Collaborative Development of Groupware Applications

Keith D. Swenson
Fujitsu Open Systems Solutions, Inc.
3055 Orchard Dr. San Jose, CA, 95134
kswenson@ossi.com

Groupware aimed at coordinating the actions of people is faced with a dilemma. In order to support a group at work, the groupware must be configured into applications that accurately reflect how the group works. Yet, organizations are constantly changing both their structure and the way they work, leaving the groupware application inaccurate, or at least out of date. Implementers of groupware applications are finding that the cost of implementation comes not from the hardware or the software, but the effort needed to capture and codify their organizational processes, and then to maintain the applications to keep them up to date.

This paper explores how the implementation of groupware is a collaborative activity. Methods to define processes, such as BPR, are group activities. A powerful way to solve this is to develop the groupware applications collaboratively. In a sense, this is groupware to help in the development of groupware. Four key capabilities are described that are needed to supports collaborative development of applications. Collaborative development of groupware applications can reduce the cost of implementing groupware, and can result in a better fit between the application and the organization.

1 Introduction

The workplace is changing. Management is learning the benefit from defining work along lines of processes. Information technology is finally getting wide acceptance in the white collar workplace, and as it does, it creates even more change in the way that we work. Advanced organization need powerful means to support their work, and to also support the changes in the way that they work, so that they do not lose the flexibility required to remain competitive.

While some groupware tools are ready to function "out of the box" (e.g. simple email), most groupware tools need to be configured to meet the needs of a particular business or office. High end e-mail sorting and filtering tools need to have rules that fit the communication patterns for a particular person in order to be a benefit to that person. Workflow tools need to have process definitions that fit the way that particular organization work, before the workflow is useful. Data sharing tools, (e.g. Lotus Notes) need to have forms and scripts configured to support particular modes of interaction of specific groups. These configurations of groupware tools can be called "applications[1]" because they are an application of the groupware tool for a particular purpose.

This paper postulates that the forces reshaping our corporations are also creating an opportunity for a new aspect of groupware, termed Collaborative Development. This

[1] The term "application" is not optimal here because it conjures up images of a program that has been developed by a programer or a specialist who is trained to do this. In some cases groupware does need to be configured by a programmer or other specialist, but for the lack of a better term, I refer to the kind of configurations that can be readily made within many groupware products that are designed to meet a needs of a group of individuals, as applications, in the same way that a particular spreadsheet template might be considered an application to accomplish a certain calculation. The terms "configuration" and "application" are used interchangeably to distance the discussion from the kind of applications that are developed by programmers.

paper will explore how collaborative development of groupware applications will help organizations respond to the dynamics that they face.

1 Changes in the Workplace

The central reason for flexibility in groupware applications is organizational change. As the way that an organization works is changed, the groupware applications that support this work must also be changed. There are two kinds of change which are important to consider: "improvement" and "innovation." The difference is not simply magnitude -- it is possible to have a complete redesign of a small department, and to have a large incremental improvement. The difference is really whether the change is imposed from outside or inside the organization being changed, and whether the change redefines basic job functions.

1.1 What is Process Innovation?

Process innovation refers to a major change of an organization as demonstrated by the examples of BPR success where the main processes are reevaluated from the point of view of the customer and redesigned from the ground up. Such changes must be instigated from the top of the organization and supported throughout the organization.

Davenport advises that an organization wishing for successful process innovation should: 1) identify processes for innovation, 2) identify change levers, 3) develop process visions, 4) understand existing processes, and 5) design and prototype the new processes. A process innovation program can not be successful unless it is followed up by a program of continual process improvement. A company that is not successful at continual process improvement will not be successful at process innovation.[6]

1.2 What is Process Improvement?

Process improvement, like the Japanese term "kaizen," refers to the continual fine tuning of processes that are already in place. For example a particular group may decide to handle its piece of a process in a different manner, while providing the same basic function. Process improvement can be motivated from the bottom up by incorporating suggestions from workers in a manner reminiscent of Total Quality Management (TQM). In order for workers to be empowered they need to have control over the processes they participate in, and to improve them when they identify a problem.

1.3 What Slows Groupware Application Change?

Being creative and innovative enough to come up with the optimal process redesign and implementing the groupware is only part of the work. The largest amount of labor will go to working with the people that will be using the application, helping them to understand and change their habits to the new way of working, and possibly retraining some of them to perform new and different functions. The greater number of people involved, the more difficult it is to implement, and the greater risk for failure. It stands to reason therefore that any means which can isolate the changes to a smaller set of people, or possibly the minimal set, will be advantageous when it comes time to implement the changes.

Another roadblock for easy groupware application improvement appears when a small part of the organization wishes to make a change just to their part of the application. If the application is represented as a single monolithic definition, then all of the people and groups that use that application need to be part of the change approval process, even if their own functions are unaffected by the change. The bureaucratic overhead may be considerable. In some cases the advantage gained from some very small adjustments to the application may be outweighed by the cost and trouble of getting all the approvals, thereby forming a barrier to the introduction of small improvements.

If the ability to change is difficult, the problem is compounded by the difficulty of getting the groupware application right the first time. The people involved in group work are often unaware of exactly how the work is done. Shoshana Zuboff points out that skills

picked up on the job form tacit knowledge which the worker is unable to express verbally either because they are unaware of exactly what they do, or because they lack the vocabulary and experience of discussing work habits.[29] An example is the difficulty of describing how to balance on a bicycle. To develop a groupware application, workers are usually interviewed to discover the current process, but since the interview is limited by the workers ability to describe the work, the application can be flawed. Even when this method is complimented by having the process specialist observe activity, the rational for a particular action is not observable. Another approach is needed, and collaborative development might be the answer.

1.4 Do All Groups Work Identically?

While change usually refers to a difference in the work processes over time, there is also evidence for needing a difference in the process across the organization. Different groups work in different manners. It stands to reason that for each group to operate optimally they should be able to have their own customized version of a given groupware application. Greenberg has pointed out that groupware that treats all individuals and groups the same has a high probability of failure.[9]

Having a customized process is exactly what process orientation is about. Sale of a large computer system and sale of a reference manual should not be forced to follow the same groupware application merely because they are both sales. Taken to the logical extreme one could imagine a company which has a unique groupware application for selling each different product; this may be appropriate. Yet all of these applications are related because the result should be in some sense the same: a sale. Therefore it is not sufficient to let different teams have different application, they must be able to have different versions of the same application.

2 Human Aspects of Workflow

Information Technology (IT) will change work processes in two major ways. The first is the ability to remotely access information which allows a single individual to do many things directly. Since information can be delivered anywhere, it should be delivered to wherever it can be of the most use, and that is usually wherever the customer is. Many companies are seeing tremendous process improvements by providing information systems that allow salespeople to, for example, directly check a customers credit, verify inventory of items, schedule manufacturing time, or generate complex quotes on the spot. All of this can be done while the customer is waiting.

The second way is when the work process involves a number of people, IT can be used to coordinate their interactions. As information access becomes location independent, teams become geographically dispersed and become more dependent upon electronic communication. Workflow software is needed to provide this communication.

Groupware applications must take into consideration many social factors. The office environment is more complex than it seems. Many interactions happen at an unconscious level or in a natural way that belies their importance. Introduction of new technology into a social setting is a veritable mine field until the users and technology adapt to each other, assuming that they are able to adapt. The next few sections point out a number of land mines that we uncovered in the course of development and early testing of our prototype. This can not be a complete listing because collaborative development is a immature area and is still largely untested.

2.1 Why Not Simply "Program" the Organization?

It is all too easy to view the steps in the groupware as being like a program to be executed by the organization. A programming approach to workflow or other groupware attempts to control the people by forcing them

to do the right thing at the right time. This is, in effect, de-skilling the work. An early supporter of this approach was Frederick Winslow Taylor and his principles of scientific management.[26] While Taylor expounded more than eighty years ago the importance of attention to process, his ideas fall far short of BPR because of his assumption that there is a single ideal design for any work process, and his view that people should be subordinate to the process. Robert Howard points out that "Taylorism is alive and well in the assumptions of many technology managers [who] see computer technology, first and foremost, as a means to eliminate, or at least minimize, the human element in work."[15] Besides the obvious lack of appeal to the workers, there are other serious problems that result from trying to program the organization.

Programmed organizations are unable to handle exceptions beyond those provided for in advance. This means that during the investigation and design of the process, much attention must be given to every possible point of failure, and a programmed response must be provided. A side effect of having to provide for all of the possible exceptions up front is that the cost of introduction of the new process is significantly higher. This forms a barrier to adoption. Since so much effort goes into initially creating the process it gains a sort of momentum and becomes very hard to change. This decreases the ability of an organization to respond to change from external sources.

In a programmed organization the worker has lost control of the work processes. There is no room for creativity and experimentation; key elements that would lead to spontaneous process improvement. Most workers find this unpleasant and are unmotivated.

2.2 How Can Users Be Empowered?

Andrew Clement makes a strong case for the need to use information technology to empower the user.[2,3] He recognizes two meanings of empowerment in the context of the workplace. The first meaning comes out of TQM and BPR and refers to the additional responsibilities that a worker gains as the organizational hierarchy is flattened. "Empowerment means that operational decisions will not be made hierarchically, that knowledgeable workers must feel comfortable in making decisions, that managers must provide counsel rather than directives, and that information and our conceptual models of how to use it are the source of decisions"[4]

The second meaning might be termed "democratic empowerment" because it emphasizes the ability for an individual to have control over his or her own situation. Workers empowered in this way might be able to make improvements in any process in which they play a role. Clement presents three examples where such empowerment lead to changes in the work environment and processes which "TQM and business process reengineering advocates would be proud of." Given the ability to change, they made improvements far beyond those that had been considered by the traditional management. The conclusion is that process support needs to be "supporting" instead of "enforcing."

An example of a product widely criticized for its enforcement of a set process is the Coordinator.[8] This product embodied a static model for negotiation based upon a finite state machine. Perhaps the greatest reason cited for the low adoption rates of the Coordinator was the negative reaction that users had to being constrained to a particular set of moves, and the feeling of being forced to do something they did not want to do.

Traditional groupware has been often criticized for its lack of flexibility. Too often the applications are too simple to handle the richness of real world problems. A study by SRI found that lack of flexibility was a common complaint among collaboration technology users.[17]

2.3 Intelligence behind the Rules

Failures at automating office work may arise from the attempt to literally implement the rules of the office. Workers in an office work under the assumption that there are a set of rules that they are following. While the rules exist, the workers have a great deal of flexibility in implementing them. It is the intelligence behind the use of the rules that allows them to work. Upon close examination

studies[14] have found mutually contradictory office rules; yet the office still functions on the judicious application of these rules. Blind coding of the office rules into a groupware application that lacks the common sense of the workers may yield disastrous results.

A key point to keep in mind is that office workers currently enjoy a great deal of flexibility in how they implement office procedures. This flexibility may account for a large part of the effectiveness of each worker. It is critical that any replacement system allow for the same degree of flexibility and empowerment.

2.4 Why should the Applications be Designed by the Users?

One theme common to all approaches to reengineering organizations is to involve the worker as much as possible at every stage.[6][12][10][18][29] Major process innovation is inevitably top-down and is imposed from "outside" to organization. This type of change is easy to implement in any workflow system because a new process definition can always be designed separately and introduced to the organization.

The real difficulty is supporting bottom-up process continual improvement. The separation of the user from the planner forms a barrier to process improvement. When a small improvement to a process is identified, the bureaucratic cost of involving a second person, or more likely a separate department, will outweigh any potential benefits. Jonathan Grudin has noted that group support systems fail when there is a separation of the person doing the work and the person gaining the benefit.[11] A system that is designed to allow the user to make changes to the application is the only solution. Studies have shown that only about 10% of spreadsheet users actually modify spreadsheets. In a similar vein, not everyone is expected to actually make changes to plans. Rather, it is the potential for change by anyone, and the ability to trade templates, that makes the difference to allow incremental process improvement.

3 Collaborative Development of Groupware Applications

Development of groupware application is used here to mean the activity of configuring a groupware product so that it supports the specific work requirements or processes of a situation. This would include the creation of process definitions for a workflow application, or the configuring of forms, scripts, or rules in other groupware environments. This is the extra work that an end user must go to in order to coordinate future work. There exist two sorts of configuration: instances and templates.

The term "instance" is used here to denote a specific configuration for a process which has begun and is “in progress”. It involves specific people and organizations, and includes details of the situation. The application to satisfy a particular customer with a specific order is a process instance.

The term "template" is to refer to a definition that is prepared in advance to anticipate a kind of situation before the details of the actual instance is known. For example, an emergency evacuation plan is prepared in advance of any real emergency so that when it is needed it can be instantiated quickly. An application created for a repeatable situation, such as sales lead tracking, which has elements designed to support a way that an organization does a particular job, is a template. It serves as a starting point for the specific configuration.

3.1 What is the Process that creates a Groupware Application?

All groupware systems have, necessarily, some form of application definition. Collaborative development is concerned with how the applications themselves are created. For most systems the activity of application development is outside of the system itself, something to be done at a different time (before) and usually by a different person (a programmer) than those involved in the work. Collaborative development should include features to

support the development activity, including the interactions between people that necessarily take place during the creation and evolution of appplications.

Groupware vendors are beginning to realize the limitations inherent in the separation of developer and user and are starting to produce groupware systems with collaborative development capabilities. The goal of this paper is to identify the common features that all such systems require in order to be truly considered collaborative development tools.

4 Requirements for Collaborative Development of Applications

There are four general requirements for a collaborative development of groupware applications. First, the groupware tool must designed so that application can be configured by an average user. Second, it must have some features to support multi-user editing of the application structures. Third, it must be able to support modification of active application structures. Finally, it needs to support differentiation of application configurations for different groups.

4.1 Support for End Users

In order for a group to collaborate in the development of groupware applications, end users must be able to configure such applications without specialized training. An analogous situation is the production of financial reports compared to the introduction of spreadsheets. Spreadsheets did not enable any capabilities that were not already available to those willing to hire a programmer. What the spreadsheet offered was a way for end users to create reports and calculations without having to program.

The key to making a groupware product configurable by end users lies in a having a clean graphical representation of the way the group interacts. A number of workflow tools are appearing that support a graphical representation of process definitions. Most project management tools have at least a PERT chart representation. Workflow tools that only display the application graphically, without allowing editing, stop short of a real solution. End users expecting a WYSIWYG environment will want to construct and edit the application graphically.

4.2 Support for Multi-User Application Development

Responsibility for the work and responsibility for the application are closely related. Michael Hammer states: "Companies that have reengineered don't want employees who can follow rules; they want employees who can make their own rules. As management invests teams with the responsibility of completing an entire process, it must also give them the authority to make the decisions needed to get it done."[13] Involving a group in the configuration activity implies that a group is involved in the responsibility of the task.

While empowering multiple people to create and modify applications simultaneously, the tool must give the users *control* over the changes. A user who is responsible for the result of an application fragment must have some assurance that his or her configuration has not been improperly modified. Reflexively, that user should not be able to change someone else's application fragment.

We draw as a conclusion from this that the groupware tool must support some form of application fragmentation where different capabilities can be assigned to different people for different fragments. At a very minimum the application fragment should have an owner who is allowed to change all aspects of the application, and who can assign other capabilities to others. The requirement for fragments seems to come directly from the desire to allow multiple people to be involved in the planning process, and not from any specific implementation goal, much in the same way that file ownership and access controls are a fundamental requirement for a multi-user file system.

4.3 Support for Groupware Application Improvement

The end users involved in the configuration applications will neither be experts in how to program, nor will they be able to devote a large amount of time. Typically, they will not get the configuration correct from the start. Occasions will arise for which the application is not prepared.

Changes to the applications must be allowed on-the-fly after starting enactment. Without the ability to change process plans on-the-fly, the end user would be burdened with having to produce very complete very formal application configurations, or to turn to specialists to assure completeness, which would interfere with the collaborative aspect of the tool.

Being able to change a groupware configuration is not enough. The user must also be able to change this at any time without undesirable side effects in active processes. An instance of a process by its nature will persist for a length of time, in some cases for years. If the organization is continually improving there may be several versions during the time the instance is active. We have found the proper solution to this to be a separation of the application instance from the definition template. In general, once a process is started, to remain consistent it should complete with the same version of a process. The change of a template should leave the instances unaffected. The system must be able to support instances with different versions of a process being enacted at the same time.

Collaborative development tools must accommodate users who have been familiar with a particular process and who unexpectedly discover that the process has changed. The process must be accompanied with enough explanation so that a user confronted with an unfamiliar step in a process can immediately find enough information to be able to able to take the appropriate action.

4.4 Support for Individual and Group Optimizations

Since groups work in differing manners, a groupware application should support a way for each group or individual to specify their own versions of application definitions which are optimized for their group.

The groupware environment is directing workers' actions, so the application definition carries some authority while it is being executed. It must not be possible for a person to forge a new application, and then to run it at the authority of someone else. The system must enforce a proper mapping between authority of an application and the designer of that application.

In practical terms there will be a large number of application definition types. While the system allows each person to have their own version of an application, it would be tedious if everyone was forced to have every kind of application. From experience we have found there needs to be a mechanism to allow users to specify another place to "inherit" applications from.

5 Benefits of Collaborative Development

5.1 Better Applications

As a company enters into continuous improvement, collaborative development allows better applications to evolve. Each member of the organization can improve their own application when ever a potential improvement is identified. In accordance with the principles of TQM, ideas for application improvements may come from the people doing the job that would not have occurred to a centralize application development effort. March and Simon argue that decentralized planning is always at least as good as centralized planning, and usually much better.[18] Removing the overhead of making a change empowers people to make the change. The result is an organization that is able to learn from the way it works, and to

improve, much like the learning organization suggested by Senge.[21]

5.2 Ability to Experiment

It is easy for people to test new ideas. If the application is determined to be faulty, the ability to modify applications on the fly allows for recovery. Through experimentation in live situations people may come across solutions that are not apparent in a more abstract situation. Since different teams are allowed to have different applications for the same situation, applications that are optimal for each particular team can be found.

5.3 Process Capture

Collaborative development can help when the process is unknown, but the workers know what to do. By adding to the application as it proceeds, one can end up with a record of what happened as well as a preliminary template for the next time. March and Simon point out that definition of new processes is one of the most important activities of any organization.[18]

5.4 Improvement at all Levels

Experimentation and improvement can be made at all levels. Top level applications can be modified to make use of different combinations of services from lower levels. Simultaneously the services of the lower levels can be improved. This situation has been compared to the self-similar aspect of fractals in that you see the same activity happening at the same time at different levels of granularity.

5.5 Save Planning Time

Collaborative development has an advantage over traditional development in that the single user bottle neck is avoided. Each individual or group is responsible for keeping their own applications up to date, effectively distributing this activity away from a central control out into the work force closer to where the work is being done.

The application fragment templates allow complete applications to be constructed quickly and automatically. The complete application is customized for the groups that are involved in this process instance.

5.6 Better Feedback

Tools supporting collaborative development must be easy to understand. If it succeeds in this goal, more people from the organization can be involved in the development process, and feedback from all people involved is more accurate.

Collaborative development surrounds the problems associated with process discovery through interviews and observation by allowing users to directly manipulate and experiment with processes. Tacit knowledge is learned by trial and error; collaborative planning allows processes to be discovered and improved by trial and error, in order to tap some of that tacit knowledge.

6 Regatta Technology

The Regatta Project was started in 1991 within Fujitsu to explore groupware technologies. The three goals are to develop software to 1) support coordination of work, 2) help users understand how their group works, and 3) support the change and improvement of work processes. With these goals in mind, and generous sponsors in Japan, Regatta Technology was developed.[22, 25]

Regatta Technology has now been incorporated into a product by ICL called TeamWARE Flow. This new product is a workflow component for their TeamWARE suite of groupware tools.

6.1 Shared Space for Collaboration

Primarily, TeamWARE Flow offers a shared space for collaboration[19] which contains data, artifacts, and a collection of plan fragments. Each plan can act as an application or configuration of interaction between people to support a particular work process. Access to the space is allowed only

to participants which have been specified. The system provides a list of all active tasks on the plans, as well as the options for each of those tasks. A record of user interaction is kept to provide a history of the process to participants. Platform independent forms allow for consistent presentation of data on any client platform.

6.2 Visual Process Language

TeamWARE Flow includes a graphical representation of a plan that is elegant and easy enough to use, yet powerful enough to handle all the needs of an average business user. TeamWARE Flow's graphical planner (editor) allows an end user to edit Visual Process Language (VPL) diagrams directly. The same VPL diagram is used to display the current status of the enactment.

Zisman did early work in the area of representing office procedures with Petri Nets.[27,28] VPL is a parallel language which has properties similar to Petri Nets but it has been specialized for the needs of collaboration and work processes. VPL is quite similar to Ellis' Information Control Nets[7] except VPL is simplified by omitting the document flow.

Plans are composed of stages which represents task to be done. A stage can be active or inactive to indicate whether it is time for that task to be done. Stages can be programmed to respond to particular events, and to send events to another stage in order to activate and deactivate each other. Stages are represented by a bisected ellipse with the role assigned to the task in the upper part, and the description of the task in the lower.

Stages are programmed by placing options (another graphical object) on them. An option looks like a small circle on the edge of the ellipse with an arrow pointing to the stage that will receive the event. When the stage is active, the option appears as a menu item to the user. Choosing that menu item triggers the option, sending an event to another stage, possibly activating it, and optionally deactivating the originating stage. Four other nodes of lesser importance exist to fill out the capabilities. Experience has shown that VPL is simple to understand, yet powerful enough to represent a wide variety of processes. More completed descriptions of VPL can be found in [23, 24].

6.3 Plan Binding

TeamWARE Flow supports plan fragments. A user who is assigned to a stage can invoke a plan template containing subtasks to complete that stage. The subplan for a stage forms a different plan fragment for which that user is the owner. Plan owners may make any modification in the plan through the graphical editor at any time during the process enactment. The enacted plan is a copy of the template, so changes to the template, do not effect the active process, and vice-versa. A subplan for a task can also be in a separate collaboration space thereby assuring privacy, and only the result is communicated.

TeamWARE Flow does not require a plan in all situations. Any stage can be handled manually. Each user who goes to the trouble of providing or improving their templates for a particular task will also receive the benefit. It is important for success that the same person receive the benefit that does the work.[11]

6.4 Status

Regatta Technology has been in beta test in real office use since June 1993 in a number of sites within Fujitsu. The largest installation involved a team of 18 software engineers who used Regatta to track over 300 independent modules through 5 phases of development which lasted 8 months. Quality assurance was tracked through each phase. Within weeks of installation Regatta was supporting development processes. The ability to modify plans was shown to be essential to keep up with the changing understanding of the process. The result was that productivity was substantially improved, the final quality was assured by the knowledge that every module had been correctly processed.

TeamWARE Flow is in beta test now at selected TeamWARE customer sites. The full general release of the product will be in Q4 1995. The workflow server will run on NT

3.5+ and on Sun Solaris. The client modules run on MS Windows 3.1+.

7 Conclusion & Summary

Groupware technology is a strong option that corporations have to help them support work processes. A program of continual improvement of groupware configurations is needed to assure the organization remains on top, but is difficult to implement in most current groupware systems which support a more centralized style of development. Collaborative Development is a way to empower users to have control over their own parts of the groupware application, while integrating into the work processes of their organization. There are four basic requirements to support collaborative development: end user configurability, editing by multiple users, support for application change, and different applications for different groups. TeamWARE Flow is an example of an existing collaborative development tool which has been shown to be effective in a live environment. As the BPR movement gains momentum, the need for Collaborative Development tools will become more clear, and more example will appear.

8 Acknowledgments

The author wishes to acknowledge many of the contributions that have made this paper a reality: Robin Maxwell, Kent Irwin, Allen Chang, & others for developing and testing the software and ideas; Yoshida, Yamamoto, and Kondoh of Fujitsu Laboratories for their insight and helpful recommendations; Tsutsui, Takiuchi, Watanabe, Iwakata, Hamano, Matsushita, Araki, and Fukao of Fujitsu Ltd. for their support; DeNicola & Chiya of FOSSI for support, and to Shuitsu Yoshida for starting it all.

9 References

[1] Jim Bair, Del Langdon, Integrating Process Re-Engineering and Workflow, Tutorial session at Groupware `93, San Jose, Aug 1993

[2] Andrew Clement, Computer Support for Computer Work: A Social Perspective on the Empowering of End Users, *CSCW 90 Proceedings*, ACM Baltimore MD, 1990

[3] Andrew Clement, Computing at Work: Empowering Action by `Low-level Users', *Communications of the ACM*, 37(1):53-63 January 1994

[4] R Benjamin, Michael S Scott Morton, Reflections on effective application of informatoin technology in organizations, In *Personal Computers and Intelligent Systems: Information Processing 92*. R H Vogt, Ed., North Holand, Amsterdam, 1992, p 131-143

[5] Thomas H Davenport, James E Short, The New Industrial Engineering: Information Technology and Business Process Redesign, *Sloan Management Review*, Summer 1990.

[6] Thomas H Davenport, *Process Innovation: Reengineering Work through Information Technology*. Harvard Business School Press, Boston, 1993

[7] Clarence A Ellis, Gary J Nutt, Office Information Systems and Computer Science, reprinted as reading 9 in *Computer Supported Cooperative Work*, Irene Grief ed., Morgan Kaufman, San Mateo, 1988

[8] Tom Erickson, An eclectic look at CSCW 88, *ACM SIGCHI Bulletin*, 20(5), pp 56-64, July 1989

[9] Saul Greenberg, Personalizable Groupware, *Proceedings of the 2nd European Conference on Computer Supported Cooperative Work*, Kluwwer Academic, Amsterdam, Sept 1991

[10] Jonathan Grudin, Obstacles to user involvement in software product development, with implications for CSCW, reprinted in *Computer Supported Cooperative Work and Groupware*, Harcourt Brace Jovanovitch, Academic Press, 1991

[11] Jonathan Grudin, Why CSCW Systems Fail, Proceedings of the *1988 Conference on Computer Supported Cooperative Work*, ACM, p85-93, Portland Oregon, 1988

[12] Michael Hammer, Re-engineering Work: Don't Automate, Obliterate, *Harvard Business Review*, July/August 1990

[13] Michael Hammer, James Champy, *Reengineering the Corporation*, Harper Collins, New York, 1993

[14] Carl Hewitt, Offices are Open Systems, *ACM Transactions on Office Information Systems*, 4(3):271-287, July 1986

[15] Robert Howard, *Brave New Workplace*, Viking, New York, 1985.

[16] Simon M Kaplan, William J. Tolone, Douglas Bogia, and Celsina Bignoli, "Flexible, active support for collaborative work with Conversation Builder", *Proceedings of the 1992 Conference on Computer Supported Cooperative Work*, ACM, 1992

[17] David Kolbus, et. al. A multi-client study in Collaborative Technology Environments, SRI International, 1992-93

[18] James March, Herbert Simon, *Organizations (Second Edition)*, Blackwell, Cambridge, 1993

[19] Michael Schrage, *Shared Minds: The New Technologies of Collaboration*, Random House, New York, 1990

[20] Michael S Scott Morton, The Corporation of the 1990s, *Information Technology and Organizational Transformation*, Oxford University Press, New York, 1991

[21] Peter M. Senge, *The Fifth Discipline: The Art and Practice of the Learning Organization* Doubleday/Currency, New York, 1990

[22] Keith D Swenson, The Regatta Project, *Proceedings of the First International Conference in Technologies and Theories for Human Cooperation, Collaboration, and Coordination*, Applica `93, March 1993

[23] Keith D Swenson, A Visual Language to Dscribe Collaborative Work, *Proceeding of the International Workshop on Visual Languages*, Bergen Norway, August 1993

[24] Keith D Swenson, Visual Support for Reengineering Work Processes, *Proceedings of the Conference on Organizational Computing Systems*, ACM press, Milpitas California, p130-141, November 1993

[25] Keith D Swenson, Kent Irwin, Robin Maxwell, Toshikazu Matsumoto, Bahram Saghari, A Business Process Environment Supporting Collaborative Planning, to appear in *The Journal of Collaborative Computing*, 1(1) 1994

[26] Frederick W Taylor, *Principles of Scientific Management*, Harper & Row, New York, 1911

[27] Michael Zisman, *Representation, Specification, and Automation of Office Procedures*, PhD Thesis, University of Pennsylvania, 1977

[28] Michael Zisman, Office Automation: Evolution or Revolution, *Sloan Management Review* 19(3):1-16, Spring 1978

[29] Shoshana Zuboff, *In the Age of the Smart Machine*, Basic Books, New York, 1988