Rapid Software Development through Team Collocation

Stephanie D. Teasley, Lisa A. Covi, *Member*, *IEEE Computer Society*, M.S. Krishnan, *Member*, *IEEE Computer Society*, and Judith S. Olson

Abstract—In a field study conducted at a leading Fortune 100 company, we examined how having development teams reside in their own large room (an arrangement called *radical collocation*) affected system development. The collocated projects had significantly higher productivity and shorter schedules than both the industry benchmarks and the performance of past similar projects within the firm. The teams reported high satisfaction about their process and both customers and project sponsors were similarly highly satisfied. The analysis of questionnaire, interview, and observational data from these teams showed that being "at hand," both visible and available, helped them coordinate their work better and learn from each other. Radical collocation seems to be one of the factors leading to high productivity in these teams.

Index Terms—War rooms, collocation, rapid software development, metrics, software engineering, productivity.

1 Introduction

As firms prepare to compete in the e-business era, the ability to produce quality software on time is emerging as an important source of competitive advantage. Because software development is notoriously slow, firms have been experimenting with new approaches to software application development to meet current business needs. In response to these needs, firms have introduced quality management approaches, automated software design and development tools, and process improvement initiatives to their traditional methodologies [30]. In spite of these efforts, the software industry is still plagued by poor schedule, cost, and quality numbers [21]. The reason behind this could be due to either the true efficacy of the methods and tools or the organizational challenges in successful adoption of these proposed solutions or both. Or, these are not the solutions.

Communication delays and breakdowns have been identified as one reason behind such schedule and cost overruns in software projects [29]. In this paper, we present a field study of a new approach to software development where the entire project team is radically collocated. By collocation, we mean that the project team (including a customer representative) is located in a single physical room, called the *war room*.[1] We hypothesize that this collocation significantly reduces communication hurdles in software projects. As a consequence, we expect that projects using this approach will show significantly better productivity and shorter schedules when compared to projects that do not locate the entire project team in a war room. Note that, in order to derive all the benefits of collocating the entire software development team in a war room, the size of the team and, consequently, the scope of the project are also limited, a point we return to in our discussion.

This paper addresses the following research questions:

- Does collocated software development lead to higher productivity?
- Does collocated software development lead to a significantly shorter schedule?
- Does collocated software development lead to high customer, sponsor, and project team satisfaction?
- What in the team's activity accounts for any improvements in project performance?

We report on a field study conducted at a leading Fortune 100 company where we tested the war room approach on system development. The productivity and schedule results of the collocated projects were significantly better than both the industry benchmarks and the performance of past application development projects within the firm. We also found that, satisfaction of the project team, customer, and the project sponsor were high with this new approach to development, suggesting quality was not sacrificed to speed. Our analysis of the questionnaire, interview, and observational data from these teams provides evidence in support of the significant effect of collocation in enhancing software productivity, cycle time,

- *S.D. Teasley is with the Collaboratory for Research on Electronic Work, School of Information, University of Michigan, 1075 Beal Ave., Ann Arbor, MI 48109-2112. E-mail: steasley@umich.edu.*
- *L.A. Covi is with the School of Communication, Information and Library Studies, Rutgers University, 4 Huntington St., New Brunswick, NJ 18901-1071. E-mail: covi@scils.rutgers.edu.*
- *M.S. Krishnan is with the Business School, University of Michigan 701 Tappan St., Ann Arbor, MI 48109-1234. E-mail: mskrish@umich.edu.*
- *J.S. Olson is with the School of Information and Business School, University of Michigan, 701 Tappan St., Ann Arbor, MI 48109-1234. E-mail: jsolson@umich.edu.*

Manuscript received 27 Sept. 2000; accepted 8 Nov. 2001.
Recommended for acceptance by K. El-Emam.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 112914.

1. The term "war room" comes from the practice in WWII of having major leaders confer in special rooms outfitted with key maps and other information as well as the key figures "at hand."

and user satisfaction. Subsequently, the firm has adopted this approach for all its application development needs.

The contributions of this paper are threefold.

1. Most studies on software productivity focus on tools, people, or the development process used to develop applications. Our focus in this paper is on the effect of collocating the entire software project team (including a customer representative) in a war room, affecting the communication among team members.
2. We study the effect of team collocation and the new software environment on end user satisfaction. User satisfaction is an important dimension that reflects the perceived quality of the software product in the eyes of the customer. We believe that only a few studies in the literature have analyzed both productivity and some dimension of quality measure in the same sample of projects [31].
3. We support our findings with qualitative data from observations and interviews, similar to the research methods advocated in [34]. These data help us better understand the specific reasons behind the improvements found.

This paper is organized in the following way: In the next section, we discuss the theory behind why collocation of the project team and customers in a war room might increase productivity and satisfaction measures. We also describe the new approach to product development undertaken at our research site in this section. In Section 3, we discuss our research design and measures, followed by our findings in Section 4. We conclude our paper with directions for future research.

2 Theory

2.1 Software Development Methods

The traditional systems development life cycle method (also referred to as the waterfall model) is still used in some organizations for development of large and complex systems [32], [41], [42]. This is a formal methodology that divides the systems development life cycle in to six stages: project definition, feasibility study, design, programming, testing and installation, and systems support. This approach is resource intensive and inflexible, thus not suited for more recent systems with unstructured and fluid requirements. In addition, the physical separation of tasks, such as design, coding, systems, and integration testing, hinders communication, leading to cost and schedule overruns due to rework.

New methods were introduced, including rapid application development approaches such as prototyping, JAD (Joint Application Development) [1], [2], [7], [16]. The prototype and iterative development method is effective for capturing the user interface requirements, but it works less well for capturing requirements for more complex systems whose inner workings are not reflected at the interface.

2.2 The Role of Communications in Software Projects

In large software projects, most of an individual programmer's time is spent in communicating with or looking for information from other team members. Past studies have indicated that less than 30 percent of a software development programmer's time in large projects is spent on traditional programming tasks and less than 20 percent of the time is spent on coding [6]. The rest of the programmer's time is spent in design meetings, resolving problems within the team, resolving specification misunderstanding with the customers, other communications with the customer and product testing, etc. Breakdowns in communication among development team members and with the customers often cause schedule delays, especially from rework when the delivered system is not fitting the users' needs. In addition, unplanned interruptions constitute a significant time sink, as does the time lost in context switching [36].

In traditional software development, formal artifacts and documents produced by one group are assumed to be sufficient to provide all of the information needed by another group. However, this is often not true [13]. The fact that tasks such as design, coding, systems, and integration testing are carried out by different groups hinders communication of both technical information as well as its rationale, resulting in project delay and rework. Even the physical separation of the software development team within a building can create several forms of communication breakdowns. Specifically, there is a logarithmic decline in communication with increased distance between collaborators, where any distance over 30 meters produced the same low probability that team members would talk to one another [4], [28].

Numerous ethnographic studies of teamwork reveal the subtle, hidden nature of the features that make communication effective. For example, chapters by Suchman, Hutchins and Klausen, and Heath and Luff in Engestrom and Middleton [14] examined transcripts of conversation from work practices to show how group work and workspaces are mutually constituted. Team members systematically communicate information both through various posted messages, their glances, the arrangement of chairs and desk, etc., to help accomplish the group task. Hutchins [22] has proposed a theory of "distributed cognition" to provide a theoretical framework for understanding how people exploit features of the social and physical world as resources for accomplishing a task (also termed "socially shared cognition" in [38] or "situativity theory" in [19]). In this view, communication creates mutually held representations of the work that allow activity to proceed successfully within a complex and ever changing context [23], [46]. The concept of common ground is used to describe how conversations proceed by creating shared understanding between participants [11]. This shared understanding is essential to conducting any joint activity.

Communication breakdowns occur in a number of ways. For example, members of design teams can occasionally mistakenly assume that the others share a common understanding of an issue when in fact they do not. These confusions usually arise when each team member starts with an unstated assumption and is not able to immediately

resolve the problem once a conflict is detected. When the team members are physically separated, this error will often not be detected until the following design review meeting when the group meets face-to-face. Meanwhile, other members of the team would have continued their work assuming that there was no error. A design change to rectify this mistake at a later stage will further add to the project delay. Such situations are not uncommon in any large software project.

The same phenomenon is also observed when there is a communication breakdown between developers and the customers of the project. Ambiguity of customer specifications and misunderstandings of specifications are prevalent in customized application development. Once again, these problems can be attributed to conflicting unstated assumptions that are not resolved immediately. We believe that collocation of the project team and customers in a war room can be effective in reducing such communication breakdowns and facilitating speedy resolution of conflicts. By improving communication, productivity and timeliness of the projects will also improve.

2.3 Team Collocation through "War Rooms"

In corporate America, collocation is achieved in what are called "war rooms." The term generally refers to an immersive environment where experts, technology, managers, and new products come together in a "nerve center" to facilitate interactive information sharing with a minimum of outside distraction. In this context, the war room connotes the centralization of all the best resources into one location to promote efficiency and timely work output. These dedicated project rooms have also been called "skunk works" [43] or "team rooms" [45]. There have even been several software implementations of "virtual war rooms" [18] in an attempt to achieve these same ends while being remote, a point we will return to in the discussion.

Driving the popularity and perceived significance of war rooms are several factors. First, it is economical to locate people in the same place at the same time. Although today's technology provides workers with a growing number of tools for interacting over distance (e.g., e-mail, message pagers, instant message systems, videoconferences), there is an enduring preference for face-to-face interactions [10]. Further, with collocation, every team member can be aware of all aspects of the project development without the need for scheduling status meetings and circulating written progress reports. Finally, corporate culture associates the worker's status with size of a workspace, proximity to high status co-workers, and other locally relevant perceived advantages of a workspace [12]. Therefore, projects given a dedicated space are seen by outsiders as places for high intensity, important activities. Team members selected to participate in war room projects are usually hand picked by managers because of their particular skills and the perceived importance of the project over the worker's routine tasks.

Several evaluations of workspaces lead us to hypothesize that collocating members of software development teams will enhance productivity. In an investigation of integrated product teams, Poltrock and Englebeck [37] described how physical collocation facilitated collaboration and coordination within teams via both scheduled meeting and opportunistic interactions. Sawyer and his colleagues found that team rooms helped focus the activities of the work group and isolated them from interruption from people outside the project [43]. Becker and Steele at Cornell's International Workplace studies program surveyed a number of case studies on collaborative teams and found that the way an office environment is organized influenced work processes such as coordination, work patterns, and communication internal and external to the team [8]. Kraut et al. [28] studied collaboration between scientific researchers and found that the physical distance between offices influenced the development of collaborative relationships and the execution of the work. Allen's [4] investigation into communication patterns in R&D laboratories found that engineers are more likely to communicate with the individuals nearest to them and that people tend to communicate more with people from the same group than from other groups. Although these studies provide evidence that war rooms should increase communication and facilitate efficient flow of work, they lack formal indicators of measurable performance outcomes of a project.

Some of these communication-related issues are resolved partly in the various approaches to systems development discussed above. However, we believe that collocation of the entire team in a war room (with its concomitant limiting of the scope of the project) can combine several advantages to overcome the breakdowns. For example, collocating the customer with the designers and developers brings in the advantages of quick customer feedback and fast resolution of requirement questions, as in prototyping and joint application development. Collocating the team also helps in resolving misunderstanding among designers and developers and also helps to ensure adherence to formal procedures and quality standards, as in the traditional waterfall model. Although collocation will increase interruptions, the interruptions are about the project itself, producing only minimal loss from context switching.

3 Research Overview

The research reported here is a combination of case study and empirical evaluation at a Fortune 100 automobile company. Based on the literature on software development methods, the auto company adopted a methodology that included having the customer on the team, using a project scoping method called "timeboxing," and using iterative development. Having seen the success reported about the war room facilities in the Sun Java Factory [47], the company had the teams use makeshift war rooms. For the purposes of this paper, the facility will be called the Rapid Software Development Center (RSDC), consisting of six war rooms, additional conference rooms, and hotelling[2] areas.

A number of measures were taken to assess the success of the pilot teams:

- productivity indicators, standard measures including time to market, and function points per staff month,

2. "Hotelling" refers to the practice of having cubicles nearby in which one can work individually. The cubicles are not owned by anyone, but can be used for a short period of time, much as a hotel room is not owned by anyone but can be reserved and occupied for short periods of time.

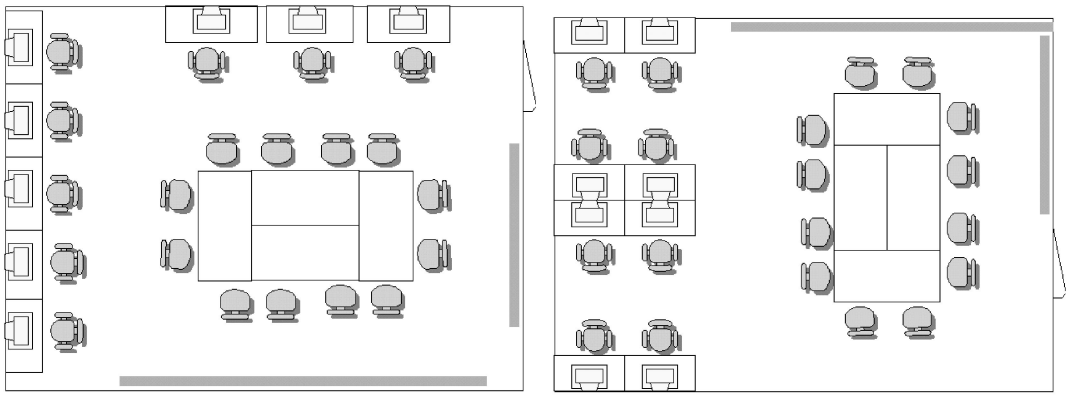

Fig. 1. The two kinds of room layouts at the trial facility for software development.

- questionnaires, administered at the beginning, asking all team members about their predictions about their satisfaction with the facilities and again, at the end of the project, assessing their actual satisfaction with the facilities,
- observations of two teams in depth from visits with them about 8-10 hours a week for the duration of the projects and interviewing the team members at project completion,
- questionnaires, administered at project completion, assessing team satisfaction, customer satisfaction, and sponsor satisfaction for all pilot projects.

These measures are described in detail below.

3.1 The Setting

The facilities at the RSDC included a dedicated war room for each software development team, conference rooms nearby, and various hotelling cubicles for more private work away from the team. Fig. 1 shows the general layout of the rooms.

The dedicated war rooms were outfitted with individual workstations for each of the team members. Workstations were arrayed along the outside walls, shown in the lefthand panel of Fig. 1, or in a "E" shape, shown in the righthand panel of Fig. 1, more like individual cubicles, but without any walls. In the middle of the room was a worktable, the walls had whiteboards, and flip charts on easels were available as needed. Several rooms had printing whiteboards. Near the war rooms were conference rooms, available on a first-come-first-served basis. One conference room was outfitted with video conferencing for use in remote meetings with others as needed. "Hotelling" space—unassigned, more private, quiet cubicles with workstations and phones—was also available nearby.

These facilities contrast with the company standard of relatively large individual cubicles with 6-foot walls near cubicles with people who may or may not be associated with the same project. The company benchmark statistics come from people in such cubicles.

3.2 The Teams

The six teams ranged in size from six to eight people. Each team consisted of a manager, three to four contract employees for the programming, and two to three business partners from within the company who were the intended (internal) customers of the applications. The teams shared the services of methodologists, technical architects, database experts, and testing specialists. For the duration of the RSDC project, the team members were not sharing their time with any other projects.

The teams in this study differ from the teams included in the company benchmark statistics on only these factors. They were neither individually selected for the teams nor newly hired for the experiment.

3.3 The RSDC Projects

The six projects selected as pilots for the RSDC were developed on client-server and mainframe platforms. The projects came from several areas of the company, including manufacturing, finance, market and sales systems, and purchasing. The projects included a global financial database, a system to measure equipment effectiveness, a retail website, a system to support competitive analysis, a bill of material audit, and a database. The projects ranged in size from 326 to 880 function points (a standard measure of size described in Section 3.5).

3.4 The Software Development Method

The software development method used at the RSDC was Fusion[3] with several new features adopted from IBM's Rapid Application Development (RAD) methodology. These features were intended to be project accelerators for achieving rapid development. Fusion is a variant of the waterfall model that was customized within the organization and primarily included all the phases in the waterfall model.

First, all projects were originally scoped to be between 600-1,000 function points and staffed to comprise about 24 staff months (usually, six people for four months). Projects used a scoping method, called "Timeboxing," which attempts to hold the time and staffing constant and requires customers to determine the functionality they most highly value that can be developed in the designated time. The customer designates a coherent subset of features that would produce a valuable product. These constraints contrast with traditional scoping, which allows the

3. Fusion is a proprietary methodology of a large IT consulting organization and this methodology has evolved with new software development technologies and approaches (Ernst and Young and Microsoft Alliance Background, www.microsoft.com/indonesia/enterprise/alliances/ernstyoung_bg.htm.)

functionality to be the driver, with the team estimating how many people and the length of time they believe it will take to finish this. It is believed that the old method produced systems that were large and difficult to finish mainly because, as the time increased, the feature set increased as well. The customer's needs changed while the project was being programmed. By fixing the time to approximately four months, it is less likely the customer's needs will have changed, making the project more likely to be completed. This is not a solution to all programming projects because some have to be larger and more complex than 600-1,000 function points. The hope is that these larger projects can be broken up into subgoals, completed in the facility, and then reassembled into coordinated code. We readdress this issue again in the discussion section.

The software development method also included prototyping and iterative design. Although one of the potential problems with prototyping is that the prototypes are not thrown away but, rather, become the first partial version of the final product, this was not a serious issue at RSDC. A customer representative was always available with the project team to assess the real product and provide his/her inputs. The projects adhered to formal project management planning and documentation procedures. The project management method had suggested paths for different kinds of projects and had formal processes for issue, risk, and change management.

As noted earlier, prior to RSDC, the firm used the Fusion methodology for software development, but the teams were not physically collocated and did not use process accelerators such as timeboxing and prototyping. Hence, the company baseline measures are computed using data collected from projects that used the Fusion methodology.

3.5 Measures

3.5.1 Productivity Metrics

A number of measures have been used in previous research to assess the output delivered from a software project relative to the effort put in. These measures of output quantity include lines of code, function points, feature points, object points, etc. [9], [3], [5], [24]. The choice of a productivity measure often depends on the domain of software projects and the languages used. For example, although lines of code is a measure of output, this metric is suitable only when comparing projects developed in the same language. Lines of code are not comparable across different languages; a line of code in COBOL is not equivalent to a line of code in Java. Similarly, object point metrics are more appropriate when the projects being compared all use object-oriented design. The projects at our research site used several different languages for applications development, including both traditional third generation languages such as C and COBOL and more recent object-oriented and web application development languages such as C++, Perl, and Java. Since only a few projects in the sample used object-oriented design and several languages were used for application development, neither lines of code nor object point metrics are appropriate for productivity measurement for our research site.

Function point measure is an abstract but workable surrogate measure for the output produced by software projects that does apply to heterogeneous projects. This metric is appropriate for measuring productivity and the performance of application software projects and is widely used in software organizations [25], [3]. Function points are the weighted sum of five different factors related to the application's functionality. These factors are: Inputs, Outputs, Logical files, Queries, and Interfaces. We used the approach specified by the IFPUG (International Function Points User Group, version 4.0) to compute the function points of the applications in our data set. In this approach, a raw score of function points is computed initially based on a weighted count of the number of the five factors in the application. This raw score is then adjusted to control the inherent complexity in the projects based on inputs collected on 14 complexity factors. These factors range from complexities in the use of data communication features, transaction rate, and data volume, differences in performance objectives, online data updating features to complexities from multiple sites, and reusability of the application. Our productivity measure is defined as the number of adjusted function points per staff month. In this metric, we include the effort of all the project team members, project manager, business analyst, architects, technical writers, and other external experts, such as database specialists, whose services were used in the projects.

For the projects used in our samples, one Function Point expert did all the counting. He has over 20 years of industry experience. This expert is also IFPUG certified and has managed several software metrics projects using Function Points and other size measures.

3.5.2 Cycle Time

Our product schedule measure is defined in number of months from the start of the project (i.e., when the project specifications are agreed upon and signed off by the customer and the project manager) to the time when the project is completed (i.e., customer acceptance testing whether the project is complete). Cycle time was measured in the number of days between the start and end dates of the project. Since the industry benchmarks and company baselines measured cycle time in months, for ease of interpretation and meaningful comparison, we converted our cycle time measure from number of days to months, retaining the precision up to two decimal places. For relative comparison of the schedule time performance across projects, we normalize the schedule time data for the size of the projects, i.e., function points in our analysis. Hence, our cycle time measure is defined as the number of months per 1,000 Function Points.

3.5.3 User Satisfaction

Our team satisfaction measure had questions about the organization of the team and the roles of each individual member, satisfaction with the war room facility, and satisfaction with facilities provided outside the war room. Responses were obtained from multiple team members for each project. The interrater reliability index ranged from 0.28 to 0.86. It was also noted that the responses from some of the team members had no variance, with a constant high satisfaction rating on all the items. One of the reasons for the low reliability index in some projects may also be due to the differences in the roles played by these team members. Cronbach alpha for pooling the constructs under team

satisfaction together ranged from 0.79 to 0.98. These measures of construct reliability exceed the threshold recommended in the literature [35]. For ease of interpretation, the composite score on team satisfaction was computed as the average of item scores.

Our customer satisfaction measure had questions about satisfaction with the system data, system performance and functionality, ease of use, and system documentation and training. The composite score on customer satisfaction is the average of scores for these items. Although an attempt was made to collect customer satisfaction data from more than one end customer for each project, the response rate was poor in some of the pilot projects and the final data ended up from a single end customer contact. In the case of projects where data from multiple respondents was available, the interrater reliability index ranged from 0.42 to 0.53. We learned from the managers at our research site that one of the reasons for this relatively low level of reliability could be due to different application exposure and technical background of these end users.

Our sponsor satisfaction data includes overall satisfaction of the sponsor with the project, cost, schedule, quality, and periodic information updates about the project. The composite score on sponsor satisfaction is the average of scores for these items. The individual item scores for sponsor satisfaction was not available.

3.5.4 Measures of Team Experience

Questionnaires were used to measure the teams' experience with and preferences for various kinds of workspaces and tools. The entrance questionnaire had 103 items and took about 20 minutes to fill out. This questionnaire asked about the person's prior *experience* with various facilities and technologies using a 5-point scale anchored with "not at all" to "very frequently." This questionnaire assessed team members' predictions of how frequently they *would use* new facilities like war rooms and hotelling space, conference rooms, etc., at the RSDC, as well as new technologies available to them at the RSDC. The questionnaire also asked team members to assess how well they *liked* to work in various facilities or with various tools, using a modified 5-point Likert[4] scale with "strongly dislike" to "strongly like" as the anchors. The questionnaire also asked for team members to assess their preferred work styles, again using 5-point Likert scales, with "strongly disagree" to "strongly agree" anchors. The latter items are listed in Table 1. Nine items, starred in the table, are those intended to assess the kinds of preferences that people have about characteristics we thought the new facilities would engender, such as being busy and collaborative.

The exit questionnaire contained 71 items of similar content as in the entrance questionnaire. However, instead of predicting the future frequency and preference, these asked how often team members *did* use various spaces and technologies and how they *liked* or *disliked* them. This questionnaire also asked the same questions shown in Table 1.

To better understand the reasons behind their opinions, all the team members from two of the pilot teams were interviewed individually. These team members were encouraged to talk freely about the advantages and disadvantages of the war rooms, the aspects of team dynamics, their attitudes about the layout and equipment in the rooms, why and when they used hotelling space, and other features they would change to make the RSDC a better environment. Interviews took about one hour to 1 1/2 hours.

For the two teams studied in-depth, we asked each individual to fill out a short 9-item questionnaire biweekly, asking both for some quantitative assessment of the team and war rooms, but also some narrative responses to back up their numerical responses, answering in each case "Why or why not?" Although we will use a sample of their statements in the results, the numerical data is not analyzed because the response rate was variable and we only assessed this in two groups.

We observed the same two teams for about 8-10 hours over the course of the projects. We sat in on meetings and conference calls, watched teams solve various kinds of problems, and photographed their use of various artifacts and tools in their rooms. Observation notes were transcribed immediately after each observation session and then clustered by two of the researchers into major categories of actions/statements relating to the research question.

4 Results

4.1 Project Outcomes in Pilot Projects

We compared the productivity of the pilot teams with two benchmarks, shown in Table 2. We compared the Function Points/Staff Month and the Cycle Time to both the industry standard for projects of comparable sizes and the baseline for the company. The company baseline numbers for productivity and cycle time were computed in the following way: A consulting firm that specializes in software metrics was hired to select a sample of past software projects from multiple domains and functional areas within the organization. No outlier projects, i.e., extremely large in size or with long duration, were selected. Ninety-three projects were selected from different functional areas in the organization and the productivity and cycle time numbers of these projects were used to arrive at the company baseline measures. The sample of projects were divided into two subsamples based on the two major platforms for development, i.e., mainframe and client-server used in the organization. The baseline measures for mainframe and client-server projects were computed using simple average of the data collected from these projects.

Since the software process used prior to RSDC involved all the stages in the waterfall model and the RSDC projects started with a well-documented design, we adjusted the baseline productivity and cycle time numbers to include only the stages covered in RSDC.[5]

4. Likert scales are those that assess the respondent's agreement or disagreement with a statement such as "This was an excellent course," with a 5-point scale below it reflecting the possible responses "Strongly agree, agree, neutral, disagree, and strongly disagree."

5. While verifying whether the numbers given to us were comparable, we found differing recollections. We therefore adopted a conservative approach; we adjusted the company baseline numbers, assuming that the given company baseline numbers were not adjusted, and included all the stages in the waterfall model right from requirement specification. The software engineering literature reports that the front-end investments (in terms of effort and time) for the stages up to the detailed design in the waterfall model range from 25-35 percent of the total development effort [9]. We conservatively assumed that 35 percent of the time and effort was expended in the stages up to the detailed design and adjusted the given baseline numbers by multiplying the given number by 1/65 percent, i.e., 1.66.

TABLE 1
The 16 Items Used in the Assessment of Workstyle

1. *I prefer to work independently as much as possible.
2. *I am easily distracted by activities taking place in the same room as me.
3. I would like to work in a paperless office.
4. People working on special projects need to stay connected to their reporting manager.
5. I prefer reading from print on paper than reading on the screen.
6. *Regular meetings are necessary for my work.
7. I would like to personalize my workspace (e.g. photographs, personal items).
8. I like working on tasks defined by others.
9. *I like to collaborate as much as possible.
10. *I think that it is important for people on special projects to work near each other.
11. *I like working in a busy environment.
12. I would like to work at different locations during the day.
13. *I am concerned about how my contribution to the project will be recognized.
14. I prefer participating in defining the tasks I work on.
15. *Running into people I know in the hallway is a convenient way of interacting.
16. *Technology will replace the need for people in the same project to work near each other.

TABLE 2
Comparative Statistics on Productivity Measures

	Pilot Teams	Company Baseline	Industry Standard
FP/Staff Month *(higher is better)*	29.49 (SD = 7.88)	14.18	11.25
Cycle Time per 1000 FPs *(lower is better)*	7.64 (SD = 3.37)	12.74	11.75

The industry standard numbers were obtained from the SPR database after adjusting for the size of the projects used in our sample [26]. SPR presents industry benchmarks for projects implemented on mainframe and client-server platforms. These benchmarks are presented for project sizes ranging from 200 to over 10,000 function points and are also adjusted to include only the stages after detailed design in the development process. We used these adjusted industry benchmarks, often referred to as physical function point measures, for projects with 600 function points in our comparison shown in Table 2. We selected 600 function points since the mean size of projects implemented in RSDC is 600. The sample of pilot projects included an equal number of mainframe and client-server projects and, hence, we computed the average of mainframe and client-server benchmark numbers for our comparison.

The pilot teams produced double the number of function points per staff month from the previous company baseline and more than double from the industry standard. Using the variance of the measures among the six pilot teams in an estimate of the variance of the means, the pilot team metrics are significantly different from the means of both the company baseline (t = 5.67, p < .001) and the industry standard (t = 4.76, p < .001). The pilot teams did not have a lower cycle time than either the company or industry standard.

The team, sponsor, and end user satisfaction measures are shown in Table 3. These scales ranged from 1 = very dissatisfied to 5 = very satisfied. Although we do not have baseline information with which to compare, the scores are all high.

In summary, the pilot teams were remarkable in their productivity while not sacrificing team, sponsor, or end user satisfaction with the resulting products.

4.2 Is There a Sample Selection Bias in the Pilot Teams?

Because the six teams studied here were pilot teams for the RSDC, there is the possibility that they were unique in some way that led to these significant productivity gains. To explore this possibility, the same performance measures

TABLE 3
Satisfaction Measures for our Pilot Teams

Team Satisfaction	4.15
Sponsor Satisfaction	4.56
End User Satisfaction	3.68

were gathered from seven teams subsequently using the facility. We ensured that these projects had the same mixture of mainframe and client-server platforms as our pilot projects.

The results are shown in Table 4. Subsequent teams were even more productive than the pilot teams. Specifically, the Function Points per Staff Month almost doubled (t (8) = 2.56, $p < .03$),[6] whereas the Cycle Time stayed the same ($t(11) = 0.47$, *n.s.*).

The satisfaction of the subsequent teams was very similar to that of the pilot teams, as shown in Table 5. (T tests comparing pilot teams with subsequent teams were nonsignificant for all measures of satisfaction.)

The productivity gains shown by the pilot teams appear not to be unique because subsequent teams using the RSDC appear to show the same, or even greater, productivity. We can only conjecture about why the second increase. There may have been learning in the team members who had already worked in the RSDC facility and were now part of the follow-on teams. There may also have been learning in the support people, including people who consult with teams on process.

4.3 Behavioral Results

The questionnaires, interviews, and observations give insight as to what was likely to have caused this productivity increase.

4.3.1 Team Members Liked the War Rooms

Based on responses to the entry questionnaires, team members had very little experience with war rooms prior to being assigned to work at the RSDC. Using ratings on a scale from 1 to 5, where 1 = not at all and 5 = very frequently, our sample rated their prior war room experience at 2.17. The entry and exit questionnaires also asked team members to rate how much they like working in various types of workspaces (where 1 = strongly dislike to 5 = strongly like). As shown in Table 6, working at the RSDC significantly increased their preferences for war rooms (t (5) = 2.59, $p < .05$) and decreased significantly preferences for working in cubicles (t (5) = 2.82, $p < .05$).

4.3.2 How Did the Teams Work in the War Rooms?

From the observations of the teams and the exit interviews, war rooms were found to benefit software development by supporting interactive, continuous communication between the team members. The close quarters of war rooms supported impromptu communication and allowed people to overhear each other. For example, as shown in Fig. 2, the team members easily moved from two small meetings to one involving everyone because they could overhear each other. One team member wrote: "It's great having everyone in the same room; so people can be brought into discussions on the fly and conversations can be overheard and commented on when necessary."

War rooms also provided other kinds of awareness. For example, if someone was having difficulty with some aspect of the coding or design, then others walking by, seeing the activity over their shoulders, stopped to help the individual. Also, when one team member was explaining something to another, other team members could overhear and spontaneously interject commentary, clarifications, or corrections. One team member commented that it was a good environment in which to learn. Coming into the team late, he was able to come up to speed very quickly. Another commented that she was able to learn by osmosis; she said that, even if she didn't listen carefully, she could make mental notes of what people were saying.

We also noticed that living in close proximity with others seemed to promote horizontal communication between team members so that team unity and coordination was facilitated. One team member commented that it was easier to develop a "group mentality." Another team member noted that he felt he worked harder in this space; he was too embarrassed to search on the web or read e-mail too much, let alone make personal phone calls. Yet another wrote: "For the most part, being located together in a small room tends to be productive—keeps us on track/focus and keeps outside disturbances to a minimum."

On the other hand, the fact that team members were in interactive, continuous communication had several drawbacks. First, overhearing distracts those doing work that requires concentration. Programmers discussed their desire to work in a "flow state," which they described as requiring quiet for intensive concentration during debugging, problem solving, or coding. One programmer wrote: "I would be much more productive if I had time to myself." Another reported: "I need to hypnotize myself to zone in on what I'm doing. If anyone talks about something I have some expertise on, I have half an ear to it."

When there was more than one informal meeting going on, the noise level would increase significantly. The rooms often appeared strikingly chaotic and active to observers and visitors. In addition, individuals reported being distressed occasionally by the lack of privacy. To explain why he had given the war room a low rating, one team member wrote: "Lack of privacy. Eavesdropping is unavoidable—constantly getting sucked into side conversations. No place for refuge. I can't imagine what would happen if [manager] had to reprimand a team member—who would then have to go back in and sit with them!"

Team members also reported being uncomfortable about exposing half-baked ideas in front of the customer or revealing that they had chosen to take various shortcuts in the implementation. In addition, the increased visibility of work led them to report that they felt their work was being monitored too closely by their project manager. Living in close quarters enhanced motivation by facilitating team successes, but it also occasionally exacerbated tension and bad feelings when things were not going well on the project.

When the war rooms got very noisy, team members sometimes went to the nearby hotelling area or to a

6. The t test for Function Points per Staff Month as calculated adjusted for unequal variance, $F = 5.53$, $p < .04$, which also alters the degrees of freedom.

TABLE 4
Comparison of the Pilot with Subsequent Teams in the RSDC

	Pilot Teams	Subsequent Teams	*significance*
FP/Staff Month *(higher is better)*	29.49 (*SD* = 7.88)	49.28 (*SD* = 18.52)	*p<.01*
Cycle Time *(lower is better)*	7.64 (*SD* = 3.37)	6.34 (*SD* = 5.93)	*n.s.*

TABLE 5
Comparison of the Satisfaction of Pilot and Subsequent Teams in the RSDC

	Pilot Teams	Subsequent Teams	*significance*
Team Satisfaction	4.15 (*SD* = .25)	4.30 (*SD* = .40)	*n.s.*
Sponsor Satisfaction	4.56 (*SD* = .55)	4.29 (*SD* = .54)	*n.s.*
End User Satisfaction	3.68 (*SD* = .52)	3.97 (*SD* = .51)	*n.s.*

TABLE 6
Comparisons of Entry versus Exit Questionnaire Data

	Entry	Exit	*significance*
Preferences for war rooms	3.53 (*SD* = .36)	4.0 (*SD* = .46)	*p<.05*
Preferences for cubicles	3.86 (*SD* = .46)	3.43 (*SD* = .46)	*p<.05*

conference room to work. Although quiet and private, the person lost the awareness that comes from overhearing teammates and was not immediately available to them. In addition, once out of the war room, there was no immediate access to shared materials like the diagrams and lists on whiteboards or flip charts. The major deterrent of the hotelling area was that, in order to use it, the team member had to physically withdraw from the team. This behavior was informally discouraged as it was interpreted as "escaping" or "hiding out" from the rest of the team, with self-deprecating comments from the team members who were left behind in the room. For example, team members noting someone was packing up to leave would smile and say, "Was it something I said?" Some programmers did leave occasionally, but they reported that they felt they were *supposed* to work in the war room. One team member who occasionally worked outside the war room wrote: "Need to get away sometimes and think, but it is not easy to switch environments quickly. You get used to an area." Several programmers reported that they would achieve this desired quiet by coming in early or staying after others had left in order to work with some peace and quiet *in* the war room where they could view all the flip charts and diagrams on the whiteboard.

4.3.3 How the Facilities Impacted the Collaboration and Communication

The entry and exit questionnaires also revealed that attitudes about the activity in the room changed over time. As shown in Table 7, at project completion, individuals were significantly less distracted by the presence of others in the room (t (5) = 3.64, p < .01).

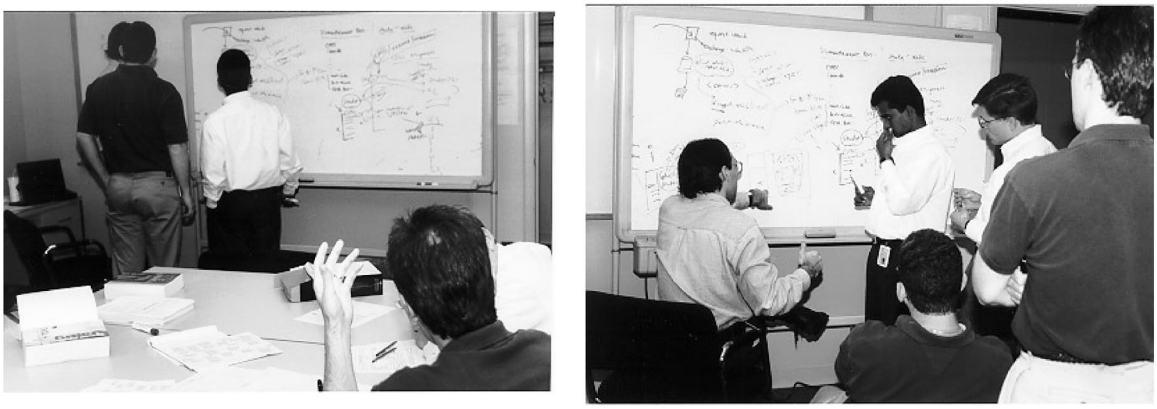

Fig. 2. Two phases of interaction in the RSDC: The team moving from two separate meetings to one central one, from overhearing.

TABLE 7
Changes in Reported Attitudes about Activity in the War Rooms

	Entry	Exit	*significance*
Susceptibility to distraction	3.37 (*SD* = .49)	2.68 (*SD* = .43)	*p*<.001
Concern about individual recognition	2.87 (*SD* = .31)	3.29 (*SD* = .49)	*p*<.025

Comments made in the interviews suggest why these changes occurred. First, team members reported that they learned to work in the noisy environment and were able to "tune in" and "tune out" of the simultaneous activities. As the project progressed through different stages, the advantages of interaction became more relevant to the project and to each individual's work. Over time, team members felt that the familiarity with each other increased the ease and enjoyment with which they could work in close proximity, so that the lack of privacy and increased visibility became less troublesome. Finally, as the project progressed, there was less uncertainty about the project outcome, members' roles, and the mechanics of the overall team process. As a consequence, team members reported that concentration was easier and there was less need to consult with the project manager.

On the other hand, team members were increasingly worried about having their individual contributions recognized by their manager. They worried about how working so closely as a team might detract from the perception of their individual competence. They worried about how the company's individual-based merit structure would acknowledge the success of the group. Team members reported being uncertain about the future once they left the RSDC and returned to their regular positions. These concerns suggest that the company reward structure might have to be changed from individual to group-based if employees are to feel comfortable leaving their regular positions to participate in these kinds of group projects.

4.3.4 What Did Teams Use to Support Collaboration in the War Rooms?

The whiteboards and flip charts located in the war rooms supported collaboration by providing large public workspaces that served as a visible permanent record of group activity and decisions. Knowledge workers in particular can benefit from sharing external representations of their expertise to articulate their work in progress and to solicit feedback from others [34]. At the RSDC, the collaboration tools were used for synchronous work, such as when several team members would gather at them for small meetings, and for asynchronous work, such as when an individual would change or add something for others to see. Teams used the whiteboards to keep the current status of the project plan visible to everyone. In addition, "to-do" lists were also posted on whiteboards or flipcharts to indicate progress and to note when an individual team member was assigned responsibility for an item. The large and public characteristics of these tools helped the teams view progress at a glance. Teams also made use of the printing capability of the whiteboards to capture information posted there and make use of it at their own desks. As one team member wrote: "Electronic white board rules." Another wrote: "I'm in love with the electronic whiteboard. It's getting to where I can't think or speak without it."

4.3.5 Facilities to Support Various Modes of the Work

Based on the many hours spent observing the teams as they worked in the war rooms, hotelling cubicles, and conference rooms, we found that the teams worked in a number of different modes through the life of the project. Synthesizing the reports of the kinds of activity they were involved in at various points and in various rooms,[7] we identified nine different kinds of work, shown in Table 8.

Being in a war room supported most modes of work because the space was large enough to monitor (or not) what was appropriate for each individual at any given time. However, war rooms were not ideal for all nine modes of work. On the occasions when the group activity did not involve everyone (e.g., discussion about customer input, training), people attempting to work alone while activity surrounded them could be distracted. On some occasions, meetings of a subset of team members moved to conference rooms nearby. In addition, some war rooms were larger than others, putting distance between the solo worker and the subteam meetings. The distance reduced the noise and allowed more parallel activities to occur without having to change rooms. One team member said that he thought being collocated would support some phases of development more than others, that rapid debugging and integration were especially appropriate for this style of work.

Privacy was the second issue that affected the match between work mode and facilities. Private conversations with outsiders and conversations about political issues were typically moved outside the war room. In fact, the most frequent use of the hotelling area was to make private phone calls. Political or sensitive issues were moved to conference rooms so that there was no inappropriate overhearing.

5 Discussion

The data from this study are striking. Groups working in the RSDC showed significantly improved productivity and high levels of satisfaction by everyone involved, from team member to customer. The significant improvements in productivity over the company baseline are most likely

7. Lisa Covi, the main observer of these teams, has a number of years of experience in systems analysis and design, as well as PhD training in observational methods and codification. Because of her systems background, she knew the domain; because of her social science training, she carefully followed the method of informed observation.

TABLE 8
Nine Kinds of Work

1. Discussion to acquire customer input
2. Discussion of a political issue
3. Problem solving at the whiteboard
4. Status meeting using the to-do list (usually on a flip chart or the whiteboard)
5. Team building discussion (social)
6. Training
7. Simultaneous problem solving meetings (subsets of team members)
8. Working solo (typically coding)
9. Private conversations with outsiders

due to the tight fit between the development method (timeboxing) and the collaborative facilities. War rooms contributed to the enhanced software development because they constituted a collaborative information system, which facilitated communication and continual awareness [23].

There has been a follow-on to the pilot projects at this company. Because of the positive results for the pilot teams, the company has built 112 war rooms in a facility solely dedicated to software development. Groups of eight war rooms constitute a "neighborhood" in which a central area houses the support services (e.g., the database expert, the methodologist, etc.). The rooms themselves are both large and configured in the "E" style used in a few of the pilot teams (see Fig. 1). The entire end wall of the room is whiteboard (and, indeed, they ask for stepladders so they can use the entire surface). In informal training or status meetings, teams project a laptop from a portable stand onto the whiteboard, which has a semigloss surface suitable for viewing.

This paper has shown that, when people are radically collocated on a software development team using a variant of Fusion that includes timeboxing, productivity goes up and timeliness increases. Collocation brings interactive, continuous communication, which allows overhearing and awareness of teammates' activities. This helps for clarification, problem solving, and learning. It also enhances team building. Although there are disadvantages to the interactive, continuous communication—in that there is little privacy, work is exposed, and interruptions occur—people perceive that the value of the coordination outweighs the liabilities. This is an important result. There may be tools to enhance software development and ways to choose expert team members. But, one of the simplest solutions might be to allow the team members the kind of easy access to each other that occurs when they are collocated.

This is a serious challenge in the face of today's push to constitute teams across geographic locations, called "virtual collocation." There are clear problems with remote teams. There is evidence that software engineering, specifically the closing of open issues, is done significantly more slowly when it has to be coordinated across locations [20]. People do not trust each other as much if they are from remote locations [40]. And, partially distributed teams will reconfigure work practices so that the tightly coupled work is all being done by collocated members [34]. The challenges to remote work are summarized in a recent paper with a conclusion for a title, "Distance Matters" [33]. There may be technologies in the future that provide the kind of interactive, continuous communication that radical collocation gives. But, until that time, we will still see losses when teams are not collocated.

Although this study presents empirical evidence and valuable insights on the potential benefits of team collocation in software projects, there are some limitations. Because our sample size is relatively small, the results need to be validated using a larger sample. Our data sources are too insufficient to rule out various possible confounds. We do not have measures of the teams' incoming capabilities or experience, we do not know if there was an altered emphasis on designing for reuse, and we do not have evidence to allow us to claim that this speed-up did not result in poorer quality software, other than that the sponsors and customers were satisfied. Also, note that the industry benchmarks used for comparison in this study are from the SPR database. Although we selected the appropriate benchmark figures noting the size of projects studied in this sample, the SPR data is known to have some noise in it due to multiple data sources and aggregation bias.

It is also possible that a major source of the productivity enhancement came from the fact that the projects were merely "timeboxed." However, the fact that the company invested substantial money in a new facility for team collocation is a strong indicator that they felt the software method alone was not driving the success of the RSDC teams. In fact, referring to the new facility, a company Vice President said: "The building is an essential tool for achieving speed and productivity [because] it is all about facilitating communication." In addition, we believe both manageable project scope and smaller team size may be *required* to reap the full benefits of team collocation. Collocation of large teams may not be feasible and can also add to the complexity.

This raises a related issue. It is not clear how the results of this study apply to large projects. Future research will have to assess the company's current approach to large projects, given these results. Faced with this issue, the company is splitting large projects into smaller subsystems and collocating these subsystem teams, putting associated

teams in rooms in the same "neighborhood," a hallway of war rooms with shared resources in the hall area and lounge areas nearby.

We also do not have accurate information on the quality of the software delivered by RSDC facility. There is a need to assess the problems and defects encountered in both the prerelease and postrelease of these software products to end users to ensure that the up-front productivity and cycle-time benefits are not at the cost of downstream defects.

Also, we do not know without future research whether the increased productivity was due to the Hawthorne effect, where people merely worked harder because of the novelty of the situation and the fact that we were collecting data. Only by tracking additional follow-on teams for whom this approach is not novel anymore can we discern the answer to this. Long-term research is also the only way to tell whether the teams were "sprinting," working extra hard because the project was only a few months long. As Kidder pointed out in "The Soul of a New Machine," however, this kind of sprinting leads to burnout and loss of the best people [27]. We did hear comments that included: "If all my projects used this methodology, I'd look for a different job." In contrast, however, recall that the general sentiments were significantly more positive than their going-in expectations and that there were a number of very positive comments about how effective it was to have team members "at hand."

ACKNOWLEDGMENTS

The authors are grateful for the access and support provided by the automobile company involved in this study. Additional support for the work was provided by Steelcase, Inc., as a grant to the Collaboratory for Research on Electronic Work (CREW) at the University of Michigan. A shorter version of this paper was presented as "How Does Virtual Collocation Help a Team to Succeed" at the ACM Conference on Computer Supported Cooperative Work (CSCW) 2000, December, Philadelphia [44]. That paper reported on some of the results here, with particular focus on how to support face-to-face groups in general and implications for supporting long-distance teams to work as well as these teams did. This paper has more theory, more results from the interview data, and addresses the issues having to do with software engineering in particular.

REFERENCES

[1] M. Alavi, "An Assessment of the Prototyping Approach to Information Systems Development," *Comm. ACM,* vol. 27, no. 66, 1984.

[2] M. Alavi, R. Nelson, and R. Weiss, "Strategies for End-User Computing: An Integrative Framework," *J. MIS,* vol. 4, no. 3, 1988.

[3] A. Albrecht and J. Gaffney, "Software Function, Source Lines of code, and Development Effort Prediction: A Software Science Validation," *IEEE Trans. Software Eng.,* vol. 9, no. 6, pp. 639-647, Nov. 1983.

[4] T.J. Allen, *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information within the R&D Organization.* Cambridge, Mass.: MIT Press, 1977.

[5] R.D. Banker, S. Datar, C.F. Kemerer, and D. Zweig, "Software Complexity and Software Maintenance Costs," *Comm. ACM,* vol. 36, pp. 81-93, Nov. 1993.

[6] D. Barstow, "Artificial Intelligence and Software Engineering," *Proc. Ninth Int'l Conf. Software Eng.,* vol. 9, no. 5, pp. 541-561, 1987.

[7] R.L. Baskerville and J. Stage, "Controlling Prototype Development through Risk Analysis," *MIS Quarterly,* vol. 20, no. 4, 1996.

[8] F. Becker and F. Steele, *Workplace by Design: Mapping the High Performance Workscape.* San Francisco: Jossey-Bass, 1995.

[9] B.W. Boehm, *Software Engineering Economics.* New York: Prentice Hall, 1981.

[10] C.V. Bullen and J.L. Bennett, "Groupware in Practice: An Interpretation of Work Experiences," *Computerization and Controversy,* second ed., R. Kling ed., pp. 348-382, 1992.

[11] H.A. Clark, *Using Language.* Cambridge, U.K.: Cambridge Univ. Press, 1996.

[12] L.M. Covi, J.S. Olson, E. Rocco, W.J. Miller, and P. Allie, "A Room of Your Own: What Do We Learn About Support of Teamwork from Assessing Teams in Dedicated Project Rooms," *Cooperative Buildings: Integrating Information, Organization and Architecture: Proc. First Int'l Workshop,* 1998.

[13] B. Curtis, H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems," *Comm. ACM,* vol. 31, no. 11, pp. 1268-1287, 1988.

[14] *Cognition and Communication at Work,* Y. Engestrom and D. Middleton, eds. Cambridge, U.K.: Cambridge Univ. Press, 1996.

[15] N.E. Fenton, *Software Metrics: A Rigorous Approach.* New York: Chapman and Hall, 1996.

[16] M.D. Fraser, K. Kumar, and V.K. Vaishnavi, "Strategies for Incorporating Formal Specifications in Software Development," *Comm. ACM,* vol. 37, no. 10, 1994.

[17] W.W. Gibbs, "Software's Chronic Crisis," *Scientific Am.,* pp. 86-95, Sept. 1994.

[18] C.M. Giglio, "Business Is War, But You Can Fight Back with a Killer APP—A Virtual War Room" *Infoworld,* vol. 21, no. 18, p. 72, 1999.

[19] J. Greeno and J. Moore, "Situativity and Symbols: Response to Vera and Simon," *Cognitive Science,* vol. 17, no. 1, pp. 49-59, 1993.

[20] J.D. Herbsleb, A. Mockus, T.A. Finholt, and R.E. Grinter, "An Empirical Study of Global Software Development: Distance and Speed," *Proc. 23rd Int'l Conf. Software Eng.,* pp. 81-90, May 2001.

[21] D.J. Hoch, C.R. Roeding, G. Purkert, and S.K. Lindner, *Secrets of Software Success.* Boston: Harvard Business School Press, 2000.

[22] E. Hutchins, *Cognition in the Wild.* Cambridge, Mass.: MIT Press, 1995.

[23] E. Hutchins, "Constructing Meaning from Space Gesture and Speech," *Discourse, Tools, and Reasoning: Essays on Situated Cognition,* L.B. Resnick, R. Saljo, C. Pontecorvo, and B. Burge, eds., pp. 23-40, 1997.

[24] J. Johnson, "CHAOS: The Dollar Drain of IT Project Failures," *Application Development Trends,* vol. 20, no. 1, pp. 41-44, 1995.

[25] C. Jones, *Applied Software Measurement: Assuring Productivity and Quality.* New York: McGraw-Hill, 1996.

[26] C. Jones, *Software Assessments Benchmarks and Best Practices,* New York: Addison-Wesley, 2000.

[27] T. Kidder, *The Soul of a New Machine.* New York: Avon Books, 1982.

[28] R. Kraut, C. Egido, and J. Galegher, "Patterns of Contact and Communication in Scientific and Research Collaboration," *Intellectual Teamwork: Social and Technological Foundations of Collaborative Work,* J. Galegher, R.E. Kraut, and C. Egido, eds., pp. 149-171, Hillsdale, N.J.: Erlbaum, 1990.

[29] R.E. Kraut and L.A. Streeter, "Coordination in Large Scale Software Development," *Comm. ACM,* vol. 38, no. 7, pp. 69-81, 1995.

[30] M.S. Krishnan and M.I. Kellner, "Measuring Process Consistency: Implications for Reducing Software Defects," *IEEE Trans. Software Eng.,* vol. 25, no. 6, pp. 800-816, Nov./Dec. 1999.

[31] M.S. Krishnan, S. Kekre, C.H. Kriebel, and T. Mukhopadhyay, "An Empirical Analysis of Productivity and Quality in Software Products," *Management Science,* vol. 46, no. 6, pp. 745-759, June 2000.

[32] K.C. Laudon and J.P. Laudon, *Management Information Systems: New Approaches to Organization and Technology: New Approaches to Organizations and Technology.* New York: Prentice Hall, 1998.

[33] G.M. Olson and J.S. Olson, "Distance Matters," *Human Computer Interaction,* vol. 15, pp. 139-179, 2001.

[34] J.S. Olson and S.D. Teasley, "Groupware in the Wild: Lessons Learned from a Year of Virtual Collocation," *Proc. Conf. Computer Supported Cooperative Work,* pp. 419-427, Nov. 1996.
[35] J.C. Nunnally and I.H. Bernstein, *Psychometric Theory.* New York: McGraw-Hill, 1994.
[36] D.E. Perry, N.A. Staudenmayer, and L.G. Votta., "People, Organizations, and Process Improvement," *IEEE Trans. Software,* vol. 20, no. 7, pp. 36-45, July 1994.
[37] S.E. Poltrock and G. Engelbeck, "Requirements for a Virtual Collocation Environment," *Information and Software Technology,* vol. 41, no. 6, pp. 331-339, 1999.
[38] *Perspectives on Socially Shared Cognition.* L.B. Resnick, J.M. Levine, and S.D. Teasley, eds. Washington: APA Press, 1991.
[39] B.R. Rich and L. Janos, *Skunk Works.* Boston: Little, Brown and Company, 1994.
[40] E. Rocco, T.A. Finholt, E.C. Hofer, and J.D. Herbsleb, "Out of Sight, Short of Trust," *Presentation at the Founding Conf. European Academy of Management,* Apr. 2001.
[41] W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," *WESCON Western Electronic Show and Convention,* 1970.
[42] W. Royce, *Software Project Management: A Unified Framework.* Addison-Wesley, 1998.
[43] S. Sawyer, J. Farber, and R. Spillers, "Supporting the Social Processes of Software Development Teams," *Information Technology and People,* vol. 10, no. 7, pp. 46-62, 1997.
[44] S.D. Teasley, L. Covi, M.S. Krishnan, and J.S. Olson, "How Does Radical Collocation Help a Team Succeed?" *Proc. ACM Conf. Computer Supported Cooperative Work (CSCW '00),* pp. 339-346, Dec. 2000.
[45] S.D. Teasley and J. Roschelle, "Constructing a Joint Problem Space: The Computer as a Tool for Sharing Knowledge," *Computers as Cognitive Tools,* S.P. Lajoie and S.D. Derry eds., pp. 229-258, 1993.
[46] *Java Technologies: Case Studies,* Edward Jones: Retirement Planning System: http://www.sun.com/java/javameansbusiness/edwardjones, 1999.
[47] "How to Jump-Start Java Computing Initiatives," *Sun J.,* Nov. 1999, http://www.sun.com/SunJournal/v1n4/global2.html.

Stephanie D. Teasley received the BA degree from Kalamazoo College, the PhD degree in psychology from the University of Pittsburgh, and did postdoctoral research in the Psychology Department at the University of Michigan. She is a senior associate research scientist at the Collaboratory for Research on Electronic Work (CREW) and the School of Information, both at the University of Michigan. She has also been affiliated with the Learning Research and Development Center in Pittsburgh and the Institute for Research on Learning in Palo Alto, California. Her current research focuses on technology use to support key aspects of collaboration for both collocated groups and distributed groups. She is the collaboratory director for the Great Lakes Regional Center for AIDS Research. Her work has appeared in several journals, including *Science*, and she is the coeditor of *Perspectives on Socially Shared Cognition.*

Lisa A. Covi earned the BS degree in mathematics from Carnegie-Mellon University, Pittsburgh, and the MA degree in higher education administration from Columbia University, New York. She received the MS and PhD degrees in information and computer science from the University of California, Irvine, in 1996 and conducted postdoctoral research with Teasley and Olson at the University of Michigan, Ann Arbor. She is an assistant professor in the School of Communication, Information and Library Studies (SCILS) at Rutgers, The State University of New Jersey. Her work has addressed social aspects of computerization and networked information through the study of digital libraries, collaborative technologies, and service-provider collaboratories. She has published research findings in *The Journal of the American Society for Information Science & Technology*, *Information Processing & Management*, and *The Information Society*, and serves on the editorial board of the *Journal On Digital Information (JODI).* She has been a leader in the development of Rutgers' new undergraduate major in Information Technology and Informatics and she has given talks and workshops on teamwork and collaboration at more than 80 colleges and universities. She is a member of the IEEE Computer Society.

M.S. Krishnan received the PhD degree in information systems from the Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, in 1996. He is the Mary and Mike Hallman e-Business Fellow, the director of eLab, and an associate professor of computer information systems at the University of Michigan Business School. He was awarded the ICIS Best Dissertation Prize for his doctoral thesis on "Cost and Quality Considerations in Software Product Management." His research interests include the modeling of issues related to customer satisfaction, quality and productivity in information systems, business value of information technology, return on investments in software process improvements, software engineering economics, metrics and measures for quality, and customer satisfaction for products in software and information technology industries. His research articles have appeared in several journals, including *Management Science, Information Technology and People, Harvard Business Review, IEEE Transactions on Software Engineering, Decision Support Systems, Information Week*, and *Communications of the ACM.* He is a member of the IEEE Computer Society.

Judith Olson received the BA degree from Northwestern University, Chicago, the MS and PhD degrees from the University of Michigan, Ann Arbor, and did a year's postdoctoral research at Stanford University, Palo Alto, California. She is the Richard W. Pew Chair in Human Computer Interaction at the University of Michigan Business School, the School of Information, and the Psychology Department. She was a faculty member at Michigan in psychology for 10 years before moving to a managerial position at Bell Labs in human computer interaction. She returned to the University of Michigan Business School and became a charter member of the new School of Information. She is active in CREW (Collaboratory for Research on Electronic Work) and has served on a number of national committees, notably, the National Research Council on Human Factors. She has more than 70 publications, with current research on the nature of effective electronic support for remote group work.