Managing Organizational Handoffs with Empowered Teams

Kathleen K. Glass
Lucinda M. Sanders

High morale, strong teamwork, and proactive thinking (instead of reactive thinking) are essential catalysts for the development of products that meet or exceed customer expectations. The traditional organizational structure into which development work is segmented can either impede or enhance the ability of developers to form effective teams. Because of their usual hierarchical form and often rigid boundaries, organizational structures can also inhibit the development of a strong sense of ownership, thereby stifling creative, proactive problem solving. This paper describes a way to form empowered teams around traditional organizational boundaries, i.e., by organizing the team into a *subproject team*. The subproject concept was used effectively in the development of a hardware, firmware, and operating system platform for a new PBX. We did not create the usual artificial deliveries between hardware and software organizations through contractual obligations. Instead, we formed a subproject that encompassed all the development work needed to formulate, test, and deploy the new PBX vehicle. Quality metrics data for the subproject showed an increase in customer satisfaction, high team morale, and a reduction in necessary rework.

Introduction

Many development handoffs between organizations are achieved via written *contracts*, in which the sending and receiving organizations specify their delivery expectations. These contracts are appropriate at some handoff points. However, we believe that such contracts can also have a downside. By their very nature, contracts are often legalistic, and can cause finger pointing and accusations when a certain clause is not met. This, in turn, can lead to teamwork and ownership that are less than optimum among developers.

The subproject team concept is an alternative to rigid, organizational product development. In this paper, we describe the subproject concept and how it was deployed in the development of a large, complex PBX for the AT&T DEFINITY® Communications System product line. We also give a brief overview of the subproject methodology and present quality metrics from the PBX development. In addition, we share some comments from developer and customer retrospectives to highlight the keen sense of ownership and teamwork within the subproject. (Panel 1 defines acronyms and terms.)

It is important to understand that many of the subproject methodologies we describe are techniques that good team builders and developers use everywhere in AT&T. But the subproject concept goes one step further. It allows us, in essence, to form an organization that is based on the actual work at hand, rather than try to fit the work into the existing organizational structure.

Although this paper describes the use of the subproject concept for developing products across the boundary between hardware and software organizations, the concept can and should be used at other traditional organizational boundaries as well.

Project Background

Throughout this paper, we will draw on examples to illustrate the subproject

concept. Because we took these examples from a real PBX product development, we present some background information on the product before describing the subproject organizational technique.

Scope of the Project. The RISC (reduced instruction set computer) platform[1] subproject was chartered in late 1987. Its purpose was to coordinate the design, development, testing, and delivery of the RISC processor complex for the Generic 3 Release of the DEFINITY Communications System.

The hardware for the RISC processor complex consisted of the:

- Processor board (which is based on RISC technology).
- Memory boards.
- Duplication interface board. (This board provides the connection between the two fully redundant processor complexes that are used to ensure highly reliable operation for the DEFINITY system.)
- Mass storage boards (i.e., controller, disk, and tape).
- Network interface boards.
- System maintenance board.

Several of the boards had large firmware packages (i.e., built-in programming for the processor, network interface, system maintenance, and mass storage).

In addition, a real-time operating system was ported to the hardware. (*Ported* means existing software was adapted and translated to work on the new processor.) Significant development effort was needed on the operating system (i.e., kernel, drivers, etc.) to accommodate the new processor and peripheral boards. (The *kernel* is a set of programs for executing an operating system's most primitive or basic functions. A *driver* is the set of instructions a computer uses to transfer data to and from a particular peripheral, e.g., a memory device or a printer.)

The operating system also was enhanced with a new debugger that was compatible with the RISC compiler's optimization techniques. (A *debugger* is software that is designed to help detect errors in a computer's programs.)

The RISC subproject team designed and developed all the hardware, firmware, and software (i.e., operating system) needed for the new processor complex.

The Subproject's Customers. The RISC subproject team had several internal customers to whom deliveries were made (see Figure 1). These internal customers were functionally organized teams within our business unit, who added additional development before the final product was released to market.

Panel 1. Abbreviations, Acronyms, and Terms

customer-supplier model — a work process that emphasizes relationships between the customer and supplier, process input and output, and requirements and feedback

KNCSL — thousand noncommentary source lines of C-language code; does not include comment lines in the code

methodology — the processes, metrics, and documentation developed for a particular task or technique

MR — modification request

PBX — private branch exchange

PQMI — process quality management improvement; a seven-step methodology for process management and continuous improvement

RISC (pronounced *risk*) — reduced instruction set computer; an architecture in which the instruction set is reduced to a minimum to increase processing speed. The instructions used most often are built in; others are done by combining the built-in instructions.

Our largest customer was the application-software development community for the DEFINITY system, which consisted of about 115 developers. This customer required that the subproject software, firmware, and hardware deliveries be stable, so that development of the extensive application software could proceed on schedule.

Another major customer was the PBX testing organization. Its role was to certify the quality of the entire PBX product before the product was shipped to external customer sites.

Finally, we viewed the AT&T Denver Works in Colorado as an important customer. This customer (which we refer to as *the factory*) expected our designs to fit smoothly into its *just-in-time* manufacturing processes,[2] and to be easy to build and maintain.

The RISC Subproject Team. Over the life of the project, the subproject team consisted of 40 to 60 people who came from about 15 supervisory groups. We (the authors of this paper) were the co-supervisors for the entire subproject, and interfaced with all the developers

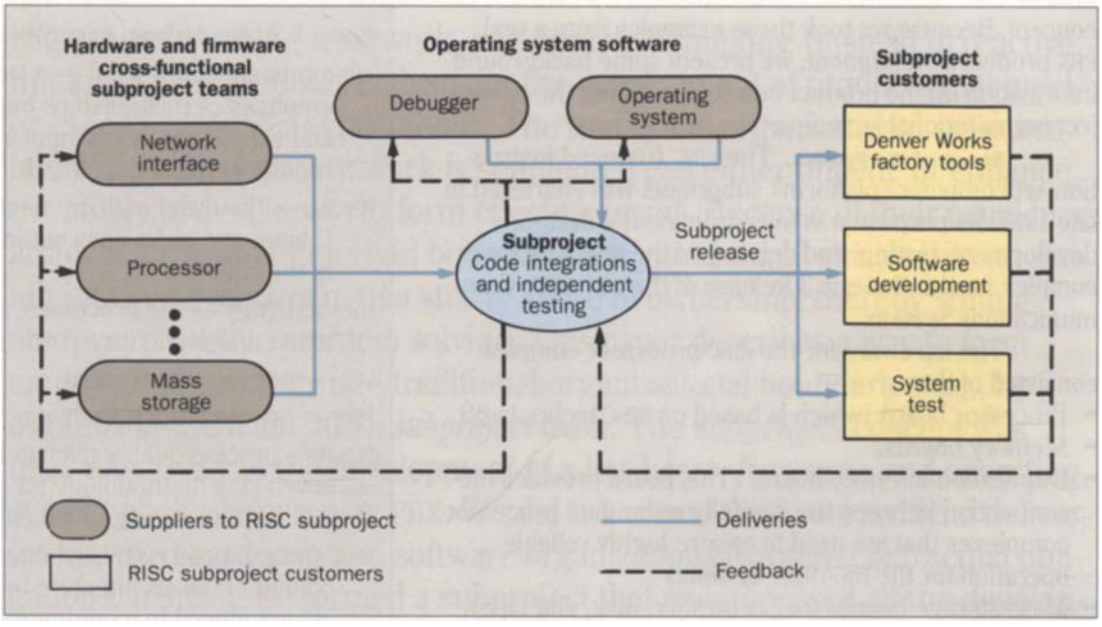

Figure 1. The subproject functional structure shows the deliveries to the subproject team from the various cross-functional teams and from the operating-system software developers. These deliverables are integrated together and then tested as a unit before being released to the subproject customers.

and managers involved in the development.

The RISC platform and the DEFINITY system software were developed in parallel, and the software was ported to the new hardware reasonably late in the development program. Hence, the software development community used interim hardware to develop the software on a parallel development track, while the subproject team developed the RISC platform. The parallel effort was designed to reduce the overall project-development interval, yet have state-of-the-art hardware incorporated in time for the product release date.

What Is a Subproject?

Now that we have provided background on the DEFINITY system project, we are ready to discuss the more structural and managerial aspects of subprojects. Here and in the sections that follow, we present the generic concept of a subproject and illustrate it with examples from the RISC subproject.

A subproject is nothing more than a team that is managed by a small number of supervisors and is responsible for a broad, easily separable, functional part of a product. Hence, a large product-development effort can have multiple subprojects that operate within the product boundaries.

In a subproject, organizational boundaries are meant to be ignored, and team members frequently come from many different organizations. The subproject team members *own* the development and are empowered to solve issues. Unfettered by rigid organizational boundaries, this sense of ownership leads to creative problem solving, high expectations, and improved employee satisfaction and morale.

In our case with the RISC processor complex, every team member's job was to identify and resolve *any issue* that was related to introducing the new processor complex boards into the DEFINITY system product. (The issues could be maintenance code and algorithms, duplication, factory tools, application interfaces, etc.) Our primary mission was to make sure that all issues about the processor complex were resolved, and that our subproject team made high-quality deliveries on time.

Subproject Principles and Techniques

It is important to acknowledge that just forming a team and calling the team a subproject is not enough. Just as important is to have team norms and values. When subprojects are started, such *guiding principles* should be stated explicitly by the subproject management. These principles tend to be reiterated many times by team members and, thus, become the main themes of the subproject.

As an example, these were some of the guiding principles for the RISC subproject:

- There will be no "throwing things over the wall" to another organization. We want to minimize the myopic tendencies of traditional handoffs between hardware and software, and we want to maximize accountability and ownership of issues. (*Throwing things over the wall* refers to the practice of creating and handing off a product or process whose design does not consider the needs or limitations of the other design, development, or manufacturing organizations and processes.)
- Because teamwork is of the essence, supervisors *must* act as the role models. Minimal escalation of problems beyond the supervisory level is a key ingredient to teamwork.
- Technical decisions must be made at the developer level, with consensus and buy-in. Empowered teams own their work!
- We will make use of subject-matter experts to improve quality and shorten intervals. As an example, while the developer is designing the circuit, independent people can be entering the circuit schematic into the computer, simulating circuit operation, and designing the board layout. Likewise, all models (including the first hardware prototypes) can be ordered and assembled by factory personnel.
- We will strive for perfection in all we do, providing excellent response—timely, knowledgeable, professional—to issues and questions that our customers raise. There is no substitute for a competent team—people make the difference!

Along with the guiding principles, subprojects must use specific *organizational techniques* to structure their work. The specific subproject structure is not as important as the agreed-on organizational techniques that have been well communicated and have the team's commitment.

The RISC subproject used the following organizational techniques:

- Our work will be viewed as a combined delivery to our customers and consists of hardware, firmware, operating system, and software tools. The subproject deliveries will have been completely tested as a unit by independent testers, according to extensive test plans.
- To promote further cooperation among subproject team members, we will form cross-functional teams within the subproject. Each such team will have milestones and demonstrations for which all members of the team are responsible.

For example, we formed a cross-functional team for each board of the RISC processor complex. The appropriate hardware, firmware, and software developers were members of each such team. For the *mass storage team*, for example, a key function to be demonstrated was: *saving PBX translation information to disk*. Even though, at the highest level, this was a software function, the hardware and firmware members of the team were just as responsible as the software member for achieving the milestone.

These demonstrations were important focal points for team members, and allowed them to broaden their scope beyond their traditional development responsibilities.

Figure 1 shows the functional organization for the RISC subproject. This structure followed the *customer-supplier model*,[3] where each member of the subproject team was a supplier and the subproject served as the single point of contact for the customers (i.e., the Denver factory, the software development organization, and the test organization).

Notice how the structure in Figure 1 minimized organizational interfaces and produced a package rather than pieces. This package was then delivered as a platform to our customers in software development, testing, and the factory. Early in the development cycle, both the customers and the suppliers agreed to the quality level and delivery date for platform releases.

Subproject Planning, Tracking, and Methodology

In addition to guiding principles and a team organizational structure, subprojects—like most AT&T projects—need rigor in their planning and tracking. This section describes some of the planning and tracking policies used by the RISC subproject.

While the ideas that we present in this section are considered a Best Current Practice in many AT&T development organizations, sometimes these ideas are shoved aside or overlooked. Hence, reiteration is appropriate. (In AT&T, a *Best Current Practice* is a practice that covers one part of the total development process, has substantial favorable experience associated with it, and is considered competitive by most practitioners. AT&T Bell Laboratories has a quality initiative that identifies, documents, and deploys these practices.)

One of the first things that the subproject supervisors should do is use process quality management improvement (PQMI)[4] to chart the subproject's

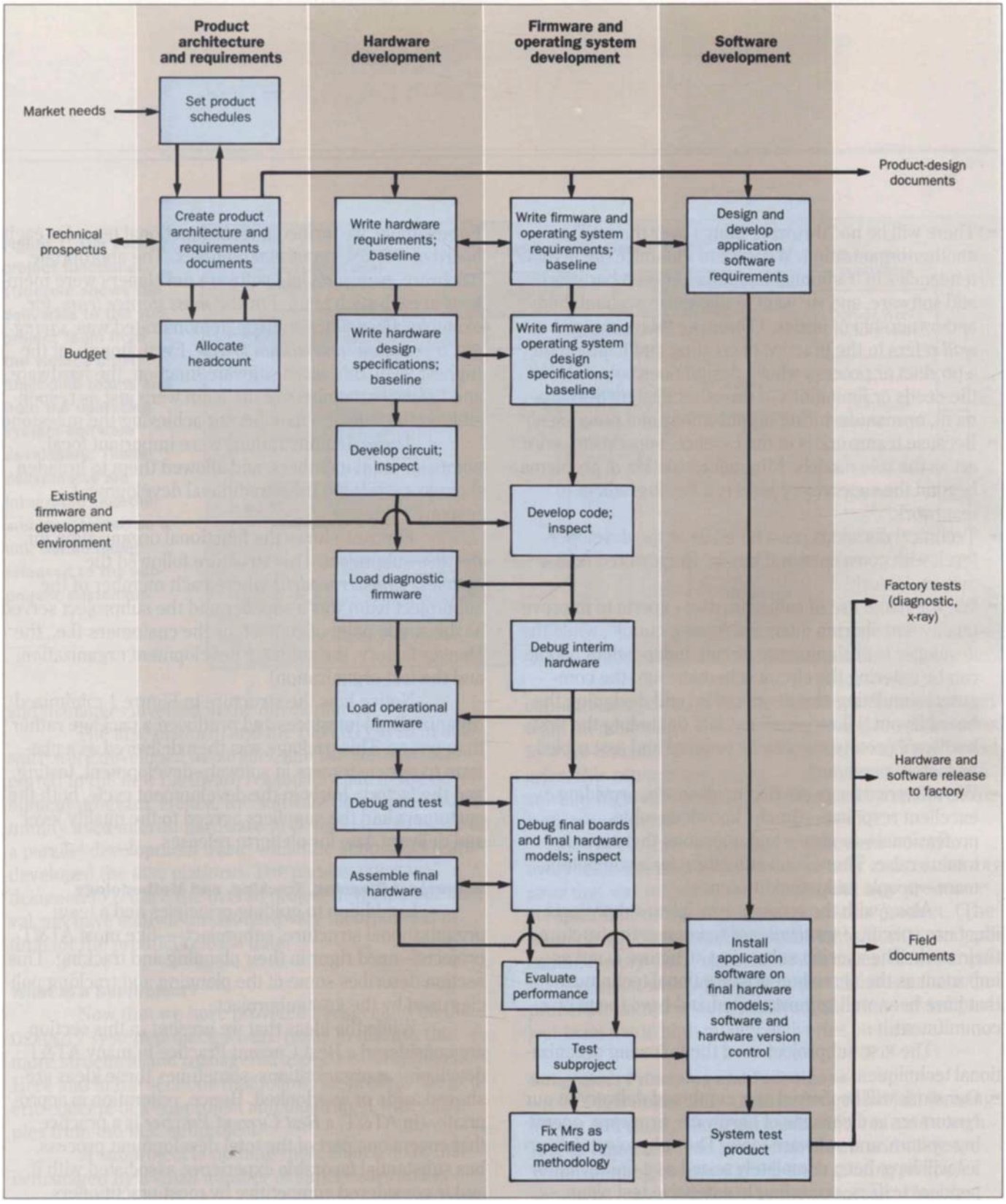

Figure 2. The RISC subproject PQMI diagram shows organizational handoffs for the RISC subproject development. Extensive cooperation between organizations was needed during design and development. Application software and the platform were developed in parallel, so a subproject test was needed before the platform was released to the application.

organizational handoffs. Such diagrams help managers broaden their thinking, leading to better planning and organization.

As an example, Figure 2 contains a high-level PQMI diagram that was created by the RISC subproject supervisors.

First, notice the extensive cooperation needed between organizations (i.e., for hardware, firmware, and operating system) during the design and development phases. By looking at the number of necessary interactions, we realized that the concept of not throwing things over the wall and the concept of a subproject for the hardware, firmware, and operating system effort could greatly improve the quality of our design and the initial development.

Second, we realized that, because the application software was being developed in parallel on this project, we needed a form of subproject test before we released the platform to the application customers.

Finally, as a result of drawing this diagram, we extended our concept of *customers*, broadening it to cover system test and the factory.

Besides having an understanding of the subproject's organizational boundaries, each subproject should also have a basic subproject development plan. This plan describes such things as deliverables, responsibilities of each subproject team member, and external dependencies on people outside the subproject. The plan should also document the subproject's demonstrations of key functions (e.g., *first phone call on RISC hardware*) and when they should occur. Subproject integration schedules, which show key synchronization points for the subproject, should also be included. In addition, methodology should be briefly described in this plan.

We recommend that the subproject supervisors write this plan and that supervisors in the subproject's customer organizations review it. However, we believe it is critical that the members of the subproject team set the dates contained in the plan.

Another important area for subprojects is the exchange of design and implementation ideas among team members. Much of the team's sense of ownership stemmed from the weekly, self-managed, design team meetings. In this forum of peers, team members discussed architectural issues, tracked action items, and refined the methodology. The designers followed the Best Current Practice development methodology (i.e., requirements, architecture, design documents, inspections, etc.) and were heavily involved in peer-level reviews at every step.

Finally, subproject teams must take the initiative in helping to head off customer issues before they happen. Therefore, it is imperative that the subproject supervisors be intimately involved with the various subproject customers. For the RISC subproject, for example, we helped write the overall PBX project plans, attended all project meetings, and tracked milestones in the project tracking database.

Quality Metrics from a Real Subproject

By now, readers should have some understanding of the organizational and planning aspects of a subproject. Therefore, we can take a quantitative look at the quality improvements that come from such an empowered subproject team.

This section provides a range of metrics from our RISC subproject. When selecting the quality metrics and results to be presented here, we decided to show a variety of metrics that we felt reflected overall quality. For example, we present traditional fault rates and customer-filed modification requests (MRs) that document troubles or problems found by product users. (We refer to these as *nonenhancement MRs*. An MR may also be filed to request an enhancement.) But we also believe that the high-quality design work we did up front was reflected in our ability to demonstrate key functions ahead of schedule. Hence, we present some schedule data as well.

Part A of Panel 2 illustrates the size (i.e., the complexity) of the RISC platform. This information is needed to understand fully the scope of the development effort and the significance of the quality metric results.

Part B of Panel 2 contains the subproject's MR data. It represents all nonenhancement MRs that were filed by customers after the first subproject delivery on February 1, 1990. (MRs filed to request a product enhancement are not included in the data.)

In Part C of Panel 2, we show cumulative software fault rates (since February 1, 1990) for the RISC platform firmware, operating system, and debugger. These rates are about ten times less than the typical software fault rate of two to three faults per thousand

Panel 2. RISC Subproject Quality Metrics

Here, we present quality metrics for the RISC processor development effort to illustrate the benefits of the subproject concept.

A. Project Complexity

This table illustrates the size or complexity of the RISC platform. KNCSL stands for *thousand noncommentary source lines of C code.*

Area	Size
Hardware	18 board codes
Firmware	131.5 KNCSL
Operating system	110.17 KNCSL
Debugger	38.14 KNCSL

B. Cumulative MR Data

Since February 1, 1990 (i.e., since the first subproject delivery), customers have filed the following MRs for changes other than enhancements:

Package	MR count
Hardware	0
Firmware	16
Operating system	38
Debugger	7
Total	61

C. RISC Subproject Fault Rates

Since February 1, 1990, the cumulative software fault rates for the RISC platform firmware, operating system, and debugger have been about ten times less than the typical software fault rate for similar software:

Unit	Fault rate
Firmware	0.12/KNCSL
Operating system	0.34/KNCSL
Debugger	0.18/KNCSL

D. RISC Platform Demonstrations

The subproject approach has enabled the developers of the RISC processor complex to meet or beat schedule dates:

Milestone	Original date	Actual date
Operating system boot	5/6/89	5/2/89
Operating system operational	5/15/89	5/2/89
Boot from disk	6/15/89	5/22/89
Boot from tape	7/1/89	5/22/89
Start application port	8/15/89	8/14/89
Primary port network phone call	9/15/89	8/17/89
Extended port network phone call	11/1/89	8/24/89
Start application development	1/1/90	11/1/89

noncommentary source lines (KNCSL) of C code.

Another important metric centers around the models conversion program where the new RISC hardware replaced the interim hardware that the software development community had been using. The hardware conversion went flawlessly. About 15 systems were involved; and each was converted successfully, with only about 2 hours of downtime needed to replace a processor complex. Since then, the models have been up and running 100 percent of the time. This represents a vast improvement over previous development efforts that required substantial rework during the prototype program, resulting in long spans of instability and model downtime.

A final metric shows the subproject team's ability to meet (or beat) expected schedules. Part D of Panel 2 contains our startup view of the important subproject milestone dates. Notice that it took only three days (i.e., August 14 to 17) to bring up an entire PBX application, instead of the projected one month (i.e., August 15 to September 15). Also notice that the subproject customers (i.e., the application software developers) were able to start work two months sooner than expected.

Retrospectives on Subprojects

We have discussed subproject concepts and philosophies and looked at some subproject quality metrics. Now, it is time to share some customer and developer comments about working with and on a subproject team. These remarks point out the customer focus that subprojects can have and the sense of empowerment team members can feel.

The DEFINITY system's project manager provided the following piece of customer feedback, which is representative of what our other customers have told us: *The quality of the delivery was outstanding. Historically,*

the creation of a new platform of this complexity is a project manager's nightmare because it requires close integration of hardware, firmware, and software. The traditional approach of each organization doing its part and handing it off to the next generally leads to many unpleasant surprises. As project manager, I was never aware of any problems or disconnects within the RISC subproject or between the subproject and the main project. When the RISC product was finally delivered, it met or exceeded all our goals and expectations for quality and performance. Interfacing with the subproject was a pleasure. The team had a personal goal of striving for excellence in everything it did—not only technically, but organizationally as well.

After we had completed our first phase of development (i.e., around April 1990), we conducted a developer retrospective, or survey, to obtain feedback about the subproject. These remarks were drawn from that retrospective, and from a later survey:

- *While this successful development used several tried-and-true quality practices, the overriding commitment and ambition to drive for a quality product was achieved through genuine caring and teamwork among peers.*
- *I enjoyed being involved with this project. There was seemingly low overhead and independence of subgroups. I liked the fact that we decided the milestone dates and were left responsible for managing the time in between. I also liked working in small, containable groups.*
- *The team became essentially self-motivating and, to a great extent, self-governing.*
- *We started as a rather unorganized group, trying to figure out what to do and how to proceed. Perhaps, in a way, letting the team struggle through this initial period of uncertainty on its own was itself a team-building exercise. Later, writing documents that required a lot of interaction between people from different departments and disciplines generated a strong sense of team participation.*
- *Organizing the project by functional subprojects, rather than along departmental lines, fostered teamwork by encouraging people from different areas to work together at an earlier stage in the project.*
- *It may sound silly, but an almost family-like feeling has developed among the team members.*

Conclusion

There are many ways to manage a team. This paper described the benefits of organizing around a subproject where organizational boundaries are minimized and the customer becomes the major focus. By using this approach, the DEFINITY system's RISC processor team achieved excellent results in terms of quality deliveries, customer satisfaction, and improved morale among team members.

Others who are interested in using the subproject approach can do so by scrutinizing their product development and identifying large, easily separable pieces of development work that span organizational boundaries. These work areas can then be organized into subprojects.

Acknowledgments

We did not invent the subproject concept. The technique has been used successfully at AT&T on other PBX development projects, perhaps on a smaller scale. It is also just a good, common-sense technique that good team builders intuitively understand and deploy. We acknowledge previous uses of what we are calling a subproject.

We would also like to thank Tonia Wright, Ken Roberge, and Tom Fisher for their keen management insights; and the members of the RISC processor subproject for their sense of ownership, energy, and hard work. Tom Fisher deserves special thanks for managing our subproject testing efforts.

References

1. J. R. Mashey, "RISC, MIPS and the Motion of Complexity," *Proceedings of the UniForum Conference*, February 4–7, 1986, Anaheim, California, USR Group, Santa Clara, California, 1986, pp. 115–124.
2. J. D. Carboy, G. Foo, L. P. Jones, L. E. Kinney, and D. C. Krupka, "Striving for Manufacturing Excellence at the Denver Works: A Summary," *AT&T Technical Journal*, Vol. 69, No. 4, July/August 1990, pp. 5–18.
3. R. B. Ackerman, R. J. Coleman, E. Leger, and J. C. MacDorman, *Process Quality Management and Improvement Guidelines: Issue 1*, Select Code 500–028, AT&T Customer Information Center, Indianapolis, Indiana, 1987.
4. AT&T, *PQMI: Tips, Experiences, and Lessons Learned*, Select Code 500–446, AT&T Customer Information Center, Indianapolis, Indiana, 1986.

(Manuscript received October 23, 1991)

Kathleen K. Glass *is a supervisor in the Integrated Systems Development Department with AT&T Bell Laboratories in Denver, Colorado. She is responsible for the development of high-performance processor subsystems for AT&T's line of* DEFINITY *Communications Systems. Ms. Glass joined the company in 1979. She has a B.S.E.E. from Virginia Polytechnic Institute and State University (Blacksburg, Virginia) and an M.S.E.E. from the University of Colorado (Boulder, Colorado).*

Lucinda M. Sanders *is a supervisor in the Advanced Systems Department with AT&T Bell Laboratories in Denver, Colorado. Her group develops software for the Generic 3 Release of the RISC platform. Ms. Sanders joined the company in 1977. She has a B.S. in computer science from Louisiana State University (Baton Rouge, Louisiana) and an M.S. in computer science from the University of Colorado (Boulder, Colorado).*