactly what users and customers want and do. Field research methods are designed to provide such in-depth understanding. These methods emphasize openness to user experience and create a dialog with users about that experience. Direct contact with users at early stages of design is viewed as an essential step, and the barrier between users and designers has been cited as a significant cause of suboptimal design.[4,5]

Quantitative Research Given that discovery is the first stage to effective design, the next stage is decision.[6] Most likely, a team will not be able to respond to all user needs. Thus, it needs a systematic and objective way to make decisions. Quantitative methods provide a basis for decisions because they establish a dimension along which features can be compared.

Grounded Design Unfortunately, many designs have an insufficient basis. Third-hand information, brainstorming, anecdotes from trade shows, and

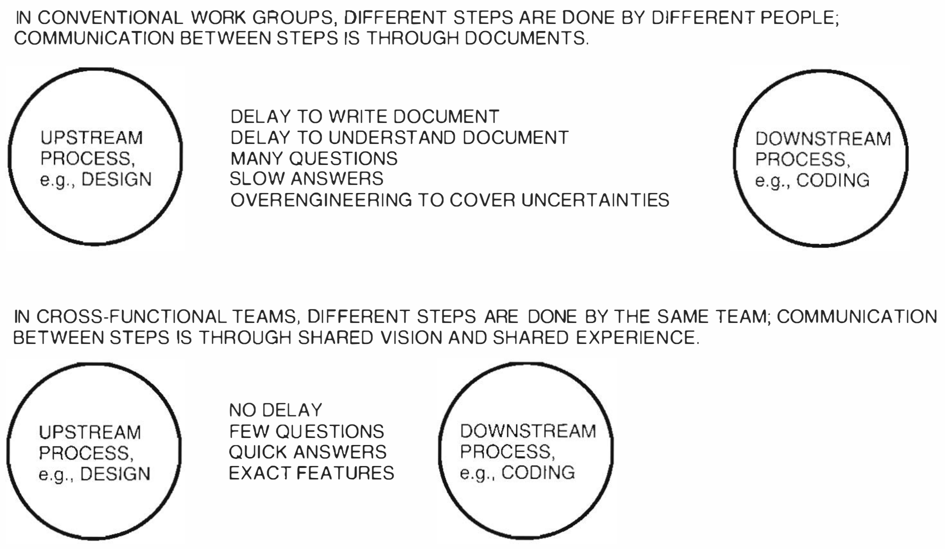

IN CONVENTIONAL WORK GROUPS, DIFFERENT STEPS ARE DONE BY DIFFERENT PEOPLE; COMMUNICATION BETWEEN STEPS IS THROUGH DOCUMENTS.
UPSTREAM PROCESS, e.g., DESIGN
DELAY TO WRITE DOCUMENT
DELAY TO UNDERSTAND DOCUMENT
MANY QUESTIONS
SLOW ANSWERS
OVERENGINEERING TO COVER UNCERTAINTIES
DOWNSTREAM PROCESS, e.g., CODING
IN CROSS-FUNCTIONAL TEAMS, DIFFERENT STEPS ARE DONE BY THE SAME TEAM; COMMUNICATION BETWEEN STEPS IS THROUGH SHARED VISION AND SHARED EXPERIENCE.
UPSTREAM PROCESS, e.g., DESIGN
NO DELAY
FEW QUESTIONS
QUICK ANSWERS
EXACT FEATURES
DOWNSTREAM PROCESS, e.g., CODING

Figure 1 How Cross-functional Teams Speed Up Work

speculative talk about "what the customer really wants" within an isolated team all contribute to designs that do not meet customer needs and designs that do not reflect customer work. To ground a design means that all aspects of the design are rooted in customer data rather than in speculation. Providing mechanisms for this grounding is critical to producing an effective design.

Design Your Product Based on What You Learn

Demand pull, customer involvement, and design metaphors all contribute to a customer-focused product design.

Demand Pull Using customer interaction to pull design features out of the development team greatly reduces the number of design decisions and the time required to make these decisions. A customer focus on work essentials and not on "bells and whistles" provides unambiguous feedback that supports direct decisions.[7]

Customer-driven Design Design is a process of refinement and elaboration embedded in a cycle of creation and evaluation. Customer-driven design involves the evaluation of a tentative design (the creation) with the customer's evolving understanding of their work vis-a-vis the product.

Design Metaphors Metaphors are an effective way to generate a design from customer work and technical capabilities. Examples include the "desktop" metaphor that drives much user interface design today. Although often criticized, metaphors have been shown to be very powerful and fundamental to human thought.[8,9,10]

Refine Your Product with Customers

Using an iterative approach to product design combined with prototyping helps refine the product design.

Iterative Requirements The need to break the development of complex software into manageable pieces has led to schemes such as "separation of concerns," "top-down development," and "step-wise refinement." Iterative design addresses this problem with a "basics first" approach. A basic idea is embodied in a prototype implementation and reviewed with customers. The iterative approach allows solutions to come into being and quickly converge to finished products under the influence of user interaction, even while users are discovering what they need. Detailed requirement specifications are not necessary to begin implementation, so there is no time lag between gathering requirements and providing solutions. This approach minimizes miscommunication and eliminates obsolete requirements.[11]

Prototyping Prototyping supports a customer-driven design process, providing customers with an effective medium to respond to current system thinking.[12] For instance, user interface designs embody a theory about the way users work.[13] The most straightforward way to get feedback on the theory is to express it in a prototype. A prototype allows users to try the system directly instead of translating their work into an unfamiliar symbolic language.[14]

What We Did

The project team developed customer partnerships early in the project life cycle. Through Contextual Inquiries, focus groups, and artifact walk-throughs, the team internalized customer needs and requirements. The new data helped establish a shared understanding among team members and manifested itself in a new product design. Vector Comparative Analysis (VCA) data summarized team learnings and provided the foundation for new designs. Figure 2 diagrams this process.

Find Out What Your Customer Needs

Cross-functional Teams The team comprised product managers, engineering managers, engineers

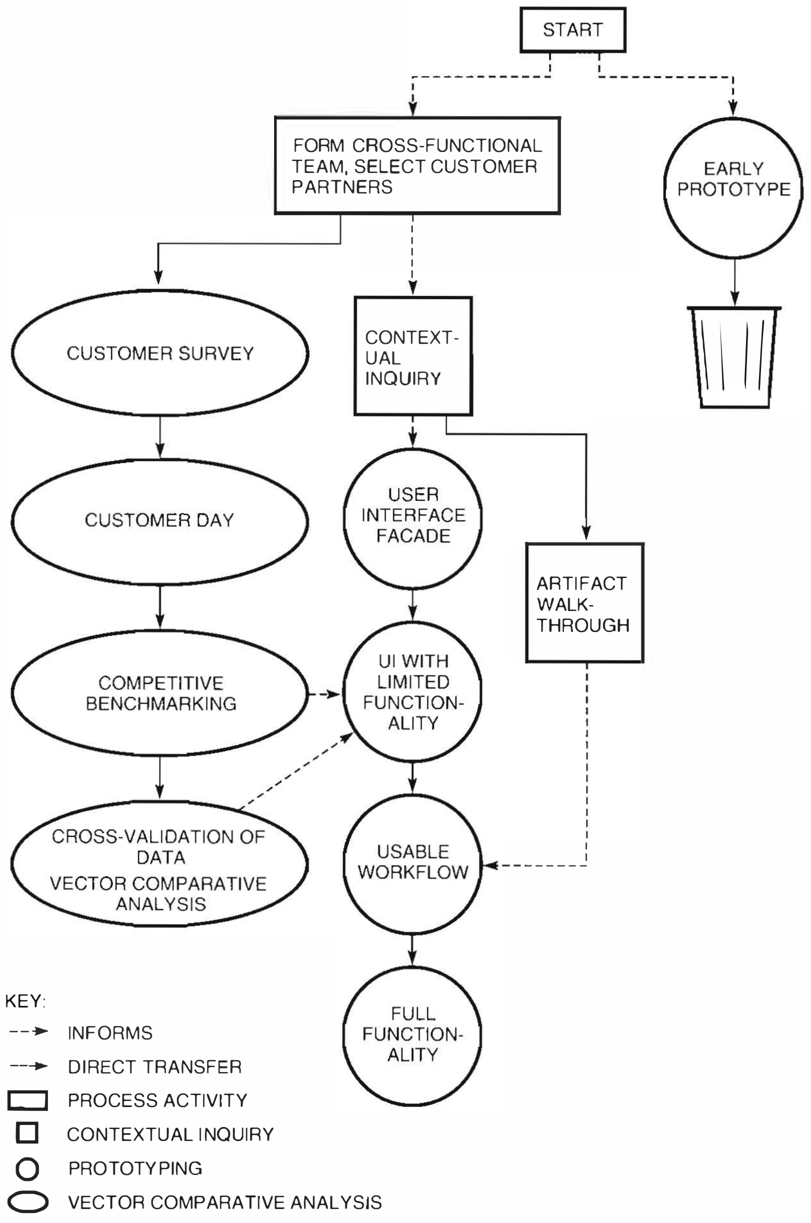


Figure 2 Overall "Find Out" and "Refine" Activities

(including some from companion products), account managers and support people, customer personnel, and specialists in marketing, human factors, graphic design, user publications, and competitive analysis. This cross-functional team took training, visited customers, analyzed data, and made decisions as a whole or in cross-functional subgroups. The mutual understanding that grew out of the shared experience and the shared data enabled faster, more stable decisions and shorter schedules.

Customer Partners We formed product-life-cycle partnerships at the start of the project with customers who represented the four industries that most heavily use PCs on the desktop: U.S. government contractors, manufacturing, pharmaceuticals, and banking. Within these industries, we identified Digital customers from the office partner group who used Macintosh PCs. Working with the account teams and the customers themselves, we selected partners who represented their industries. Each partner designated a specific person to coordinate their participation.

These partnerships allowed more interaction, better follow-up, clearer communication, and more consistent direction. For example, we could model their work in detail in later versions of the prototypes, and the partners could perform complex evaluations. Since we were familiar with their work and they were familiar with our product, no one experienced a high cost of learning at any stage of the project.

Contextual Inquiry We decided to train the team in Contextual Inquiry methods so that they could interact more effectively with customers. Contextual Inquiry techniques are adaptations of the methods used by anthropologists and sociologists to understand other cultures. The Contextual Inquiry framework emphasizes three principles: (1) context, i.e., study user work in its natural environment; (2) partnership, i.e., engage customers as co-investigators to help develop your understanding; and (3) focus, i.e., clarify your interests and assumptions and be willing to change them based on what customers tell you.[15] Contextual Inquiry techniques have been used widely at Digital and have shown a positive impact on market penetration and revenue.[16]

Customer Survey Information from customer visits was organized into a single hierarchy with benefits and needs at the top and desired capabilities and features at the bottom. A questionnaire was created to obtain quantitative customer importance weights for each node and leaf of the hierarchy. The questionnaire was sent to the customer partners. We encouraged multiple responses from each partner to get data from both Information System professionals and end users. We also collected importance weights from an industry consultant and additional customers beyond the partners. Figure 3 shows a typical question from the questionnaire.

ALLOCATE 100 POINTS AMONG THE FOLLOWING CHARACTERISTICS TO INDICATE THEIR RELATIVE IMPORTANCE TO YOU AS COMPONENTS OF "SUPPORT PERSONAL DIARY."

___ PROVIDE TIME, TASK MANAGEMENT
___ SUPPORT SEARCHING CALENDAR FORWARD, BACKWARD IN TIME
___ PROVIDE QUICK, SIMPLE NAVIGATION TO ANY DATE
___ PROVIDE VARIED CALENDAR VIEWS

Figure 3 Sample Questionnaire Question

Customer Day Representatives from the four customer partners brought completed questionnaires to a customer day. We inquired about their experience with the questions, looking for omissions and refinements. We asked them to describe their top 10 issues and explain why they are important in their environment. The customer day information provided additional insight into user needs as well as a sanity check of the quantitative survey data.

Competitive Benchmarking We created a score sheet from the features at the lowest level of the hierarchy developed for the customer survey. Engineers on the TeamLinks project, an industry consulting firm, and customers scored our existing products, alternative versions of our planned product, and competing offerings. The scoring by engineers directly contributed to their understanding of customer requirements. The information also fed the VCA process. Figure 4 shows a typical question from the score sheet.

Cross Validation To minimize investment risks and to maximize the return on the wealth of information obtained from the data-gathering exercises, we revalidated the information to determine its applicability to the project. The information was cross-validated by comparing multiple sources, including the competition, industry consultants, SCORE EACH OF THE FOLLOWING FEATURE CATEGORIES FROM 0 TO 5, BASED UPON THE DIMENSIONS OF COMPLETENESS AND GOODNESS AS COMPONENTS OF "SUPPORTING DIFFERENT WORK STYLES":

___ SUPPORT OFF-LINE WORK
___ PROVIDE TOOLS THAT SUPPORT CONSENSUS MANAGEMENT
___ PROVIDE TOOLS THAT SUPPORT LOCAL CULTURE
___ PROVIDE TOOLS THAT SUPPORT TELECOMMUTING

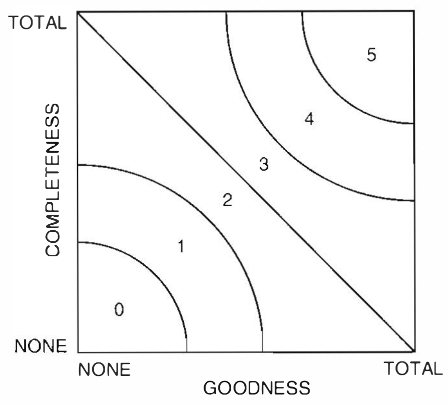


Figure 4 Sample Score Sheet Question

and customers. We verified that we could understand different responses as true expressions of different needs before we used the data.

Vector Comparative Analysis We input the customer importance weights from the questionnaire and the feature scores from the score sheet into the computer-based VCA tool.[17] This tool rolls the feature scores up through the hierarchy by a method of weighted averages to provide a score at each node. VCA can create a vector diagram for each node showing graphically how well each product satisfies the user needs represented by the node. Figure 5 shows the top few branches in the TeamLinks VCA hierarchy. Digital developed VCA for use with or as an alternative to Quality Function Deployment (QFD). For the TeamLinks project, no QFD was conducted.

Artifact Walk-throughs Based on Contextual Inquiry principles, artifact walk-throughs allow a design team to look at processes that take place over time and that occur among groups of people. The name is derived from the approach of asking customers to bring the actual artifacts of a process, e.g., notes, memos, forms, and documents, into the walk-through as a reminder of the full complexity of the process. In the presence of the artifacts, we ask for the overall process goals, any known issues and problems, and a list of process steps. For each step of the process we ask, Who makes requests? Who does work? Who approves? What is the cost in person effort, materials, and equipment? What is the normal cycle time? and What problems and issues exist with this step? Each type of information is recorded on a colored Post-it note and assembled into an annotated flow diagram of the process. Thus, these walk-throughs emphasize articulating a process in detail, grounding it in a specific customer example. We chose artifact walk-throughs as the natural approach to gathering data in order to customize our prototypes to each customer situation. At the same time, the walk-throughs uncovered additional general requirements.

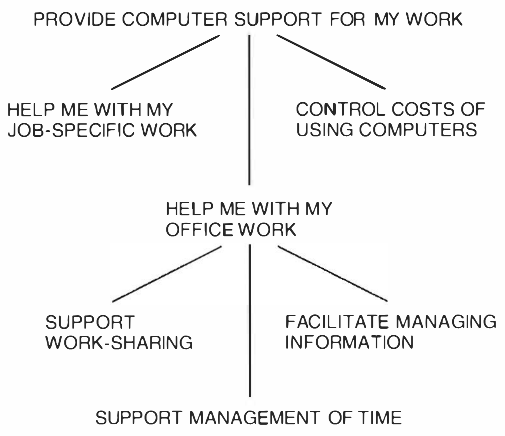

PROVIDE COMPUTER SUPPORT FOR MY WORK
HELP ME WITH MY JOB-SPECIFIC WORK
CONTROL COSTS OF USING COMPUTERS
HELP ME WITH MY OFFICE WORK
SUPPORT WORK-SHARING
FACILITATE MANAGING INFORMATION
SUPPORT MANAGEMENT OF TIME

Figure 5 Simplified TeamLinks Hierarchy

Design Your Product Based on What You Learn

Team Discussions The Contextual Inquiry results contained surprises. Even though the inquiry focus was on office products, customers expressed more requirements about cost containment than about product features. The messages, discussed in detail in the section What We Learned, were clear in the raw data and became the basis for revised plans even as the rigorous VCA was being completed. At this time, an early prototype, seen only by the development team, was redirected. Real customer data enabled rapid consensus within the team on changes to the project's direction.

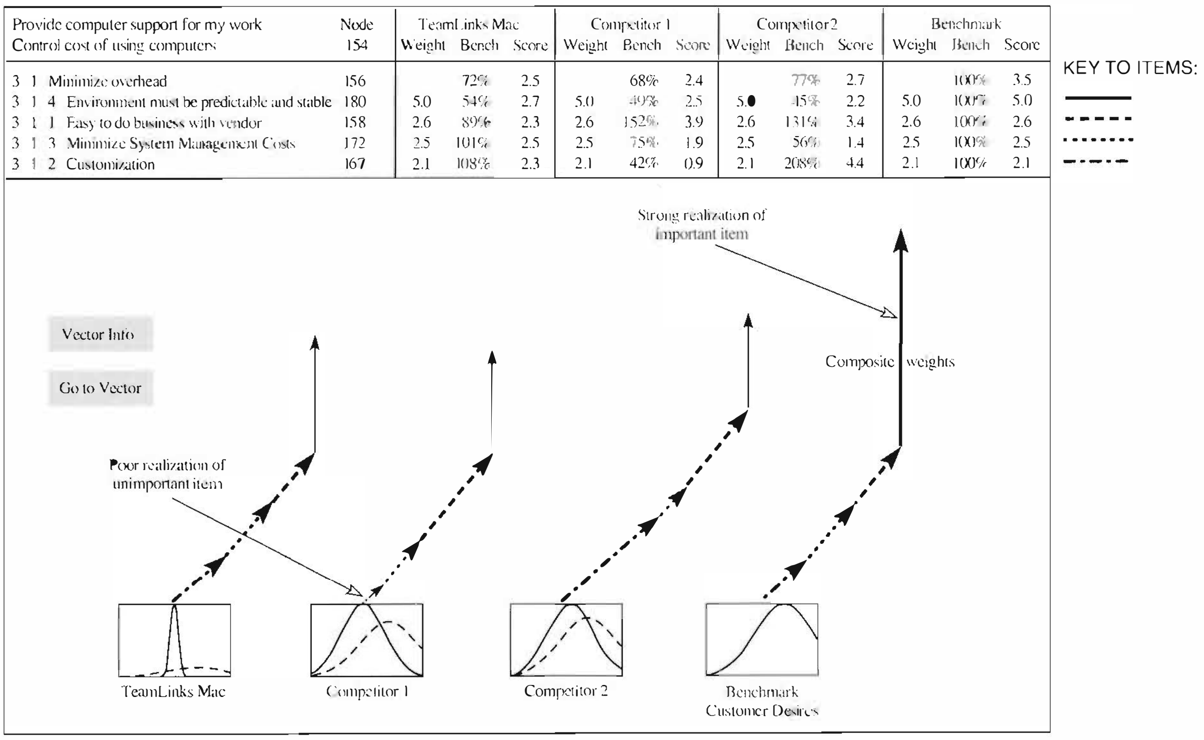
Competitive Positioning The survey and benchmark data, which was processed by VCA, allowed us to track our competitive position at all times.

We could say, for instance, "If we build this alternative, we will satisfy more customers than competitor A but will need more mail features to compete with B." In addition, when the engineers performed the benchmarking in person, they learned more than just scores. One engineer decided to keep the competitive product he benchmarked as a working tool until our own replacement product was ready, because the competitor's product was better than the tools he had been using. Such experiences challenge the engineers to build better products.

Trade-off Analysis The computer-based VCA tool allowed precise numerical comparisons to be made on demand. Many alternatives, ranging from the most probable plan, through minor variations, to wild "what-if" scenarios, could be analyzed. The graphical displays allowed the trade-offs between alternatives to be understood at a glance. Low customer-impact branches of the hierarchy could be identified and ignored during the period when basic directions were being established, thus simplifying the design process. Figure 6 is a representation of a VCA display, annotated to clarify how the charts are to be read. In particular, the importance of an item is indicated by the angle of the vector representing it—the more important the item, the nearer the angle is to vertical. The length of a vector shows how well the item is realized in a given plan—the better the realization, the longer the vector. Therefore, long vertical vectors represent important items that are implemented well, and short horizontal vectors represent unimportant items that are not implemented well.

Refine Your Product with Customers

In addition to the techniques already described to bring customer input into the design of TeamLinks for Macintosh, we used four cycles of prototyping to confirm and refine our designs. In preparation for the third cycle, we conducted artifact walk-throughs with each customer partner as described earlier. The walk-through information enabled us to simulate real processes during the final prototype cycle, thus putting our products to an ultimate test. The four cycles are shown in Table 1.



Provide computer support for my work Control cost of using computers	Node 154	TeamLinks Mac Weight	 Bench	 Score	Competitor 1 Weight	 Bench	 Score	Competitor2 Weight	 Bench	 Score	Benchmark Weight	 Bench	 Score
3 1 Minimize overhead	156		72%	2.5		68%	2.4		77%	2.7		100%	3.5
3 1 4 Environment must be predictable and stable	180	5.0	54%	2.7	5.0	49%	2.5	5.●	45%	2.2	5.0	100%	5.0
3 1 1 Easy to do business with vendor	158	2.6	89%	2.3	2.6	152%	3.9	2.6	131%	3.4	2.6	100%	2.6
3 1 3 Minimize System Management Costs	172	2.5	101%	2.5	2.5	75%	1.9	2.5	56%	1.4	2.5	100%	2.5
3 1 2 Customization	167	2.1	108%	2.3	2.1	42%	0.9	2.1	208%	4.4	2.1	100%	2.1

KEY TO ITEMS:
Strong realization of important item
Vector Info
Go to Vector
Composite weights
Poor realization of unimportant item
TeamLinks Mac
Competitor 1
Competitor 2
Benchmark Customer Desires

Figure 6 *Representation of a VCA Vector Display*

Table 1 The TeamLinks Prototyping Cycles

Cycle	Content	Presentation	Data Collection
1	User interface facade	Macintosh Powerbook	One-on-one contextual interviews
2	User interface and limited functionality	Client software only	Sample tasks (scenarios), user diaries, and phone calls
3	Usable workflow, filing, and basic mail	Client and server software	Customer forms and work tasks, user diaries, and phone calls
4	Full functionality	Client and server software	Daily use, visits by team, and phone calls

What We Learned

Significant changes in functionality and the user interface were made based on user reaction to the prototypes. This section discusses these changes.

Unlearning Things We Thought We Knew

Throughout this paper, we focus on three main themes: (1) find out what your customer needs, (2) design your product based on what you learn, and (3) refine your product with customers.

The previous section of the paper discussed tools and techniques that we used to achieve these goals. Before actively gathering data, we developed a set of assumptions about our customer's needs and preferences for working. On subsequent visits we discovered that some of our assumptions were flawed and that we needed to change our original plans to better satisfy customer demand. In this section, we describe our initial assumptions, discoveries made throughout the data-gathering process, and new designs derived from our discoveries. Table 2 lists a comparison of our original and revised designs.

Lesson One

Our initial assumption was that customers need an information manager to navigate and to view file cabinets. TeamLinks for Windows provided an information manager to assist Windows users in viewing, naming, and navigating the ALL-IN-1 IOS and DEC MAILworks file cabinets. The file cabinet is a logical container based upon the physical metaphor of a filing cabinet. It enforces a hierarchical relationship, providing drawers that contain only folders and folders that contain only documents. The file cabinets represent the central storage areas for all objects within the TeamLinks environment.

To parallel the TeamLinks for Windows environment, the team proposed an information manager for the Macintosh product. Figure 7 shows the proposed information manager window. Users would be presented with a single, world view of the file cabinets through the information manager. This proposal adds an additional document management layer on top of the native document management. The team planned to display the information in a manner as similar as possible to the Macintosh desktop display.

However, our customers stressed: "Document management should look and work like the Mac." The Macintosh desktop presents a single, world

Table 2 Comparison of Original and Revised Designs

Original Design	Discovery	Revised Design
Mail		
Develop new X.400 TeamLinks mail client for Macintosh.	"Build one mail client and do it right."	Leverage existing X.400 mail client and focus on developing mail-enabled workflow applications.
Workflow		
Develop information manager application that contains routing services.	"Help us utilize our available desktop resources." "Build a 'real' Mac product."	Develop independent components that work well with existing Macintosh applications.
Filing		
Develop information manager application, in addition to Mac file system.	"Document management should look and work like a Mac."	Provide access to ALL-IN-1 IOS file cabinet as an extension of the Macintosh file system.

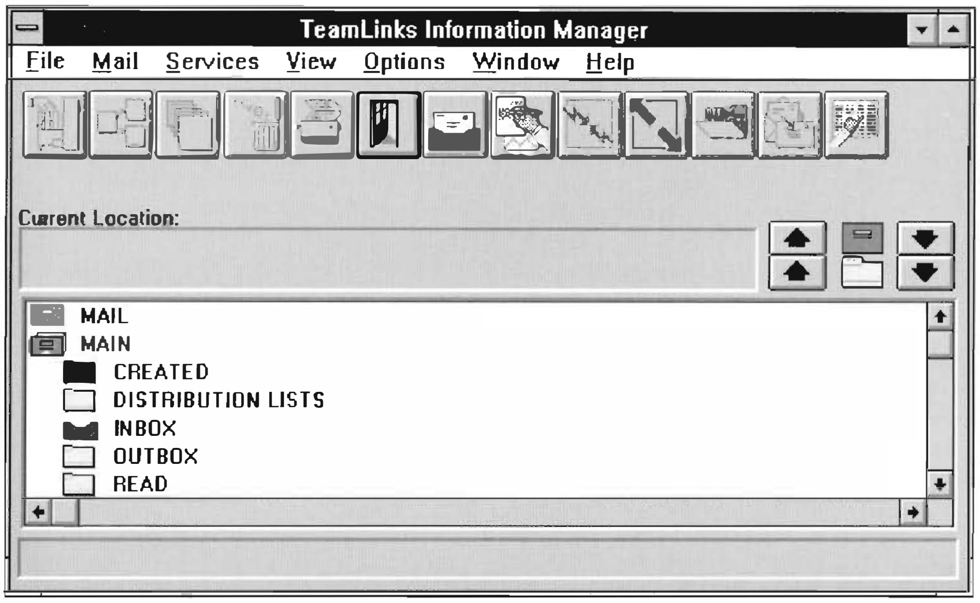

TeamLinks Information Manager
File Mail Services View Options Window Help
Current Location:
MAIL
MAIN
CREATED
DISTRIBUTION LISTS
INBOX
OUTBOX
READ

Figure 7 TeamLinks for Windows Information Manager

view to the users. They do not want a replacement. Our partners urged us to support document views and navigation that is native. After attending the Apple Developers Conference, the project leader also concluded that we would build a noncompetitive application if we followed our proposed plans.

The team decided *not* to build an integrated information manager. The revised design in Figure 8 shows how users can access the remote ALL-IN-1 IOS file cabinet as they do remote network volumes. In this approach, the ALL-IN-1 IOS file cabinet becomes an extension to the file system. This paradigm builds upon the Macintosh user's prior knowledge, making the interface comfortable and familiar.

Lesson Two

Our initial assumption was that we should follow the TeamLinks for Windows lead and create one tightly integrated application. Given the TeamLinks for Windows working model, the team proposed to develop a similar application for the Macintosh platform. Original plans detailed a large, integrated application. The information manager window would provide the central world view of the file cabinet. This window would have its own set of menus and a tool bar. All other services would be available through the information manager menus and tool bar. Mail messages, workflow packages, and other documents would be stored in file cabinet folders. Users would open these objects by double-clicking to invoke the appropriate editor.

Each service would be represented by its own window with unique menus and a tool bar. Services would include mail, workflow, address book, directory lookup, and distribution list editing.

Rather than enhancing the existing X.400 mail client, DEC MAILworks for Macintosh, the team planned to create a new mail client for the TeamLinks product. This decision would have resulted in two competing mail clients.

However, our customers stressed: "Help us utilize our available desktop resources." Digital's office products need to work with existing Macintosh applications. Customers want to use their existing word processing, graphics, and other business applications while working with our office applications. The customers emphasized that TeamLinks components must work well together.

Throughout our interviews we heard: "Build a real Mac product." Our customers stressed that our Macintosh office products must look and feel like Macintosh applications as well as adhere to the Apple Human Interface Guidelines. They encouraged us to take advantage of color, direct manipulation, and point-and-click paradigms. In following these standards, we enable users to transfer their skills from one application to another, thus reducing training costs.

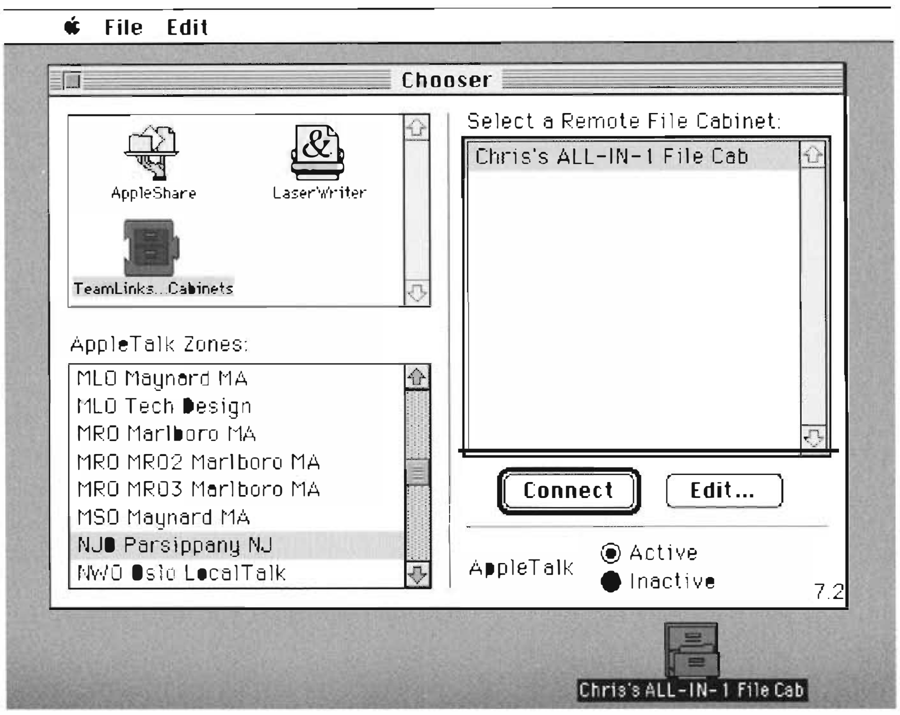


Figure 8 New Design for ALL-IN-1 IOS File Cabinet Access

We also heard: "Build one mail client and do it well." Customers want consistency across our applications. If two Digital office products provide X.400 mail support on the Macintosh platform, each should present the same user interface. This practice will help reduce customer costs by eliminating additional user training. From Digital's perspective, it makes good business sense to take advantage of existing products and resources where appropriate. Our customers cautioned against developing a new X.400 mail client for the TeamLinks product when DEC MAILworks for Macintosh already exists. They encouraged us to direct resources toward developing a single, strategic mail application that is simple to use, X.400 compliant, reliable, and available for the popular desktop computers. They mentioned mail-enabled applications, such as workflow, conferencing, and time management.

The team decided to take advantage of existing components. Rather than build a new mail client, the TeamLinks and DEC MAILworks for Macintosh project teams collaborated to enhance the existing DEC MAILworks client and provide workflow support. The TeamLinks team focused on developing the workflow component that would assist users with routing forms and documents for review and approval. As a result, the TeamLinks design migrated from a large, integrated application to components that work well together and allow users to exchange information that they have created with other popular Macintosh applications. Depending upon specific needs, customers can purchase a mail-only package, a workflow package, or a comprehensive package with mail, workflow, remote ALL-IN-1 IOS file cabinet access, and conferencing applications. Throughout development, the team refined designs, adhered to Macintosh guidelines where possible, used color to add value, and implemented point-and-click paradigms.

Lesson Three

Our initial assumption was that time management is important, but we still have time before missing the opportunity to implement this feature. Although time management was viewed as an important product requirement, the team did not fully appreciate the consequences of not implementing a time management solution. Due to limited resources, the team relied on another internal group to deliver these services. If a time management product were to become available before the TeamLinks release date, it might be integrated into the package.

However, our customers stressed: "Help me manage my time." Customers often described their struggle in trying to schedule a meeting with a group of people and quickly followed this description with a request for time management support. People spend a great deal of time trying to manage their calendars. Two of our four partners rated time management support as their top priority. People want to browse one another's calendars, get assistance in finding common meeting times, and schedule resources and events across their organization or company.

One partner stated that they would not be able to migrate their ALL-IN-1 IOS users to TeamLinks for Macintosh until a time management solution was in place. VCA data indicated that if TeamLinks for Macintosh had an integrated time management model, the product would be in better competitive standing.

An office industry consultant told us that we had only six months to release an integrated time management module. If we delayed any longer, we would miss the opportunity.

The team had been considering third-party time management providers, but negotiations had stalled. The team decided to reemphasize negotiations. A contract was signed within a short time.

Lesson Four

Our initial assumption was that we would port TeamLinks for Windows to the Macintosh platform and Mac users would like the results. We originally planned to port the TeamLinks for Windows application first and then retrofit a Macintosh user interface. The team proposed an initial design that contained a rich set of functions identical to those in TeamLinks for Windows but gave little thought to what Macintosh users really wanted from a groupware office application. The importance of simplicity and ease of use was not clear to all team members.

However, our customers stressed: "I don't learn new functions unless I see clear value to my work." "[The] most valuable tool is the one you [already] know how to use." "Less is better." "All I want to do is create mail and read it." "Build a real Mac product."

People use tools and applications to simplify work tasks. Tools should support existing work rather than create new work. People use tools if they add value; otherwise, they quickly abandon them. Customers want simple, elegant solutions.

Porting TeamLinks for Windows to the Macintosh platform would not succeed even if a user interface that resembled an actual Macintosh user interface were provided. Macintosh users easily spot and freely reject a ported Windows application. Vendors who have ported Windows applications to the Macintosh platform have failed to gain product acceptance.

The team decided to adopt simplicity as a theme. Although mail and workflow add value, they must be simple to use. We decided to take advantage of our users' previous knowledge of electronic mail and the postal mail metaphor in the design of our workflow package. The team first concentrated on designing the most frequently used functions and then on refining them.

Our VCA results indicated that we had an opportunity in the workflow area but that the window of opportunity was quickly closing. To complete our designs and develop customer-specific templates for prototyping, we needed to learn more about our customers' business processes. We used artifact walk-throughs to study three workflow examples: a manufacturing procurement request, a pharmaceutical regulatory submission, and a banking credit approval.

Rather than port the Windows application, the team created a new design utilizing user interface prototyping tools. We adhered to Macintosh guidelines, incorporating standard system fonts, point-and-click selection, standard text selection routines, standard menus and accelerators, consistent button placement, and dialog layout.

Discovering Delighters

Through the discovery process, several of our initial assumptions proved to be inaccurate or misguided. As a result, the team changed plans to better satisfy customer requirements. We learned from the experience and adapted appropriately. The team also discovered that certain product attributes delighted customers.

Button Bar Surprisingly, the button bar or tool bar within the TeamLinks components is a delighter among customers. The buttons provide point-and-click access to frequently used mail and workflow functions, reducing menu navigation and recall of keyboard accelerators. Colorful icons indicate button function. Context-sensitive help is also available as users pass the mouse pointer over buttons in the bar.

Workflow Automation Data from Contextual Inquiries, artifact walk-throughs, and VCA revealed that business process reengineering and automation is an emerging opportunity within the office automation market. Today, businesses lose time and money tracking materials through approval life cycles. Tools that support workflow automation can potentially yield substantial savings for a corporation. In some industries, trimming one hour from a process can save millions of dollars.

One customer expressed his interest in workflow support as follows: "It will mostly save everyone's time which is now wasted in tracking down who has the material and who still needs to sign it. It should speed up things, because it doesn't have to physically be sent from office to office (sometimes even different states) for approval. I would think it could save time at year end for summary reports."

The development team capitalized on this information, focusing the corporate office strategy on developing leadership workflow tools. Rather than provide a set of "me too" features, the team decided to concentrate on a specific customer problem and provide a simple, well-done solution. The TeamLinks Routing product is the outcome of these efforts, and the group intends to focus the marketing message on its tracking capabilities. Six months later, leading competitors are now hastening to announce workflow product offerings.

Refinement during Prototype Review Our VCA results indicated that customers place great value in ease of use. Items from the benefit hierarchy such as "Make the product usable—match the way I work," "Make the UI consistent within itself," and "[Make a] product [that] adds value to my work" were all rated as highly important by our customer partners. Users are specifically interested in minimal keystrokes, consistent interfaces and functions across components, point-and-click paradigms, adherence to Macintosh user interface standards, and short-cut keys.

The team focused on satisfying these requirements within the TeamLinks components. We employed a design methodology that involved users throughout the development life cycle, allowing users to see product improvements on a monthly basis. During early prototyping, the team conducted one-on-one sessions with users to study concept learning and ease of use. Feedback from these sessions was used to progressively change the design. Subsequent testing revealed that the design modifications improved ease of use. A summary of specific design changes follows.

Redesign of Main Window for TeamLinks Routing A user receives new packages for review and approval in the mail in-box folder. To view the package, the user double-clicks on the package in the in-box folder, opening a window. The original screen design for the TeamLinks Routing package window appears in Figure 9.

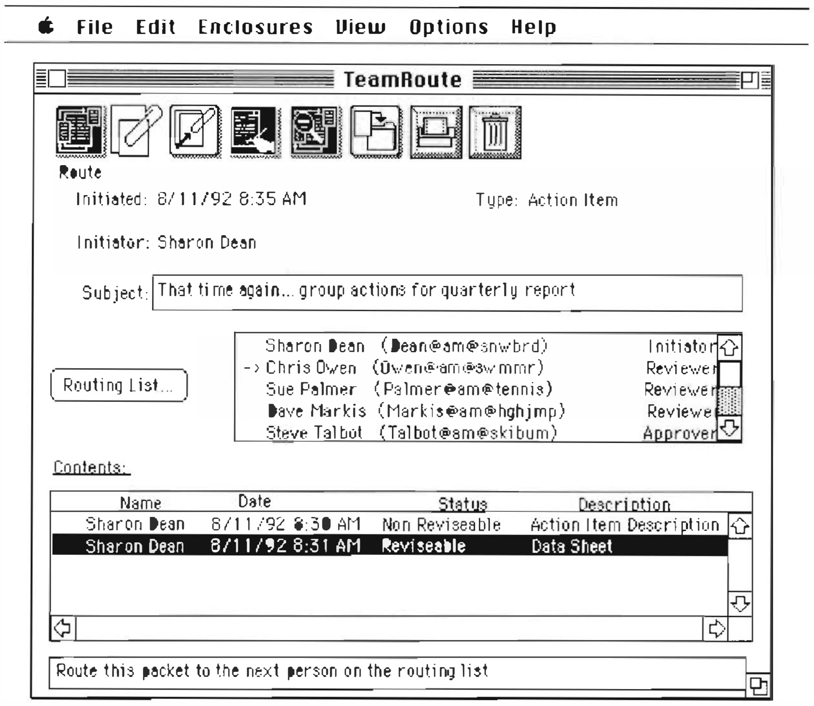
Prototype testing demonstrated that users had difficulty focusing on important information in this window. The button bar immediately caught their attention, and their eyes were then drawn to the distinctive "Routing List..." button and the corresponding list of names. Several users overlooked the list of attachments at the bottom of the window. Many users were unable to locate their role instructions, which outlined their specific tasks. Finally, several users commented that important information, such as, What do I have to do with this? When do I have to respond? and What's my role? was not visible on the main screen.

Users had difficulty understanding that the window represented a package that contained several attachments and signatures. Users were familiar with mail messages. They easily understood the concept of message attachments and the postal metaphor as it relates to electronic mail. They associated a workflow package with a special type of mail message that needed approval, yet the package window did not resemble the familiar message window.

Users overwhelmingly liked the button bar, because frequently used functions were more accessible and visible.

After going through several design iterations, the package window now appears as shown in Figure 10. The team applied the mail metaphor to workflow, rearranging some of the information to create distinct header and attachment areas as seen with mail messages. The header contains Initiator (From), Initiated (Date), To, and Subject fields. Additionally, we added a Role field to the header in response to user requests. Text labels are displayed in a bold font to improve readability and to help users focus their attention.

We simplified the window by removing noncritical information. For example, although the data in the routing list is important to users, they do not require this information in the main window, as long as it is available with a single mouse click. Therefore, we added an Edit/View-Routing-List button on the left-hand side of the tool bar. Users are also able to quickly view the routing list by double-clicking on the To field. In addition, we



Initiated: 8/11/92 8:35 AM

Type: Action Item

Initiator: Sharon Dean

Subject: That time again... group actions for quarterly report

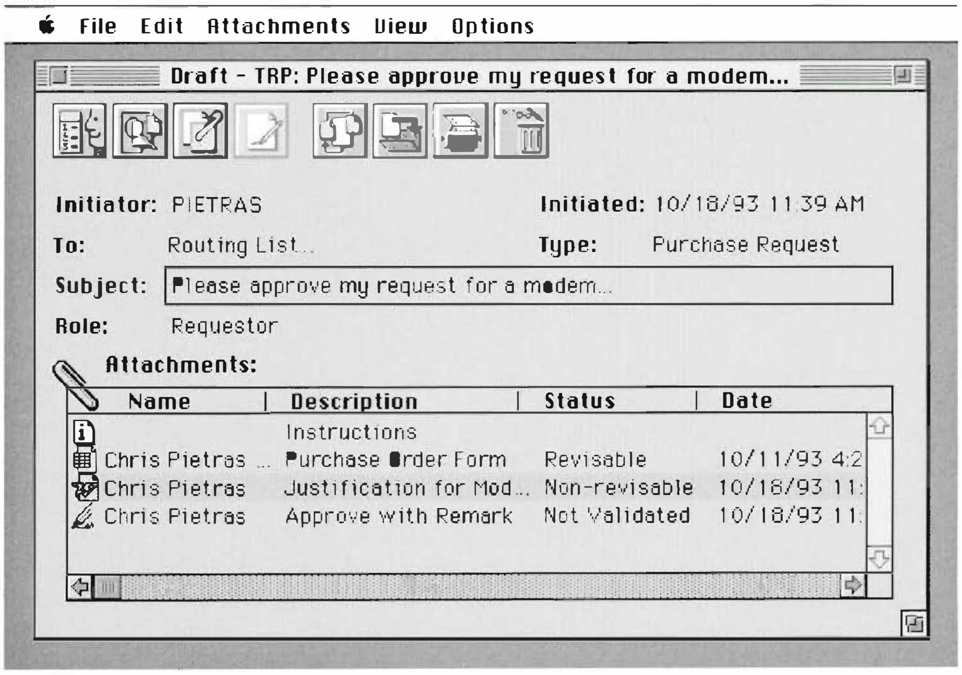
Sharon Dean	(Dean@am@snwbrd)	Initiator
-> Chris Owen	(Owen@am@swmmr)	Reviewer
Sue Palmer	(Palmer@am@tennis)	Reviewer
Dave Markis	(Markis@am@hghjmp)	Reviewer
Steve Talbot	(Talbot@am@skibum)	Approver

Contents:

Name	Date	Status	Description
Sharon Dean	8/11/92 8:30 AM	Non Reviseable	Action Item Description
Sharon Dean	8/11/92 8:31 AM	Reviseable	Data Sheet

Route this packet to the next person on the routing list

Figure 9 Original TeamLinks Routing Design


File Edit Attachments View Options
Draft - TRP: Please approve my request for a modem...

Initiator: PIETRAS

Initiated: 10/18/93 11:39 AM

To: Routing List...

Type: Purchase Request

Subject: Please approve my request for a modem...

Role: Requestor

Attachments:

Name	Description	Status	Date
	Instructions		
Chris Pietras ...	Purchase Order Form	Revisable	10/11/93 4:2
Chris Pietras	Justification for Mod...	Non-revisable	10/18/93 11:
Chris Pietras	Approve with Remark	Not Validated	10/18/93 11:

Figure 10 New TeamLinks Routing Design

removed the Routing List button, which needlessly distracted users.

The graphic designer created smaller buttons and used subtle shades of gray to create a three-dimensional look. Shading was used to invite users to press the buttons. Icons were designed to be understandable in international settings. Below the header, shading was used to define the attachments area, and a paper clip icon was added to reinforce the metaphor.

To address the difficulty users had in locating role instructions, we placed them in the attachments list. If instructions are present, they always appear as the first attachment and are denoted by a distinct document icon. Users simply double-click on the list entry to find out what they need to do with the package.

In subsequent evaluations with the prototype, customers commented: "I think it's pretty good. Once you get into it, it's pretty easy to use, pretty logical." "I was already somewhat familiar with it because I saw base-level one. It was pretty easy coming back to it. Just from using it the first time, it became familiar. I had some problems with the last one [base-level one], and I think you've solved a lot of the problems with this one [base-level two]." "Anyone familiar with a Mac shouldn't have a problem."

In designing the package window to look more like a mail message, we enabled users to transfer their mail knowledge to workflow. The concept of creating a package could be related to the concept of creating a mail message, namely, addressing the workflow package, attaching documents to the package, and typing in a subject. These changes help to reduce the need for user training.

By simplifying the main window, we enabled users to focus on important information, i.e., their role instructions and the attached work materials. Providing icon buttons for frequently used functions helps to minimize keystrokes and save time.

Terminology Review The original TeamLinks Routing product used a series of technical terms in the title bars of package windows to identify packages and states. These terms were not very meaningful to users. The original terms are listed in column one of Table 3.

Team members working on the Windows and Macintosh platforms agreed to review terminology with the goal of reaching consensus on simple terms that users could immediately identify. The team reflected on the traditional terminology for routing paper packages to develop the new terminology. The new terms are listed in column two of Table 3.

By using terms that reflect the paper process, users can immediately identify packages they receive and understand the appropriate actions to take. The terms Template, Original, Carbon Copy, and Routing Copy describe both package type and status in simple, familiar terms rather than in technical terms. The package name is placed in the title bar of the package window and is readily visible to the user. The revised terms help to minimize new learning and reduce frustration. Consistent use of terminology across platforms allows users to speak in common terms with colleagues using alternate desktop systems.

Focus on the Package The team made a concerted effort to focus on all components of the TeamLinks Office package: mail, workflow, filing, and conferencing. As discussed earlier, the process of iterative design yielded excellent results with TeamLinks Routing. Studies of prototypes demonstrated that the use of buttons, color, larger fonts and professional graphics, the mail metaphor, and

Table 3 TeamLinks Workflow Terminology

Original Title Bar	Revised Title Bar
TeamRoute – Template	Template – <document title>
TeamRoute – (Master, Routing)	Original – <document title>
TeamRoute – (Master, Completed)	Completed Original – <document title>
TeamRoute – (Master, Unsent)	Draft – <document title>
TeamRoute – (Master, Sent)	Original – <document title>
TeamRoute – (Routing Copy, Pending)	Routing Copy – <document title>
TeamRoute – (Routing Copy, Sent)	Carbon Copy – <document title>
TeamRoute – (Carbon Copy, Read)	Carbon Copy – <document title>
TeamRoute – (Tracking Report, Read)	Latest Copy – <document title>

adherence to Macintosh standards all contributed to ease of use and acceptance of the TeamLinks Routing product.

VCA results indicated that our customers viewed consistency across components as essential to minimizing training and increasing accessibility. Given this information, our goal was to produce a family of products with a consistent look and feel. The team spent six weeks working on mail enhancements, modifying the screens to be more consistent with TeamLinks Routing. For example, the graphic designer created more meaningful icons for the buttons, adding color to reinforce metaphors and make the buttons more distinct from one another. The team agreed on consistent button placement across components, moving all buttons to the top of mail windows. Similar font styles and sizes were used across components to increase readability. Figure 11 shows the original mail file cabinet window. Figure 12 shows the same window with the enhancements just mentioned.

In addition to focusing on consistency across user interfaces for mail, workflow, filing, and conferencing, the team employed the same graphic for the on-screen "About" boxes and for the packaging and documentation cover designs.

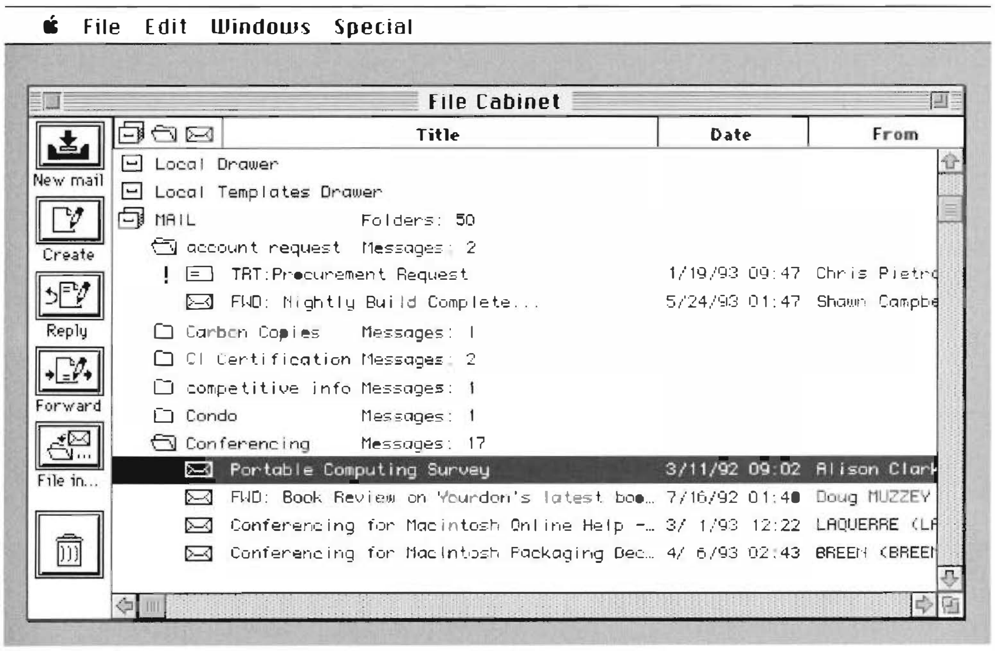
Consistency across product components and with other Macintosh applications received rave reviews from customers: "I liked the buttons across the top real well. Real nice." "The fact that it's consistent with other Mac applications is the best news." "Support for point-and-click—you did a good job here."

By creating a similar look and feel across components, the team reduced customer training needs by increasing the transfer of learning. Employing the same graphics for all components created a recognizable product identity for the TeamLinks family.

Filing

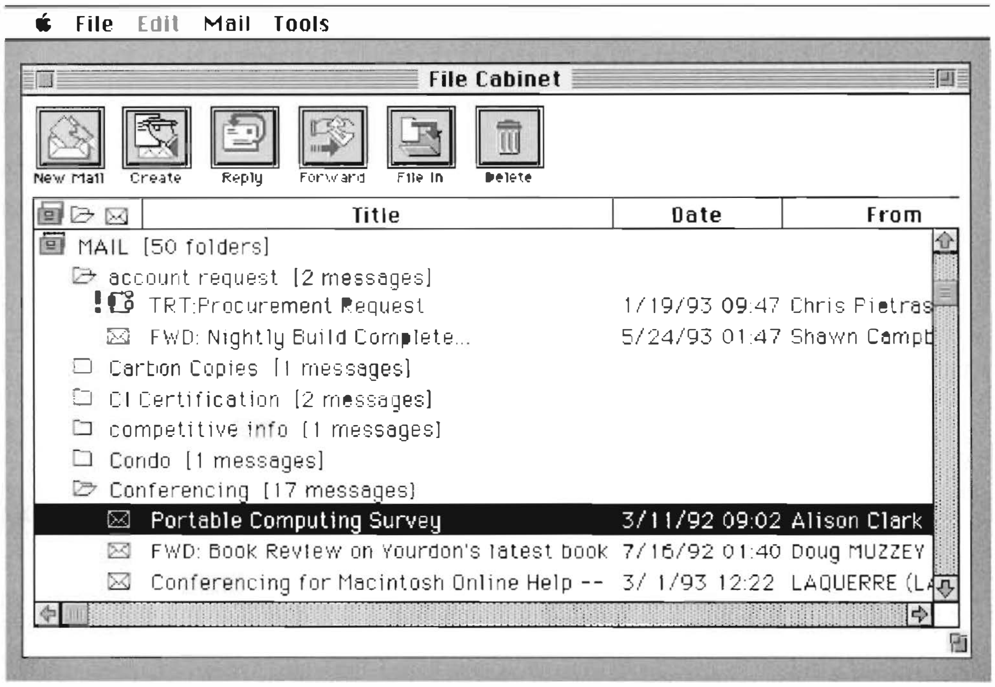
The original design to access the remote ALL-IN-1 IOS file cabinet on the Macintosh replicated the TeamLinks for Windows information manager. The VCA process demonstrated that this design would not be competitive nor would it satisfy customer needs.

The team developed a more viable solution by visualizing the ALL-IN-1 IOS file cabinet as an extension of the Macintosh file system. Team members developed a TeamLinks file cabinet extension. Users connect to the ALL-IN-1 IOS file cabinet through the chooser window. Once a user is connected, a volume, visually represented by a file cabinet icon, appears on the user's desktop. The user double-clicks on the file cabinet volume to view the



Title	Date	From
Local Drawer		
Local Templates Drawer		
MAIL Folders: 50		
account request Messages: 2		
! TRT:Procurement Request	1/19/93 09:47	Chris Pietro
FWD: Nightly Build Complete...	5/24/93 01:47	Shawn Campbe
Carbon Copies Messages: 1		
CI Certification Messages: 2		
competitive info Messages: 1		
Condo Messages: 1		
Conferencing Messages: 17		
Portable Computing Survey	3/11/92 09:02	Alison Clark
FWD: Book Review on Yourdon's latest boo...	7/16/92 01:40	Doug MUZZEY
Conferencing for Macintosh Online Help -...	3/ 1/93 12:22	LAQUERRE (LA
Conferencing for Macintosh Packaging Dec...	4/ 6/93 02:43	BREEN (BREEN

Figure 11 Original TeamLinks Mail Design



```
Title                                              Date            From
MAIL [50 folders]
  account request [2 messages]
    ! TRT:Procurement Request                      1/19/93 09:47   Chris Pietras
    FWD: Nightly Build Complete...                 5/24/93 01:47   Shawn Campb
  Carbon Copies [1 messages]
  CI Certification [2 messages]
  competitive info [1 messages]
  Condo [1 messages]
  Conferencing [17 messages]
    Portable Computing Survey                      3/11/92 09:02   Alison Clark
    FWD: Book Review on Yourdon's latest book      7/16/92 01:40   Doug MUZZEY
    Conferencing for Macintosh Online Help --      3/ 1/93 12:22   LAQUERRE (L
```

Figure 12 New TeamLinks Mail Design

contents in a new window. ALL-IN-1 IOS drawers and folders are visually depicted as their real-world counterparts, as seen in Figure 13. Users can manipulate files in a familiar fashion.

By using the standard Macintosh user interface to manipulate drawers, folders, and documents in the ALL-IN-1 IOS file cabinet, users do not need to learn a new paradigm. This approach minimizes new learning, increases accessibility and ease of use, and adds value. This design is compatible with the future Apple Open Collaborative Environment (AOCE) and will create a better return on investment for the program team.

Conclusions

The success or failure of any product can normally be attributed to the product's initial plans and the implementation of those plans. For this project, one can evaluate the development strategy against the initial project goals and against the customer needs.

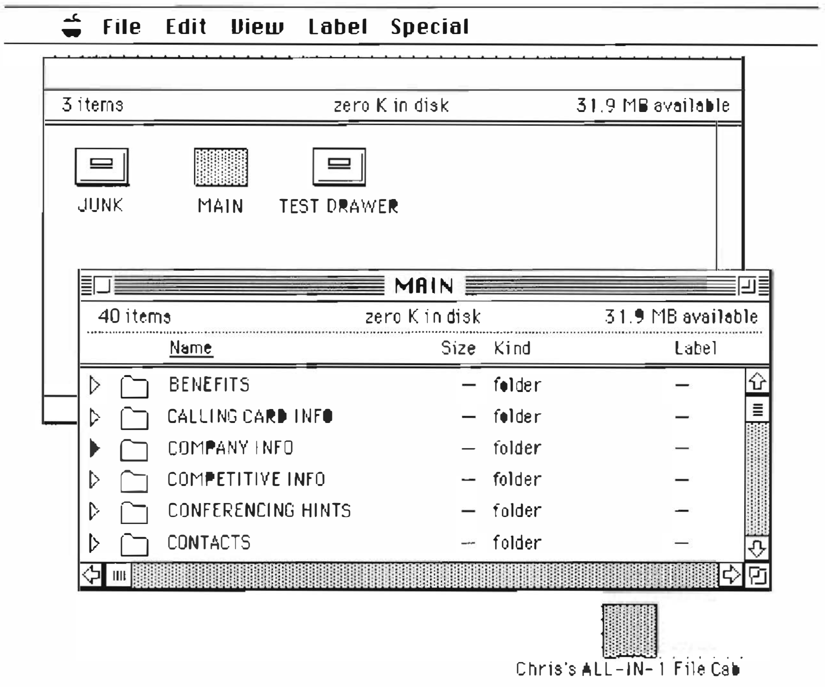
The development strategy satisfied the program's goals. The initial version of the product was delivered in less than a year of development time and with minimal resources. By-products of the development strategy allowed the team to take additional "informed" risks (seven months into the project, the team received additional responsibility for delivering the mail client), to deliver three separate products with minimal resources, and to better engage and motivate the development team through consistency of purpose.

As for the customers, they say it best in their own words:

Major government contractors: "I thoroughly enjoyed testing the product. I am definitely going to buy it—our company is committed to TeamLinks...." "Excellent adherence to Mac Interface."

Major manufacturing companies: "Simple enough to use and it works." "I'd say yes [in response to a question regarding whether they would purchase the product], it ties in well with ALL-IN-1 and meets the needs."

Major pharmaceutical companies: "Logical enough to use without the need to read documentation." "We're very excited and encouraged by these changes. Looks like a Winner!!!!" One customer stated publicly in *ComputerWorld* that TeamLinks/DEC MAILworks is their standard.


File Edit View Label Special
3 items zero K in disk 31.9 MB available
JUNK
MAIN
TEST DRAWER
MAIN
Chris's ALL-IN-1 File Cab

```
40 items            zero K in disk         31.9 MB available
      Name                         Size  Kind        Label
▷  BENEFITS                          —   folder       —
▷  CALLING CARD INFO                 —   folder       —
▶  COMPANY INFO                      —   folder       —
▷  COMPETITIVE INFO                  —   folder       —
▷  CONFERENCING HINTS                —   folder       —
▷  CONTACTS                          —   folder       —
```

Figure 13 Browsing the ALL-IN-1 IOS File Cabinet

Selected government agencies: "Really like mail; like the graphic UI, color, bit buttons, the file cabinet...." "Easy to use." "I love this! Our whole branch will want this." "It is exactly what I've imagined and desired for months." "They [customer's users] are going crazy over it. They love it!"

Acknowledgments

Many people were involved in the development of TeamLinks for Macintosh from its inception to its shipment. The authors would like to acknowledge the following contributors: Dave Brown, Dave Burns, Gary Floyd, George Gates, Sabrina Prentiss, Janna Rhodes, Charles Robbins, David Stutson, John Wise, Nam Hoang, and Eunice Zachry (account and support managers); Jennifer Dutton, Nina Eppes, Peter Laquerre, Terry Sherlock, Ricky Marks, Barb Mathers (documentation); Paul Clark, Debbie Christopher, Beth Doucette, Jim Emmond, Mark Grinnell, Steve Hain, Dean Jahns, John Lanoue, Bruce Miller, Stan Neumann, John Quimby, Tom Rogers, Larry Tyler, and Steve Zuckerman (engineering); Peggy Doucet, Mike Pfeiffer, and Beverly Schultz (management); Robert Lehmenkuler, Steve Fink, and Steve Martin (marketing); Phil Gabree and Meg Lustig (product management); Keith Brown, Tina Boisvert, Rick Palmer, Tim Sagear, and Tony Troppito (quality assurance engineering); and Peter Mierswa, for leading the team to develop customer-focused products.

Special thanks to our customers, without whose involvement none of this would have been possible.

References

1. D. Ziemer and P. Maycock, "A Framework for Strategic Analysis," *Long Range Planning,* vol. 6, no. 2 (1973): 6-17.

2. P. Naur and B. Randall, eds., "Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee," North Atlantic Treaty Organization, 1969.

3. J. Womack, D. Jones, and D. Roos, *The Machine That Changed the World,* ISBN 0-89256-320-8 (New York: Rawson Associates, 1990).

4. J. Gould, S. Boies, and C. Lewis, "Making Usable, Useful, Productivity-enhancing Computer Applications," *Communications of the ACM,* vol. 34, no. 1 (January 1991): 75-85.

5. J. Grudin, "Systematic Sources of Suboptimal Interface Design in Large Product Development Organizations," *Human Computer Interaction,* vol. 6, no. 2 (1991): 147-196.

6. T. Gilb, *Principles of Software Engineering Management* (Reading, MA: Addison-Wesley, 1988).

7. T. Ohno, *The Toyota Production System* (Cambridge: Productivity Press, 1988).

8. F. Halasz and T. Moran, "Analogy Considered Harmful," *Human Factors in Computer Systems Proceedings* (March 1982): 383-386.

9. S. Pepper, *World Hypotheses* (Los Angeles: University of California Press, 1966).

10. G. Lakoff and J. Johnson, *Metaphors We Live By* (Chicago: University of Chicago Press, 1980).

11. Subcommittee on Investigations and Oversight, Committee on Science, Space, and Technology, *Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation* (Washington, D.C.: Government Printing Office, September 1989).

12. J. Wilson and D. Rosenberg, "Rapid Prototyping," *Handbook of Human-Computer Interaction* (New York: North-Holland, 1988): 859-873.

13. J. Carroll and R. Campbell, "Artifacts as Psychological Theories," *Behavior and Information Technology,* vol 8. (1989): 247-256.

14. P. Ehn, *Work-Oriented Design of Computer Artifacts* (Stockholm: Arbetslivscentrum, 1988).

15. K. Holtzblatt and S. Jones, "Contextual Inquiry: Principles and Practice," Technical Report DEC-TR 729 (Maynard, MA: Digital Equipment Corporation, October 1990).

16. D. Wixon and S. Jones, "Usability for Fun and Profit," *Human Computer Interface Design: Success Cases Emerging Methods and Real World Context* (San Mateo, CA: Morgan Kaufman, Spring 1994, forthcoming).

17. J. Gilmore, Jr., "A Quantitative Comparative Analysis Technique for Benchmarking Product Functionality and Customer Requirements," *Eleventh International Conference on Decision Support Systems* (Providence, RI: Institute of Management Sciences, June 1991).

John A. Hrones, Jr.
Benjamin C. Jedrey, Jr.
Driss Zaaf

Defining Global Requirements with Distributed QFD

Obtaining valid data on customer needs and translating it into optimum product functionality is always a challenge, but especially so when the customers are geographically, culturally, and functionally diverse. Digital's Corporate Telecommunications Software Engineering (CTSE) used groupware techniques supported by the distributed use of Quality Function Deployment (QFD) to identify product features that meet customer needs. By linking engineers, customers, and product personnel from across the globe, CTSE redesigned the QFD model to optimize the use of local and global groups in defining product requirements. During one year, three software products, including Automatic Callback version 2.1, were defined using the Distributed Quality Function Deployment (DQFD) technique. Lessons learned from each interactive session were applied to continuously refine the approach to improving process. The critical follow-up steps after the DQFD ultimately determine the success or failure of the effort.

The Challenge of Global Requirements

Corporate Telecommunications is responsible for managing Digital's worldwide telecommunications resources including voice, video, and data networks. The engineering organization within Corporate Telecommunications develops tools, applications, and solutions to optimize the use of telecommunications services. Developing the right product for a customer depends largely on the accuracy of the requirements defined, which in turn depends on the approach used to gather information about the customer's needs. Traditionally in Digital's Corporate Telecommunications Software Engineering (CTSE), product managers have obtained customer requirements from various geographies by using electronic mail or electronic conferencing. This method was deficient in the delivery of a customer-focused product in several ways.

- Input did not come from all the corporate geographies that used the product.
- CTSE had no direct contact with the customer.
- No data was available on the importance of customer requirements.
- There was no clear correlation between product features and customer needs.

This paper discusses the approach taken by CTSE to improve the process used to define customer needs and product features worldwide.

Commitment to Improving the Process of Defining Requirements

In January 1992, CTSE made a commitment to utilize Total Quality Management (TQM) as the foundation for the development and maintenance of their products. As part of this commitment, CTSE began a set of initiatives to increase customer and user satisfaction with Digital's worldwide telecommunications products and services.

CTSE customers are from three internal geographies: the United States, Europe, and the Asia/Pacific and Americas (APA) (formerly General International Area [GIA]). Each area has its own business needs and practices. Product development must ensure that technical solutions meet the common needs of each group. CTSE recognized that the creation of successful products is based on the quality of the requirements against which these products are created. Consequently, CTSE mandated the use of the Quality Function Deployment process for all scheduled projects.

Quality Function Deployment

Quality Function Deployment (QFD) is a structured approach for proactive planning. QFD provides product planners with a process that translates customer needs into prioritized product features. This method emphasizes the use of quantitative techniques to evaluate various product features based on the impact each has on providing benefits to the customer.

QFD has been used extensively as a product planning tool for companies both in the U.S. and in Japan. Digital, Hewlett-Packard, AT&T, Ford, and Toyota are but a few of the companies that have successfully applied the QFD process to ensure that they are building products that meet customer needs.

As practiced at Digital, the QFD process begins by assembling a cross-functional team that includes customers, customer experts who have timely data on customer needs, and technology experts who know the product capabilities and the competition. The team gathers for a concentrated and focused meeting, usually two or more days in duration.

Team activities during the QFD include

- Brainstorming. Attendees state as many customer needs and product features as they can and document each need or feature without regard to merit.
- Affinitizing. The team associates and categorizes the customer needs and product features into appropriate groupings.
- Value setting through consensus. The team evaluates customer needs according to various attributes, such as customer value, goals, and improvement targets, and assigns a weight to each need.
- Correlation analysis. The team correlates the needs with the features to determine which features impact which needs and to what extent.

Throughout the QFD, a chart called the "House of Quality" (see Figure 1) graphically displays the work of the team. The customer needs become the rows of the House of Quality, and the features become the columns. The House of Quality allows you to view directly the relationship between any customer need and product feature.

The final result of the QFD is a prioritized list of features, each with an associated numeric sum of weights. This list is often displayed as a Pareto chart, which is a bar graph of the total weights in left-to-right descending order. Figure 2 is an example of such a chart.

The Distributed QFD Concept

Traditionally, the QFD process is conducted with all participants in one physical location, thus allowing constant personal interaction. This scheme works well when participants are not widely scattered; however, Digital develops most of its products for the global marketplace. Busy schedules and the high cost of travel prevent all QFD participants from gathering in one location at the same time. The challenge was to overcome the one-location issue and utilize the QFD process in a modified manner to get people in various locations working together.

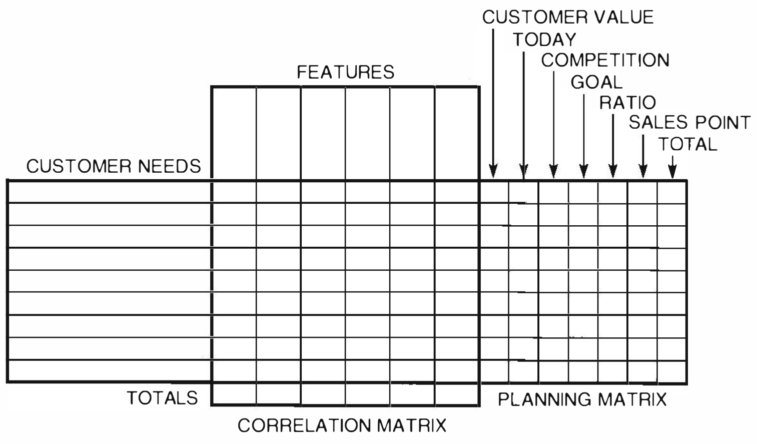


Figure 1 House of Quality Chart

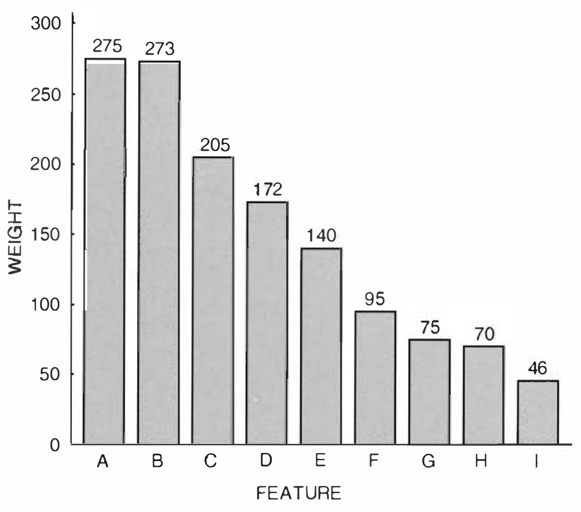


Figure 2 Prioritized QFD Product Feature Weights

CTSE calls the practice of running a QFD with involvement from multiple physical locations a Distributed QFD (DQFD).

Instead of conducting a regular QFD session at one site, the DQFD session is conducted simultaneously at the two or more sites where the participants are located. Every site has a facilitator. At each site, the DQFD participants are organized into teams connected by means of teleconferencing or videoconferencing equipment. These teams work together through the regular QFD process administered under the control of the designated "primary facilitator."

In the DQFD process, distributed team members discuss product requirements during interactive sessions moderated by facilitators. The result of these discussions is the QFD data (i.e., features, ranks, and needs) and documents (e.g., the House of Quality and the data spreadsheet). The primary facilitator and the organizer collect and process the information from all participating sites and prepare the final QFD documents, such as the product business requirements.

Before starting the session, the organizer and the primary facilitator develop the schedule and the agenda. They select the list of participants, which should include all geographies and span the involved functions such as engineering, sales, support, service, and customers. Often, a questionnaire is distributed to the participants. This questionnaire describes the customer information that is important, such as the tools they use and what unfulfilled needs they have, and should therefore be gathered and brought to the DQFD session. If solid customer data is missing in certain areas described, participants then have the opportunity to collect additional information during the weeks leading up to the DQFD. The best data comes directly from the customer while the customer is actively involved in the activity that the product or service will support. Digital has fostered a technique called Contextual Inquiry, in which the product developers visit the customer's workplace and observe and interview various users while they are engaged in their normal work activities. This technique yields timely and detailed data that often is not forthcoming in surveys, problem reports, and other passive approaches to data gathering.

In addition to the important issues of cultural differences, business relationships, and working environments, the time zones of participating sites are a major consideration when developing the schedule logistics for the DQFD. The DQFD process usually takes two or three sessions (working days). Therefore, while developing the DQFD workflow and schedule, the DQFD organizer and the primary facilitator must review the QFD process with respect to site requirements/time zones and determine the activities that best suit the participating sites.

The DQFD Model

Figure 3 portrays the basic steps of the DQFD model. Though similar in appearance to a typical QFD, the DQFD differs in the areas of logistics and training of participants, and in the order and manner in which the actual QFD sessions are conducted. The DQFD model uses videoconferencing and teleconferencing for the overview meeting and throughout the three-day DQFD itself.

Preparation

Preparation is a key element of a successful DQFD. Some important parts of the preparation are

- Planning. The primary facilitator and the organizer determine the goals and feasibility of the DQFD, the most appropriate participants, and the logistics that will work best.
- Training the team. A short (one-half day) tutorial in the basics gives the participants sufficient background in the process to contribute effectively.
- Gathering customer data. The need for accurate, complete, and current customer data as input for the DQFD cannot be overstated. Many

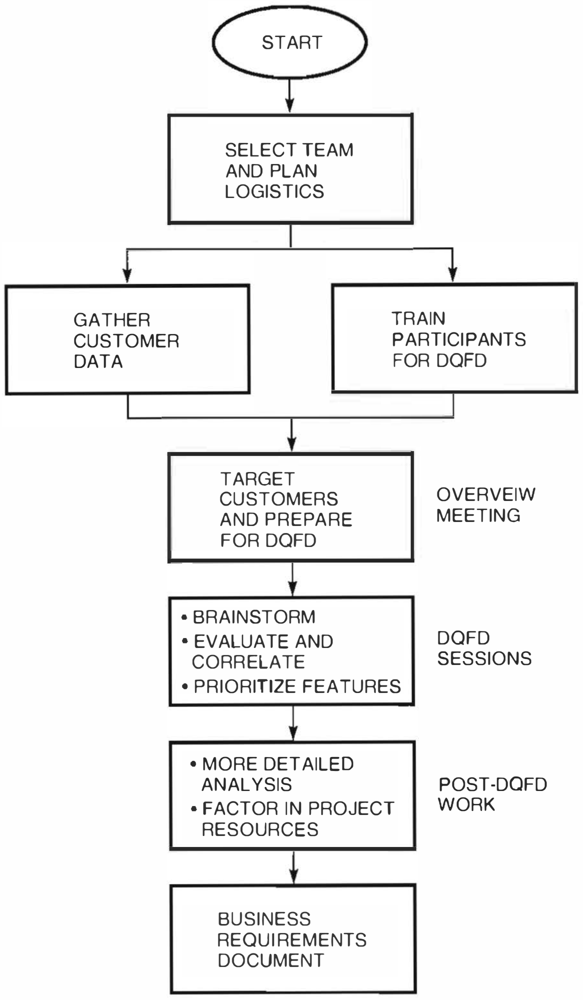

START
SELECT TEAM AND PLAN LOGISTICS
GATHER CUSTOMER DATA
TRAIN PARTICIPANTS FOR DQFD
TARGET CUSTOMERS AND PREPARE FOR DQFD
OVERVEIW MEETING
• BRAINSTORM
• EVALUATE AND CORRELATE
• PRIORITIZE FEATURES
DQFD SESSIONS
• MORE DETAILED ANALYSIS
• FACTOR IN PROJECT RESOURCES
POST-DQFD WORK
BUSINESS REQUIREMENTS DOCUMENT

Figure 3 Logistics of the Distributed QFD

techniques are useful for collecting data, including surveys, interviews, problem reports, suggestions, and free-form interview.

Overview Meeting

The overview meeting serves several main purposes. This meeting

- Helps the participants from the various sites to get to know one another.
- Provides participants with an understanding of the DQFD process and their roles in the process.
- Gives the planners an opportunity to summarize the project at hand and the issues that the DQFD is intended to address.
- Allows the team to decide who the customers are for the product or service and, furthermore, which customer is to be considered "primary" for the purposes of the DQFD. Distinguishing the primary customer can help avoid conflicts in the development of the House of Quality.
- Informs the participants about the preparation required.
- Answers questions about the logistics and mechanics of the forthcoming DQFD meeting.

The two options for handling the overview meeting in Distributed QFDs are videoconferencing and teleconferencing. CTSE prefers videoconferencing for several reasons.

- Participants from the various sites who will be working together can see one another, possibly for the first time. The visual image thus created will enhance communication during the DQFD meeting.
- Participants gain an understanding of the working styles of the facilitators at each site, which helps to move the process along.
- The visual aspects of the meeting help promote the feeling of "teamness," which fosters cooperation in the subsequent activities.

QFD Meeting

In the western Europe-eastern U.S. DQFD model, the QFD meeting spans three days. The major sites involved in the CTSE meeting described in this section were Valbonne, France, and Littleton, Massachusetts. A six-hour time difference exists between the two locations, so we scheduled the mutual meetings for mornings in the U.S., i.e., 8:00 A.M. to 12:00 M. (noon) eastern standard time (EST), and afternoons in Europe, i.e., 1400 to 1800 coordinated universal time (UTC) (known as Greenwich mean time).

Although undoubtedly inconvenient for some participants, DQFDs are possible in locations where the time difference is greater than six hours. During an earlier DQFD, one team member resided in Australia and worked with the rest of the team from 10:00 P.M. to 2:00 A.M. his time. A better approach is to schedule the DQFD over six days with overlapping sessions of two hours, as described in the section Observations about the DQFD Model.

Figure 4 shows a design of the western Europe-eastern U.S. DQFD model, as managed by the U.S. Note that the two sites work together for four hours each day. Working in overlap for just one half of each workday provided the following advantages:

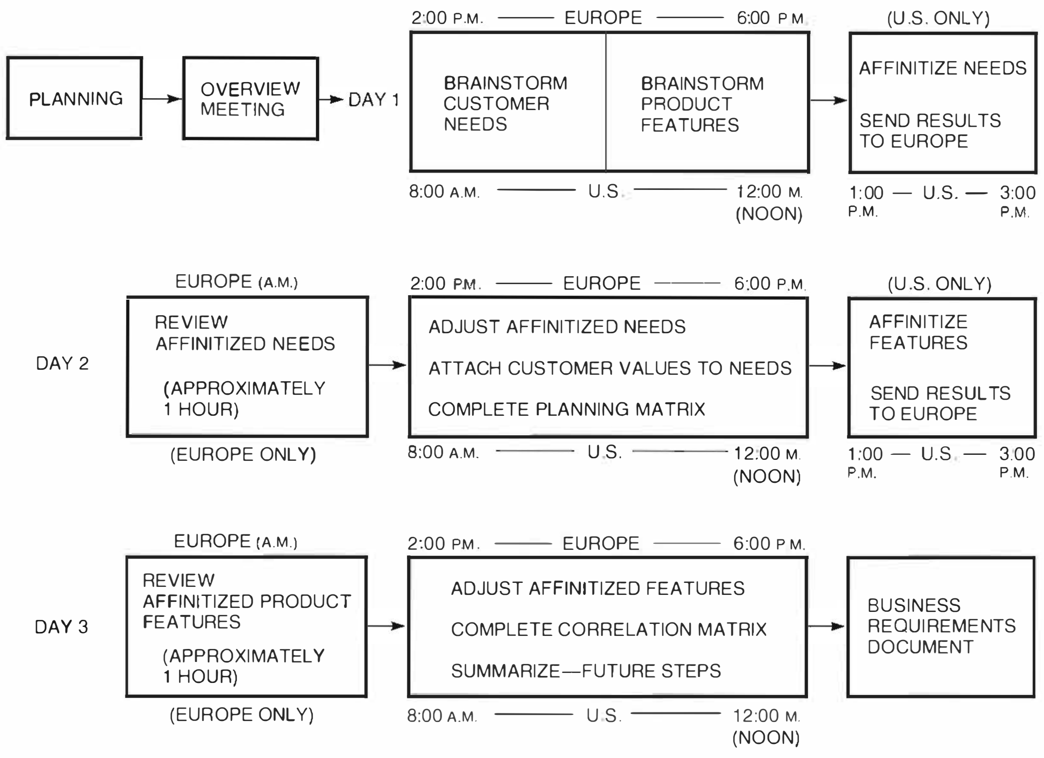

PLANNING
OVERVIEW MEETING
DAY 1
2:00 P.M. — EUROPE — 6:00 P.M.
BRAINSTORM CUSTOMER NEEDS
BRAINSTORM PRODUCT FEATURES
8:00 A.M. — U.S. — 12:00 M. (NOON)
(U.S. ONLY)
AFFINITIZE NEEDS
SEND RESULTS TO EUROPE
1:00 P.M. — U.S. — 3:00 P.M.
DAY 2
EUROPE (A.M.)
REVIEW AFFINITIZED NEEDS
(APPROXIMATELY 1 HOUR)
(EUROPE ONLY)
ADJUST AFFINITIZED NEEDS
ATTACH CUSTOMER VALUES TO NEEDS
COMPLETE PLANNING MATRIX
AFFINITIZE FEATURES
SEND RESULTS TO EUROPE
DAY 3
REVIEW AFFINITIZED PRODUCT FEATURES
ADJUST AFFINITIZED FEATURES
COMPLETE CORRELATION MATRIX
SUMMARIZE—FUTURE STEPS
BUSINESS REQUIREMENTS DOCUMENT

Figure 4 DQFD Model, as Managed from the U.S.

- Since interactive QFDs are concentrated efforts, meeting for only four hours per day allows the team to devote their peak energy to this part of the QFD.
- The schedule allowed part of each day for those sections of the QFD that could not be performed through teleconferencing, such as affinitization, administration, and computer logging of the results. The team that is managing the particular DQFD performs most of these activities.
- Each team had time between the larger group sessions to review the work of the previous day and to make a list of issues for discussion and resolution.
- At the start of each group session, participants have a hard copy of the House of Quality with data derived from the previous day's session.

A description of each of the three days of the DQFD follows.

DQFD—Day 1

The first day of the DQFD starts in the morning for the eastern U.S. and in the afternoon for western Europe. As is typical for all Digital QFDs, the team begins by brainstorming to identify customer needs. Participants contribute ideas alternately, one from Europe and one from the U.S. Both sites record each idea and the contributor's initials on a Post-it note. Later in the QFD, the contributor may be asked to clarify the content of the Post-it note. The team also marks each note sequentially with a number for easy reference. The odd numbers represent the ideas that came from Europe, and the even numbers represent those from the U.S. Once the brainstorming session is complete, a so-called scrubbing process takes place to ensure a common understanding of the content of each Post-it note. The team examines each idea statement and rewrites it if the idea is not clearly understood by all participants. No evaluation of the idea takes place during scrubbing.

At this point, the DQFD diverges from the standard QFD, which would now move to the Planning Matrix. Extending the DQFD to four days would preserve the normal sequence of QFD. To complete our work in three days, however, we elected to follow the customer needs brainstorming session with a similar brainstorming exercise for product features, which are the columns of the House of Quality. Again, we scrub the ideas after completing the brainstorming. The initial mutual session between Europe and the U.S. is now over. We did not find that the change in sequence had an impact on the process.

While the European team goes home for dinner and to sleep, the U.S. team meets during the afternoon to affinitize the customer needs specified in the morning. Affinitizing is a free-form method of grouping like ideas together into categories that will become the rows of the House of Quality. Affinitizing is a highly interactive activity involving constant physical movement of the Post-it notes. Affinitizing would have been difficult across continents without supporting hardware, so we elected to confine this work to a single site.

After completing its afternoon session, the U.S. team sends the results of the affinitization to Europe in an electronic message. When computers are not available, information can be transferred using facsimile machines.

DQFD—Day 2

On Day 2, while the U.S. team sleeps, the European team reviews the affinitization of the customer needs and compiles a list of questions and issues. When the two teams meet during the European afternoon and the U.S. morning, they raise issues about the customer needs and negotiate to resolve the issues.

The combined group now assigns customer values to each need and enters these values in the first column of the Planning Matrix, which is on the right side of the House of Quality. Next, the group fills in each row of the Planning Matrix with corresponding values for how the customer rates our current product, how the customer rates our competition, our goal for the next product release, and a sales point that indicates the natural attractiveness of the customer need. The group can now calculate weights for each need as input to the Correlation Matrix. Once the Planning Matrix is complete, the team can add ideas to the product features and scrub them.

After the European team departs at approximately 1800 UTC, the U.S. group goes through an affinitization exercise for the product features and again sends the results to Europe.

DQFD—Day 3

During their morning of Day 3, the European team members review the product features' affinitization and compile a list of questions and issues, which are addressed with the U.S. team later that day. The major joint activity for the third day is completing the Correlation Matrix, which is at the center or "heart" of the House of Quality. For each (feature, need) pair, the teams decide how much the feature, if implemented, will contribute to satisfying the need. Each correlation is then multiplied by the weight for that need. The sum of the weights is entered at the bottom of each column.

Now, all the information is available to build a Pareto chart of prioritized features. This chart, which is the desired end product of the DQFD, provides an informed basis for future product direction. The teams do a sanity check of the chart results. If the results appear rather different than expected, the teams may review the steps that led to the results to ensure that those steps were completed accurately, and to understand what data led to the results. In some cases, accurate results lead to counterintuitive but valid conclusions.

At the conclusion of the DQFD, the teams review the issues list, assign action items as appropriate, and then enumerate the next steps. These steps may include determining the resources needed to implement various features and perhaps doing follow-on QFDs to determine more detailed information about the various features.

Observations about the DQFD Model

- In the model design just described, the U.S. team did all the affinitizing. This scenario best suited the particular circumstances, i.e., the scheduling constraints and the fact that the most experienced facilitator was located in the U.S.
- The DQFD could have been managed from Europe with all the affinitization performed there, as illustrated in Figure 5. If the European team members were to do both affinitizations, these activities would take place during their morning hours of the second and third day. Note that using this approach, the U.S. participants must begin no later than 7:00 A.M. EST in order to be ready to meet with the European team at 8:00 A.M. EST.
- A third approach would have been to have one affinitization take place in Europe and the other in the U.S., as shown in Figure 6.
- The model described in detail earlier in this section is appropriate for DQFDs between the eastern U.S. and western Europe and can be used in other instances where the time difference is six hours or less. DQFDs across locations with a

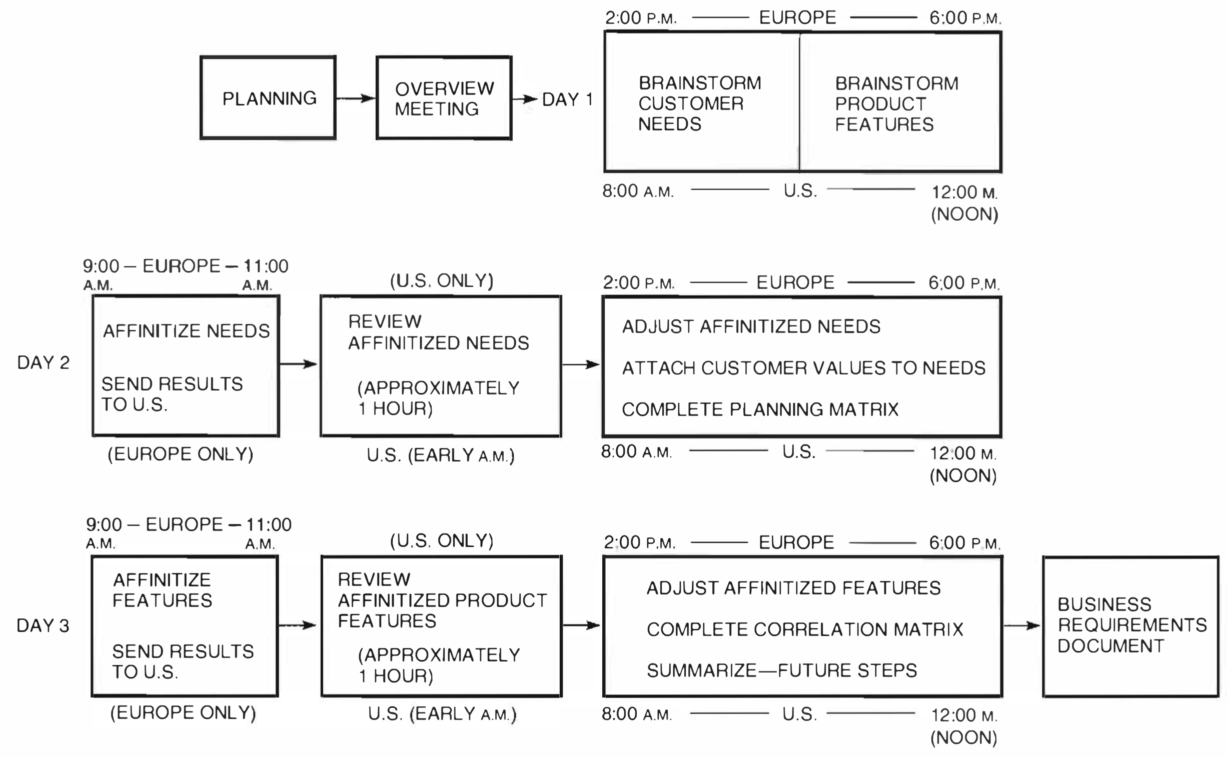

PLANNING
OVERVIEW MEETING
DAY 1
2:00 P.M. — EUROPE — 6:00 P.M.
BRAINSTORM CUSTOMER NEEDS
BRAINSTORM PRODUCT FEATURES
8:00 A.M. — U.S. — 12:00 M. (NOON)
DAY 2
9:00 A.M. — EUROPE — 11:00 A.M.
AFFINITIZE NEEDS
SEND RESULTS TO U.S.
(EUROPE ONLY)
(U.S. ONLY)
REVIEW AFFINITIZED NEEDS
(APPROXIMATELY 1 HOUR)
U.S. (EARLY A.M.)
ADJUST AFFINITIZED NEEDS
ATTACH CUSTOMER VALUES TO NEEDS
COMPLETE PLANNING MATRIX
DAY 3
AFFINITIZE FEATURES
REVIEW AFFINITIZED PRODUCT FEATURES
ADJUST AFFINITIZED FEATURES
COMPLETE CORRELATION MATRIX
SUMMARIZE—FUTURE STEPS
BUSINESS REQUIREMENTS DOCUMENT

Figure 5 DQFD Model, as Managed from Europe

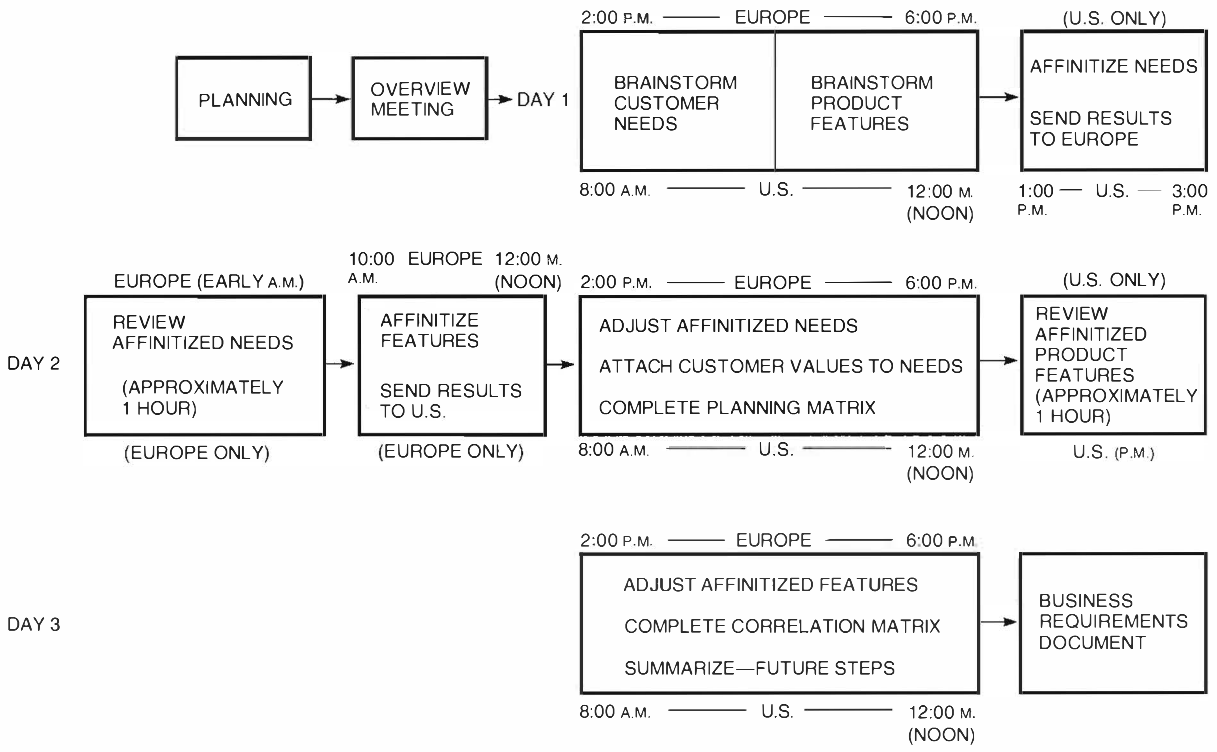
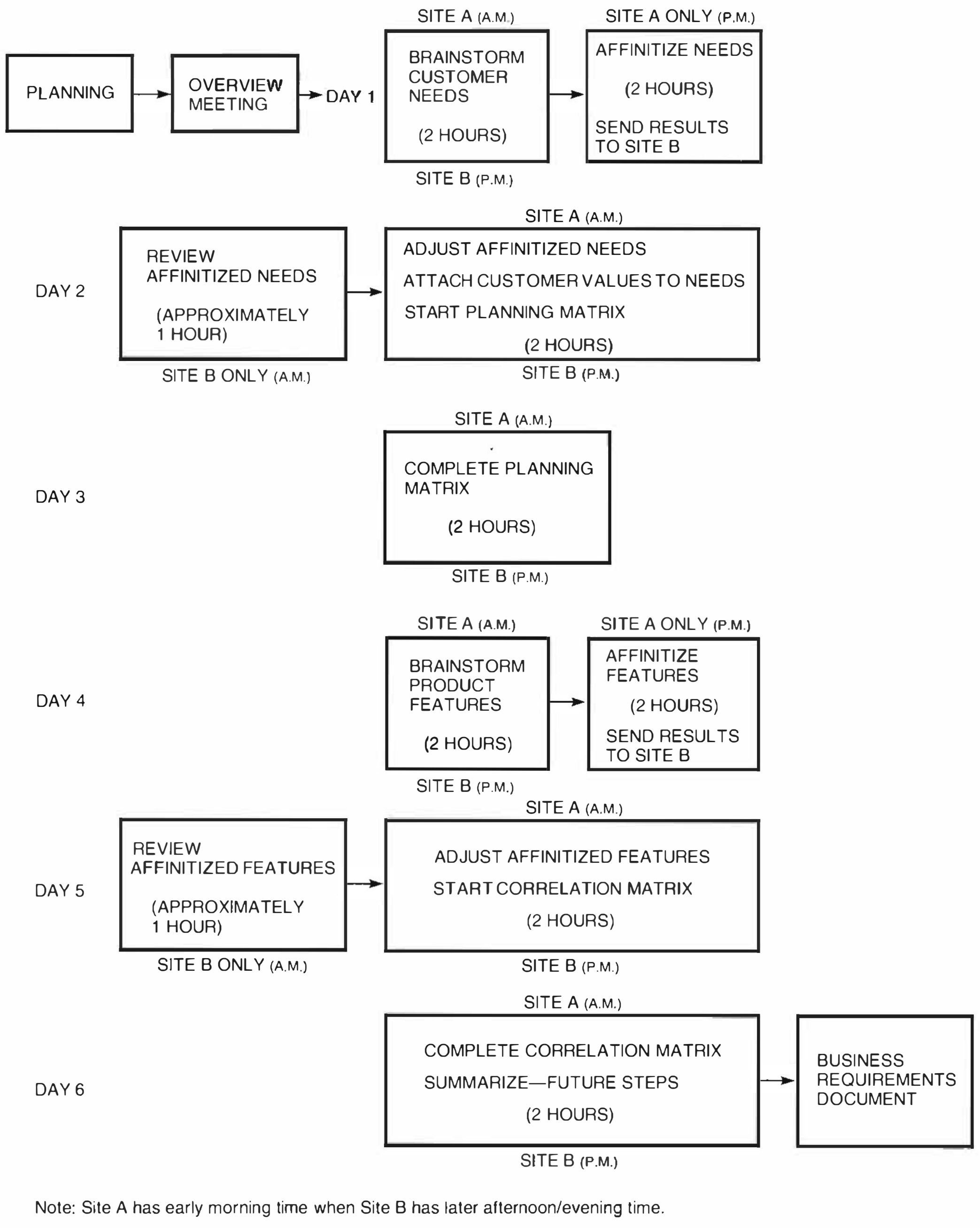

PLANNING
OVERVIEW MEETING
DAY 1
2:00 P.M. — EUROPE — 6:00 P.M.
BRAINSTORM CUSTOMER NEEDS
BRAINSTORM PRODUCT FEATURES
8:00 A.M. — U.S. — 12:00 M. (NOON)
(U.S. ONLY)
AFFINITIZE NEEDS
SEND RESULTS TO EUROPE
1:00 P.M. — U.S. — 3:00 P.M.
DAY 2
EUROPE (EARLY A.M.)
REVIEW AFFINITIZED NEEDS
(APPROXIMATELY 1 HOUR)
(EUROPE ONLY)
10:00 A.M. EUROPE 12:00 M. (NOON)
AFFINITIZE FEATURES
SEND RESULTS TO U.S.
ADJUST AFFINITIZED NEEDS
ATTACH CUSTOMER VALUES TO NEEDS
COMPLETE PLANNING MATRIX
REVIEW AFFINITIZED PRODUCT FEATURES (APPROXIMATELY 1 HOUR)
U.S. (P.M.)
DAY 3
ADJUST AFFINITIZED FEATURES
COMPLETE CORRELATION MATRIX
SUMMARIZE—FUTURE STEPS
BUSINESS REQUIREMENTS DOCUMENT

Figure 6 DQFD Model, as Managed Alternately from the U.S. and Europe

time difference greater than six hours are possible but require that the sessions be conducted over more days and the daily overlap in work be confined to a shorter time period of two hours. Even with the expanded schedule, the teams must be willing to work during the early morning and the evening hours to accommodate the time difference. Figure 7 displays the possible organization of activities for long-distance DQFDs. The DQFD is spread out over six days. Note that the team that meets in the early morning hours does the affinitization work. In order for the team at the other location to perform the affinitization, participants at that site would have to work earlier morning hours or the DQFD would take longer than six days to complete.



PLANNING
OVERVIEW MEETING
DAY 1
SITE A (A.M.)
BRAINSTORM CUSTOMER NEEDS (2 HOURS)
SITE B (P.M.)
SITE A ONLY (P.M.)
AFFINITIZE NEEDS (2 HOURS) SEND RESULTS TO SITE B
DAY 2
REVIEW AFFINITIZED NEEDS (APPROXIMATELY 1 HOUR)
SITE B ONLY (A.M.)
SITE A (A.M.)
ADJUST AFFINITIZED NEEDS
ATTACH CUSTOMER VALUES TO NEEDS
START PLANNING MATRIX
(2 HOURS)
SITE B (P.M.)
DAY 3
SITE A (A.M.)
COMPLETE PLANNING MATRIX (2 HOURS)
SITE B (P.M.)
DAY 4
SITE A (A.M.)
BRAINSTORM PRODUCT FEATURES (2 HOURS)
SITE B (P.M.)
SITE A ONLY (P.M.)
AFFINITIZE FEATURES (2 HOURS) SEND RESULTS TO SITE B
DAY 5
REVIEW AFFINITIZED FEATURES (APPROXIMATELY 1 HOUR)
SITE B ONLY (A.M.)
SITE A (A.M.)
ADJUST AFFINITIZED FEATURES
START CORRELATION MATRIX
(2 HOURS)
SITE B (P.M.)
DAY 6
SITE A (A.M.)
COMPLETE CORRELATION MATRIX
SUMMARIZE—FUTURE STEPS
(2 HOURS)
SITE B (P.M.)
BUSINESS REQUIREMENTS DOCUMENT
Note: Site A has early morning time when Site B has later afternoon/evening time.

Figure 7 DQFD Model for Sites Located Far Apart

Case Study: Automatic Callback Version 2.1

The Automatic Callback (ACB) software product provides customers, both internal and external, with remote host access and user authentication from personal computer platforms. A goal of the planned update release, ACB version 2.1, was to support the increasing number of customers who use mobile computing solutions while traveling or while otherwise remote from their home offices. A cross-functional team of product developers, planners, technical experts, and user representatives from Valbonne, France; Geneva, Switzerland; and Littleton, Massachusetts, was given the responsibility of developing the product priorities through the DQFD technique.

Planning

Several weeks before the DQFD, while in the U.S., the technical project leader and facilitator for the seven-person Valbonne contingent met with the primary facilitator of the five-person Littleton team. They planned all sessions and created ready-to-go materials, such as flip charts with the House of Quality and appropriate matrices predrawn. This preparation helped ensure that the sets of visual materials used at both sites were exactly the same.

An overview meeting took place one week prior to the DQFD using videoconferencing media. After a discussion of the process, the team discussed the customer base for the product and decided on "security managers" as the major customer.

Logistics

The DQFD took place over three days, with combined Valbonne-Littleton sessions lasting four hours, as described in the section The DQFD Model. Using teleconferencing, the two teams alternated between site-based activities, such as brainstorming, and interactive activities, such as attaching customer values, goals, and correlations. Throughout the DQFD, the project manager kept track of issues important to the project but not those that would be resolved at the DQFD meeting itself. At the end of the three days, the team associated action items with these recorded issues. The team then conducted a sanity check on the House of Quality results shown in Figure 8. The figure does not contain the detailed subcategories of features and needs that the brainstorming produced. The project team used this additional information after the DQFD to make specific detailed product decisions. The project leader assigned further work to figure in cost-benefit data and to subdivide the prioritized product features.

Post-QFD

The cross-functional alliances forged at the DQFD continued into the design and development phases of the project. Concurrent engineering was applied to deliver ACB version 2.1 on schedule within a nine-month time frame.

Lessons Learned

ACB was the first successful DQFD conducted by CTSE, in terms of the participants getting what they sought out of the process. To repeat that success, CTSE examined the factors that helped the process. At the conclusion of the Automatic Callback DQFD, CTSE conducted a short postproject review, asking what went right, what went wrong, and what might be improved. The following are some lessons learned:

1. Planning. The detailed planning done prior to the overview meeting and the DQFD eliminated potential problems and helped the process run smoothly. It is essential that the facilitators at each site understand the process as it has been modified to function in the DQFD setting. Though not an expert at QFD, the technical project leader's experience working in team situations balanced the primary facilitator's QFD expertise.

2. Automated tools. This DQFD was the first in CTSE to use the QFD/Capture tool in real time during the QFD sessions.[1] After each day's activities, the Littleton site sent a PostScript file or a facsimile of the results of that day's work to the other site. Each site entered the results on the flip charts used to display the information. The automated tool performed all the calculations and displayed the results in an easy-to-read graphical format. CTSE now sees the QFD/Capture tool or a similar tool as a necessity for a smooth-running DQFD.

3. Issues list. Maintaining an issues list accessible to all sites allowed the teams to remain focused. Topics that might sidetrack the discussion were duly noted by the project manager, and the DQFD moved ahead.

4. Videoconferencing. Most participants were impressed with the use of videoconferencing and would have preferred that the entire DQFD,

CUSTOMER NEEDS	FEATURES	1 INSTALLATIONS	2 SUPPORT NEW PROTOCOLS ENVIRONMENT	3 ENHANCED NEW END-USER FUNCTIONALITY	4 EXTEND CAPACITY	5 SECURITY ENHANCEMENTS	6 REPORTING AND BILLING	7 DISTRIBUTED MANAGEMENT TOOLS	8 CLIENTS	9 DOCUMENTATION	10 PC SECURITY	11 MANAGEMENT TOOLS	12 FIELD MARKETING AND DISTRIBUTION	13 PC FUNCTIONS	14 RENAMED PRODUCT	CUSTOMER VALUE	ACB V2.0 RATING	ACB V2.1 GOAL	IMPROVEMENT RATIO	SALES POINTS	RAW WEIGHT	
ENHANCED INTEROPERABILITY	1	1	5	1	3		1		1						1	5	2	4	2.00	1.2	12.00	1
SAVE MONEY AND PROVE IT	2	3	3	3	3	1	5	3	1			1				5	2	4	2.00	1.2	12.00	2
ENHANCED SECURITY	3	3	1	1		5			1	1	5			1		5	3	4	1.33	1.5	10.00	3
EXPAND/ENHANCE ACB CLIENT BASE	4	3	3	5	3	1	1		5	1			3	3		4	2	3	1.50	1.2	7.20	4
EASE OF ACB SERVICE ADMINISTRATION	5	5	1	3		3	1	5		3		5	3	1		4	2	3	1.50	1.2	7.20	5
BETTER DOCUMENTATION	6	1		3						5			1			3	2	3	1.50	1.0	4.50	6
FEATURES RAW WEIGHT		140.1	134.8	129.1	93.6	90.8	86.4	72.0	70.0	61.3	50.0	48.0	47.7	38.8	12.0							
FEATURE RANKING		1	2	3	4	5	6	7	8	9	10	11	12	13	14							

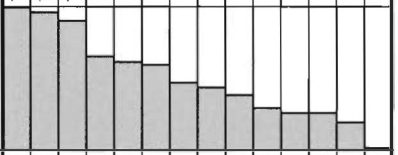


Figure 8 Automatic Callback Version 2.1 DQFD Results

not just the overview meeting, take place via videoconferencing. Something is lost when you do not see the person with whom you are talking.

5. Competitive data. Although the teams had good customer data, they did not have much information about competitive offerings. Such competitive data would have helped the teams establish more accurate weightings to the customer needs.

Recommendations

With each Distributed QFD conducted, CTSE learns more about how to improve the process and applies this knowledge to future DQFDs. The following are some CTSE recommendations for conducting successful DQFDs:

1. All participants should be educated in the QFD process, i.e., know their roles and the kinds of results to expect. Unknowledgeable participants only add to the confusion of the DQFD. CTSE has developed two half-day learning modules, "Introduction to QFD" and "Improving the Effectiveness of QFDs." Participants who complete the first module consistently contribute effectively at our DQFDs. Those that complete both modules help organize and lead the DQFD and follow-on activities.

2. Designate a primary facilitator. Although it is important to have facilitators at each site, specifying one primary facilitator, with the responsibility of designing and managing the organization of the meeting, works best. Having two "expert" facilitators who independently "know what is best" and who implement their separate ideas can have a disastrous effect because information may not be in a compatible form for the concurrent sessions.

3. Use a computerized QFD package. Having a support package is nearly essential in DQFD to provide an accurate and quick way to ensure that

each team is viewing the same information. We have used QFD/Capture and take advantage of both facsimile and electronic communication to mail updated versions of the House of Quality each day.

4. Be clear about the target customer before the DQFD sessions begin. If the product or service has an array of customers, be sure to specify a primary customer at your overview meeting. Doing so will help you decide which customer or set of customers to differentiate between, should there be a conflict during the DQFD.

5. Encourage attendance throughout all sessions. The work of the DQFD is most effective if all who participate in the overview meeting attend each day of the DQFD. People who arrive for later sessions but have not participated in earlier ones usually have difficulty contributing effectively without extensive updating and rehashing of the work of previous sessions. In addition, latecomers may have trouble adjusting to the different format of the DQFD.

6. The preparation is as important as the DQFD itself. In order for the DQFD to be fruitful, the customer information must be current and accurate. Such data helps establish goals that are competitive in the key areas about which the customer is concerned. When the list of participants is being selected, special care should be taken to ensure a diverse and comprehensive representation of customer interests and corporate functions.

Summary

The Distributed Quality Function Deployment technique provides an efficient and effective mechanism to bring together customers and multifunction representatives from across the globe into an interactive setting to exchange information and prioritize product actions in real time. The success of the DQFD rests on a sound implementation model, trained facilitators and participants, preparation and planning, and a team willing to work toward solutions through brainstorming and consensus building. Flexibility is important because adjustments must be made throughout the process to accommodate the multiple physical sites involved.

Corporate Telecommunications Software Engineering has defined and refined a set of DQFD techniques that has successfully met the goal of establishing consistent and valid product features to meet the business needs of its customers. DQFD has been adopted as a standard part of Digital's software development process.

Reference

1. *QFD/CAPTURE* (Milford, OH: International TechneGroup Incorporated, 1988).

General References

Y. Akao, *Quality Function Deployment: Integrating Customer Requirements into Product Design* (Cambridge, MA: Productivity Press, 1990).

D. Clausing, *Quality Function Deployment: Applied Systems Engineering* (Cambridge, MA: MIT Press, 1991).

L. Cohen, "Quality Function Deployment: An Application Perspective from Digital Equipment Corporation," *National Productivity Review* (Summer 1988).

J. Hauser and D. Clausing, "The House of Quality," *Harvard Business Review* (May-June 1988).

B. King, *Better Designs in Half the Time* (Methuen, MA: Goal QPC, 1987).

B. Jedrey, Jr., *Planning and Conducting Distributed Quality Function Deployment* (Littleton, MA: Digital Equipment Corporation, Corporate Telecommunications Software Engineering, 1992). This internal document is unavailable to external readers.

L. Tse, J. Bolgatz, and R. June, "Using Quality Function Deployment to Design the Colormix Widget," Technical Report DEC-TR 739 (Maynard, MA: Digital Equipment Corporation, November 1990).

G. Van Treeck and R. Thackeray, "Quality Function Deployment at Digital Equipment Corporation," *Concurrent Engineering* (January/February 1991): 14-20.

Ernesto Guerrieri
Bruce J. Taylor

DEC TP WORKcenter: A Software Process Case Study

DEC TP WORKcenter is Digital's object-based production system development environment for Application Control and Management System TP applications. Goals for the DEC TP WORKcenter project were to meet customers' requirements, to provide superior product quality, and to maintain schedule predictability. Modern software process techniques helped to achieve an appropriate balance in resolving the inevitable conflicts between project goals. A critical analysis of each software process shows its effect on the engineering team, the product, and the project schedule. Changes to the process were implemented based on the team's experience and quality metrics. Recommendations to other project teams are offered based on the conclusions drawn from the DEC TP WORKcenter project.

The DEC TP WORKcenter product is an interactive production system application development environment specifically customized for Application Control and Management System (ACMS) transaction processing (TP) applications.[1] Development of the DEC TP WORKcenter object-based development environment started in 1991 in response to requests from a number of Digital's ACMS customers. They wanted a tool that could help them to

- Perform configuration management of ACMS application components
- Track ACMS application components
- Obtain a more efficient build mechanism for ACMS applications

The product development team consisted of a team leader, an architect, six software engineers, a quality engineer, two test engineers, and two documentation writers. The average experience of the team was seven to eight years of industrial experience (with at least three members having over ten years of experience) in a wide variety of software industries, including defense-oriented developments. This breadth of experience was important in the creation and adoption of the development process.

The key goals of the project were to provide

- Customer-defined product requirements
- Compliance with the product requirements specification
- A high-quality product
- Delivery on schedule

For the customer satisfaction goal, we describe our use of Contextual Inquiry, Quality Function Deployment, conceptual modeling, and rapid prototyping. We also describe a formal requirements documentation technique to analyze requirements and guide later software phases.

For the quality goal, we describe the use of the requirements document, the interface and design review process, and the use of inspections. We mention functional testing as guided by the requirements document.

For the schedule goal, we discuss the organization of the team into working groups and the use of the requirements document to ensure coverage of a requirements matrix.

Finally, we describe several management processes for balancing conflicting goals and assessing project dependencies and risks through process metrics. From this experience, we have formulated a collection of recommendations that we feel are true not only for the DEC TP WORKcenter project but for all projects.

Theme

Every engineer on the DEC TP WORKcenter development team had experience with formal or semiformal software development processes. The positive experiences came from projects that were developed smoothly and without incident. The negative experiences stemmed from projects that ended in disaster in spite of (or because of) formal development methodologies. The entire engineering team, however, was enthusiastic about formal policies, as long as the team could be in control of the process. The team's unofficial motto was

"Use the process, but
don't let the process use you."

Throughout the development cycle, we looked for formal techniques to control various parts of our work, and then tried to adapt these techniques to the particular requirements and capabilities of our development team. In some instances, we were able to install a formal mechanism with little or no modification; but for most cases, we had to refine the mechanism, using the following steps.

1. Document the mechanism.
2. Test it on a realistic sample task.
3. Collect objective measures of how well it worked.
4. Adapt the mechanism.
5. Repeat until satisfied.

We never used complex metrics, software physics, or deep analysis; the key to any success was to keep the process simple and to continually adapt it to fit the nature of the task and the team. Once we were satisfied with the process, we tried to apply it uniformly and consistently across the product development.

Design Requirements

Because the DEC TP WORKcenter product was the result of a customer-driven process, we were faced with a number of challenges, which can be categorized into the following three areas.

- Gathering customer requirements efficiently, accurately, and objectively
- Capturing and integrating the requirements of several customers into a single, coherent specification
- Recording the requirements specification so that it could be used as a reference during design and testing phases

With the help of Digital's Software Engineering Technology Center (SETC), we focused on two techniques for gathering requirements: Quality Function Deployment and Contextual Inquiry. Furthermore, we utilized a formal requirements specification document to capture the results of these techniques. We also utilized prototypes to validate our understanding with the customers and documented this in another document, the *DEC TP WORKcenter Conceptual Model.*

Quality Function Deployment

Quality Function Deployment (QFD) is an exercise in forming consensus among team members (including customers and development partners) for identifying key requirements.[2,3] In a previous project, QFD techniques had been performed for many of the same functionalities of the DEC TP WORKcenter product. We evaluated the validity of the data and results of QFDs for that project to determine if they could be applied to the overlapping features in the DEC TP WORKcenter product. This method allowed us to take advantage of valid QFD data and results without incurring the cost of producing them.

Apart from the reuse of valid QFD results, we found QFDs to be a fairly expensive way to gather requirements. The QFD techniques involve a great deal of preparation, customer participation, and analysis. The results, however, justified the effort expended. We emerged from the QFD process with a prioritized list of requirements. For each requirement, we also identified (1) how well the current products satisfy the requirements, and (2) how well the competition satisfies the requirements.

All of these factors were expressed as numbers and could be readily ranked for importance, cost, and benefit. Once the requirements were ranked, we determined the features to be included in the product based on resources and projected market dates. These decisions were then validated by the customers who had been involved in the initial requirements gathering.

Recommendation: Reuse QFD data. Existing QFD data (either QFD input data and/or requirements resulting from the QFD) may be reused upon assessment of their validity.

Contextual Inquiry

Acting on the advice of the SETC, we used Contextual Inquiries (CIs) to gather requirements.[4,5] CIs are structured visits to selected customer sites to record exactly how the customer develops ACMS applications today, and exactly how a proposed solution could improve the customer's productivity. This technique involved a great deal of analysis and was an expensive way to gather requirements. We feel it was worth the cost because it gave us confidence in our requirements list. We were able to compare the requirements against actual customer activities to determine:

1. Those requirements on the list that would not be used by the customers
2. Those customer activities that would not be supported by the product as described in the requirements list

Both the CI and QFD techniques yielded firm, objective requirements specifications that could be compared, ranked, and further analyzed.

In retrospect, the CIs that had the most impact were the ones that were properly documented for future reference immediately after the CI visit.

Recommendation: Document Contextual Inquiry data. In order to trace information to the CI and/or reuse its data, the CI visit needs to be formally documented.

Requirements Specification

We needed an effective way to capture and combine the product requirements into a formal specification that could be used as a benchmark for development. Several engineers on the team had a background in programming for the Department of Defense and were familiar with the DoD-STD-2167A development process.[6] These engineers convinced the team that the process is beneficial if it is simplified and streamlined.

Accordingly, the team analyzed the DoD-STD-2167A Software Requirements Specification format and modified the format to the project's needs. As a result, the team produced a requirements specification document that matched the scope of the project, reflected the background of the team members, and traced the origin of the customer requirements. The final document was 40 pages of semiformal prose and has remained current for the duration of the project.

We have used the requirements document as an important data source in later development phases. During software design, we compared design features to the requirements document to eliminate unnecessary design frills and to detect requirements that were not met. We referred to the requirements specification to develop a test suite for complete testing of all product features. To ensure the use of the requirements specification, the documentation should be kept as short as possible, as concise as possible, and as descriptive as necessary.

Recommendation: Customize the requirements specification. The level of formality of the requirements specification should reflect the purpose of the document. Furthermore, it should be as short as possible, as concise as possible, and as descriptive as necessary.

Prototypes and Conceptual Model

While we were preparing the requirements specification, we also built two prototypes of the human interface for the DEC TP WORKcenter environment. The first prototype existed only on paper as a series of Motif windows that illustrated how we imagined the main functions of the DEC TP WORKcenter would operate. We showed this paper prototype to customers, asked for their feedback, and made extensive modifications based on their reactions. We repeated this process at least three times. In retrospect, it was an expensive way to refine the interface, but it gave us confidence that we were building the correct interface to our product. This paper prototype was captured in a formal document called the *DEC TP WORKcenter Conceptual Model* and would later support the *DEC TP WORKcenter Functional Specification* and the user interface design.

To demonstrate that the product was practical and to get some initial performance results, we also constructed an executable prototype of a few product functions. This activity was valuable in demonstrating feasibility, but it had two unfortunate side effects. First, it distracted the team from the design process, which caused the schedule to slip. Second, we did not have the sense to discard the prototype after it served its purpose. The engineering prototype suddenly became the first base-level code and entered the main line of development. Eventually, we had to rewrite most of the prototype code, which was a more costly procedure than starting with a clean design. The engineering prototype can be a valuable step if it has a well-defined purpose and if it is discarded when that purpose is served.

Recommendation: Restrict prototype usage. The engineering prototype can be a valuable step in product development, if it has a well-defined purpose and if it is restricted to that purpose.

Design Phase

We used several techniques during the design phase, including

- Feature-based working groups
- Electronic design notebook
- Layered approach to object-oriented design
- Detail-level design header files

The feature-based working groups allowed the team to develop the high-level design in parallel in a concentrated period of time. The output of each feature-based working group was kept in an electronic design notebook and formed the evolving high-level design. Once the high-level design was completed, the team reviewed the design to validate consistency and integrity to product requirements and between interacting or dependent product features.

A layered approach to the object model was used to describe the design of the product. The layered approach allowed for easy separation of the object-oriented design from the object-oriented features of the product. After the high-level design was completed, header files were used to define the detail design of the product.

Feature-based Working Group Technique

During the design phase, we defined the major features of the product and determined which requirements affected which feature. We then formed feature-based working groups (FBWGs) to develop the design of each feature with respect to its associated product requirements. Team members participated in the FBWG of interest to them, and a designated responsible individual (DRI) led each FBWG. Since the number of team members was less than the number of working groups, team members participated in more than one FBWG. There were approximately twice as many features as there were team members. Consequently, each team member was a DRI of approximately two FBWGs and participated as a member of approximately six other FBWGs. Once membership of the various FBWGs was established, the FBWGs met, depending upon the availability of the members. Meeting conflicts were avoided by tracking FBWG meetings on a white board.

Table 1 illustrates the team members' participation in the various FBWGs for the DEC TP WORKcenter project. The columns in Table 1 represent the various FBWGs, and the rows represent the project team members. The entries in the table indicate the role that a specific team member played in the specific FBWG. The load column indicates the overall role (number of FBWG DRI roles, number of FBWG member roles) the team member played across all FBWGs.

Dependencies or interactions between product features needed to be managed. If a team member's participation overlapped with the interacting features, that person provided a means of communicating among the associated FBWGs. Otherwise, the corresponding DRIs provided this exchange of information. Also, the project leader and the architect attempted to attend all meetings to guarantee consistency across the various FBWGs. This allowed us to resolve many issues consistently, but we would have benefited from a more formal mechanism for settling design disputes.

The FBWGs continued to a lesser extent during the detail-level design, but the issues were narrower in nature and were dealt with by the FBWG DRI and the affected component DRIs.

In conclusion, the FBWGs provided clear assignment of responsibility and guaranteed that the design was covered by more than one team member. Due to their parallel nature, the FBWGs had no adverse affect on the schedule. Unfortunately, even for small groups, the FBWG generated too much specialization of knowledge.

Recommendation: Adapt the design process. The design process should be adapted to meet the schedule and resource constraints.

Electronic Project Notebook

The minutes and draft/final design of each FBWG were recorded in an electronic project notebook. The electronic project notebook provided a means of communicating the evolving design of the product among the team members. Once entered into the notebook, the information was made available to the team. Also, the entries posted in the notebook during the day were collected and mailed electronically to the team members every night so that the team remained current on all design issues and decisions. This proved an efficient method for communicating the information to the entire team as well as for recording the information for later use.

Without a goal to produce a formal design document, the team members were not as careful in documenting their design. Furthermore, the design was dispersed over a set of notebook entries that created issues in two areas:

Table 1 Feature-based Working Group Matrix

Team Member	Load D/P	WG 1	WG 2	WG 3	WG 4	WG 5	WG 6	WG 7	WG 8	WG 9	WG 10	WG 11	WG 12	WG 13	WG 14	WG 15	WG 16	WG 17	WG 18	WG 19
Engineer 1	1/11	P			P	D	P	P		P	P	P	P		P					P
Engineer 2	2/5				D		D			P		P				P				
Engineer 3	2/8	P				P	P	D					D					P	P	P
Engineer 4	2/9	P	P			P				P	D		P		P			D	P	
Engineer 5	4/8	D	P	D		P		P	P						D	D				
Engineer 6	1/9		P		P	P	P				P	P		D		P	P			
Engineer 7	1/2																P			D
Engineer 8	3/4								D			D						P	D	
Engineer 9	2/7	P	D						P	D	P		P	P						
Document Writer 1	0/1																P			
Document Writer 2	1/3		P								P						D			

Notes:
D – Designated responsible individual for the WG
P – Participant in the WG

- Configuration management: Which notes formed the current set of design notes?
- Inspection difficulty: Which version of a design note was a source document?

The electronic project notebook was not limited to the design phase but was used to record and exchange information throughout the phases of the product development life cycle.

Recommendation: Capture project information. The electronic project notebook is an easy way to share knowledge and exchange ideas, issues, solutions, futures, etc., about a project.

Recommendation: Generate formal design specifications. Although the electronic project notebook contained the design, it is not a substitute for a formal design specification.

Layered Approach to Object-oriented Design

Since the product would be object-based, we used object-oriented design (OOD) techniques. Due to the inexperience of some team members, the distinction between abstraction levels was not always clear. To allow the team to recognize the different abstraction levels, we used different languages for the two levels of abstraction. At the product level, object-oriented terminology was used. At the product architecture level, a constrained layered model was used in which the constraints allowed a simple mapping into an object-oriented model.

The following constraints were applied to the various layers in the model.

1. Each layer provides one and only one specific type of resource.
2. Each layer provides a set of services to manipulate that resource.
3. The resource and/or its services may use other layers to provide needed resources and services.

These rules allowed the team to distinguish between the design of the product and the data model of the objects manipulated by both the product and its object-based operations. Although this layered approach to OOD was formulated to make use of the team's background, the resulting design was not a pure OOD.

Recommendation: Understand the purpose for modifying a process. Although the layered approach to OOD attempts to bridge traditional design methods to OOD methods, it should represent only a phase in a planned transition to OOD techniques.

Detail-level Design Header Files

During the detail-level design stage, we refined the various layers required to implement the resources and services to support the product features. This included determining the final interface of each layer, defining the resource controlled by the layer, and describing the functionality of the services provided by each layer.

To optimize consistency and effort, the detail-level design was represented as a C header file that provides the services of a layer implemented in a C module. Furthermore, if a module represents an object, then the header file consists of the visible operations that can be performed on the object.

The header files were placed under configuration control while issues and resolutions concerning a layer were recorded in the electronic design notebook.

Since several features required the services of a specific layer (later implemented as a C module or component), we captured the relationships in a feature/component matrix. Table 2 gives the feature/component matrix for the DEC TP WORKcenter product. The columns in Table 2 indicate the various product features, and the rows indicate the components of the product. An entry in the matrix indicates that the component implements or supports part of the product feature.

A DRI was assigned to each header file to coordinate the needs of the various features on that layer. The component DRI met with several FBWG DRIs to ascertain the needs of each feature and present

Table 2 Feature/Component Matrix

Components	Features																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1																3						
2			3	3	3	3	3	3		3		3		3								
3			3	3	3	3	3	3		3		3		3					3			
4		2	2	1+	2+	3	3	2+		3	2+	3	2				2+					D
5		3	3	3	3	3	3	3		3	3	3	3						3			D
6		2	2	2+	2+	3	3	2+		3	2+	3	2	3								D
7		2															2+					
8			2	2+	2+	3	3	2+		3	2+	3										D
9										3	2+											
10														3								
11									2+													
12									2+													
13			1														2					
14			1+	1+	2+	3	3	2+	2	3	2						2+					
15	2		1+	1+	2+	3	3	2+	2+	3	2						2+					
16																						
17														3								
18													1+				2+					
19																			1+			
20													1+									
21	2						3															
22	2+																	2+		2+		
23																				2+		
24															3				3		3	

Notes:

1	: Base Level 1
1+	: Base Levels 1 and 2
2	: Base Level 2
2+	: Base Levels 2 and 3
3	: Base Level 3
D	: Deferred

a satisfactory interface. On the other hand, each FBWG DRI needed to coordinate with several component DRIs to provide the capability for the associated feature.

Recommendation: Share information across development phases. The use of header files as part of the detail-level design provided (1) a centralized location for all interface information about a module, (2) no redundancy of interface information, and (3) an up-to-date interface in the corresponding code.

Design Reviews

The entire team reviewed the high-level design for consistency across the various product features and for integrity of the dependencies between features. Due to time constraints and the amount of design information, this review was inefficient and was not formally completed. Marathon high-level design review did not work since it was too intense and too long. We concluded that the review process must be streamlined.

The detail-level design was represented as C header files. Consequently, they were targets for code inspection.

Recommendation: Review the design in manageable pieces. Divide the high-level design into modules so that its review is manageable.

Code Inspections

Although inspections were used for the requirements document and the data model design, most of the inspections occurred during the DEC TP WORKcenter coding phase. The technique was modified to deal with time constraints and the amount of coding, and to gain the acceptance of the team on the usefulness of inspections. Basically, we defined a formal inspection and a semiformal inspection.

The formal inspections follow the guidelines as described by Fagan.[7,8] The semiformal inspections had the following restrictions:

1. Only two engineers participated in the inspection.
2. The moderator was also the reader.
3. The author was also the recorder.

The following criteria were established to decide which type of inspection would be performed.

1. Complex code was formally inspected.
2. Critical code was formally inspected.
3. Remaining code was informally inspected.

The complexity of the module was determined by computing the McCabe cyclomatic complexity of the module.[9,10] The threshold for complex code was initially set at 7 and would be periodically adjusted based on feedback on the effectiveness of the inspections. Note that the literature has usually determined 10 to be this threshold. At 7, approximately 17 percent of the code was considered complex. This may be attributed to either the tendency of modules to represent objects in the design or the use of the X Window System and Motif as the graphical user interface.

The project leader determined the critical code according to the nature of the code or intermodule dependencies in the system. This information was available from the detail-level design. One example is DEC TP WORKcenter parsers, where the flow of control is based on pattern triggers rather than on sequential execution of statements. Consequently, the DEC TP WORKcenter parsers were deemed to be complex.

All remaining code was inspected using semiformal techniques. To discourage the engineers from artificially constraining their code to be noncomplex, the project leader could randomly choose code for formal inspections (this was never needed).

As another refinement to the inspection process, we reduced and adapted the set of codes used to characterize a defect according to the type of document being inspected. This technique allowed us to accelerate the inspection and continue to capture the information of interest.

In another attempt to refine the inspection process, the recorder defined the defect codes. This accelerated the semiformal inspections but slowed the formal inspections.

Recommendation: Understand the purpose for modifying a process (revisited). Under schedule or resource constraints, consciously decide how to formally relax the inspection process and understand the consequences.

Recommendation: Choose tools to support the process. Given unbiased criteria to select the level of inspection, choose the appropriate tools to support the decision process.

Scheduling

Project scheduling played an important role in managing the project. Scheduling tools associated with personal computers (such as program evaluation and review technique [PERT], critical path method [CPM], precedence network, and resource leveling capabilities) were used to manage the schedule. Tasks were classified as either process-related or product-feature-related. The process-related tasks covered activities such as Digital's Phase Review Process or customer interactions. The product-feature-related tasks were activities directly related to the design, implementation, and testing of product features.

One distinction of the DEC TP WORKcenter product is that most of the product-feature-related schedule was determined from the feature/component matrix (see Table 2). When a specific feature was planned to be added into the product, the components supporting that feature were also scheduled to be added. The entries in the matrix in Table 2 indicate in which code base level the component implements or supports the product feature.

The engineer(s) assigned to a task submitted an estimate of the time needed to accomplish the task to the project management. If the estimates were considered unreasonable based on past engineering experiences, an in-depth analysis was performed to understand the discrepancy. These discrepancies were due to either a misunderstanding by the project management of the complexity of the task or an inefficient solution plan by the engineer to build upon existing components or processes.

Recommendation: Share information across development phases (revisited). Use requirements analysis and design information to define the schedule.

Recommendation: Get team support for the schedule. For any schedule, obtain commitment from the team.

Efficiency Factor

We also calculated an efficiency factor to account for activities that would lower the efficiency of engineers in performing their tasks. These activities included periodic mail reading, attending non-project-related meetings, sick time, jury duty, and code inspections. We revised all work estimates to reflect the engineer's efficiency factor. Initially, the efficiency factor for most of the engineers was calculated to be 60 percent. Although the efficiency factor was intended to achieve the most realistic schedule possible, it was the cause of several problems:

- The efficiency-related activities were counted twice if the engineer's estimates included these activities.
- There is an assumption that the efficiency-related activities are spread uniformly over all tasks. This is true for repetitive activities that occurred within the resolution of the tasks being estimated, but other efficiency-related activities occurred rarely (e.g., sick time) or were associated with a specific phase of the project (e.g., code inspections).

As a result, the schedules needed to be refined and adjusted frequently.

Recommendation: Understand the factors that impact the schedule. The efficiency factor attempts to capture those separate activities that were not worthwhile but impact the efficiency of other activities.

Unplanned Tasks

During the initial phase of the project, the project management recognized that schedule predictability was highly influenced by unplanned tasks. To better understand the nature of unplanned tasks, the project management participated in a Software Metrics In Action (SMIA) course offered by the SETC. The SMIA course was applied to our problem of unplanned tasks over the next phase of the project. To our surprise, we concluded that, no matter how well one plans, one always has an additional 20 to 25 percent of unplanned tasks. This included new tasks, existing tasks that took longer, and existing tasks that were completed.

Recommendation: Understand the impact of unplanned activities. No matter how well one plans, one always has an additional 20 to 25 percent of unplanned tasks. This includes new tasks, existing tasks that took longer, and existing tasks that were completed.

Milestones

The difficulties of estimating tasks and the existence of unplanned tasks would sometimes render the schedule invalid. Milestones within the project schedule allowed the team to meet the associated deadlines. Milestones also caused two events that affected the project:

- Unplanned tasks were prioritized against planned tasks, causing readjustment of milestones based on the prioritization criteria.
- Engineers became more efficient, causing the efficiency rating to be revised and allowing some of the unplanned tasks to be included without impacting the schedule.

Recommendation: Define milestones. The team works best when well-defined milestones for goals are established.

Feature "Hit List"

Toward the end of the design phase, we determined that the planned date for completion could not be met unless we reduced the functionality of the product. We created a feature "hit list" in the electronic project notebook in which we listed the candidates for elimination from the product. The feature hit list was used in a Pugh process to determine, in a structured manner and with group consensus, the features to be eliminated in order to meet the projected market date.[11]

Some of the features that we eliminated through our hit-list technique originated in the QFD process. During field test training, customer feedback indicated that some of the eliminated features were needed for a viable product. This event caused us to reevaluate and readjust the projected market date in order to include the missing features. Thus, we reaffirmed the validity of the results supporting our customer satisfaction goal.

Furthermore, the readjustment of the projected market date had high management visibility, but the utilization of the customer satisfaction processes permitted us to adequately document the rationale for and justification of the readjustment.

Recommendation: Manage and adapt the change process. When making a change that is visible to the customer and/or management, one needs (1) a formal process for defining the change, (2) consensus among the team, (3) traceability to facts supporting the original decision and its change, (4) impact analysis of change, and (5) agreement from customer and/or management.

Final Phase

In the final stages of the DEC TP WORKcenter product development, we conducted field tests at customer sites, identified defects, and determined the final changes to be made to the product.

Field Test Advocacy Program

During field test, we took a proactive approach in our relationship with the customer field test sites. Under our Field Test Advocacy Program, an engineer is assigned to monitor the progress and to resolve any issues or problems at the customer's field test site. The engineer monitors the customer's software problem reports (SPRs) in the field test SPR database to understand (or be aware of) any patterns in SPRs.

In one example, a customer raised a series of feature suggestions that were all attempts to use the DEC TP WORKcenter environment for an unsupported object type. Although the suggested features would be useful, they would not be as important if the main feature was provided. Monitoring customer SPRs provided us with an understanding of how the customer was testing and assured the customer that the engineering team understood the customer's concerns.

Recommendation: Adopt useful processes. Adopt processes in which the benefits outweigh the costs, but understand the time frame of both.

Tracking Defects and Monitoring Fixes

As the product was being developed, all (internal and external) problems were tracked using a problem tracking tool. Every problem was entered into the problem database and given a unique identifier. This allowed the engineer to associate a fix with the corresponding problem identifier. Furthermore, the problem tracking tool allowed us to monitor the defect identification and fix rate on the project. Figure 1 shows both the number of problems entered over time as well as the problems fixed over time.[12] Interesting points in the graph are the slopes, plateaus, change in slope, and vertical distance between the two lines.

The tracking tool also allowed us to verify that the priority of the fixes was consistent to the severity of the problem. Figure 2 shows the same graph for the two highest severity classes and indicates that the problems with the highest severity classes were monitored closely and fixed immediately.

Tracking the problems worked well to identify issues during the DEC TP WORKcenter product development. More analysis, however, was needed to understand trends as soon as possible.

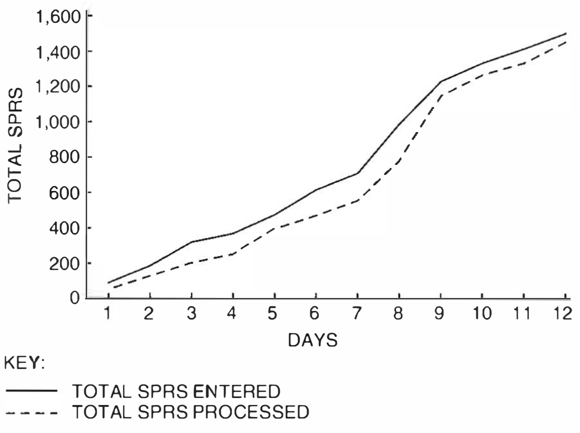


Figure 1 Trend of Total SPRs Entered and Processed

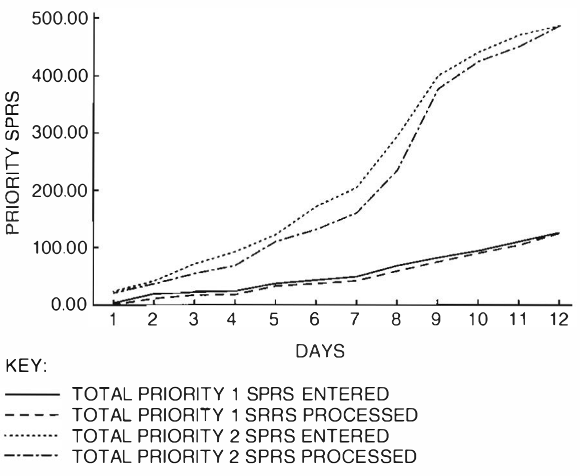


Figure 2 Trend of Priority 1 and 2 SPRs Entered and Processed

Recommendation: Adopt processes to collect valuable metrics. Understand the rationale for adopting a metric and set up a process that achieves the goal of the metric.

MUST-DO Lists

As we approached major code freeze dates, we prioritized the defects to be fixed and compared them to our MUST-DO criteria. Usually the criteria consisted of the following.

- The defect was a priority 1 or 2.
- The defect impeded testing efforts of critical functionality.
- The defect represented a regression from a previous stable version of the product.

The defects were added to the MUST-DO list if they met the criteria. This list indicated backlogs of defects that needed to be resolved prior to declaring a code freeze. Figures 3 and 4 show MUST-DO count patterns prior to reaching code freeze. The solid line (total) indicates the outstanding MUST-DO items over time.

Recommendation: Define valuable metrics (or focus on important issues). The MUST-DO list helps prioritize the tasks that require focus during a specific activity and provide well-defined goals for the team.

Product Stability

Once the product had reached feature freeze, a change control board was put in place to guarantee the stability of the product and to avoid any major regression that would impact the schedule. The board approved the inclusion of any defect fix after (1) review or inspection of the code modifications, and (2) adequate testing.

Furthermore, we monitored the defect discovery rate to determine if it was stable enough to warrant a code freeze.[12] In this case, we measured a running total of the number of MUST-DO items added over the last five days. Figures 3 and 4 show this metric. The broken line (five-day cumulative) indicates the five-day running total and measures if the changes are stabilizing.

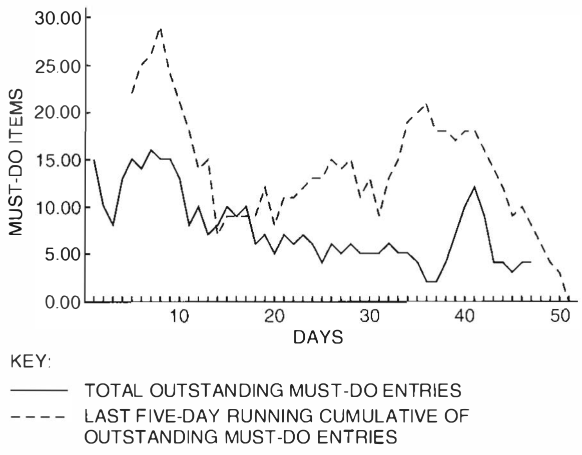


Figure 3 Trend for MUST-DO Processing and Stability Metric for Code Freeze 1

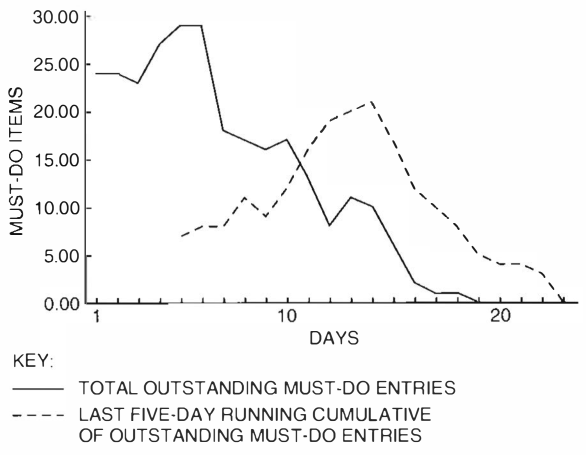


Figure 4 Trend for MUST-DO Processing and Stability Metric for Code Freeze 2

Recommendation: One can always improve. It is never too late to set up a change control board to reduce the introduction of new problems and regressions.

Conclusions

The DEC TP WORKcenter object-based development environment (version 1) was developed over approximately 24 months. During this time, we were presented with a variety of situations that could have impacted our project goals. This paper presents several of the processes that the team adopted to meet the project goals. Table 3 summarizes the recommendations based on our experiences on adopting processes to support our goals. In retrospect, we see that the project functioned smoothly when all of the following conditions were met.

- Everyone understood what development phase was in progress.
- We identified a set of processes to govern each phase.
- We adapted the process to suit the project team.
- We adapted the process to the realities of the project schedule.
- All the team members understood and accepted the process.
- We followed the process conscientiously.

In short, the entire experience of the DEC TP WORKcenter project can be summed up as:

- Software development processes should be as simple as possible.
- The team should formally adapt the processes to its own needs.
- The team should understand the consequences of modifying the process.

Although these rules of thumb do not ensure a smooth, productive project, the DEC TP WORKcenter team found them to contribute to a successful conclusion.

Our recommendations can be adopted by any project team; however, the team would benefit by taking part in a similar process of identifying its goals and supporting them with appropriate processes.

Acknowledgments

The authors would like to acknowledge the members of the DEC TP WORKcenter team, past and present, that helped define and adopt the various processes presented in this paper. Also, the detailed comments from the many reviewers were very helpful.

Table 3 Recommendations Based on the DEC TP WORKcenter Development Project

1. Reuse QFD data.
2. Document Contextual Inquiry data.
3. Customize requirements specification.
4. Restrict prototype usage.
5. Adapt the design process.
6. Capture project information.
7. Generate formal design specification.
8. Understand the purpose for modifying a process.
9. Share information across development phases.
10. Review design in manageable pieces.
11. Choose tools to support process.
12. Get team support for the schedule.
13. Understand the factors that impact the schedule.
14. Understand the impact of unplanned activities.
15. Define milestones.
16. Manage and adapt the change process.
17. Adopt useful processes.
18. Adopt processes to collect valuable metrics.
19. Define valuable metrics (or focus on important issues).
20. One can always improve.

References

1. T. Speer and M. Storm, "Digital's Transaction Processing Monitors," *Digital Technical Journal,* vol. 3, no. 1 (Winter 1991): 18-32.

2. J. Hauser and D. Clausing, "The House of Quality," *Harvard Business Review,* vol. 66, no. 3 (May-June 1988): 63-73.

3. L. Cohen, "Quality Function Deployment: An Application Perspective from Digital Equipment Corporation," *National Productivity Review,* vol. 7, no. 3 (Summer 1988): 197-208.

4. K. Holtzblatt and S. Jones, "Contextual Inquiry: Principles and Practice," Technical Report DEC-TR 729 (Maynard, MA: Digital Equipment Corporation, 1990).

5. D. Wixon, K. Holtzblatt, and S. Knox, "Contextual Design: An Emergent View of System Design," Technical Report DEC-TR 686 (Maynard, MA: Digital Equipment Corporation, 1990).

6. "Military Standard Defense System Software Development," Technical Report DoD-STD-2167A (Washington, D.C.: U.S. Department of Defense, 1988).

7. M. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal,* vol. 15, no. 3 (1976): 219-248.

8. M. Fagan, "Advances in Software Inspections," *IEEE Transactions on Software Engineering,* vol. SE-12, no. 7 (July 1986): 744-751.

9. T. McCabe, "A Software Complexity Measure," *IEEE Transactions on Software Engineering,* vol. SE-2, no. 6 (December 1976): 308-320.

10. T. McCabe and C. Butler, "Design Complexity Measurement and Testing," *Communications of the ACM,* vol. 32, no. 12 (December 1989): 1415-1425.

11. S. Pugh, *Total Design: Integrated Methods for Successful Product Engineering* (Reading, MA: Addison-Wesley, 1991).

12. R. Grady, *Practical Software Metrics for Project Management and Process Improvement* (Englewood Cliffs, NJ: Prentice-Hall, 1992).

Neil L. M. Davies
Margaret M. Dumont

SEI-based Process Improvement Efforts at Digital

The Software Engineering Institute is chartered with advancing the state-of-the-practice of software engineering to improve the quality of the systems that depend on software. Digital has based its software process improvement program on the Capability Maturity Model and Software Process Assessment developed by the SEI. As software organizations gain process maturity, they produce higher-quality products. Case studies report the experiences and learnings of two software organizations at Digital that have introduced the SEI framework and methods into their process improvement efforts.

During the late seventies and early eighties, the state-of-the-practice of software development and management at Digital improved significantly. Examples of these improvements include the following.

- Software and hardware architectures, notably the VAX VMS and the Digital Network Architectures, were developed.
- Higher-level languages (BLISS and C) were introduced into common use in systems development.
- Debuggers and language-sensitive editors were developed and used widely.
- Code management systems were introduced into widespread use.
- The phase review process for managing software projects was used extensively.

Although the complexity of software development projects has grown exponentially over the last few years, relatively few changes have occurred in the practice of developing and managing software projects. The lack of effective process management techniques impacted Digital's ability to predictably deliver quality software products that satisfy customers' expectations both in feature and time-to-market needs.

This paper describes the use of software process methods to improve the quality and predictability both in time and function of Digital's software products. Specifically, it describes the approaches of two organizations actively involved in software process improvement efforts. In addition, it presents the conclusions drawn from case studies of their process improvement programs as well as the challenges to be faced in the future.

Software Process Improvement Program

The software process improvement program at Digital is based on the framework developed by the Software Engineering Institute (SEI). The SEI is a federally funded organization chartered with advancing the state-of-the-practice of software engineering to improve the quality of the systems that depend on software.

The SEI promotes the belief that software productivity and quality gains can be achieved through a focused and sustained effort toward building a process infrastructure of effective software engineering and management practices.[1] Case studies on process programs at Hughes Aircraft and Raytheon support this premise.[2,3] Although the importance of a quality process to the end quality of the product is gaining acceptance, this idea is not prevalent within software organizations. A strong fear still exists that development of a process structure is equivalent to the creation of a bureaucracy.

We chose the SEI's framework as the basis for our process improvement efforts because its focus is specific to software organizations. A key element of improving software process is the ability to develop effective structures and the discipline to manage the process. The SEI has developed a process framework and method that deal specifically with the complexity of software practices and organizations.

SEI Capability Maturity Model

The framework, known as the Capability Maturity Model (CMM), asserts that a project is an instantiation of the organizational processes in which it was developed. Therefore, to improve a project's predictability or quality, one must improve the structure and discipline of the process (or develop the process maturity) in which the project is developed. The capability of a process to deliver a quality product predictably is determined by how well the process is defined and how consistently it is applied.

As shown in Figure 1, the CMM framework defines five levels of maturity: Initial, Repeatable, Defined, Managed, and Optimizing. Each level is a building block for the next level. To see improvements, organizations must proceed from the lowest level to the highest level. Since each level is a precondition for the next, the organization cannot skip a level. Organizations can determine their process maturity and the processes they should develop by undergoing an SEI process assessment.

SEI Process Assessment

The SEI has developed a method called the SEI process assessment to enable organizations to determine their process maturity. The assessment is used

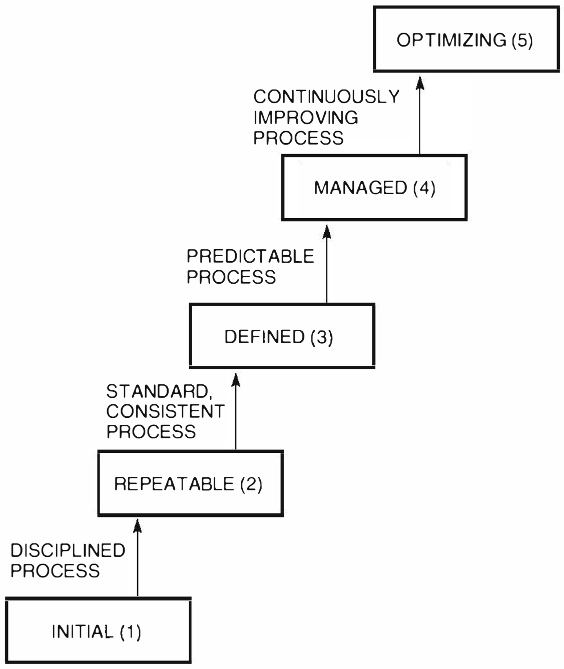


Figure 1 Five Levels of Process Maturity as Defined by the CMM

to determine process awareness in the organization and to devise an action plan for improvement of the process. The assessment involves all levels of the organization in a structured method aimed at building consensus on the primary problems the organization faces. A by-product of a well-run assessment is organizational agreement on the actions of how to address the problems. For more information on the process maturity framework and assessment, see *Managing the Software Process* by Humphrey.[4]

SEI Guidelines for Process Improvement

Once the organization decides to introduce a process improvement program based on the SEI model and method, two questions require answers: (1) What does this mean? and (2) How do we get started? Process improvement work is unique and involves a level of abstraction beyond the usual work done in software organizations. This effort must be staffed with individuals who can blend organization knowledge with process improvement techniques. Unless the organization is serious about applying adequate resources to the effort, including a substantial amount of time and commitment from management, we suggest that the effort not be undertaken. The SEI has developed guidelines on staffing a Software Engineering Process Group (SEPG).[5]

In the next two sections, we offer our different experiences in implementing SEI-based process improvement programs as case studies from which other organizations can learn. In the first case study, an organization started with a small bounded improvement and used that to launch a process improvement effort that started with an SEI assessment. In the second case study, an organization built SEI concepts into existing quality processes to gain momentum for a process improvement program based on the SEI framework and SEI assessment.

Case Study 1: Using an SEI Assessment to Initiate the Process Improvement Program

Undertaking an SEI-based process improvement effort is a huge task. The effort officially begins with an SEI assessment; however, we have found that months or years may be needed to prepare for an assessment. In our case, nine months passed from the time we began work to improve our processes until we considered an SEI assessment. Another four months was needed to complete the assessment. As our first step, we sought commitment for change within the organization. To this end, we initiated a test involving a small bounded improvement plan.

Obtaining Commitment for Change

Often there is a perception in the organization that it is easy to change. In our experience, however, it is a difficult process even when an organization wants to change. To prepare for the larger process improvement effort, we devised a small bounded improvement effort to evaluate if the organization was ready to change. The test is beneficial in two ways. First, it gives the organization experience in dealing with change. Second, it creates energy for process improvement and helps to enlist sponsors within the organization.

The first improvement was to update the code management system. The organization had recently undergone changes in organizational structure and product strategy. These changes put new requirements on the system we used to build and integrate our sources. The improvement was to choose a new source management system and to establish its use in the development and release processes within one product release.

The success of our improvement plan was measured in two ways. First, the introduction of the code management system did not impact the schedule of the release in which it was introduced. Second, during the retrospective of the release, the new code management system was viewed positively by both the release management and engineering organizations. In addition, 30 percent of the people involved in the retrospective responded that updating the code management system was the highest positive change we made to the process. As a result of this success, we proceeded to the SEI assessment and SEI-based process improvement program.

Choice of SEI Model and Method

We chose to use the CMM and SEI assessment as part of an overall effort to improve the software development environment in our organization for two major reasons.

First, the CMM provided a framework for prioritizing process improvement efforts to develop the organization's capabilities. In the months prior to adopting the CMM, we tried unsuccessfully to agree on the priority of improvement in the organization. In time, we reached the point where we agreed that use of the CMM and SEI assessment would enable us to establish priorities for improvements. The major benefit we saw was that the assessment involved all levels of the organization from senior managers to individual contributors in the prioritization and implementation of changes. In addition, we considered the cross-functional involvement to be essential to sustaining the effort.

The second major reason we chose the CMM was its focus on the software industry. In the future, we hope to be able to benefit from the programs in risk management, software education, and software measures, now being developed at the SEI.

The assessment is designed to help determine the process areas that the organization must address in order to move up the capability levels of the CMM. In our case, the assessment was led by a trained SEI facilitator and a team of people within the engineering organization. We tapped the knowledge of approximately 60 people from within the organization through questionnaires, interviews, and free-form meetings. The data collected was analyzed and developed into a findings and recommendations document that was presented to senior management. This document is the basis for process improvement work in the organization. It is required reading for new managers at the staff level.

Extensions to the Framework of the CMM The CMM has its roots in the government systems and defense-oriented areas of the software industry. It has only recently made inroads into the commercial software industry. Although it is the most complete method available for software process improvement, it makes certain assumptions about software development organizations that may not be true in the commercial sector. While implementing our software process improvement project, we found it necessary to extend the CMM.

As stated earlier, the CMM provides a set of levels that allow an organization to determine the maturity of its processes. Each level defines a set of key process areas (KPAs) required to reach that level's capability. For example, there are six KPAs at the Repeatable Level 2:

- Subcontractor management
- Software project planning
- Software project tracking and oversight
- Software configuration management
- Software quality assurance
- Requirements management

Each KPA is defined by a set of practices that cover the goals, the abilities and commitments to perform the process, the activities the organization must perform, and the mechanisms to measure and verify those activities.

The first extension we made to the CMM occurred during the assessment process. The CMM does not address resource management and development, that is, employee development, changes in the way resources are applied to new processes, and communication within the organization. These are necessary to develop the practices required to implement a KPA. For example, to develop a project plan, one must be able to negotiate effectively to share resources among interdependent projects; or, to verify that an activity is performed, feedback loops must exist in the organization's communication processes.

Our findings indicated that the areas of commitment and communication needed improvement. The CMM describes attributes for these areas in each KPA; however, it provides no guidance on the goals, activities, and abilities of commitment and communication as process areas in their own right. We have some activity in each of these areas but have not successfully developed them into an integrated plan for the organization.

The next extension to the CMM required us to implement processes from the Defined Level 3, even though we had not achieved the Repeatable Level 2. First, we needed to establish an SEPG to carry out the activities to improve the process. Second, we needed to establish guidelines and methods for a training program. Without a training program, we could not ensure that the organization would have the abilities to perform KPAs at the Repeatable Level 2. Third, we needed to define the processes used in the organization. Definition of process and training are perceived by the organization as major causes of frustration. These areas tend to embody the organization's recognized need to change and its overall resistance to change. These two areas involve problems related to understanding how other functions in the group work, developing good peer-to-peer communications, and transferring responsibilities between people.

Finally, we introduced a KPA for the definition of the software development process. The CMM is based on first providing a good management framework and then developing the engineering framework. The assumption is that, as engineers, we tend to focus first on the engineering process for improvements. In implementing process improvement, we found that we needed a process model specifically for development of software components within our overall software product process.

Turning Recommendations into Actions Our experience has shown that with organizations assessed at the Initial Level 1 of maturity, two aspects of turning recommendations into actions need to be considered. The first is the skill set of the people who develop the process improvements; the second is the framework for developing and delivering process improvements to the organization. We found that the individuals and teams who deliver process improvement must possess project management skills and organizational development skills.

Project management skills are essential because the environment does not otherwise foster the discipline or ability to create a set of plans from a set of recommendations. We structured the process improvement work into a project with a set of goals, objectives, and deliverables. The high-level goals and objectives were integrated into a set of long-range milestones. Currently, each person working on process improvement has a set of project plans that describe individual deliverables based on the project goals. The next step for the project is to attain the same level of detail in all the plans so that we can integrate the work as a single set of deliverables into the organization. Our recommendation to anyone starting a process improvement effort is to staff the effort with a strong emphasis on project management skills.

Organizational development skills are also essential. The process improvement team needs to assess the organization to determine the root cause of problems, to determine the rate of change for the process improvement efforts, and to institute feedback mechanisms to measure progress. In addition, the team needs to understand how to overcome resistance to change, to deal with change at all levels of the organization, and to sustain change at a manageable rate.

Our experience has convinced us that a framework is essential to develop and deliver process improvement to the organization. Our process improvement framework has three aspects:

- Skills development
- Process definition and improvement
- Operational environment and technology enhancements

For example, we had been working in the area of improving the organization's planning processes. After evaluating the existing planning processes, we determined that we would have to develop the organization's planning skills. First, we introduced a tool to enable people to implement schedules. Second, we developed requirements for the operational environment for the tool and process, specifically for access, archival, and retrieval of project-related information such as project plans and schedules. Third, we determined the requirements for training based on the needs of key individuals in the organization. Finally, we defined the organization's planning process and developed continuous improvement cycles for the process.

Each of our process improvement efforts included the three factors from our project framework. These efforts were tracked by the organization to ensure that the schedule and resource needs of the work were met. In addition, process improvement work was prioritized according to the organization's business needs. The delivery methods for the process improvement work must be agreed upon and understood at all levels of the organization. This provides the context and enables the work to be better understood in the day-to-day routines of the organization.

Case Study 2: Building Support for a Formal SEI-based Process Improvement Program into Ongoing Projects

Initially, the amount of engineering time needed for a formal SEI-based process improvement program was intimidating to management and engineers. To demonstrate that the process could benefit the organization, we took several introductory actions. First, since the organization was already committed to project retrospectives, we introduced the basic SEI concepts into the existing retrospective process. Second, we worked with engineering management to ensure that formal quality planning was undertaken at the start of each project so that quality goals and processes were consciously selected. Third, we designed a metrics program to support our quest for maturity.

Project Retrospective

We developed a retrospective process based upon the principles in the SEI model for process improvement and applied it to our most recent product release. We wanted to ensure that we covered all the key elements in the SEI model (sponsorship, organizational preparedness, employee involvement, working first on KPAs at the Repeatable Level 2). As shown in Figure 2, the process was designed by the forerunner of the SEPG.

First, the SEPG met with the sponsor (the head of the engineering organization) to define the particular attributes of the SEI process we wished to integrate into our retrospectives. They included clear sponsorship, employee involvement in all aspects of the process, and creation of action teams to make improvements. The sponsor communicated to her organization the goals of the enhanced retrospective and her commitment to act on any findings.

Next, we designed and distributed a survey aimed at obtaining a broad view of what worked or did not work on the most recent large release. The retrospective team was assembled and conducted

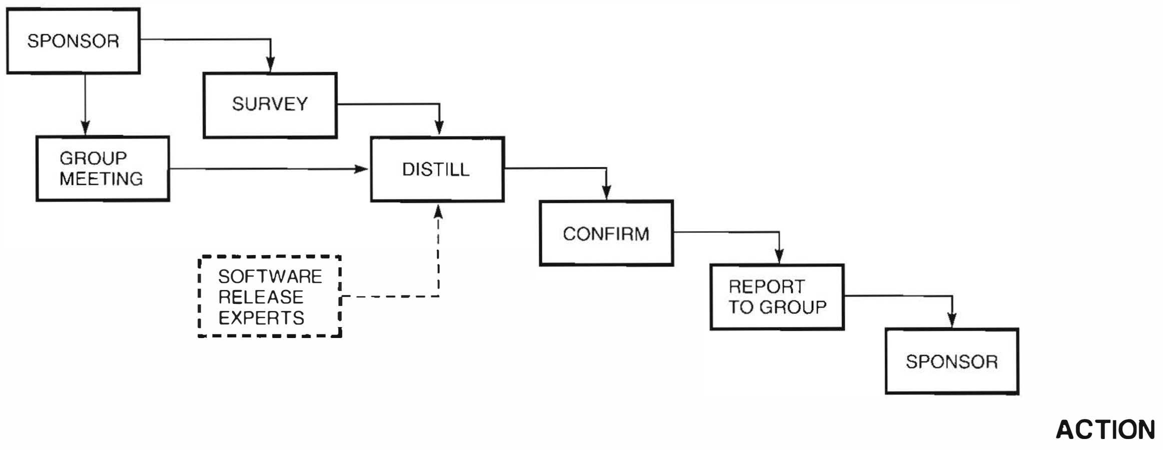


Figure 2 Release XXX Retrospective Process

a facilitated meeting of the larger group to obtain an alternate view of what had happened during the project. The team used the findings from this meeting and the survey to develop a prioritized list of problems.

The following problems were identified as being applicable to both hardware and software.

- Design continued during debugging.
- Component quality ranged from faultless to untested.
- Check-in criteria were inconsistent.
- Check-in criteria were unclear and changed as the project progressed.

Team members discussed the problems in a series of structured interviews with the key people concerned with the release. The interviews focused on identifying the root causes of the problems. Sample root causes are listed below.

- Different assumptions were made about code freeze.
- Changes to check-in criteria were not communicated.
- Hardware was not available for tests early in the project; builds and tests were time consuming.
- Consistent success or failure was not rewarded or fixed.
- Known problems were allowed to continue.

The team then distilled these root causes into a set of findings that were fed back to the originators for confirmation and then to the sponsor for action. The findings from the retrospective team were the following.

- We planned only one release at a time.
- The overall testing model was unclear.
- Check-in procedures were unclear.

The final list of findings can be mapped to the Initial Level 1 of the CMM. The latter two issues relate to software quality assurance (SQA), and the first issue relates to the requirements definition.

The enhanced retrospective boosted our process improvement program. It showed that management needed to sponsor the project, that employee involvement facilitated the improvement plans, and that an SEPG was required to handle the results. In addition, the enhanced retrospective produced better results than a traditional retrospective. We recommend this process to other groups conducting process improvement programs.

Serendipitously, our retrospective was led by the manager of the next release. As we discussed the project's problems, he was heard to say, "We are doing the same thing in my release; I'd better talk to...." We could not have asked for faster implementation! Furthermore, we changed our process to recommend that the manager of the next release participate in all retrospectives. We also believe that too much intuition was at work during the retrospective. At our next retrospective, we will closely compare the problem list with the key practices for our CMM level before we produce a list of findings.

Quality Planning

Often the action plans from SEI, from other process improvement task forces, or from total quality control (TQC) teams are not carried forward to day-to-day project activities. A new technique is invented and prototyped by the action team and then turned over to the SEPG for widespread implementation. At this point, the process improvement usually ends. In other cases, a small group improvement activity may create an improved engineering process, but its success is unknown outside the immediate team.

Ideally, quality planning selects the processes to be used at the start of each project. Quality (process) plans close the gap between improved processes and project activities. We have asked each subsequent team to prepare a quality plan. The process for institutionalizing practices works well at our current CMM level. After we complete our first full SEI assessment and improvement cycle, we should see the necessity of these activities to achieve process maturity. The best quality plans are fully embedded in the release or project plan prepared by each team. We do not require a separate quality plan for each release, merely that the following questions are answered for each new release:

- What attributes of quality are important for this release?
- How will those quality goals be measured before and after the release?
- What are the goals for the product before and after the release?
- What processes will be put in place to ensure that the goals are met?

- What are the expectations for each component in a release and at what milestone?

For example, if the release is to have 10 percent fewer defects than the last release, then the questions above might be answered as follows. The defect reports from customers are important. The goals might be to have 10 percent fewer defect reports per 100 customers, to increase pre-release test coverage by 10 percent, and to continue testing until a rate of less than 1 defect per 1,000 hours of testing is achieved.

To ensure that the goals are met, formal code inspections for 100 percent of all new code would be introduced and regression testing coverage increased by 15 percent. All components would be required to meet this standard 2 weeks before integration.

Our early experiences with quality plans have confirmed our need for a more mature software engineering process. We have seen a tendency to "abandon quality to the quality person"; alternately, some plans have been rejected as "trying to tell engineering how to do its job." It is difficult to separate the testing plans from the quality plan. As a result, the early quality plans have focused on release criteria and have included large sections of background information justifying their very existence.

In the long term, we believe that the quality plan should cease to exist as a separate document and should be included in the overall project plan. In the future, quality plans will be created from known good practices in engineering. As we climb the maturity ladder, we will more and more use a repository of good practice as the basis for creating these plans. An SEPG will be chartered with maintaining the repository (or life cycle as we know it). The life cycle will be updated based upon SEI assessments, retrospectives, small group improvement activities, and so on.

The SEPG is aimed at long-term process improvement across multiple projects. The quality plan is the document to connect these general process improvements to day-to-day project work. Every project or release now has a person designated as responsible for quality. This person is responsible for liaison with the SEPG and bringing the best practices into the teams.

The Software Metrics Program

As shown in Figure 3, full benefit from metrics is experienced only when the processes are under real control, as at the CMM Managed Level 4 or above. In addition, measured SQA is one of the major criteria for attaining the Repeatable Level 2. Therefore we created a metrics program with a dual thrust: we instituted project- and release-related metrics of doneness, or SQA. We also created a metrics program throughout the organization to measure and track our long-term intent for process improvement. These process metrics are not pure because the underlying processes are not under rigorous statistical control; however, they provide a point of focus for the organization's improvement efforts. Our early efforts showed that the organization did not think in terms of processes whose yield can and should be measured over time. We need to start these metrics today so that we will have an effective collection system when we reach the Managed Level 4, and we will also have a population familiar with process management.

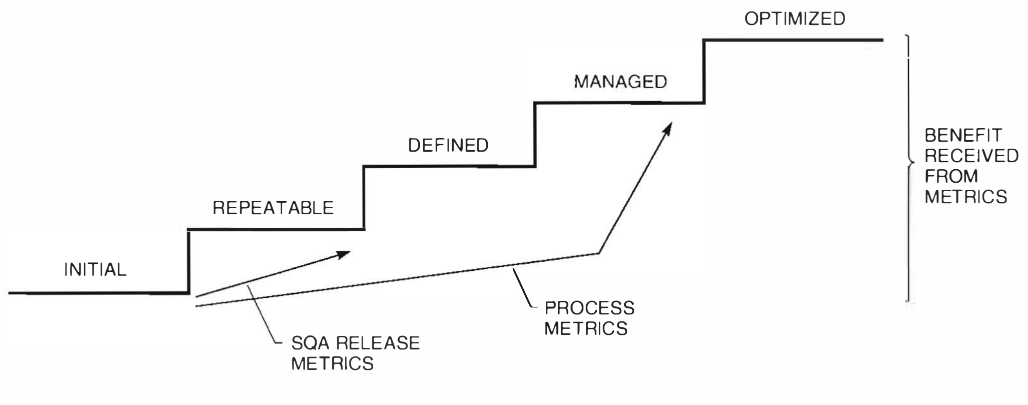
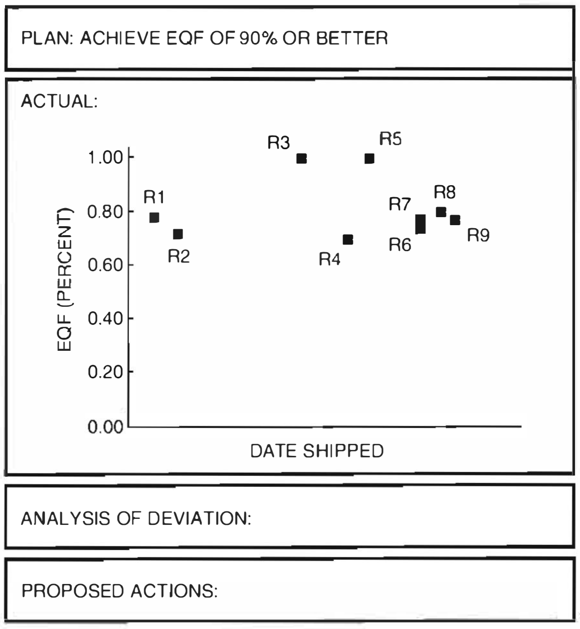


Figure 3 SEI Benefits of Metrics by Level

Organization-wide Metrics We have tried to ensure that our metrics provide a business focus for our improvement activities throughout the organization. We have also tried to present the metrics in such a way as to promote continuous process improvement. We have metrics for product reliability, performance, predictability of schedule, i.e., estimating quality factor (EQF), responsiveness to customers, and cost-effectiveness. Each of the metrics is displayed in a format that embodies the Shewhart/Deming cycle (plan, do, check, act) as shown in Figure 4. In future quality planning sessions, we will review each plan for its impact on these metrics. The SEPG is responsible for preparing and analyzing these metrics.

SQA Metrics Our SQA metrics are relatively simple and are based upon a convergence during a series of checkpoints at the end of our testing cycles. We are measuring test coverage, time under stress without failure, incident arrival rates, and unresolved incidents in the classic way. These measurements ensure that the product has been tested enough to ship. We are now starting to measure early quality indicators such as design stability, which predicts eventual SQA problems. The SEPG is defining improved metrics and is analyzing the effectiveness of our test programs. Day-to-day project decisions as to whether or not to ship are the responsibility of the project teams.


PLAN: ACHIEVE EQF OF 90% OR BETTER

ACTUAL:

ANALYSIS OF DEVIATION:

PROPOSED ACTIONS:

Figure 4 Organizational Metrics

Conclusions Drawn from Both Case Studies

We have drawn two conclusions based on our experiences using the SEI framework. Both conclusions apply whether the organization begins its process improvement efforts with an SEI assessment or uses the SEI framework in support of existing quality activities. First, involving people in the change process is important. At the Initial Level of the CMM, organizations are characterized by ad hoc processes. The processes are not described or enforced, and there is a high dependence on heroic efforts to meet schedules. At the Initial Level of maturity, people are the process. Lack of focus on the importance of people in improving the process causes confusion and chaos in the organization. Examples include:

- A process is not adopted or becomes a "jump through the hoop" exercise when people are unsure of how the change benefits their goals.
- Confusion and conflicts arise when the people involved in carrying out the process are not included in making changes to the process.

By involving people in the change process, we have found that new processes are adopted more quickly and are better suited to the work that people perform. In fact, the introduction of new processes becomes transparent to the organization.

Second, the use of the alternate method bolsters the primary process improvement method. For example, when we started with an SEI assessment in the first case study, we found that incorporating the SEI framework into our product retrospectives raised the group's awareness of the SEI methodology. The SEI framework continued to reassert the importance of process improvement within the organization. In the second case study, we incorporated the SEI framework into ongoing activities. We concluded that, for future process improvement efforts, an SEI assessment would align the organization behind a single common vision and set of priorities.

Current State and Future Challenges

In this section we describe our current state and some of our next challenges in implementing the SEI-based process improvement programs.

Case Study 1—Formal SEI-based Process Improvement Program

As previously described, the process improvement program provided the assessment, an action team was formed, and we introduced improvements based on its recommendations. Our major learning from this program is that actual process change is risky to introduce in spite of strong organizational commitment and difficult to keep on track because factors that interact with the organization are changing. The change in business goals and restructuring within the organization had the highest impact on our process improvement efforts.

In implementing our process improvement efforts, we found that it was important to tie the improvements in our product process to the business goals of the organization. When the business goals changed, we were required to realign our priorities to meet those changes. For example, we set a business goal to meet the first revenue ship date for key hardware products. This required us to move from a sequential product release model to a concurrent release model, where we might have the development of several releases occurring in parallel, e.g., one or more functional releases and one or more hardware releases. This placed new requirements on our processes; as a result, we had to shift the priorities within the process improvement efforts.

Of the two changes, restructuring the organization had a greater impact for us. As a Level 1 organization, we had the practice of overreliance on a small number of people with special skills to perform critical functions. They understood and supported the process improvement work. The restructure resulted in these people leaving the organization or changing positions. Since many of the key sponsors for the process improvement work left the group, we had to rebuild support and sponsorship within the new management and organization structure. This had an impact on both the priority and the methods to deliver the process improvement work.

The basic problem in both changes was that we had no way to transfer knowledge or skill sets during changes. We expect that the system in which we work will continually change and shift. Our major future challenge is to develop process improvements and support for these improvements that transcend changes to the system in which the organization exists. We intend to continue to bolster our SEI activities with the addition of metrics and quality planning to ongoing organization activities.

Case Study 2—Adding SEI to an Existing Process Improvement Program

Currently, the organization is focused on delivering two key products and on developing a new organizational structure. As a result, it has been difficult to maintain progress on major process improvements.

The retrospective process is now in use on all major releases of our products with positive results. The first action plans from the retrospectives took a long time to complete and are only being implemented today (August 1993). Metrics and quality plans are now in use by 100 percent of our releases.

We could have made faster progress throughout the improvement program if we had better fundamental knowledge about quality and process in our organization. The additional learning from retrospectives could have been more effective if we also had a broadly based education program in quality.

The retrospectives have produced real benefit and some goodwill toward process improvement. In addition, they have acted as an excellent way of educating their participants about the fundamentals of process management. We recently held the first meeting for the formal SEI program; both attendance and enthusiasm were high. The prototyping work with the retrospectives, however, has not overcome the concerns of the organization. For example, concern remains that an SEPG will take ownership of the process away from the engineering groups despite repeated assurance that it will not. The full benefits of quality planning and the metrics program and their connection to our breakthrough productivity objectives remain to be achieved.

We believe that the visible commitment for an SEI assessment is needed to galvanize the organization to achieve breakthrough levels of process improvement and higher benefits, and we are continuing with our formal SEI program. The initial organization-wide training is scheduled for the first week of September 1993, and the assessment is tentatively scheduled for April 1994.

Acknowledgments

Neil thanks his team partners Bryan Jones, Brian Porter, Tom Saleme, Nick Craig, and most especially his sponsor Laura Woodburn.

Both authors acknowledge Barbara Kelczewski, who helped edit this paper. She turned a random collection of thoughts and experiences into a format for communication.

References

1. M. Paulk, B. Curtis, M. Chrissis, and C. Weber, *Capability Maturity Model for Software V1.1* (Pittsburgh, PA: Carnegie-Mellon University, Software Engineering Institute, Technical Report, CMU/SEI-93-TR-24 ESC-TR-93-177, February 1993).

2. W. Humphrey, T. Snyder, and R. Willis, "Software Process Improvement at Hughes Aircraft," *IEEE Software* (July 1991).

3. R. Dion, "Process Improvement and the Corporate Balance Sheet," *IEEE Software* (July 1993).

4. W. Humphrey, *Managing the Software Process* (Reading, MA: Addison-Wesley Publishing Company, 1989/1990).

5. P. Fowler and S. Rifken, *Software Engineering Process Group Guide* (Pittsburgh, PA: Carnegie-Mellon University, Software Engineering Institute, Technical Report, CMU/SEI-90-TR-24 ESD-90-TR225, September 1990).

General References

R. Ackoff, *Creating the Corporate Future: Plan or Be Planned For* (New York: Wiley, 1981).

T. DeMarco, *Controlling Software Projects* (New York: Yourdon Press, 1982).

A. Duncan and T. Harris, "Software Productivity Measurements," *Digital Technical Journal,* vol. 1, no. 6 (February 1988): 20–27.

D. Kauffman, Jr., *Systems One: An Introduction to Systems Thinking* (Future Systems, Inc., 1980).

M. Paulk, C. Weber, S. Garcia, M. Chrissis, and M. Bush, "Key Practices of the Capability Maturity Model V1.1" (Pittsburgh, PA: Carnegie-Mellon University, Software Engineering Institute, Technical Report, CMU/SEI-93-TR-25 ESC-TR-93-178, February 1993).

J. Thompson, *Organizations in Action* (New York: McGraw-Hill Book Company, 1967).

Robert G. Thomson |

Assessing the Quality of OpenVMS AXP: Software Measurement Using Subjective Data

In the absence of a well-defined development process and a set of objective metrics, subjective data can be used to assess the quality of a software release. This assessment can identify and characterize development risk, focus testing and validation efforts, and indicate where and how process management should be improved. The OpenVMS Engineering organization has developed a questionnaire, a set of quality indicators, and a data reduction methodology that implement such an assessment. This assessment approach is flexible and can be applied generally to the measurement of software quality during the evolution of a repeatable development process.

Porting the OpenVMS operating system from the VAX to the Alpha AXP architecture was a tremendous technical challenge for the OpenVMS Engineering organization. Part of this challenge was to achieve the high degree of quality that customers expect of the OpenVMS system and would require before migrating their mission-critical OpenVMS applications and operations to a new hardware platform.

To assure that this quality challenge was met before releasing the product, the engineers involved in the port needed to answer the intuitive question, How will we know that it's right? The quality assessment approach described in this paper was an integral part of the answer. Following an overview of the quality challenge and the assessment framework, the paper describes the quality indicators and assessment process used to measure software quality during the development of OpenVMS AXP versions 1.0 and 1.5.

Quality Challenge

OpenVMS Engineering considered schedule, functionality, and quality all to be critical factors in successfully porting the OpenVMS system to the Alpha AXP platform. Although both aggressive and complex, the port had several characteristics that favored its success:

- An established product with well-defined capabilities
- Carefully controlled source code and build procedures for the system
- A very experienced development team
- A consistent project management system for managing progress against the schedule

What the port lacked was a uniform development process with a comprehensive set of objective metrics for measuring software quality. As the project progressed, engineers were added when their expertise became needed. But with the engineers came a variety of engineering processes. Given the size and complexity of just the initial release of the OpenVMS AXP system, this lack of process consistency represented a significant deficiency.

The version 1.0 development effort kept to a demanding schedule spanning more than two years. During that time, more than 170 engineers made approximately 68,000 separate modifications or additions to the source code in order to port, build, and test the OpenVMS AXP system. These modifications were integrated and tested in stages with weekly software builds that resulted in roughly 1,200 system base levels. At its release for customer shipment, the base system of OpenVMS AXP version 1.0 comprised an estimated 3,045,000 lines of noncomment source statements. Yet, the existing metrics for measuring software quality were limited primarily to weekly statistics on incremental test hours, source code modifications, and problem reports.

Quality Assessment Framework

Despite its dearth of software metrics for the initial release, OpenVMS Engineering had the following clear goals for the quality of its version 1.0 and version 1.5 releases on the Alpha AXP platform:

- Correctness goals, which focused on completing all critical functionality
- Reliability goals, which focused on minimizing defect introduction, stabilizing the code base, resolving all significant defects, and meeting availability targets
- Performance goals, which focused on meeting SPECmark and TPC Benchmark A (TPC-A) projections
- Migration goals, which focused on supporting easy and reliable application porting or execution of translated images
- Usability goals, which focused on providing reliable system installation, documentation, and tuning guidelines
- Maintainability goals, which focused on supporting easy problem diagnosis

Measuring progress against these goals with objective data would have required OpenVMS Engineering to define appropriate metrics, integrate procedures for collecting metric data into the existing development process, and accumulate sufficient data to validate the collection procedures and establish baselines. The aggressive OpenVMS AXP development schedule made this approach impracticable for version 1.0.

As an alternative, OpenVMS Engineering developed an approach for assessing release quality based on subjective data. This approach built on the organization's historic reliance on the technical expertise of its engineering teams for assuring quality. At the same time, the approach laid the foundation for defining a practical set of quantitative metrics guided by experiences with the subjective data. Over time, OpenVMS Engineering can implement these metrics as part of its Continuous Improvement effort for the OpenVMS development process.

Quality Assessment Indicators

Seven quality indicators provide the framework for the process of assessing quality in the OpenVMS AXP operating system. Each indicator is intended to show the presence or absence of a meaningful characteristic of software quality. These indicators correspond to seven sets of data provided by projects that constitute a particular software release. Table 1 lists these indicators together with a summary of the subjective data and objective metrics over which the indicators are defined. The table also shows the significance of each indicator with respect to the quality assessment process. This section presents a more detailed discussion of the data sets that define the indicators and the information that these indicators provide.

Explicit Statement

A project most clearly indicates quality through explicitly stated judgments from the engineering team that the software elements

- Possess all planned functionality
- Currently pose little technical risk to the release
- Embody equal or superior implementation on the Alpha AXP platform as compared to the VAX platform
- Meet the project's criteria for release readiness

Because it most fully reflects a project's overall quality, explicit statement is the most important indicator of quality.

Element Expertise

The accuracy of a subjective measure of quality is a function of a team's expertise regarding the implementation of their project's elements. Moreover, lack of expertise may indicate a higher likelihood of introducing defects during implementation. Such expertise is based on the team's knowledge of how the project's elements were implemented and behaved on the VAX platform. The expertise is bounded by areas where a team perceives difficulty in working with the elements on the Alpha AXP platform. A project indicates high element expertise when it involves engineers who

- Have significant experience with the OpenVMS system
- Are already familiar with the elements involved in the project
- Encounter little technical difficulty in modifying project elements

Table 1 Summary of Quality Assessment Indicators

Quality Indicator	Significance	Subjective Data	Objective Metrics
Explicit Statement	Judgment from engineering team that release requirements are met	Implementation quality; outstanding risks; completeness	Source code change rate; problem report rate
Element Expertise	More accuracy in quality judgments; less likelihood of introducing defects	Experience with OpenVMS and with project elements	
Technical Ease	Less susceptibility to defect introduction; less need for element expertise	Quality requirements; portability; maintain-ability	Structural complexity
Process Consistency	Less quality variation within and across development phases	Coherence of requirements, design, reviews, and testing	
Engineered Changes	Better defect prevention; less reliance on methodical testing	Use of specifications and inspections in development	
Methodical Testing	Better defect detection; less reliance on well-engineered changes	Testing effort, regularity, variety, and code coverage	
Defect Detection	Indicates progress where change and testing processes are strong; indicates risk where they are weak	Percent of detected defects being logged; percent of logged problems that describe defects	Defect counts

Technical Ease

Project elements that are technically easier to maintain are also less vulnerable to the introduction of defects during changes. The less element expertise possessed by the project team, the more significant technical ease becomes as an indicator of quality. A project indicates technical ease if the team judges that their project has

- A relatively low priority on technical quality
- Simple functionality, code, and data structures
- Little vulnerability to instruction atomicity or memory granularity problems

Process Consistency

The usefulness of a process-related indicator of project quality depends on the consistency of the software development process that a project team employs. This consistency encompasses the team's understanding as well as their implementation of good software engineering process. A project indicates process consistency when software delivery involves

- Rating product suitability based on a good understanding of customer expectations
- Removing technical and operating risks as a precursor to release readiness
- Defining an effective development process based on requirements, design, specification, inspection, and testing
- Using tests with good code coverage for methodical testing
- Reviewing or inspecting the code developed in one-person projects

Engineered Changes

Careful engineering of changes to a project's source code can catch defects before its elements are integrated into a running system. A project indicates the quality of code ports, modifications, fixes, or additions through the extent of

- Expenditures of engineering resources on design
- Functional or design specification completeness
- Inspections or reviews of code changes

Methodical Testing

Regular and deliberate ad hoc, regression, and stress testing is needed to find the defects introduced into a project's elements through additions or modifications to its source code. The less effectively a team engineers changes to the elements to prevent defects, the more significant testing becomes as an indicator of quality. Methodical testing of a project's elements is indicated where tests

- Run each week and on each software base level
- Involve ad hoc, regression, and stress tests
- Cover a significant portion of main program code and error-handling code
- Use a significant portion of a project's total engineering resources

Defect Detection

When compared against the number of defects detected in prior releases, the number detected within a project's elements for the current release provides an indication of its current quality. A low ratio of the current defect count to the past defect count may indicate either an improved development process or inadequate detection; a high ratio may indicate the reverse. The more effectively a team engineers changes to an element and performs the element's tests, the more reliable the defect detection indicator becomes as a measure of quality.

Defect counts are available from the defect tracking system; however, defects that are readily resolved are frequently not logged. Therefore, defect counts across a release are normalized by having project engineers estimate the percentage of defects identified during inspections, debugging, and testing that they actually log in the defect tracking system.

Quality Assessment Process

The assessment process applies these quality indicators to data gathered primarily through a questionnaire, which is administered to a subset of the projects included in a software release. Applying the quality indicators to questionnaire data yields a set of quality profiles. The usefulness of these profiles for assessing quality depends both on the accuracy of the data and on the ability of the targeted projects to represent the quality of the overall release. This section describes the quality assessment process in terms of our experiences across two releases of the OpenVMS AXP system, versions 1.0 and 1.5.

Select Assessment Targets

The assessment process begins by selecting a set of projects within the software release to serve as targets for measuring the release's quality. We made this selection for a particular OpenVMS AXP release by ranking the projects based on the following factors:

- The functional areas where the project manager believed quality was critically important to the success of the release
- Whether a project provided latent, limited, or full support of ported or new functionality for the release
- The number of problem reports filed in prior releases against the elements of the project

Because the version 1.0 development effort was quite large, we focused the assessment on 57 projects, which constituted the top 17 percent of the resulting ranked list. Those projects accounted for 74 percent of the total source code involved in the release. Because the version 1.5 development effort was smaller, we targeted only 38 projects and yet encompassed more of those projects that dictated the release's quality.

Administer an Assessment Questionnaire

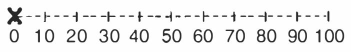
The assessment process uses a questionnaire to measure the quality of the targeted projects. Because all answers to the questionnaire are assumed to be subjective, its effectiveness relies more on the completeness of the responses than on their accuracy. With this in mind, we designed the question set for each OpenVMS AXP release to be large and varied, yet easy to answer.

For the version 1.5 release, 29 questions, some multipart, provided 75 data values for each project. The version 1.0 questionnaire was slightly smaller. Most questions could be answered by indicating on a graduated scale either a percentage value or a qualitative judgment (such as easy versus hard, or low versus high). Typically, respondents were able to complete the version 1.5 questionnaire in less than 15 minutes.

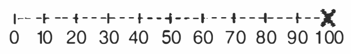
Figure 1 shows the steps involved in deriving an individual quality score and a composite quality score using a questionnaire. The three questions from the OpenVMS AXP version 1.5 questionnaire illustrated in Step 1 of the figure form the question set that provides the data for assessing element expertise. The example shows the questions as completed for Project_20.

STEP 1 ADMINISTER A QUESTIONNAIRE TO GATHER SUBJECTIVE DATA.

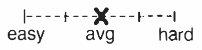
2. What percentage of the engineers were familiar with this area of the system when the OpenVMS AXP V1.5 project began?



3. What percentage of the project's engineers were experienced VAX and/or Alpha AXP developers?



4. How difficult is it to complete this project with respect to its technical obstacles?



STEP 2 NORMALIZE THE DATA AND AVERAGE ACROSS THE QUESTION SETS THAT DEFINE EACH QUALITY INDICATOR.

	QUESTION 2	QUESTION 3	QUESTION 4	AVERAGE
PROJECT_20	0	10	5	5
PROJECT_36	10	10	10	10

STEP 3 SYNTHESIZE QUALITY PROFILES USING A COMPOSITE OF THE QUALITY SCORES FOR EACH PROJECT.

	NORMALIZED SCORES					NORMALIZED, WEIGHTED, AND SCALED SCORES		
INDICATOR	PROJECT_20	PROJECT_36	RELEASE AVERAGE	WEIGHTING FACTOR	SCALING FACTOR	PROJECT_20	PROJECT_36	RELEASE AVERAGE
EXPLICIT STATEMENT	6	8	7	3	0.91	16	22	19
ELEMENT EXPERTISE	5	10	7	2	0.91	9	18	13
TECHNICAL EASE	2	10	4	1	0.91	2	9	4
PROCESS CONSISTENCY	9	7	8	1	0.91	8	6	7
ENGINEERED CHANGES	3	5	3	2	0.91	6	9	6
METHODICAL TESTING	5	2	4	2	0.91	9	4	7
COMPOSITE						50	68	56

Figure 1 Deriving Quality Indicator Scores Using Element Expertise Data for Project_20 and Project_36

To mitigate bias and uncover inconsistency within the questionnaire data, we selected a broad range of questions that measured progress against quality goals from three perspectives:

- A process perspective, which covered design, specification, coding, inspection, and testing. These process elements were measured with respect to project resource expenditures and product element coverage.
- A product perspective, which covered element size, complexity, technical risks, implementation quality, completeness, release readiness, and suitability relative to customer expectations.
- A project perspective, which covered priorities, difficulty, team size, and engineering experience.

For both releases of the OpenVMS AXP system, participation in the assessment survey was high. More than 90 percent of the project teams returned questionnaires with an average of more than 90 percent of the questions answered.

Apply the Quality Indicators

The purpose of applying quality indicators to questionnaire responses is to convert qualitative judgments into quantitative measures of quality. To facilitate database entry and quantitative analysis, we first normalized the questionnaire responses, using a scale of 0 to 10, where 10 generally represented greater contribution to product quality. Numeric answers were entered directly into the database without scaling; unanswered questions were assigned the value of −1.

Given this scale, responses of 3 or less represented low (weak) assessments and responses of 7 or more represented high (strong) assessments. A response of 5 represented an implicit norm among the development teams for what constituted an acceptable process, product, or project. All assessments were interpreted in light of how this norm related to organizational goals or prevailing industry practices.

Step 2 of Figure 1 shows the normalized and averaged data used to assess element expertise for Project_20 and also for Project_36. Note that dividing by 10 normalized the responses to questions 2 and 3. For question 4, the five gradations from easy to hard were normalized by mapping them onto the values 0, 3, 5, 7, and 10. Easy completion with respect to technical difficulties indicated greater element expertise and hence received the higher value. Averaging the normalized data across the question set yielded the element expertise quality score for each of the two projects.

Note that for the process consistency indicator, this averaging occurs not over the sum of all responses in the question set but over the differences between pairs of responses that should be close in value to be consistent. The resulting average is then subtracted from 10. For example, a project that rates its ability to meet customer expectations as 9 but its understanding of those expectations as 5 would score 10 − (9 − 5) or 6 with respect to this pair of responses.

The mean value of all quality scores for a particular indicator reveals the engineering team's collective perception of how strong the overall release is with respect to the group norm for that indicator. Ranking the quality scores and then graphing them as variances from this mean facilitates Pareto analysis of the projects by indicator. This analysis reveals those projects with a particularly strong or weak score for a specific indicator.

Figures 2 and 3 show the quality scores for element expertise and technical ease that we derived for OpenVMS AXP version 1.5. These figures suggest a relatively high perception across the projects of overall element expertise contrasted by a lower and more varied perception of technical ease. Pareto analysis of these distributions highlights projects such as Project_36, whose quality scores were high for both indicators, and Project_20, whose scores were both low.

Synthesize Quality Profiles

Because our derivation of the indicators was based on engineering experience rather than on statistical modeling, no single indicator is a reliable predictor of overall project quality. Moreover, because the quality indicators are based on inexact data, the application of a particular quality indicator may be inconclusive with respect to some projects. To overcome these obstacles to comparative assessment of project quality, we synthesized quality profiles using a composite of the quality scores for each project.

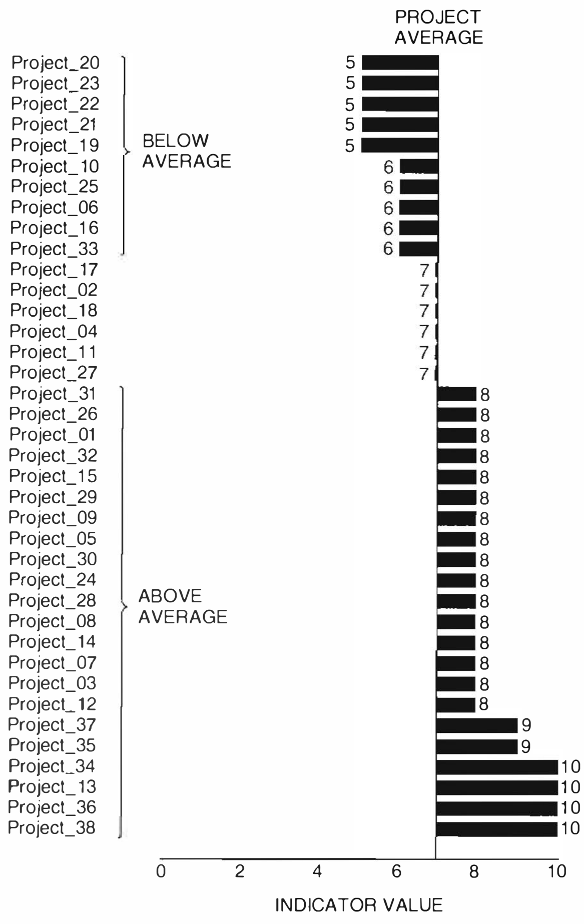


Figure 2 Assessment of Element Expertise at Alpha Test of OpenVMS AXP Version 1.5

Repeating Step 2 of Figure 1 using the responses to other question sets yields normalized scores for each quality indicator. The table presented in Step 3 shows the quality profiles for Project_20 and Project_36. Also shown is the quality profile arrived at by averaging the quality scores across all the targeted projects in the version 1.5 release.

Figure 4 depicts the quality profiles of the projects targeted for OpenVMS AXP version 1.5. These composites use six of the seven quality indicators. Due to insufficient questionnaire data regarding defect detection and removal, the corresponding indicator was not employed in the assessment. Consequently, the identification of error-prone modules and the assessment of defect removal efficiency occurred separately within the ongoing verification efforts for that release.

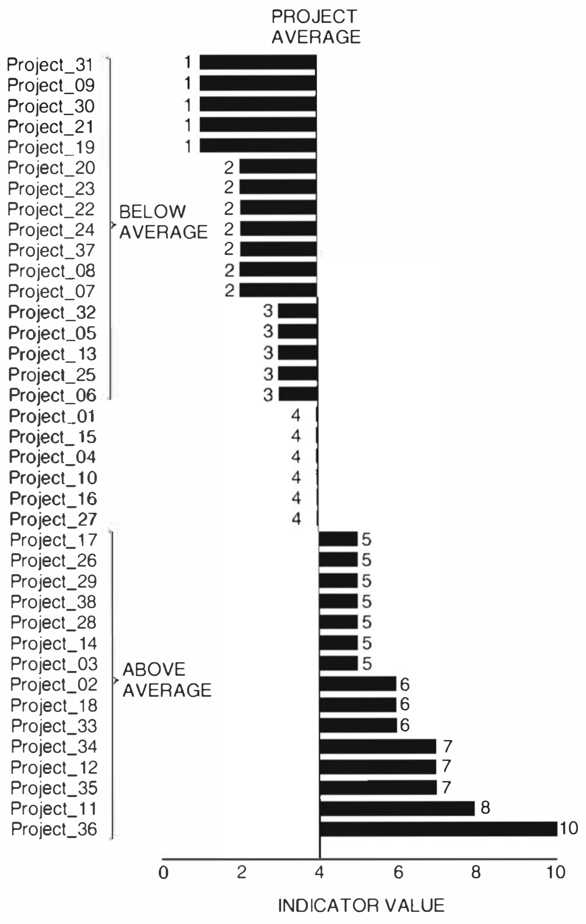


Figure 3 Assessment of Technical Ease at Alpha Test of OpenVMS AXP Version 1.5

To reflect the relative capacity of each indicator to independently provide meaningful information about project quality, we formed the composites by weighting the individual quality scores as follows:

- Explicit statement has a weighting factor of 3.
- Methodical testing, engineered changes, and element expertise have weighting factors of 2.
- Technical ease and process consistency have a weighting factor of 1.

This weighting was based on OpenVMS Engineering experience and reflects relative contribution to the assurance of quality within the current development process. Because field data regarding the actual quality of the released product was unavailable during the assessment effort, statistical analysis of the questionnaire data was inconclusive.

Using this weighting, the resulting maximum score across all six indicators totaled 110. To make the range of values for the composite quality profiles more intuitive, we further scaled this aggregate by 0.91 (100 divided by 110) so that the maximum totaled 100. Multiplying the individual scores by the weighting and scaling factors yielded the second set of scores shown in Step 3 of Figure 1. For reference, an indicator composite that consists of the maximum possible scores for these weighted and scaled indicators appears at the bottom of Figure 4. A similar composite profile of the average project scores for the release also appears.

Interpret the Quality Profiles

Clustering the projects according to their composite quality profiles highlights relative product quality, project risk, and process deficiencies. For OpenVMS AXP version 1.5, we identified nine groups of quality profiles with similar distinguishing characteristics relative to the average profile. In Figure 4, braces delimit these groups.

The average composite score for the targeted projects in the version 1.5 release was 55 out of 100, with 76 percent of the projects scoring in the range of 45 to 65. Only Project_29 scored at or above the average for each indicator; only Project_33 and Project_38 scored at or above the norm for each. Consequently, most projects fell within the Needs Ongoing Validation region of Figure 4. Scoring in this region indicated that a project required some form of validation work to improve quality prior to beta testing and customer shipment of the release.

In several instances, the questionnaire data was sufficiently scant or the quality issues sufficiently numerous to suggest that additional data on a project's actual condition was needed before completing that project's quality assessment. Because a value of −1 was assigned to each unanswered question, projects for which such a value was assigned generally exhibited low indicator composites as depicted in Figure 4 by the bars ending in the Needs Further Investigation region. Project_01 and Project_09 are examples of projects in this category.

If the quality indicators were sufficiently strong, little further assessment or validation work appeared to be needed. Projects that exhibited high indicator composites are depicted by bars ending in the Needs Final Confirmation region. Only Project_33, Project_36, Project_37, and Project_38 fell into this category.

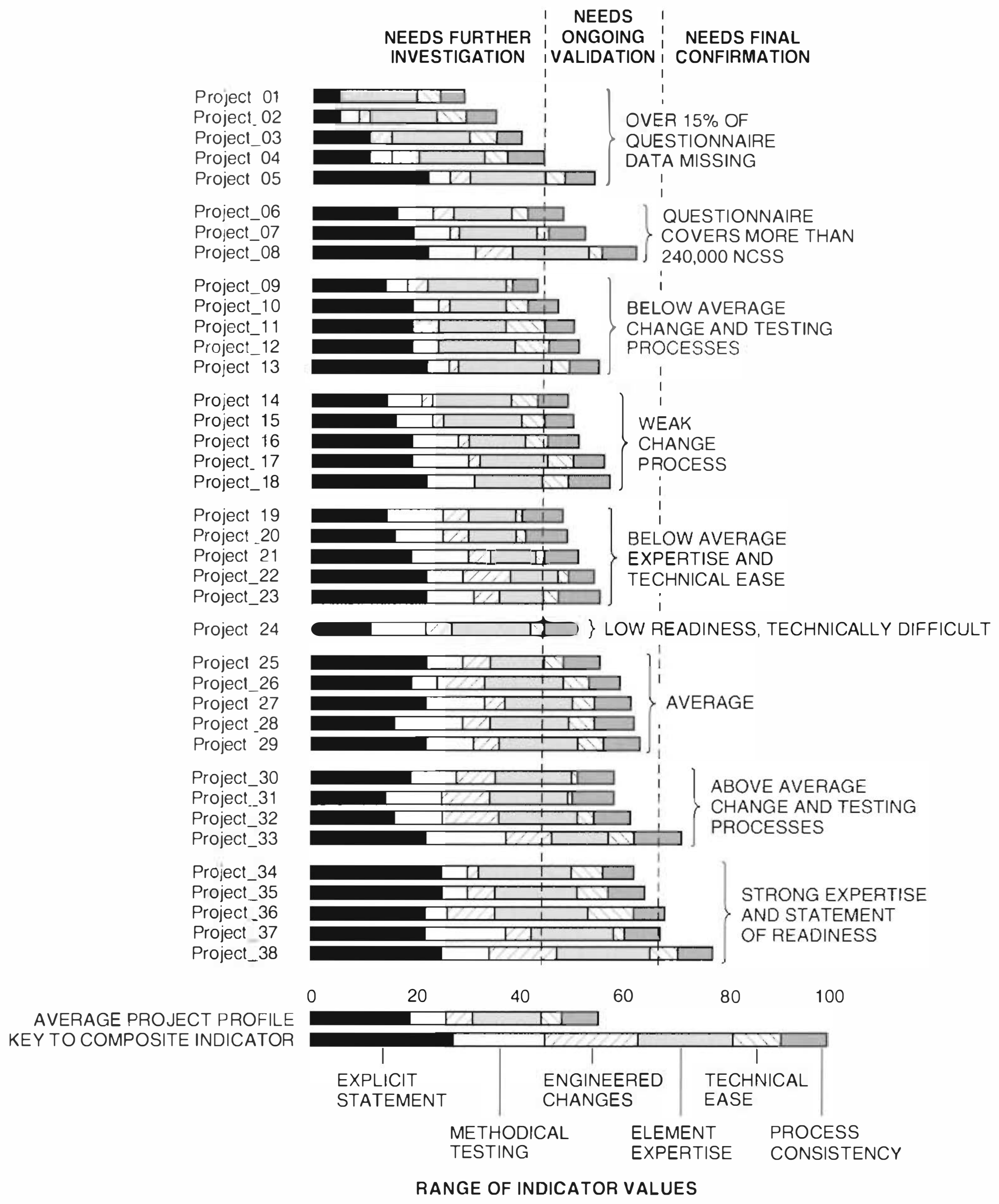


Figure 4 Composite Profile of Project Quality at Alpha Test of OpenVMS AXP Version 1.5

Quality Assessment Results

Taken together, the composite quality profiles, the quality indicator distributions, and the project questionnaire data form an assessment continuum within which to measure progress against quality goals. From a release perspective, the composite quality profiles and the indicator distributions identify process deficiencies. They also characterize areas of risk for the product. From a project perspective, a comparison of quality profiles and scores focuses ongoing verification efforts where they can have the greatest impact on the overall quality of a release. The questionnaire data itself can help determine the form this verification work takes. The results from the assessment of data obtained from the alpha test of OpenVMS AXP version 1.5 illustrate these measurement perspectives.

Identification of Release Deficiencies

The projects that made up the version 1.5 release were known to have a widely varying and typically incomplete process for engineering changes in their code base. From the quality assessment administered when alpha testing began, we clarified the following deficiencies in the process and product for that release so that steps could be taken to ensure that the release was ready for customer shipment:

- Sixteen percent of the projects had significant risk due to outstanding dependencies, unresolved technical problems, or operational instabilities.
- Although 76 percent of the project teams rated their technical capacity as high, 71 percent reported having significant difficulty completing the project due to schedule, equipment, or personnel constraints.
- Ad hoc, regression, and stress tests were regularly executed on the code of 34 percent of the projects.
- Fifty-five percent of the projects had some portion of their code implementation described by a functional or design specification.
- Thirty-seven percent of the projects were handled by just one engineer. Of these 14 projects, 5 had above-average technical difficulty and 5 expended no engineering resources on reviews or inspections.
- Twenty-six percent of the projects lacked a strong understanding of customer expectations against which to evaluate product attributes.
- Code reviews across the projects averaged only 30 percent coverage of ported source code, 40 percent coverage of rewritten or added source code, and 60 percent coverage of source code fixes.

Similar kinds of results from the quality assessment for the version 1.0 release led to the implementation of a process for enhancing product stability prior to customer shipment. The results also contributed to decisions within OpenVMS Engineering to establish more rigorous software metrics within the development process. Moreover, clarifying the process deficiencies for OpenVMS AXP versions 1.0 and 1.5 has contributed to an increased emphasis on defect prevention in the follow-on release.

Focus for Project Verification

In the context of the product risks and process deficiencies just summarized, the quality assessment results for version 1.5 provided the following framework for focusing the ongoing verification efforts:

- Project_01 through Project_05 were missing more than 15 percent of the questionnaire data. (See Figure 4.) These projects required further investigation to determine the current condition of constituent elements as well as the form, focus, and priority of needed verification work.
- Project_06 through Project_18 exhibited composite scores that were below average overall. Verification work that focused on compensating for the weak change and testing processes was a high priority for these projects.
- Project_19 through Project_24 exhibited at least average values for engineered changes and methodical testing; these projects also exhibited significantly below average values for technical ease and, in most cases, element expertise. Verification work for these projects needed to focus on the functionality that posed the greatest technical difficulty or risk given schedule and resource constraints.
- Project_25 through Project_29 exhibited average quality profiles. Their verification work needed to focus on specific portions of the code where defects may exist due to technical difficulty, inadequate changes processes, or poor test coverage or effectiveness.
- Project_30 through Project_32 had strong processes. Because their technical ease or element expertise indicator values were below average, however, verification work needed to focus existing processes on mitigating current risks and improving the product's readiness to meet customer expectations.
- Project_33 through Project_38 were evidently on-track to a high-quality release and therefore required only a confirmation of quality prior to customer shipment.

Given the limitations of the assessment data and its pervasive reliance upon engineering judgment, following all assessments with some form of verification work was important. In some cases, the data as provided and interpreted within the assessment indicated a level of quality that we knew was not actually present.

By removing defects from the product as projects completed their planned functionality, the ongoing verification effort for version 1.5 contributed to improved implementation quality relative to the VAX platform, mitigated risk due to technical or stability problems, and increased the satisfaction of release readiness criteria.

Conclusions

To assure the quality of its product while improving the quality of its development process, OpenVMS Engineering implemented a process for assessing the quality of its releases using subjective data. This assessment process has proven useful in characterizing product risks, focusing verification efforts, and identifying process deficiencies during the development of versions 1.0 and 1.5 of the OpenVMS AXP operating system. The assessment identified areas that needed attention; the resulting actions led to improved quality.

Using the Assessment Process

By focusing only on those projects key to a release's success, the assessment process described in this paper limits the cost and turnaround time for an assessment of quality without significantly diminishing its value. By focusing on subjective data, this process captures the judgment of engineers on the project teams regarding overall progress toward release readiness.

The OpenVMS AXP questionnaire covers various product, project, and process aspects of a release. The questions may be tailored for different software releases or even different software products.

Using seven quality indicators, which are defined over subsets of questions from the questionnaire, the assessment process synthesizes quality profiles for each project. These profiles are based on quality norms that are implicit within the development organization. By administering the assessment process as a release enters its alpha testing, these profiles can guide the project's movement toward its quality goals for the release.

Improving the Assessment Process

Several opportunities exist for improving the usefulness of this assessment process. As the process is repeated across successive software releases, the organization can

- Validate the predictive value of the assessment process through statistical analysis of quality indicators and questionnaire data against selected quality results when a release begins shipping to customers
- Refine the questionnaire to ensure that the questions remain relevant to the development process, unambiguous, and internally consistent
- Complement the developer assessment administered during alpha testing with a similar customer assessment during beta testing

As an organization's software measurement process matures, subjective measures should be replaced with objective metrics for which data can be economically and reliably collected. Such metrics should reduce reliance on the subjective data, but not eliminate it: the perceptions of an experienced engineer can usually add clarity to the assessment of release quality.

Acknowledgments

Development of this assessment process was sparked, encouraged, and facilitated by the members of the Quality Assessment project for OpenVMS AXP version 1.0: Jeff Pilsmaker, Pam Levesque, Ralph Weber, and Bill Goleman. Its form was refined and its usefulness validated by the members of the OpenVMS AXP Verification Group. Curt Spacht and Tim Beaudin, the validation project leaders for versions 1.0 and 1.5, were particularly supportive during the implementation and repetition of this process.

General References

As representative of current trends in the definition and deployment of software measurement processes, the following references proved particularly useful during the effort described in this paper:

W. Humphrey, *Managing the Software Process* (Reading, MA: Addison-Wesley, 1989/1990).

C. Weber, M. Paulk, C. Wise, and J. Withey, "Key Practices of the Capability Maturity Model," Technical Report CMU/SEI-91-TR-25 (Pittsburgh, PA: Software Engineering Institute, Carnegie-Mellon University, 1991).

J. Baumert and M. McWhinney, "Software Measures and the Capability Maturity Model," Technical Report CMU/SEI-92-TR-25 (Pittsburgh, PA: Software Engineering Institute, Carnegie-Mellon University, 1992).

R. Grady and D. Caswell, *Software Metrics: Establishing a Company-Wide Program* (Englewood Cliffs, NJ: Prentice-Hall, 1987).

R. Grady, *Practical Software Metrics for Project Management and Process Improvement* (Englewood Cliffs, NJ: Prentice-Hall, 1992).

Further Readings

The following technical papers were written by Digital authors:

R. Abbott, "Scheduling Real-Time Transactions: A Performance Evaluation," *ACM Transactions on Database Systems* (September 1992).

B. Aichinger, "Futurebus+ Profile B," *OPEN BUS SYSTEMS '91* (November 1991).

B. Aichinger, "Profile B Modules in a Profile A/F System," *OPEN BUS SYSTEMS '92* (October 1992).

W. Anderson, "Logical Verification of the NVAX CPU Chip Design," *IEEE International Conference on Computer Design* (October 1992).

P. Anick, A. Gunderson, A. Rewari, M. Swartwout, M. Carifio, and M. Adler, "AI Research and Applications in Digital's Service Organization," *AI Magazine* (Winter 1992).

S. Apgar, "Interactive Animation of Fault Tolerant Parallel Algorithms," *IEEE Workshop on Visual Languages* (September 1992).

J. Arabian, K. Lentz, and E. Ulrich, "The Comparative and Concurrent Simulation of Discrete Event Experiments," *Journal of Electronic Testing: Theory and Applications (JETTA)* (May 1992).

B. Archambeault, "EMI Modeling of Air Vents and Slots in Shielded Cabinets," *IEEE International Symposium on Electromagnetic Compatibility* (August 1992).

N. Arora, D. Bell, and L. Bair, "An Accurate Method for Determining MOSFET Gate Overlap Capacitance," *Solid-State Electronics* (December 1992).

S. Batra and M. Mallary, "Improved Cross-Talk Performance for Shielded Flux Sense Heads," *Thirty-seventh Annual Conference on Magnetism and Magnetic Materials* (December 1992).

J. Blanchard, V. Murthy, and D. Jones, "Quality and Reliability Assessment of Hardware and Software during the Total Product Life Cycle," *Quality and Reliability Engineering International* (September 1992).

J. Brown, D. Bernstein, R. Stamm, and G. M. Uhler, "NVAX and NVAX+: Single Chip CMOS VAX Microprocessors," *IEEE International Conference on Computer Design* (October 1992).

D. Byrne, "Computer Aided Design of High Speed Optical Data Links," *IEEE Lasers and Electro-Optics Society 1992 Annual Meeting* (November 1992).

E. Cheng, "A High-speed Open Journal System in a Distributed Computing Environment," *IEEE Second International Computer Science Conference (ICSC '92)* (December 1992).

J. Clement, "Vacancy Supersaturation Model for Electromigration Failure under DC and Pulsed DC Stress," *Materials Research Society Symposium Proceedings* (April 1992).

M. Comard and J. Cuéllar, "Rapid Development, in a Manufacturing Environment, of a 1um Triple-Level Metal CMOS Process Through the Use of Cross-Functional Teams," *IEEE/SEMI Advanced Semiconductor Manufacturing Conference* (September 1992).

D. Dossa, "Piezomodulated Reflectivity of Asymmetric and Symmetric Alx1Ga1-x1As/GaAs/Alx3Ga1-x3As Single Quantum Wells," *Applied Physics Letters* (June 1992).

B. Doyle, K. Mistry, and D. Jackson, "Examination of Gradual-Junction p-MOS Structures for Hot Carrier Control Using a New Lifetime Extraction Method," *IEEE Transactions on Electron Devices* (October 1992).

M. Elbert and T. Weyna, "Performability Analysis of Large-Scale Packet Switching Networks," *SUPERCOMM/International Conference on Communications '92* (May 1992).

R. Evans, "The Close Attached Capacitor: A Solution to Switching Noise Problems," *IEEE Electronic Components and Technology Conference* (May 1992).

C. Gordon, "Time-Domain Simulation of Multiconductor Transmission Lines with Frequency-Dependent Losses," *IEEE International Conference on Computer Design* (October 1992).

T. Guay, "Object-oriented Diagnosis," *Journal of Object-Oriented Programming* (October 1992).

G. Hoglund, E. Valcarce, L. Jansen, and L. Baillie, "ESSENSE: An Experiment in Knowledge-Based Security Monitoring and Control," *UNIX Security Symposium III Proceedings* (sponsored by the USENIX Association) (September 1992).

T. Hongsmatip and S. Hsu, "Influences of Backside Gold Conditions on Silver/Glass Die Attachment," *International Electronics Packaging Society Conference (IEPS '92)* (September 1992).

J. Ide and R. St. Amand, "PGA Failure Analysis Using Acoustic Microscopy and a Novel Sample Preparation Technique," *Eighteenth International Symposium for Testing and Failure Analysis (ISTFA '92)* (October 1992).

R. Jain, "A Comparison of Hashing Schemes for Address Lookup in Computer Networks," *IEEE Transactions on Communications* (October 1992).

R. Jain, "Myths About Congestion Management in High-Speed Networks," *Proceedings of the IFIP TC6 Fourth International Conference on Information Network and Data Communication* (March 1992).

A. Ladd, "Measuring Process Migration Effects Using an MP Simulator," *Scalable Shared Memory Multiprocessors* (Norwell, MA: Kluwer Academic Publishers, 1992).

P. Martino, "Simplification of Feature Based Models for Tolerance Analysis," *ASME Computers in Engineering* (August 1992).

T. Michalka and S. Poh, " An Electrical Comparison of Multimetal TAB Tapes," *IEEE Transactions on Components, Hybrids, and Manufacturing Technology* (August 1992).

B. Mirman, "Microelectronics and the Built-Up-Bar Theory," *ASME Journal of Electronic Packaging* (December 1992).

P. Moy, M. Hannigan, S. Bradley, J. Choi, and E. Allen, "Design and Process Considerations for Heatsink Attachment," *1992 International Electronics Packaging Society Conference (IEPS '92)* (September 1992).

V. Peng, D. Donchin, and Y. Yen, "Design Methodology and CAD Tools for the NVAX Microprocessor," *IEEE International Conference on Computer Design* (October 1992).

A. Philipossian, "Fluid Dynamics Analysis of Atmospheric Thermal Silicon Oxidation Reactors Using Dispersion Models," *IEEE International Electron Devices Meeting* (December 1992).

A. Philipossian, H. Soleimani, and B. Doyle, "A Study of the Growth Kinetics of SiO2 in N2O," *IEEE International Electron Devices Meeting* (December 1992).

M. Raven and D. Wixon, "Total Quality Management Using Vector Comparative Analysis," *IEEE International Technical Communication Conference* (October 1992).

M. Register and N. Kannan, "A Hybrid Architecture for Text Classification," *Fourth International IEEE on Tools with Artificial Intelligence (TAI '92)* (November 1992).

W. Samaras, "Futurebus+ Electrical Simulation," *OPEN BUS SYSTEMS '91* (November 1991).

S. Sathaye, "Architectural Support for Real-Time Computing Using Generalized Rate Monotonic Theory," *Journal of the Society of Instrument and Control Engineers* (July 1992).

S. Sathaye, "Distributed Real-Time System Design Using Generalized Rate Monotonic Theory," *Second International Conference on Automation Robotics and Computer Vision (ICARCV '92)* (September 1992).

M. Stick, "Systems of Linear Equations," *Six Sigma Research Institute* (April 1992).

S. Swan and D. Corliss, "Micro-Trenching During Polysilicon Plasma Etch," *SPIE's Microelectronic Processing '92* (September 1992).

T. True, "Volume Warping," *IEEE Visualization '92* (October 1992).

M. Tsuk, "Efficient Techniques for Inductance Extraction of Complex 3-D Geometries," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD-92)* (November 1992).

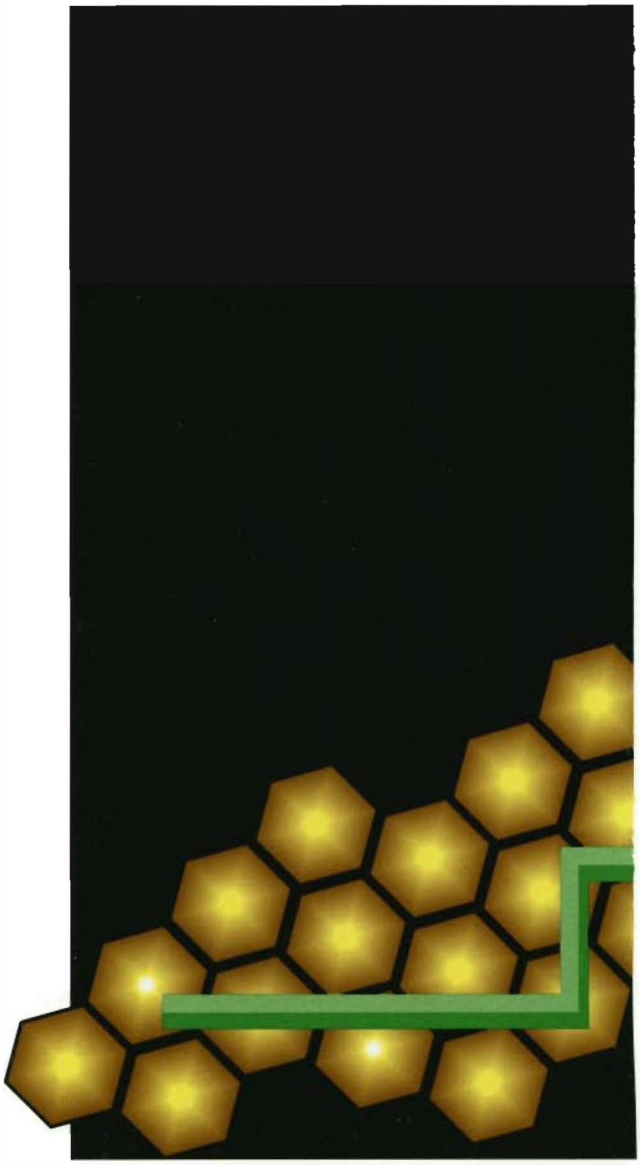
G. Wallace and R. Hagen, "Image Decompression Circuit for High-Speed Document Printing," *SPSE Sixth International Congress on Advances in Non-Impact Printing Technologies* (October 1990).

G. Wallace and B. Szabo, "Design Considerations for JPEG Video and Synchronized Audio in a Unix Workstation Environment," *USENIX Technical Conference and Exposition* (June 1991).

A. Vitale, "Issues in Speech Technology for Individuals with Disabilities," *Journal of the American Voice Input/Output Society* (July 1992).

A. Vitale, "Review of *Voice Processing* by Walt Tetschner," *Journal of the American Voice Input/Output Society* (July 1992).

E. Zimran, "The Two-Phase Commit Performance of the DECdtm Services," *IEEE Eleventh Symposium on Reliable Distributed Systems* (October 1992).



ISSN 0898-901X