Proceedings of the Second European Conference on Computer-Supported Cooperative Work
Bannon, L., Robinson, M. & Schmidt, K. (Editors)
September 25-27, 1991, Amsterdam, The Netherlands

Distributed Computing and Organizational Change Enable Concurrent Engineering

Susan Frontczak, Kathy Miner
Hewlett-Packard Company, U.S.A.

Concurrent Engineering is the latest buzzterm for bringing products to market faster. This paper explores how a distributed computing environment can enable concurrent engineering. The following topics are covered:

- Definition of concurrent engineering and how it applies to non-technical as well as technical companies.
- Case studies of companies who have harnessed their distributed computing environment to foster multi-departmental teamwork and thereby reduce design cycles.
- Suggestions for overcoming organizational and technological barriers to concurrent engineering.

1. Introduction

The purpose of this paper is to illustrate that concurrent engineering is possible through a combination of distributed computing technology and organizational change.

This paper is organized as follows: current industry awareness of concurrent engineering; definition of terms; three case studies of concurrent engineering practiced today; technological and organizational concurrent engineering enablers; and key findings.

1.1 Industry Awareness of Concurrent Engineering

Robert Glasier of Intergraph Corporation defines concurrent engineering as the teamwork approach to engineering that brings all disciplines together from the outset of the project: product development, engineering, purchasing, manufacturing, marketing and finance (Glasier, 1990). It is intended to help product developers consider all elements of the product life cycle, including quality, cost, schedule and user requirements (Edwards, 1990).

An article in the March 22, 1990, issue of Machine Design entitled "Teamwork in Real Engineering" described how concurrent engineering was used to cut the development time on General Motor's LT-5 engine for the ZR-1 Corvette. By involving engineering and other functions concurrently in the development of the engine, GM was able to cut development time from the traditional seven years to four years.

The only collaboration technology enabler described in the Corvette engine article was the use of a FAX machine! It was used for transmitting design changes between the design teams in Detroit, Michigan and Hethel, England, and the manufacturing team in Stillwater, Oklahoma (Stinson, 1990).

Another article on concurrent engineering appeared in the April 30, 1990, issue of Business Week. The article was entitled "A Smarter Way to Manufacture: How 'concurrent engineering' can reinvigorate American industry. The article states:

> "The potential advantages of concurrent engineering have been recognized for decades. But earlier calls for it were thwarted by **middle management fiefdoms** and by the **lack of computerized tools** to spur cooperation between departments. Now that such tools are emerging, top management is cracking down and forcing design and manufacturing, in particular, to collaborate." [empahsis added] (Port, 1990)

1.2 Interaction Between Technology and Organization

Much of the research on concurrent engineering emphasizes organizational issues (Hauser, 1988; Liker, 1986; Takeuchi, 1986; Turino, 1990). This paper explores the interaction between organizational issues and technology issues, especially distributed computing technology. Technological support for the teamwork required by concurrent engineering is highly relevant to the study of computer supported cooperative work.

2. Definitions

2.1 Concurrent Engineering

We define concurrent engineering as: People working in parallel toward a common project goal, who gain productivity by communicating both with each other and through a shared body of data. Our definition is broad to illustrate that concurrent engineering principles apply not only to engineering projects, but also to any product development effort, where the product could be a software package or even a financial portfolio or legal brief.

Literature on concurrent engineering typically describes three major benefits: reduced time to market, higher product quality, and reduced cost (Edwards, 1990). Accompanying these three benefits is the creation of a more innovative product that meets real customer needs. Our case studies describe benefits in terms of time, quality and cost.

2.2 Distributed Computing

Distributed computing is an environment that provides users and applications with transparent access to all resources connected via a heterogeneous network. Resources include data, applications, compute power, I/O devices, services, and knowledge held by other network users (Seybold, 1990).

We identify three levels of communication in distributed computing (see Figure 1): resource-to-resource, person-to-resource, and person-to-person.

At the resource-to-resource level the user need not know that communication is taking place at all. Examples include automatic consistency management of distributed databases, remote procedure calls in distributed applications, and object/agent communications between applications.

Person-to-resource communication implies that all the power on the network is available to the individual user, not just the power of the local terminal or workstation. Example technologies supporting this level are browsers and filters for databases, print spoolers, and compute servers.

Examples of technologies supporting person-to-person communication are electronic mail, electronic conferencing, and group calendars. Transparent access at this level implies that the users can easily reach one another, regardless of hardware or location. Person-to-person communication is addressed by various computer supported cooperative work or groupware products. However, significant strides are being made in enabling concurrent engineering through the use of distributed computing at all three communication levels.

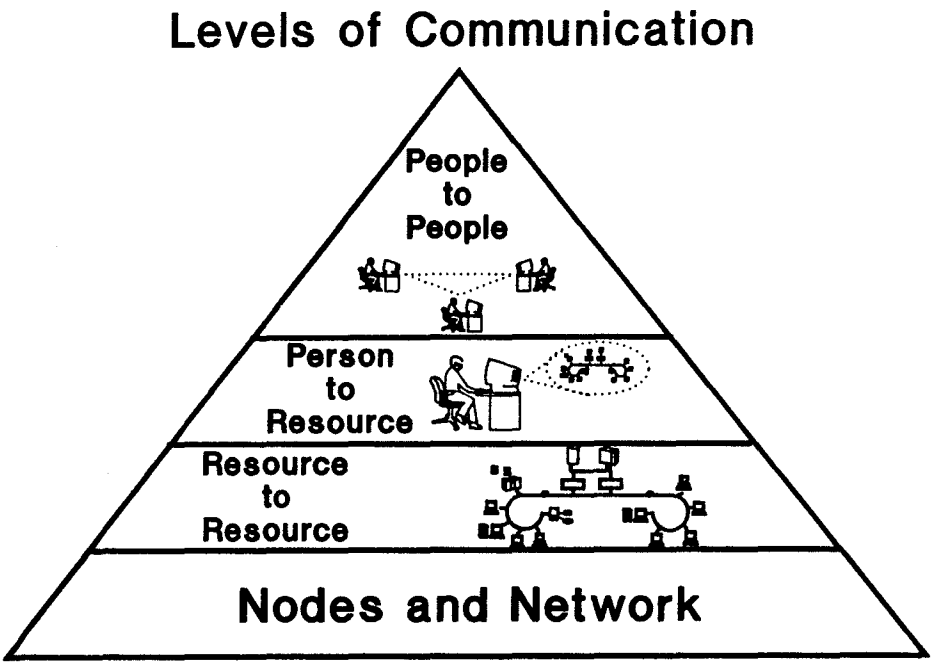

Figure 1

3. Case Studies

Our research includes contacts with 18 project teams at 11 companies that are now practicing concurrent engineering.

We present three case studies here, representing the best examples of concurrent engineering that we found.

3.1 Case Study 1: ME/EE Design

3.1.1 Context

Eight departments participate in mechanical and PC Board design review at HP's Apollo Systems Division. The departments include manufacturing, signal integrity, IC design, components engineering, and mechanical engineering. The team is highly geographically dispersed. For the Series 10000 computer, some parts were designed in Massachusetts, built in Scotland, and assembled in New Hampshire. Furthermore, if expensive ME CAD and EE CAD software are used to view the parts, licenses to this software cost from $40K to $80K per seat.

However, the reviewer of a design does not need to edit it, merely to examine it. Two tools were developed to meet the reviewer's needs: one to view the output from the ME CAD packages (CADACS) and another to view the output from the EE CAD packages (VIEWBOARD). A dialog of comments can be annotated right on the drawing, viewable by all reviewers as well as the originator. These viewing tools have some 'smarts'. For example, VIEWBOARD can highlight the path of a given signal; CADACS can render a 3-D view of a mechanical part.

3.1.2 Benefits Realized

CADACS and VIEWBOARD make information available to reviewers without requiring the expensive licenses needed by the designers who are editing the data. By eliminating the need for eight EE CAD licenses, the Series 10000 development team saved over $400K.

Savings also occur when errors are caught before parts are actually built. A technical writer noticed on one drawing that the front panel logo was upside down on a mechanical part. A simple email message saved many dollars in scrap parts.

There are over 150 active users of CADACS, and 200 active users of VIEWBOARD at Apollo Systems Division today.

3.1.3 Key Technologies

The team identifies DOMAIN's global file system as critical to their success. (DOMAIN is the operating system on HP's Apollo Systems Division computers.) For example, a designer in Massachusetts simply mails an electronic message to a reviewer in Scotland pointing to a drawing in the global file system that is ready for review. The reviewer then accesses the drawing using CADACS or VIEWBOARD as if it were available locally.

The global file system eliminates the need to copy files or worry about whether the most recent version has been sent to the reviewer. It also makes review comments visible to all since comments are recorded on the single copy of the drawing; as contrasted with markups on a copy of a drawing.

With the coming of the ANDREW FILE SYSTEM chosen by the Open Software Foundation for its Distributed Computing Environment (OSF DCE), heterogeneous environments will also enjoy the benefits of a global file system.

3.1.4 Key Organizational Factors

The desire for asynchronous, multi-disciplinary team review formed the impetus for developing CADACS and VIEWBOARD. The charter of Apollo Systems Division's tools development group, allows them to make decisions that enhance communication company-wide.

Also, due to a recent reorganization, the CADACS and VIEWBOARD teams were relocated very near each other. The co-location led to cross-fertilization of ideas. For example, inconsistencies in user interface between CADACS and VIEWBOARD

(such as different viewing mechanisms) became apparent. They have recently started an investigation of ways to view EE information (i.e. the ME viewing tool CADACS).

3.2 Case Study 2: Software Development

3.2.1 Context

This is a composite case study of software development projects at HP's Apollo Systems Division, Honeywell, and Motorola.

Software developers need to address concurrency when releasing multiple versions of code. Released versions need to be maintained while, simultaneously, one or more future versions are under development. Fixes need to be incorporated in the future software as well as added to the released software.

DOMAIN SOFTWARE ENGINEERING ENVIRONMENT (DSEE) is a software package developed at HP's Apollo Systems Division. It manages large-scale development projects involving teams of managers, engineers, and technical writers.

DSEE works at all levels of communication described in Figure 1. It supports person-to-person communication through its task manager. DSEE provides resource-to-user communication via automatic triggers that notify people (via electronic mail) about changes to specific files. And DSEE applies resource-to-resource communication by monitoring dependencies between software modules and using all the computers on the network to build the product from its many modules.

3.2.2 Benefits Realized

Developers who use DSEE at Apollo Systems Division report needing fewer face-to-face meetings. Individuals have the power to make decisions that would otherwise require discussion: the software developers know that the tool will automatically notify appropriate team members of changes, and previous versions of the software can be retrieved if necessary.

Honeywell, Motorola, and Apollo Systems Division all report dramatic time savings as a result of using the distributed build capability of DSEE (described above). The IACD division of Honeywell noticed a reduction in build time from six hours for a serial build on a Series 3000 workstation to four hours on two Series 3000's (Merrill, 1990). The build time for a product at Apollo Systems Division decreased from an hour and nine minutes to sixteen minutes (McCourt, 1990).

3.2.3 Key Technologies

DSEE, coupled with the advantages of the DOMAIN global file system and merged account registries, is the key technology used to manage the complexity of simultaneous software development.

3.2.4 Key Organizational Factors

The most important organizational factor reported by Apollo Systems Division developers is trust. Team members have learned to trust each other to make decisions on their own. Each developer is responsible for fully testing his/her own software modules prior to putting them in the pool of modules that is used by other developers. This trust in each other (coupled with trusting the tools ability to reconstruct an earlier development state), increases productivity by maximizing autonomy.

The organization at Apollo Systems Division developed simultaneously with the DOMAIN technology. Therefore the distributed computing environment and the organizational structure were formed together. When existing organizations introduce distributed computing technology, it sometimes forces the organization as a whole to coordinate. In September of 1989 Motorola combined 750 HP Apollo workstations into a network spanning Schaumburg, Illinois, Ft. Worth, Texas, and Plantation, Florida. They report:

> "We formed an active working committee for the Apollo layer to consolidate and unify security policies across the three sites. Conventions such as account creation and deletion, password maintenance, modem access, and group naming needed to be unified and made consistent." (Dolikian, 1990)

3.3 Case Study 3: Rules of Thumb Database

3.3.1 Context

An automotive manufacturer has piloted a database which contains rules of thumb for designing parts of a car. The database contains approximately 30 rules of thumb categories arranged in a hierarchy. For example, a manufacturing process category contains an assembly category.

People in various engineering disciplines enter rules of thumb into the database. An engineer designing a new part can query the database for similar parts developed in the past and discover rules of thumb used and the reasons they were used. (The reasons are important because if the reason changes, a given rule of thumb may no longer apply that may apply to their projects.)

3.3.2 Benefits Realized

Although the database is still being piloted, already it has saved the company time and improved the quality of new parts. The time savings is in reduced think time and reduced communication time. By making the collective knowledge and experience of many engineering disciplines readily available, the engineer spends less time going through the same reasoning process or searching out rules of thumb from co-workers.

The pilot program has also yielded quality improvements. On a particular part, none of the recent design errors were repeated when the rules of thumb database was used.

3.3.3 Key Technologies

The database was originally developed on a host minicomputer but was recently moved to a networked personal computer environment using an off-the-shelf distributed database package.

3.3.4 Key Organizational Factors

A major limitation to the rules of thumb effort is the relatively low priority placed on entering the rules of thumb and keeping the database up to date. It is considered to be 'above and beyond' the engineer's job to enter the data. As the database developer explained, "We tend to put these rules of thumb in our memories. Our memory and experience are valued by co-workers and managers."

3.3.5 Other Observations

Current text-based technology is also limiting the success of the database. The database developer's vision is to be able to easily enter rules of thumb in their most efficient form for information processing: pictures, audio, etc. He is using the video game as a role model for future development efforts: players, from novices to experts, intuitively learn and recall tricks and shortcuts. (Note: no manual needed!) Once they find a trick, they can combine it with other tricks to accelerate through the game.

4. Technological and Organizational Enablers for Concurrent Engineering

These case studies illustrate examples of concurrent engineering practices being implemented with today's technologies and in today's organizations. Companies are realizing concrete benefits even with pilot programs, but it requires a combination of distributed computing and organizational acceptance. Following is a discussion of technological and organizational concurrent engineering enablers.

4.1 Technological Support for Sharing

In our definition of distributed computing, we described resources as including data, applications, compute power, I/O devices, services and the knowledge available on the network. Here we concentrate on two key resources: data and knowledge. We examine how concurrent engineering can be implemented today by sharing data and knowledge.

We differentiate data from knowledge in this way: Data is the representation of the problem to be solved and the solution to the problem, the multiple inputs and outputs of the project. Knowledge is the experience and skill that are applied to solving the problem.

4.1.1 Technologies for Sharing Data

There are several benefits to sharing data, such as:

- Managing the complexity of simultaneous development
- Maintaining a single source of the data
- Widespread access

This section discusses tools and technologies to support each aspect of data sharing.

4.1.1.1 Managing the Complexity of Simultaneous Development

Software tools help manage the complexity of simultaneous development. At the basic level, UNIX provides rcs and/or sccs commands for revision control, and the make command for file dependencies. The DSEE product handles much more complex configuration management for software development.

Some of today's data management systems also handle configuration management. Examples include SHERPA (a stand alone design management system) and HP's mechanical engineering DATA MANAGEMENT SYSTEM (part of the ME10 and ME30 application packages).

4.1.1.2 Maintaining a Single Source of the Data

Sharing data eliminates the dangers and hassles of maintaining multiple copies of the same data. On a small scale, linked directories and the NETWORKED FILE SYSTEM (NFS) support sharing data between a few machines. The DOMAIN file system, and in the future the ANDREW FILE SYSTEM (AFS) support shared access to data on a large scale as well as in the small.

As Motorola reports:

> "Before the [DOMAIN] network, these activities were coordinated 'open-loop', via periodic batch updates of design files via over-night express shipments of cartridge tapes, faxes of documents, and low-speed modem transfers to update changed documents. The process was slow and error-prone, relying on engineers to manually request updates and to propagate these copies to remote sites using brute-force dumping and loading of wbak or tar archive tapes updates were sometimes not done as often as needed, leaving open the possibility of releasing inconsistent versions of software to our customers." (Dolikian, 1990)

Reviewing tools, such as the internal tools CADACS and VIEWBOARD described in the first case study, and commercial tools such as STATION SOFTWARE, allow multiple people to see and comment on a single copy of design drawings.

4.1.1.3 Widespread Access

The combination of a global file system and data management software forms the foundation for making data available to any one who needs it. Availability of data is a prerequisite to empowering individuals to make considered decisions - see discussion of trust, under "Organizational Support for Sharing", below.

But mere access to data is not enough. Some of the issues faced today in exploiting data access are:

- Security. There is always some data for which access should be controlled. This is as much an organizational issue as it is a technological one; policies must be set and specified. Technologies such as Kerberos (as part of OSF's DCE) support security requirements.
- Browsers and Filters. An overabundance of data can be as bad as the lack of data. Technology can help by providing browsers, filters, and other navigational tools to assist the user in acquiring data of interest. Ultimately we predict standards in both database interfaces (based on a Distributed Object Model) and in human navigational interfaces.
- Resource-to-Resource Access. Several application software suppliers, initiatives, and consortia are grappling today with the need to share data between databases, across heterogeneous hardware, between applications, and even across organizational boundaries (such as with vendors). Examples of software supplier solutions include the FALCON framework by Mentor Graphics and the OPUS framework by Cadence for electrical engineering. Examples of industry efforts include the CAD FRAMEWORK INITIATIVE (CFI) for electrical engineering, the PORTABLE COMMON TOOL ENVIRONMENT (PCTE) for software engineering, and the CALS/CE and DICE initiatives for product design.

4.1.2 Technologies for Sharing Knowledge

Knowledge is divided into collaborative and captured knowledge. Collaborative knowledge is knowledge brought to the project team by current team members. Captured knowledge is knowledge and experience that has been accumulated from industry experts and members of past project teams.

4.1.2.1 Collaborative Knowledge

Ideally, the concurrent engineering project team shares knowledge via frequent face-to-face meetings. However, with many of today's project teams being dispersed in time and place, face-to-face meetings are a luxury (Stinson, 1990). Case Study 1 illustrated the use of CADACS and VIEWBOARD, two internally developed distributed computing tools, to share knowledge over time and place at the Apollo Systems Division. Other tools such as electronic mail, electronic bulletin

boards and smart meeting rooms make use of distributed computing technology to share collaborative knowledge (I/S Analyzer, 1989).

Hewlett-Packard's SHAREDX is a collaboration tool that extends the industry-standard X WINDOW SYSTEM to enable real-time sharing of X protocol-based applications between two or more remote users using X WINDOW based workstations. Windows can be shared to non-HP workstations running the X WINDOW SYSTEM or to X terminals or PCs running X Window emulation mode (Hewlett-Packard Company, 1991). HP SHAREDX has been used by concurrent engineering teams for consulting on engineering designs, software debugging and joint-authoring of documents. Since HP SHAREDX operates over TCP/IP networks, it takes advantage of distributed computing technology to allow collaborative knowledge sharing.

4.1.2.2 Captured Knowledge

The rules of thumb database in Case Study 2 illustrated the use of relational database technology and distributed computing to capture knowledge of automotive company experts. Expert systems were designed to capture knowledge. Only now are expert systems finding their way into commercial applications. For instance, electrical engineering application supplier Mentor Graphics has developed the CONCURRENT DESIGN ENVIRONMENT. It contains a DECISION SUPPORT SYSTEM that allows individuals who are not programmers to create applications incorporating their design expertise (Mentor, 1990).

A set of prototype software of interest is the Bootstrap Initiative's OPEN HYPERDOCUMENT SYSTEM (OHS). The OHS was developed by Doug Engelbart, a professor at Stanford University. It is designed to capture the knowledge of a project team, much like a lab notebook does. The information is stored in corporate memory and hyperlinked for easy access by current project team members as well as members of future project teams (Engelbart, 1988).

4.2 The Effect of Technology on Communication

Computing technology is changing the way we communicate. For example, automatic bank teller machines alter routine banking transactions from a person-to-person activity to a person-to-resource activity. Essentially this allows the bank client and the bank personnel to "communicate" asynchronously. As another example, telephone switching equipment replaces a person-to-resource activity (where an operator plugs wires into a switchboard) with a resource-to-resource activity (where computers talk to each other).

In general, technology allows activities to occur at lower levels of communication (defined in Figure 1). This has two benefits: it increases the efficiency of the activity and it frees up people to spend their time performing more meaningful, higher quality communication.

Likewise, the distributed computing technologies that enable concurrent engineering support tasks at lower levels of communication (see Figure 2).

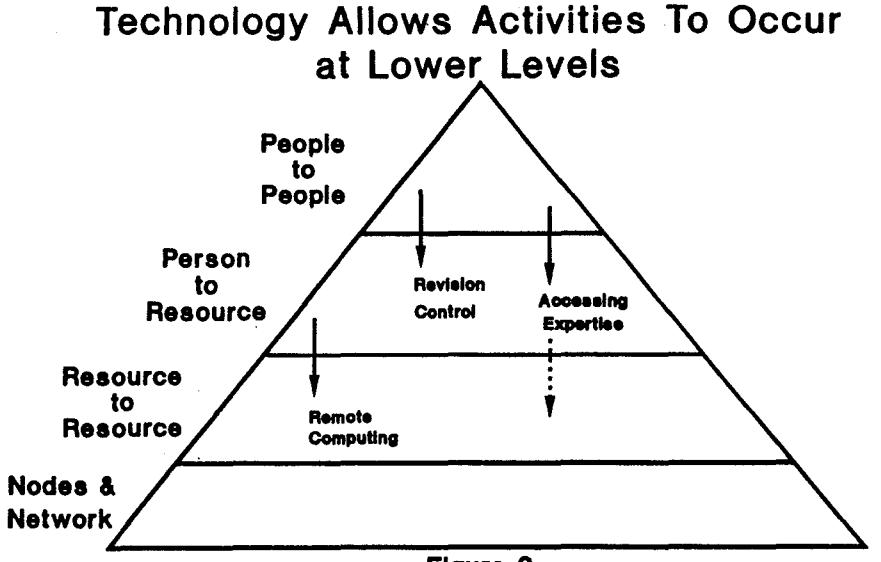

Figure 2

Some examples are:

- Tools that support data sharing, such as RCS and SCCS for revision control, eliminate the need for one person to check with another person prior to editing a jointly authored file. Person-to-person communication is replaced by person-to-resource communication.
- Knowledge sharing tools that capture expertise for later use substitute person-to-resource communication for person-to-person communication. For example, Design for Manufacturability and Design for Analysis tools capture some of the expertise of manufacturing and assembly personnel; the rules-of-thumb database, described above, captures product design expertise.
- Resource sharing tools such as the NETWORK COMPUTING SYSTEM (NCS) eliminate the need for users to learn about networking protocols, processes, etc. They can now tap into the compute resources on the whole network without becoming a computer expert. Resource-to-resource communication supplants person-to-resource communication by transferring to the computer the burden of locating, selecting, and communicating with the remote computer.

The challenge for technology developers is twofold. First, technology needs to support activities at lower levels of communication, in order to free up quality time at higher levels. Second, technology should blend the levels of communication into

a seamless continuum so that people have full access to resources as they communicate and work with each other.

4.3 Organizational Support for Sharing

Organizational support can be viewed in terms of sharing data and sharing knowledge.

4.3.1 Sharing Data

Developing a sense of trust among team members is key to data sharing. Individuals need to have the freedom to initiate additions and changes to the project data based on their expertise and roles on the project team. Certainly, computer systems will need to have revision control, notification and archiving mechanisms in place.

4.3.2 Sharing Knowledge

In the area of knowledge sharing, a key enabler is to organize for cross-functional teamwork. At Hewlett-Packard cross-functional product development teams have been the norm for several years (Nevens, 1990). Having participated on cross-functional teams doing concurrent engineering on fast-track projects, we believe that cross-functional team participation is rewarding for individual team members as well as for the company.

Another knowledge sharing enabler is to institute a reward system for recording knowledge for later use. A computerized system for capturing knowledge could easily track the number of times that knowledge is accessed. Management could recognize those people or teams whose knowledge has been accessed most often. Since design re-use reduces duplicated effort, rewards could also be given to teams who re-use designs and concepts.

All of these organizational enablers require management commitment. The organization must be willing to bootstrap itself. Bootstrap is a concept borrowed from the Bootstrap Initiative mentioned earlier. Bootstrapping implies empowering an organization to improve itself. This means setting aside resources and initiating pilot projects for organizational improvement (Engelbart, 1990). In our research we saw many examples of companies bootstrapping themselves.

We heard repeatedly that changing people was more difficult than adopting new technologies. If pilot projects are shown to be successful, people will be much more willing to try new modes of working.

5. Key Findings

There are multiple ways of working in parallel. On one hand, tools can be built and people can be organized specifically to enable synchronous work. Examples include

co-location of teams, sharing live screens, and 'smart' electronic meeting rooms. On the other hand, technology can be harnessed to reduce the need for synchronous work. Examples here include electronic mail, voice mail, and tools that migrate communication from the person-to-person level to person-to-resource level.

The challenge is to provide tools that free people to spend their time on the act of communicating and working together, rather than on understanding and managing the processes and mechanics. These tools need to:

- Support communication at a given level with communication at lower levels (Figure 2).
- Blend the communication capabilities of the different levels into a seamless continuum.

From our case studies, DOMAIN users are in the forefront of harnessing their computing environment to enable concurrent engineering. This is especially true at Apollo Systems Division, which grew up along with, and as a result of, the technology. Introducing concurrent engineering is more difficult in an established environment of serial product development and deep management hierarchies.

Frameworks are discipline-specific today. CADACS and VIEWBOARD are facing the need to communicate across the ME/EE boundary. Ultimately, either interfaces between frameworks will need to become standard or the frameworks will need to be unified.

Even with the best tools, humans still benefit from human contact. For example, the co-location of CADACS and VIEWBOARD team members led to unexpected benefits. When co-location is infeasible, as in many development efforts, we recommend that team members meet in person at the outset of the project. Throughout a project, even geographically dispersed teams can strive to maintain a level of informal communication. The promise of multimedia built on distributed computing is that it can promote the kind of communication that occurs in person (e.g., by using desktop video-conferencing and shared workspace).

Evidence of the value of concurrent engineering is being reported in terms of reduced time to market, lower cost of development and production, and improved quality. Perhaps the most important finding is that organizations can take incremental steps today toward developing a concurrent engineering environment. Sharing data and knowledge are the keys.

6. Appendix A - Acronyms

AFS	Andrew File System
ARPA	Advanced Research Projects Agency
CAD	Computer Aided Design
CADACS	Computer Aided Documentation Access System
CALS	Computer-Aided Acquisition and Logistic Support

CE	Concurrent Engineering
CFI	CAD Framework Initiative
DARPA	Defense ARPA
DB	Data Base
DCE	Distributed Computing Environment
DICE	DARPA Initiative for Concurrent Engineering
DSEE	Domain Software Engineering Environment
EE	Electrical Engineering
HP	Hewlett-Packard Company
IC	Integrated Circuit
ME	Mechanical Engineering
NCS	Network Computing System
NFS	Networked File System
OSF	Open Software Foundation
PC	Printed Circuit
PCTE	Portable Common Tool Environment - A CASE framework for data exchange.
RCS	Revision Control System
SCCS	Source Code Control System
UNIX	UNIX is a registered trademark of AT&T in the U other countries.

7. Acknowledgements

We heartily thank the following people and organizations for providing us information and participating in the case studies: Sara Beckman of HP's Product Generation Team (PGT), Tom Bower of HP's Roseville Networks Division, Alexander Cavalli of Microelectronics and Computer Technology Corporation, Dick Collette of HP's Apollo Systems Division (ASY), Ron Collett of Dataquest, Danny Cooper of Texas Instruments, Patrick Fitzhorn of Colorado State University, Takashi Fukuda of ASY, Alfred Gort of HP's Colorado Computer Manufacturing Operation, Chuck Habib of PGT, Paul Johnson of HP's San Diego Division, Steven Levy of Putnam Companies, Larry Mammon of HP, Scott McCourt of ASY, Melanie Merrill of Honeywell, Robert Quist of HP's Fort Collins Systems Division (FSY), Mike Schubert of FSY, Bob Waites of HP's Engineering Applications Group (EAG), Mitch Weaver of EAG, Peter Will of PGT, and others who prefer to remain unidentified. We wish you all the best in your current and future concurrent engineering efforts. We also thank the following reviewers for their clarifying and polishing comments: Jim Borzym, Don Brock, Jim Geer, Terrill Hurst, Rosemary Kramer and Carol McKennan.

8. About the Authors

Susan Frontczak is an R&D Software Development Engineer and Kathy Miner is a Product Manager. They both work in the Collaborative Multimedia Program within Hewlett-Packard's Workstation Group.

9. References

Dolikian, Arman (1990): "Building a cross-country Apollo Network", *ADUS Ring*, May 1990, p.1.

Edwards, Albert (1990): "Concurrent Engineering: The Design Environment for the '90s", *Instructional Television Network Course*, University of Southern California, July 19, 1990, p.1.

Engelbart, Douglas (1990): *Bootstrap Seminar*, Stanford University, June 19-21, 1990.

Engelbart, Douglas and Harvey Lehtman(1988): "Working Together", *BYTE Magazine*, December, 1988, pp. 245-252.

Glasier, Robert A. (1990): "Tying Together Engineering Systems and Data Center", *The Sixth Chautauqua Conference on Computer-Aided Engineering*, D. Brown Associates, Inc., September 10, 1990.

Hauser, John R. and Don Clausing (1988): "The House of Quality", *Harvard Business Review*, May-June, 1988, pp. 63-73.

Hewlett-Packard Company (1990): *HP SharedX Product Data Sheet*, April, 1990.

I/S Analyzer (1990): "Experiences with Workgroup Computing", July, 1990.

Liker, Jeffrey K. and Hancock, Walton M. (1986): "Organizational Systems Barriers to Engineering Effectiveness", *IEEE Transactions on Engineering Management*, Vol. EM-33, No.2, May, 1986, pp. 82-91.

McCourt, Scott (1990): DSEE Product Manager, HP's Apollo Systems Division. Interview August 31, 1990.

Mentor (1990): "Concurrent Design Environment - Design Automation in the 1990s", *Mentor Graphics Press Kit*, p. 5.

Merrill, Melanie (1990): Honeywell Corporation. Interview, September 26, 1990.

Nevens, T. Michael, Gregory L. Summe, and Bro Uttal (1990): "Commercializing Technology: What the Best Companies Do", *Harvard Business Review*, May-June, 1990, pp. 154-163.

Port, Otis, Zachary Schiller, and Resa King (1990): "A Smarter Way to Manufacture: How 'concurrent engineering' can reinvigorate American industry", *Business Week* , April 30, 1990, pp. 110-117.

Seybold, Patricia (1990): "Distributed Computing", *Network Monitor*, January, 1990, p. 2.

Stinson, Terry (1990): "Teamwork in Real Engineering", *Machine Design*, March 22, 1990, pp.99-104.

Takeuchi, Hirotaka and Ikujiro Nonaka (1986): "The New New Product Development Game", *Harvard Business Review* , January-February, 1986, pp. 137-146.