AT&T Technical Journal
Vol. 64, No. 1, January 1985
Printed in U.S.A.

System 75:

Project Development Environment

By T. S. KENNEDY, D. A. PEZZUTTI, and T. L. WANG*

(Manuscript received July 11, 1984)

The development of the AT&T System 75 office communication system required the coordinated effort of many designers working on a large number of individual components of the product. This article describes the project environment and methods created to accomplish this task. Emphasis is placed on the uncommon aspects of the project: the hierarchy of product specification documents that provided great flexibility in design decisions; the concept of a feature engineer that allowed for the vertical development of a feature by one person from feature specification to software code; the baselining and change control procedures that kept decision making at the lowest possible level; the tracking of progress so that prompt corrective action could be taken as problems arose; and the high reliance on electronic documentation and communication.

I. THE DEVELOPMENT PROCESS

1.1 Overview

The development of the AT&T System 75 office communication system spanned almost three years from product definition to introduction. The process consisted of a sequence of steps including requirements generation, external and internal design specification, imple-

* Authors are employees of AT&T Information Systems Laboratories, an entity of AT&T Information Systems, Inc.

mentation, integration, system testing, and field validation. The hardware and software components were designed in parallel to shorten the development time. Features were developed in phases so that the software architecture could be stabilized and refined incrementally. This was not unique to the System 75 project—much was learned from the experiences of other AT&T projects, such as System 85, Net 1000, *Dataphone*® II data communications service, *Horizon*® communications system, and the Transport Network, as well as evolving theories of software project management. However, in applying the theory and experience of others, methods were specifically defined to take advantage of a completely paperless office environment, to keep decision making at the lowest level, and to allow design flexibility that would accommodate innovation as the project progressed. This article provides an overview of the development process and emphasizes uncommon aspects that began with the project organization.

1.2 Project organization

The coordination of the effort of many developers working on individual parts of the project contributed to the success of the System 75 project and required as much organizational and managerial innovation as it did technical innovation. A decision was made at the start that the most effective way to organize was to minimize the interorganizational coupling and component deliveries. The functional development organization that evolved had three major *communities*: software, hardware (including both circuit and physical design), and test (including system test and field support). This organizational structure minimized coupling and fostered an *entrepreneurial* atmosphere because it encouraged the ownership of individual component designs and promoted innovation.

Figure 1 shows the simplicity of the component flow. The flow of components followed several major routes. (1) Requirements were generated jointly between the development organization and the systems engineering organization. (2) Hardware and firmware designs, originating from the circuit designers, passed through physical design to the engineering design and information organization, which generated manufacturing information. The models support group used this information to build and deliver circuit pack models to the designers for integration with the software. (3) At the same time, the physical designers transmitted hardware manufacturing information to the factory. (4) The software passed from the software designers through the integration and system test groups, and eventually the factory, undergoing testing at each step. (5) The factory delivered a complete system with both hardware and software for installation at a controlled introduction location. (6) The field support group received new releases

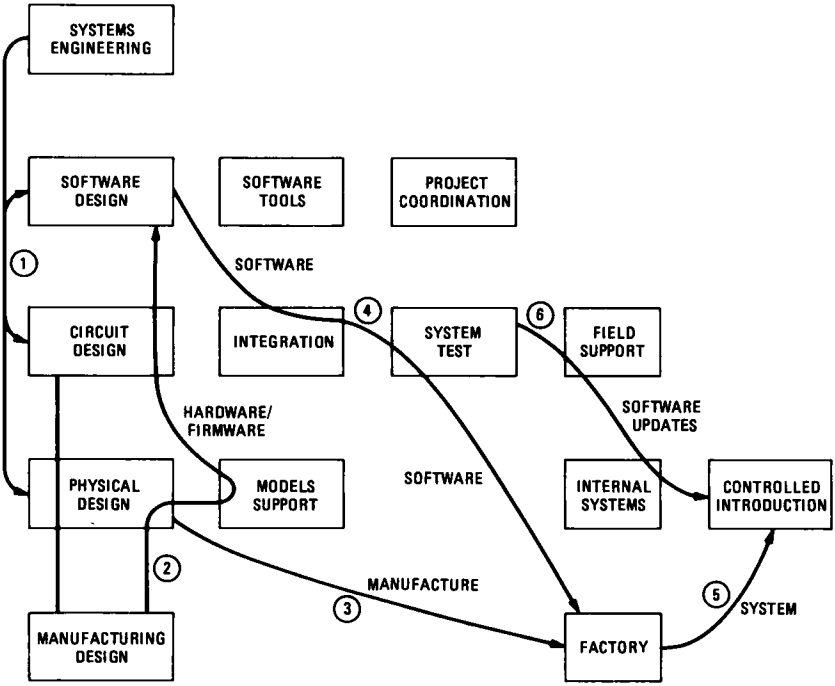

Fig. 1—Primary information transfer.

of software from the system test group, soaked the changes in internal systems, and then installed the new software in the controlled introduction location.

The component deliveries remained primarily within community boundaries. The transfer of integrated components between communities was controlled by *project documents.* The principal documents that were generated are described in more detail later in this section. These documents were generated through a process of negotiation and signature called *baselining,* and changes to the baselined documents (and other components) were closely tracked by a process called *change control.*

1.3 Planning the development

The coordination of planned activities at the project level was the function of the Project Coordination Group. To assure that all major project targets would be met, a *Project Development Plan,* which summarized major project development responsibilities, schedules, resources, and objectives, was negotiated with and signed by the appropriate management.

The major part of the plan was devoted to schedules that listed

important checkpoints or milestones for the project. To simplify the project management function, four schedule levels were defined for milestones on the project:

Level 1—A high-level view of the internal project development containing important checkpoints or milestones.

Level 2—Intermediate milestones, which represented major deliverables from one community to another.

Level 3—Detailed milestones, which represented deliverables between the individual groups within the communities.

Level 4—Internal milestones, which detailed activities of individual developers within the groups.

A computer-based tool called the Milestone Schedule Tracking System (MSTS),* which is described in Section II, was written to maintain and update these schedules.

The Project Development Plan contained Level 1 and Level 2 milestones, and estimates of the total project resource needs. Each community prepared and tracked a *Community Development Plan*, which contained Level 3 and 4 milestones, and detailed community resource estimates. The format and content of the community plans were tailored to meet the specific needs of the community rather than to follow an arbitrary, rigid project standard. By partitioning the schedules in this way, the burden of tracking schedules (i.e., the process of reporting completion of milestones and changes to estimated completion dates) was distributed. Individual communities retained control of their schedules and could adjust them as unplanned events occurred to meet the project commitments contained in the schedule for Levels 1 and 2.

1.4 Hardware development process

The hardware effort involved the design of 18 circuit packs; 2 consoles; and the cabinet, carriers, power, and interconnection arrangements to support these designs. The critical part of the hardware effort involved the design of new circuit packs and their integration with the firmware and software. The hardware community followed a traditional design program of building bread board, brass board, and prototype designs of the circuit packs. Typically, each circuit pack went through two or three design iterations before manufacturing information was transmitted to the factory.

The hardware community development plan contained Level 3 schedules for each circuit pack and hardware component. Critical milestones tracked deliveries to and from the engineering information and design organization for generating circuit pack manufacturing

* Acronyms and abbreviations used in the text are defined at the back of the *Journal*.

information, for availability of first circuit pack models, and for dates of transmittal of the manufacturing design information. The hardware community also held regular teleconference meetings with engineers from the factory to support the introduction of the new hardware designs into manufacture.

1.5 Software development process

A phased development process allowed the software to be built incrementally and avoided the pitfalls of a "big bang" integration. A total of four phases were planned, each lasting about six months. At the end of each phase, the software was delivered to system test and underwent rigorous testing. Critical performance factors and data were measured and the software structure was reviewed by the designers. Each phase was self-contained; as the product was designed and stabilized incrementally, the software architecture was refined.

Software feature completion was scheduled in two parts to avoid freezing the design too early. A Level 3 schedule, which defined the features and other capabilities to be completed during each phase, was baselined at the beginning of the development. A detailed Level 4 schedule for each phase, which defined the capabilities to be delivered per process, was built incrementally as the process design specification of each feature planned in the phase was reviewed and baselined. When the Level 4 schedules were complete midway through the phase, the design of the software components and the implementation effort were well understood.

Software integration was the focal point in scheduling and tracking. Both the Level 3 and Level 4 schedules were kept and tracked by the integration group and the milestones were established based on the integration dates.

1.6 Defining the product

A top-down design and planning process was followed throughout the development of the hardware and software. A comprehensive documentation effort was committed for the software design. The documentation hierarchy, shown in Fig. 2, reflects the two-dimensional aspects of the design process—system feature/capability specifications (external), and system architecture/structure specifications (internal). The activities in brackets <· · ·> represent the relationship of several major activities to the documentation hierarchy.

The *technical proposal*, which defined System 75 at high level and served as the major contract for the development organization, was written jointly by the development and the systems engineering organizations. The systems engineers provided input on feature content and product family planning and used data on market characteristics,

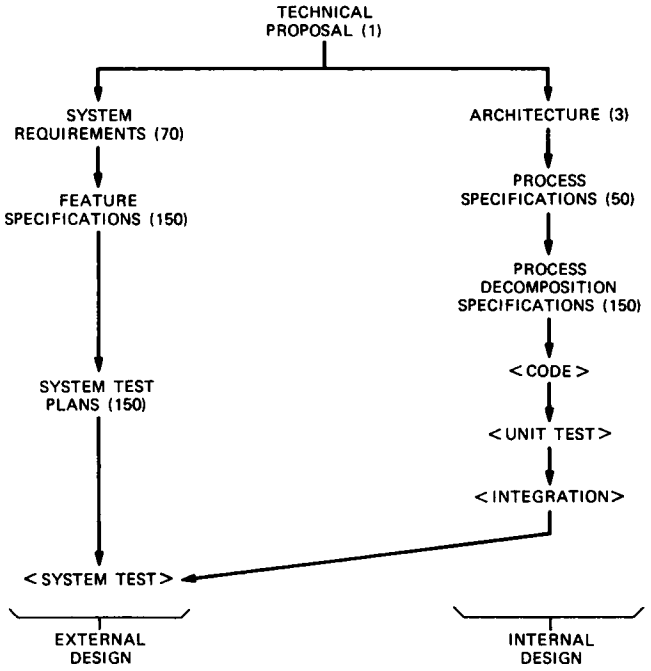

Fig. 2—Software documentation hierarchy.

profitability, and maintainability from the Line Of Business, Business Services, and Marketing organizations. The development engineers provided the technological content.

The organization of the technical proposal was structured around services and features and created the basic product structure for the remainder of the requirements. It specified the feature and system capacity at a high level. For example, it defined what terminals would be supported by the system and gave one-paragraph descriptions of features, such as call pickup, attendant recall, etc. This document was the first to be baselined, and along with the system requirements that followed, it provided the basis for later resource and schedule planning activities.

The next step in the definition of System 75 was generation of *system requirements.* This was a joint undertaking between the systems engineering and development organizations. The requirements expanded the feature definition from the single paragraph contained in the technical proposal to a few pages and captured the essence of the feature. User interfaces and other visible user details were defined with particular attention to achieving commonality with other members of the product family such as AT&T *Dimension*® System 85.

The software designers later prepared detailed external specifications, called *feature specifications*, which were reviewed and approved by the systems engineering organization for agreement with the requirements. The feature specification provided a user-level description of how the feature would be implemented. It included a definition of terms used in the feature operation, a description of feature interactions, and a list of administration requirements.

System test plans were derived from the feature specification and contained detailed scenarios for testing the feature operation.

The *architecture* document, the highest-level internal design document, permitted parallel development by providing conceptual unity in the software design by translating the technical proposal into a functional description of the system. It listed specific processes, libraries, and interface primitives, and defined key relationships between these software modules. It also described common strategies for security, reliability, recovery after failure, performance, and authorizations across the various elements of the system.

The software has a layered structure, with processes as the fundamental building blocks to enforce physical isolation. During the development, a *process engineer* was assigned to each process to review and enforce internal consistency. The *process specification* described the global aspects of a software process and fully specified all externally visible features. It defined how to initialize, invoke, terminate, and communicate with the process. The process specification was written as a set of manual pages. In general, only a few people were allowed to change the files within a process.

Each feature or user service required cooperation of many processes which in turn required the coordination of many designers, each responsible for a particular process. A *feature engineer* broke down each feature into the functionality of the various processes and specified their interfaces in a document called the *process decomposition specification.* It included a definition of how features map into the set of processes or modules, a description of internal process operation, lists of key data stored by the process, and sample sequence of message and process operation to illustrate the feature operation. The design was reviewed by the architecture team for consistency. The detailed integration schedule was committed only after the design was approved by the architecture team.

In summary, the feature engineer was responsible for the vertical design, and the process engineer was responsible for the horizontal implementation. Each was responsible for the software consistency in a particular domain and, together, they expanded the software in parallel. The software development tools[1] supported this process by allowing multiple developers to work in parallel on the software code.

1.7 Integrating hardware, firmware, and software

Several laboratory models of the System 75 were built to support the integration of the hardware, firmware, and software. These models had the major functionality of the final design. A typical model contained a control carrier, a test carrier, interconnection field, power supplies, and terminals. The control carrier had carrier slots on wider spacings to accommodate nonproduction prototype circuit packs, such as wire wrap, and to allow use of adapter and emulators for custom Very Large-Scale Integrated (VLSI) devices that were not yet being manufactured.

The basic skeleton of the models—the processing complex, carriers, power, interconnection hardware, and terminals—was delivered first. Then, the delivery of each new system component, such as a Central Office (CO) trunk circuit pack, was coordinated so that the hardware, firmware, and software elements would be integrated and tested on a single laboratory model before the remaining ones were equipped with the new component. At the start of the project, the delivery of the laboratory models was loosely controlled. As the project progressed, however, the importance of timely delivery of the models became clear and a models support group was formed with the primary assignment to build, deliver, and support the models. This group controlled the models delivery program with a document called the *Models Support Plan*, and used a computer-based inventory database to manage the process.

Figure 3 shows the typical flow of information for a circuit pack from the design phase to its use in the controlled introduction locations. This process was repeated many times as new circuit packs and firmware features were added. The models support group played a key role in assuring that the laboratory models were equipped with compatible versions of the hardware and firmware. The component flows followed the major routes outlined in Fig. 1 and show how a simple management concept becomes complicated when applied to a real problem.

1.8 Delivering the final product

Before the system could be made generally available, its quality had to be assured. It was stressed to the developers that everyone on the project was responsible for quality. However, the burden of proof fell on the system test group.

The goal of the system test group was to find as many faults in the system as possible before controlled introduction began. The system test group based its test plans on the feature specifications generated by the software community. Both manual and automated tests were performed on laboratory models.[2] This testing was supplemented by

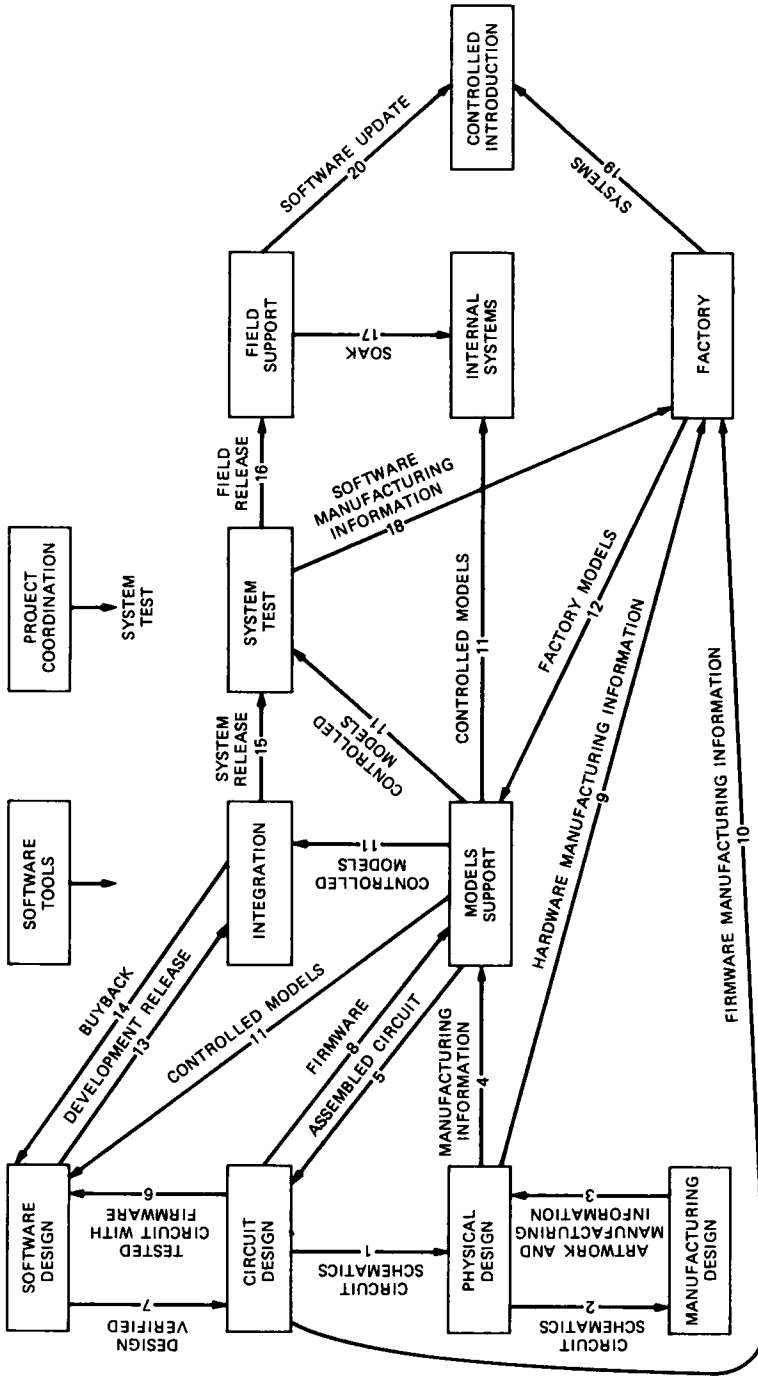

Fig. 3—Information flow for circuit pack design.

providing service to real users on two internal systems. The system test group also periodically released software to the factory. The factory quality assurance organization complemented the system test effort with its own test plans generated from the final product documentation and performed on systems in production at the factory.

The quality was evaluated quantitatively by tracking the number of validated faults against the number of predicted faults. The number of faults was initially predicted using a fault density per lines of software code that was determined empirically from prior developments. This prediction was modified as the actual fault density for System 75 was measured. The number of faults found was validated from data in the Modification Request (MR) system (described in Section II). When the predicted number of faults were found and all service-affecting faults were fixed, the software was ready for release to the controlled introduction locations.

The objective of the controlled introduction was to fine tune the product design and the delivery/support operations before manufacturing large quantities of systems.[3] In addition to evaluating product performance in a field situation, the controlled introduction was needed to test ordering, manufacturing, installation, training, and maintenance processes. Based on the actual field experience, particularly the customer reaction to the product's capabilities and the opinions of sales and service personnel, products were enhanced and/or corrected.

II. THE DEVELOPMENT ENVIRONMENT

2.1 Overview

Office automation based on a *UNIX*™ operating system computing environment was used throughout the project. Development support tools were written not only to help development work, which is commonplace for most projects, but also to improve communications and enforce project methods. Project methods were tailored specifically to be automated by computer-based tools and integrated with the communication services.

A series of regularly scheduled meetings (described in Table I) balanced the electronic communications. The meetings ranged from a semiannual project-wide review attended by all developers on the project to weekly community status meetings attended only by the supervisors.

2.2 Electronic office environment for project communications

The development of System 75 was carried out in a fully automated office environment. Every person on the project from director to clerk

Table I—Regularly scheduled meetings

Meeting	Frequency	Attendees/Purpose
Project review	Semiannual	All Set and maintain a positive mood for the project, make people aware of parts of the project they may have little contact with, and reinforce the team spirit of the project.
Supervisor review	Bimonthly	All supervisors Selected supervisors give a short presentation that focuses on the status of current and near-term deliverables with emphasis on informing the project community of any changes to previously disclosed schedules and of any existing problems that could affect future commitments.
Project status	Monthly	Selected representatives Review schedule commitments, identify problem areas, and make decisions on project-level issues, redistribution of responsibilites, etc.
Software status	Weekly	Software/system test supervisors Discuss and resolve software community issues.
Hardware status	Biweekly	Hardware supervisors/drafting representatives Review drafting status, discuss and resolve hardware community issues.

had a video display terminal, which became as important as the telephone.

The backbone of the office automation service is a collection of services named the Personal Communication Services (PCS). The PCS services are simple enough to be learned by casual computer users yet contain enough functionality to serve the needs of more sophisticated ones. These commands are characterized by a common user interface that is oriented towards a good human interface rather than one easily manipulated by a program. Extensive prompting and feedback are provided, as well as terse forms for more experienced users. Machine-dependent parameters, such as logins, system names, and directories, are hidden. The services can be customized to the personal preferences of each user.

The communication services of PCS fall into several broad categories: electronic mail service, calendar and reminder services, bulletin board service, and other miscellaneous services.

The *electronic mail* service provides for preparing, sending, reading, and filing messages between individuals and groups. Addressing is done by name (e.g., t.j.watson) rather than by the convention of system!login (e.g., hocsh!tjw). A mailing list capability aids individuals in sending information to special interest groups or organizational mailing lists.

On the System 75 project, the electronic mail service was used for a variety of purposes from a simple reminder to the transfer of technical information, either in the form of answers to questions or as a complete

document transfer. The asynchronous aspect of electronic communications eliminated "telephone tag," allowing people with busy schedules to exchange detailed technical correspondence in a timely fashion.

The *calendar* and *reminder* services provide for logging and reminding of future events. A project calendar was created to list project events. This service was also used extensively on an individual basis as a personal time management tool.

The *bulletin board* service provides for posting public messages in a central location. Many special interest bulletin boards were created to list such diverse items as meeting minutes, computer tools news, and want ads. The bulletin board was an easy way to distribute information project-wide and contributed to the feeling of camaraderie that was fostered on the project.

Other miscellaneous services include a profiler command to customize user environment, a directory assistance program, and a message display program that simultaneously displays mail messages, calendar, and a clock in windows on a locked user's terminal.

2.3 Baselining and change control

The Project Coordination Group had the responsibility of keeping the project on track—both in design content and schedule. The philosophy was to do this in an unobtrusive way and provide as much autonomy to the various communities as possible. This was accomplished by establishing formal methods for baselining design information such as requirements and feature specifications, for controlling design changes, and for tracking schedules that balanced the producer's desire for complete freedom of design and flexible schedule dates with the consumer's need for unchanging designs and firm schedule dates.

Baselining is the process whereby the current state of a design is captured to serve as a baseline against which changes can be made. The objective of any baselining process is to ensure that the design has been adequately reviewed to minimize future changes. The baselining process (and change control process) is not unique to the System 75 project; it was adapted from the experience of other projects. The baselining process (Fig. 4) begins with the assignment of a document number, goes through a review process that includes a formal sign-off step, and culminates with delivery of the approved document to the project library.

The review process was the key to the baselining process. Two types of processes were found to be effective: a design review and a circulation review. Both types of review were held at a peer level. The design review consists of a formal meeting that follows a rigid format with roles defined for a *scribe*, *moderator*, *presenter*, and *reviewers*. The

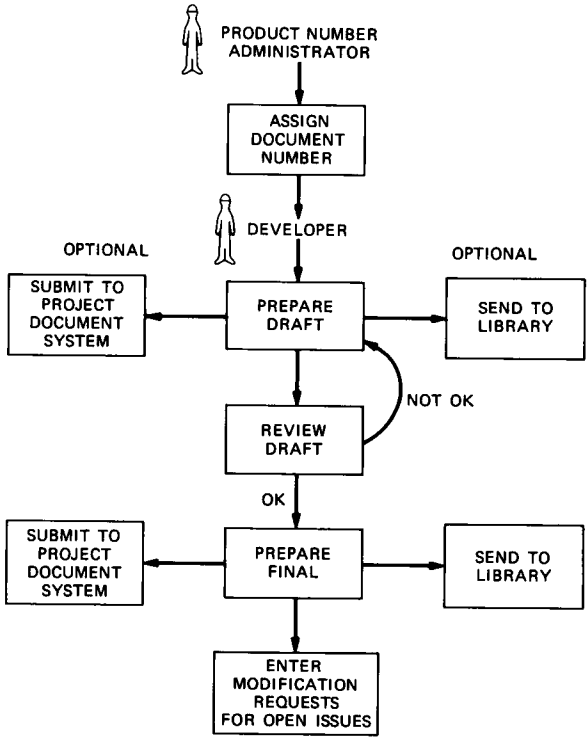

Fig. 4—Baselining process.

circulation review consists of circulating a draft of the document to *primary* and *secondary* reviewers. The primary reviewer consolidates the comments from the secondary reviewer. This reduces the writer's burden of resolving possible conflicts within a community.

The purpose of both types of reviews is to determine whether the document is acceptable (possibly with some changes), unacceptable, or acceptable with some open issues unresolved. In this latter case, the document would be baselined and modification requests entered immediately to record the open issues. This allows a designer to use information in documents that is correct without waiting for all issues to be resolved, thereby hastening the development process.

A project document library was created to serve as a repository for all baselined documents for the System 75 project. Both paper and electronic copies of these documents were kept and were readily available to persons who had a valid need for the information contained in them.

A computer-based Project Document (PD) system was created to retrieve and store copies of the electronic documents and to print a

report of both paper and electronic documents in the library. A database was used to maintain information associated with the project documents and to support the change control process. Both baselined and draft (under review) copies of documents were stored electronically on a single computer system by the PD system. Users on all systems had access to these documents over the computer communications network, providing quick dissemination of information.

Once a document or design was baselined, a formal change control process began. Requests for changes because of either enhancements or errors in the design were tracked via a modification request. An MR is simply a record that contains a description of the resolution of the problem.

The resolution process for MRs (Fig. 5) emphasized the importance of the individual developer in making decisions. In the majority of the cases (the middle route in the figure), MRs were resolved by negotiation between the person assigned and the affected persons—there was no separate review team chartered to approve the resolution of the MRs. The viability of this approach was proven since, during the course of development, developers made few incorrect decisions that required subsequent escalation to correct them.

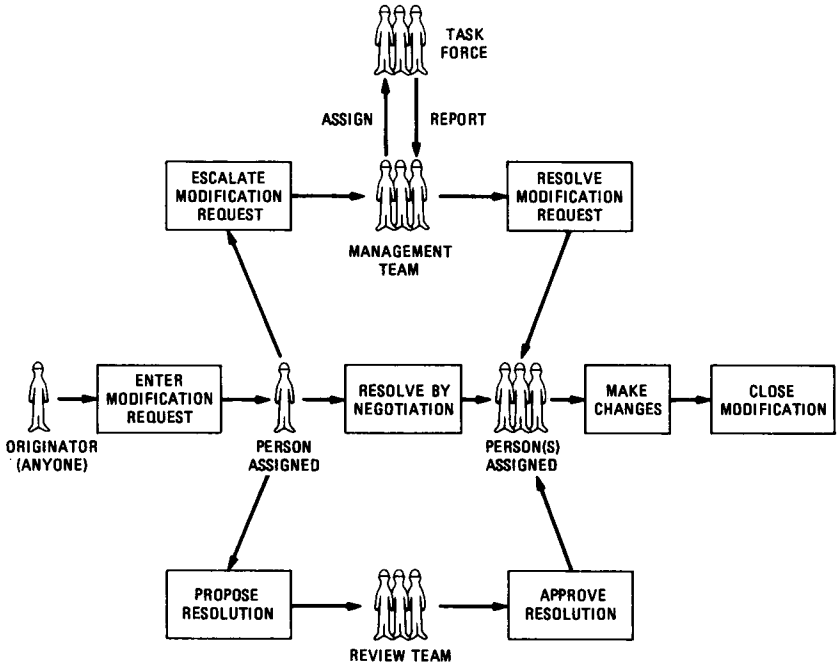

Fig. 5—Resolution process for modification requests.

Some documents, however, such as the technical proposal and system requirements, used the more traditional standing review teams to approve the resolution of MRs. Also, as the product design came closer to completion, additional review teams were formed, especially to assign priorities to fixing bugs and to decide which enhancements should be added. Representation on these boards was limited to a single individual from the relevant communities. Finally, in certain cases, the resolution of MRs was escalated to a management team that would resolve the MRs themselves or create a task force to propose a resolution.

All MRs were processed electronically using the Change Management Tracking System (CMTS) as a tool to store and track MRs. CMTS is a generic system developed by AT&T Bell Laboratories and is currently being used by many organizations throughout AT&T Information Systems Laboratories and AT&T Bell Laboratories. It provides database storage and retrieval of MR information.

To minimize user training, a collection of commands was written to provide a PCS-like interface to the MR database in place of the standard CMTS commands. These commands also allowed greater local control of the MR distribution, required less administrative overhead than the current version of CMTS, operated in a multimachine environment, and provided for completely paperless MR distribution.

2.4 Schedule tracking

Schedule tracking has received a great deal of attention in the project management community, and a large number of computer-based tools, based on an activity network analysis that calculates critical paths, early and late start dates, slack, etc., are available to assist this task. These tools provide valuable information at the start of a project, such as the dependencies of various activities, but can consume large amounts of resources if they are used to track the project as plans change. A simpler approach was undertaken to track checkpoints or milestones. The Milestone Schedule Tracking System (MSTS) was written for this task.

MSTS provides extensive reporting capabilities that have been integrated with the electronic mail service of PCS. The reports can be sorted by date or field data and milestones retrieved based on specific data values. User-defined reports can be created. A typical section of the most common report is shown in Fig. 6.

The milestone schedules were created on a cooperative basis between the various communities. Activity networks, work breakdown structures, and other scheduling techniques were used to create the initial set of consistent milestones. Once the milestones were baselined in

```
Sample                    Milestone Schedule Report                    Issue 2.0
                            Level 1-2 Milestones                        05/03/84

                                                  -----Completion Dates----  s
        Milestone              Cont/Prod/Cons  Original Previous Current  t
        ---------              --------------  -------- -------- -------  -

                            < Hardware Development >

        Interface Circuit Pack
        --------- ------- ----
hw0103  Model Assmbld  TJW/TJW/AGB              04/07/84 04/20/84 --------C
hw0104  Model Tested   AGB/AGB/AGB              04/21/84 05/04/84 05/01/84L
hw0106  EDI            TJW/TJW/dr               04/28/84 05/11/84 05/08/84
hw0107  First Ship     TJW/dr/at&t              12/31/84 -------- --------
                                              >=hw0106 + 8 months: 01/08/85J
```

Fig. 6—Sample Milestone Schedule Tracking System report.

the project development plan, they were updated and reviewed on a monthly basis at the project status meeting.

Each milestone had several key attributes: a *contact*, typically a supervisor, who was responsible for reporting the milestone completion and any changes to the scheduled date; a *description* that briefly defined what completion of the milestone meant (in practice, some incomplete definitions led to disagreement about whether a milestone was complete); and three estimates completion dates—*baselined, latest plan,* and *current estimate.* Prior to each review, the contacts provided a current estimated completion date for the milestone. If this date was different than the baselined date, the milestone was discussed at the meeting and, if all agreed, the new date was called the latest planned completion date. If the new date was unacceptable, a solution was devised to complete the milestone on time or to create an alternative plan.

This method, however, proved cumbersome in practice because it was often difficult to agree on schedule changes. MSTS has subsequently been modified to track the *original, current,* and *previous* completion date estimates for each milestone. Because this requires less coordination, the tracking interval was reduced from one month to one week to promote faster response to schedule problems. Also, the new method called attention to discrepancies between the current and previous estimates and thereby provided a timely record of project facts.

III. SUMMARY

The development of System 75 drew upon the experience of many other projects that used formal project management methods. Several ideas proved very valuable to the completion of the project: well-defined goals based on a hierarchy of product design specifications and development plans; progress tracked closely so that prompt corrective action could be taken as problems arose; baselining and change

control procedures that stressed keeping decisions at the lowest level possible; computer-based tools specifically tailored to augment the development process; and the degree and timeliness of communications obtained from an efficient, paperless electronic information management and communication service. These ideas are continuing to be used and will be improved for future work on System 75.

REFERENCES

1. T. J. Pedersen, J. E. Ritacco, and J. A. Santillo, "Software Development Tools," AT&T Tech. J., this issue.
2. C. J. Lake, J. J. Shanley, and S. M. Silverstein, "GAMUT: A Message Utility System for Automatic Testing," AT&T Tech. J., this issue.
3. M. A. McFarland and J. A. Miller, "Introduction Activities and Results," AT&T Tech. J., this issue.

AUTHORS

T. Scott Kennedy, B.S. (Mechanical Engineering), 1971, Lehigh University; M.S. (Mechanical Engineering), 1972, University of Michigan; Bell Laboratories, 1972–1983; AT&T Information Systems, 1983—. Mr. Kennedy has worked on the physical design of key telephone systems and the *Horizon*® communications system, and on the development of computer-based tools for project communications. His assignment since 1981 has been as a Member of Technical Staff, System 75 Project Coordination, where he is responsible for planning and tracking and for the development of methods and tools to support these processes.

David A. Pezzutti, M.S. (Electrical Engineering), 1970, Brown University; M.B.A., 1980, Rutgers University; Bell Laboratories, 1969–1983; AT&T Information Systems, 1983—. Mr. Pezzutti has worked on circuit design, software and firmware programming, and in management. He developed and managed the development of a variety of central office maintenance systems for electronic and electromechanical systems that are part of the CAROT System, such as the Remote Trunk Test Unit, Remote Office Test Lines, and Interrogator and Responder technologies. His assignment since 1982 is as Supervisor, System 75 Project Coordination, where he is responsible for planning, tracking, and external product information. Mr. Pezzutti holds five patents. Senior Member, IEEE; member, Sigma Xi, Tau Beta Pi.

Tse-Lin Jack Wang, B.S. (Electrical Engineering), 1964, National Taiwan University, Taiwan; M.S. (Electrical Engineering), 1969; Ph.D., 1970, University of South Carolina; Bell Laboratories, 1970–1982; AT&T Information Systems, 1983—. Mr. Wang was initially engaged in the exploratory work for business communications systems. In 1975, he joined the software development group for the initial *Horizon* communications system development and was appointed Supervisor of that group in 1978. Since 1980 he has worked on System 75 software development. He supervised switch software planning early in the project and led development groups doing call processing and maintenance software and software product delivery. Member, Eta Kappa Nu.