Journal of the Association for Information Systems

Volume 10
Issue 3 *Special Issue on Collaboration Engineering*

Article 5

3-25-2009

Toward Building Self-Sustaining Groups in PCR-based Tasks through Implicit Coordination: The Case of Heuristic Evaluation

Paul Benjamin Lowry
Brigham Young University, paul.lowry.phd@gmail.com

Tom L. Roberts
Louisiana Tech University, troberts@CAB.latech.edu

Douglas L. Dean
Brigham Young University, doug_dean@byu.edu

George Marakas
University of Kansas, gmarakas@ku.edu

Recommended Citation

Lowry, Paul Benjamin; Roberts, Tom L.; Dean, Douglas L.; and Marakas, George (2009) "Toward Building Self-Sustaining Groups in PCR-based Tasks through Implicit Coordination: The Case of Heuristic Evaluation," *Journal of the Association for Information Systems*, 10(3), .
DOI: 10.17705/1jais.00189
Available at: https://aisel.aisnet.org/jais/vol10/iss3/5

Journal of the Association for Information Systems
JAIS

Special Issue

Toward Building Self-Sustaining Groups in PCR-based Tasks through Implicit Coordination: The Case of Heuristic Evaluation*

Paul Benjamin Lowry
Information Systems Department
Brigham Young University
Paul.Lowry.PhD@gmail.com

Douglas L. Dean
Information Systems Department
Brigham Young University
doug_dean@byu.edu

Tom L. Roberts
Department of Management and Information Systems
Louisiana Tech University
troberts@CAB.latech.edu

George Marakas
University of Kansas
gmarakas@ku.edu

Abstract

Usability flaws found in the later stages of the software development process can be extremely costly to resolve. Accordingly, usability evaluation (UE) is an important, albeit usually expensive, part of development. We report on how the inexpensive UE method of heuristic evaluation (HE) can benefit from collaborative software (CSW), implicit coordination, and principles from collaboration engineering. In our study, 439 novice participants were trained in HE methods and then performed HE. Our results show that traditional nominal HE groups can experience implicit coordination through the collaborative software features of group memory and group awareness. One of the key results is that CSW groups had less duplication of effort than traditional nominal groups; these differences were magnified as group size increased from three to six members. Furthermore, because they coordinated less, traditional nominal groups performed more work in the overall process of HE. We attribute the reduction in duplication for CSW-supported groups to the implicit coordination available to them; CSW-supported groups could see violations input by other group members, but could not directly discuss the violations. These findings not only show the power of implicit coordination in groups, but should dramatically change how HE is conducted. These results may also extend to other evaluation tasks, such as software inspection and usability assessment tasks.

Keywords: *heuristic evaluation, collaboration engineering, virtual groups, virtual teams, group size, usability evaluation, human-computer interaction, thinkLets, collaboration, collaborative software*

* Robert O Briggs, Gert-jan de Vreede and Anne Massey were the accepting editors.

Volume 10, Special Issue, pp. 170-195, March 2009

Toward Building Self-Sustaining Groups in PCR-Based Tasks through Implicit Coordination: The Case of Heuristic Evaluation

1. Introduction

Organizations need to develop software for users who are increasingly distributed and diverse. Moreover, in today's competitive environment, this software needs to be usable and economical to develop. However, professional usability evaluation (UE), a major cost of software development, has become so expensive that it may not be feasible for any but the largest and most profitable firms (Mayhew, 1999). These firms spend nearly $60 billion annually on such efforts (Cusumano, 2004). Because of high costs, many software engineers avoid usability engineering techniques because they are complex and time consuming and there is a questionable return on their costs. This trend has resulted in less usable software that end users are less prone to adopt, which undermines software development efforts (Nielsen, 1994).

To counter these trends, leading researchers and software engineers are turning to "discount" usability techniques that can be executed by novice evaluators, including end users, leading to time and cost savings (Mayhew, 1999). Most of these efforts center on inspection tasks that are based on the Preparation, Collection, and Repair (PCR) process that many organizations already use to eliminate defects (IEEE, 1989). However, most organizations have not yet adopted collaborative software (CSW) and processes to help with PCR-based tasks because of the lack of effective and economical collaborative tools and processes to support those tasks.

We believe that the concepts involved in collaboration engineering (CE) can be used to improve PCR processes to make them economical, predictable, and repeatable. Collaboration engineering is an approach to designing collaborative work practices for high-value recurring tasks that allows practitioners to do the tasks themselves, eliminating the ongoing intervention of professional facilitators (Briggs et al., 2003; Briggs et al., 2006; de Vreede and Briggs, 2005; de Vreede, Kolfschoten et al., 2006). PCR-based tasks, such as UE, are of high value because they are expensive to conduct; they are recurring because they are needed frequently and can be used throughout the software-development cycle (Nielsen, 1993).

In this paper, we investigate ways to improve the popular PCR-based task of heuristic evaluation (HE), which is one form of UE. There are several reasons why we focus on HE. First, HE is an easily-understood UE technique that combines individual and group work in order to quickly evaluate user interfaces based on a series of usability heuristics (Nielsen and Molich, 1990). Second, HE can be conducted by usability experts as well as by members of the target user community (Nielsen and Molich, 1990) who are not UE experts. Nielsen (1992) showed that with limited training, novices can effectively conduct HE and can sometimes identify bugs that are overlooked by usability experts. By discovering usability problems early and quickly, HE can reduce costs and promote the efficient production of usable software (Nielsen and Molich, 1990). Third, Nielsen and Landauer (1993) found that HE is most effective when performed collaboratively in groups. However, HE has yet to fully benefit from collaborative technologies and CE.

Coordination theory suggests that coordinated groups have better outcomes than uncoordinated groups (Malone and Crowston, 1990). We posit that CE can improve PCR-based tasks by creating a process wherein groups can effectively use implicit coordination. Implicit coordination (i.e., common understanding and group memory) can be highly effective in improving group outcomes (Espinosa et al., 2001; Weick and Roberts, 1993) for many tasks; in fact, there is evidence that implicit coordination is the primary means of coordination for most groups (Gersick, 1988). However, implicit coordination requires a shared understanding. This shared understanding can be created through a variety of means including training, discussion, software tool structure, in-process feedback, and general interpersonal interaction. Novices need an efficient means of coordination and collaboration to perform HE, in part because they are novices. Simply put, HE novices need both explicit and implicit coordination. They need explicit training, instructions, process structure, and understandable tools to be able to work toward a common goal. They need collaborative support so that they can see what others are doing and avoid wasted, duplicate effort.

Our research focuses on improving the coordination of HE groups by using CSW. The CSW literature provides examples of coordination benefits related to software code inspections (de Vreede, Koneri et al., 2006; Genucthen et al., 1998; Rodgers et al., 2004; Tyran and George, 2002; van Genuchten et al., 2001) and software defect inspections (Grünbacher et al., 2003). (Though HE is similar to an inspections task, it has some important differences, as explained in the next section). Also, though several studies have investigated ways to better implement HE in order to improve software engineering and usability (Agarwal and Venkatesh, 2002; Baker et al., 2001; Garzotto et al., 1995; Levi and Conrad, 1996; Lowry and Roberts, 2003; Muller et al., 1998; Sears, 1997; Sutcliffe, 2001), very little research to date has focused on improving HE for novices through increased implicit coordination from CSW or through CE principles. This raises the question of whether implicit coordination provided by CSW could benefit novices performing HE by helping to make such groups self-sustaining and by providing and facilitating successful CE. Additionally, because potential participants in PCR-based tasks are often distributed, we examine whether implicit coordination can make distributed and larger HE groups more effective than HE groups that do not have implicit coordination. To investigate these research questions, we compare HE outcomes with novice groups in six different treatments in an HE experiment: small- and medium-sized face-to-face (FtF) non-CSW groups; small- and medium-sized FtF CSW groups; and small- and medium-sized distributed CSW groups.

This paper is organized as follows: First, we outline the basic HE processes and how they relate to the traditional PCR process for eliminating defects (IEEE, 1989). Next, using CE principles, we advance hypotheses on these conditions and describe how implicit coordination provided by CSW can improve HE. Next, we describe the treatments and the results. Finally, we discuss limitations of this study and opportunities for future research.

2. Usability Evaluation and Heuristic Evaluation

2.1. Usability Evaluation

Software engineers and researchers are continually striving to improve the software engineering process to make software not only more cost effective and efficient, but also more usable and appropriate to user requirements. Traditionally, this effort has centered on improving software usability through UE. The objective of UE is to identify and eliminate usability flaws manifest in an application's interface(s) as early as possible in the software engineering process. Usability flaws that are discovered after software is released can be costly to resolve and detrimental to customer satisfaction, customer trust, and future sales. Thus, an effective UE method must find the major usability flaws in an application before it is released, and must be efficient, fast, and easily understood. Effective UE methods are also important because software engineers typically face backlogs of work, yet are expected to produce high-quality software at a rapid pace.

Various academic and applied publication outlets advocate a wide variety of UE approaches. Some of the more widely-adopted UE approaches include cognitive walkthroughs, formal usability inspection, pluralistic walkthroughs, published guidelines, and HE. Although the majority of extant approaches share a common goal, they vary widely in regard to broad-based applicability, cost effectiveness, and ease of use (Vredenburg and Butler, 1996). Of the 14 major UE approaches in practice, HE has emerged as the most widely-adopted approach because of its relative ease of application, relatively low cost, short learning curve, applicability early in the software engineering process, and ability to generate effective UEs without the need for professional evaluators (Vredenburg et al., 2002). A further benefit is that HE is generally more effective in finding usability violations than are many other common UE approaches (Jeffries et al., 1991; Nielsen and Molich, 1990; Shaw, 1993). Table 1 summarizes the major UE methods.

Table 1. Comparison of Usability Evaluation Methods

Approach	Pros	Cons
Cognitive walk-through	• Helps designers understand the end user	• Needs a task-definition methodology • Tedious, time consuming, and costly • Misses general and recurring problems
Formal usability inspection	• Identifies the most critical problems	• Requires user interface expertise • Very time consuming and costly • Misses consistency problems
Pluralistic walk-through	• Helps developers understand the end users' needs • Involves multiple viewpoints	• Needs someone with interface expertise • Very time consuming • Tends to be limited to specific scenarios • Misses many bugs
Published guidelines	• Identifies recurring and general problems • Low cost	• Misses some severe problems • Unstructured • Inconsistent • Too much reliance on developers • No formal accountability
Heuristic evaluation (HE)	• Identifies the most problems of any approach • Identifies the most critical problems • Low cost • Takes little time • Can be used by non-experts • Easy to learn • Informal	• Requires several evaluators • Often results in duplicate bug reports • Can find trivial bugs

2.2. Heuristic Evaluation

HE involves a limited set of usability heuristics that novices can be trained to use in order to identify usability violations. HE is conducted quickly by having participants evaluate software interfaces for their compliance with established usability heuristics. It does not aim to find every possible bug; instead, HE aims to quickly find as many important violations as possible. The heuristics used in conducting HE were originally developed to improve the effectiveness of complex software evaluation. These heuristics have since been refined, based on a factor analysis of 249 usability problems to derive a set of heuristics with maximum explanatory power, resulting in a set of 10 heuristics (Nielsen, 1994). The refined set of heuristics is both widely accepted and successfully employed by the HE community.

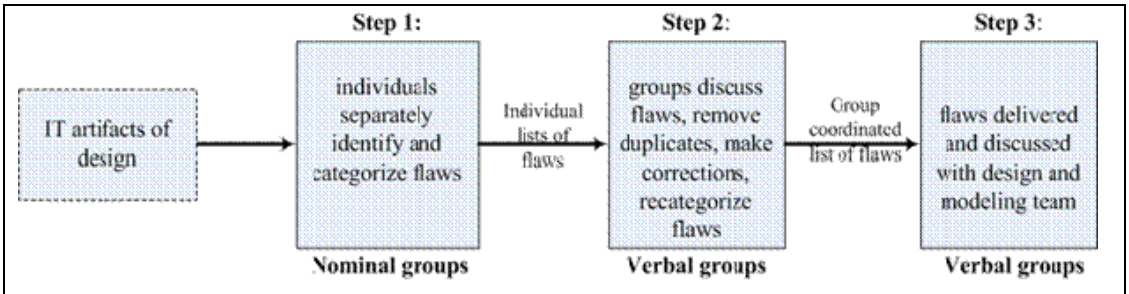

Figure 1. The Major Steps of PCR-Based Tasks on which HE Builds

The traditional HE process involves three steps that build on concepts from PCR, an iterative, three-step inspection procedure that many organizations use to eliminate defects (IEEE, 1989). Because HE builds on PCR, it is likely that the theory and hypotheses we are building for HE may generalize to other PCR-based tasks. Figure 1 gives an overview of these general steps.

HE is traditionally performed in three steps. In Step 1, though this is not explicitly required, group members work independently in nominal groups[1] (without talking to each other or seeing each other's work), with each team member individually evaluating the software interface(s) for violations of the heuristics (Nielsen and Molich, 1990). Individuals working independently use non-CSW tools such as spreadsheets, word processors, or paper and pencil to record heuristic violations (Nielsen and Molich, 1990). Recent studies with heuristic evaluation continue to employ the nominal group technique even in computer-mediated environments (Hvannberg et al., 2007; Tang et al., 2006). In Step 2, team members meet FtF to compare and discuss their evaluation results, remove duplicate bugs and false positives (FPs), and make a combined bug list to deliver to the development team in Step 3. Because this last step involves handing over and discussing the report, we do not consider it for our theoretical, empirical purposes.

It is important to note that HE differs in significant ways from other tasks that have been supported by CSW. For example, Step 1 of HE is not the same as a brainstorming task. In brainstorming, participants try to come up with as many original ideas as possible. Ideally, this is done with a free flow of uninhibited ideas that build on each other, which is why an anonymous process, which decreases inhibitions, improves brainstorming (Gallupe et al., 1992). In contrast, Step 1 of HE uses a search-and-compare task in which HE inspectors search for defects by trying to find examples of defects; in other words, they match features in the interface to patterns of known heuristics, a process that is similar to seeking solutions with analogies (e.g., Hender et al., 2002). These patterns are purposely limited in HE; traditionally, a list of only 10 heuristics is used.

HE inspections also differ from software inspections because HE inspectors are novices, whereas software inspectors are experts. Software inspectors are experts in two ways: first, they are often highly-trained software engineers who know specifically what they are looking for (e.g., many different kinds of specific software defects) (Porter et al., 1997; Tyran and George, 2002). Second, they are also often experts in the inspection process itself because they use it frequently. Novices tend to perform poorly because they lack both types of expertise. These differences make both training and utilizing an efficient means of conducting HE essential for effective outcomes.

2.3. Heuristic Evaluation in the Context of Collaboration Engineering

CE researchers have classified several general patterns of collaboration in order to categorize group activities based on the changes-of-state they produce. These general patterns include the following (Briggs et al., 2003; Briggs et al., 2006): *generate* (move from having fewer to having more concepts), *reduce* (move from having many concepts to a focus on fewer concepts), *clarify* (move from having less to having greater shared understanding of concepts and of the words and phrases used to express them), *organize* (move from having less to having greater understanding of the relationships among concepts), *evaluate* (move from having less to having greater understanding of the relative value of concepts), and *build consensus* (move from having fewer to having more group members who are willing to commit to a proposal).
Applied to HE, Step 1 is a generating pattern because participants are trying to find and document as many violations as possible. Although participants categorize their bugs in Step 1, this is not considered a reduction because the group has not collaborated via a group-level reduction effort. Thus, Step 2 involves organizing, reducing, and building consensus because groups organize bugs, remove duplicates and FPs, and seek agreement on these decisions.

The change we seek to make in the HE task is to improve Step 1 through implicit coordination so that participants not only identify bugs in Step 1, but also start the process of categorizing, reducing, and

[1] Collaboration and communication literatures have adopted a long-standing classification scheme that refers to any group working independently and separately without verbal communication as a *nominal group* (Taylor et al., 1958). Those who communicate are referred to as *verbal* or *interacting* groups.

even building consensus. We believe this improvement will increase productivity by eliminating wasteful efforts in Step 1, which will subsequently reduce the amount of effort required in Step 2. Our proposed change is contrary to extant practice with HE, which still uses nominal groups in Step 1 (Hvannberg et al., 2007; Tang et al., 2006).

2.4. HE and Group-Size Limitations

It is important to test the impact of group size and CSW on HE groups to determine whether implicit coordination can make HE groups more effective and self-sustaining. Nielsen and Landauer (1993) found that the optimal size of FtF non-CSW groups performing HE is between three and five people; in larger groups, diminishing marginal returns occurred because of duplicated effort. Diminishing marginal returns with group-size increases were also found in software-code inspections with nominal groups (Biffl and Halling, 2003). An HE process supported by CSW may allow larger groups to participate while remaining productive, as has been observed in brainstorming (Gallupe et al., 1992) and process modeling (Dean et al., 2000). Average individual productivity may also remain higher with CSW because of increased implicit coordination, as explained further in the theory section. Yet the tasks and processes of PCR activities, including HE, are appreciably different from brainstorming or collaborative modeling, which implies that the study of PCR tasks warrants separate investigation.

3. Theory and Hypotheses

3.1. Group-Level Productivity Constructs and Measures

Before proposing group-level theory and hypotheses, we will define the key constructs used in this section. False positives (*FPs*) are reported violations that turn out not to be legitimate violations (Hertzum and Jacobsen, 2001). *Duplicates* are violations that are reported more than once (Lowry and Roberts, 2003), which are a key surrogate for lack of coordination. *Usable violations* are the net legitimate violations after eliminating FPs and duplicates (Hertzum and Jacobsen, 2001; Nielsen and Molich, 1990). *Total violations* are all violations reported: the sum of usable violations, FPs, and duplicates (Hertzum and Jacobsen, 2001). *Changes* represent the additional work conducted by inspectors in Step 2 to reassign bugs to more appropriate violation categories and to decrease FPs and duplicates. These measures and their application to each step of HE are summarized in Table 2.

Table 2. Summary of Group-Level Productivity Constructs of Interest

Construct	Definition and measurement	Steps used
Total Violations	Usable violations + unusable violations (FPs and duplicates)	1 and 2
Usable Violations	Net legitimate violations after eliminating FPs and duplicates	1 and 2
False Positives (FPs)	Reported violations that are not legitimate violations	1 and 2
Duplicates	Violations that are reported more than once (key surrogate for lack of coordination)	1 and 2
Changes	Additional productive work between Step 1 and Step 2 (Increase in usable violations between Steps 1 and 2) + (Decrease in FPs and duplicates between Steps 1 and 2)	2

Because each HE step involves different processes and group interactions, we explain and predict outcomes for each step. The primary comparison in Step 1 pertains to whether there is a benefit to using collaborative support in nominal groups vs. using traditional nominal groups. In Step 2, the nominal groups become fully-interacting verbal groups and distributed interactive groups; thus, the primary comparison is between the levels of social presence.

3.2. The Effects of Implicit Coordination on Productivity in Step 1 and Step 2

Our theory challenges the conventional practice of using non-CSW-supported nominal groups in Step 1 of PCR-based tasks. We challenge these practices based on the belief that the group process losses (e.g., production blocking, evaluation apprehension, and domination) that Step 1 of HE is designed to avoid can likely be decreased by improved implicit coordination resulting from the use of CSW. Malone and Crowston define *coordination* as "managing dependencies between activities" (1994, p. 90). Coordination is necessary when interdependence exists in performing tasks and activities (Malone and Crowston, 1994). Because interdependence exists in the steps of PCR tasks, we believe coordination may be useful in Step 1. This raises the question of how coordination can be achieved while limiting process losses. We describe how this balance can be accomplished by explaining and extending coordination theory (Malone and Crowston, 1994).

The basic premise of coordination theory is that an appropriate level of coordination between interdependent actors allows them to work toward a common task more effectively. Task dependencies create coordination problems for actors working on the same tasks; coordination mechanisms are steps that actors must perform to overcome coordination problems caused by these task dependencies (Crowston and Kammerer, 1998).

In groups, coordination can be achieved by the use of explicit and implicit mechanisms to manage task dependencies effectively (Espinosa et al., 2001). *Explicit coordination* is an overt attempt to coordinate through formal task organization and group communication (Espinosa et al., 2001; Van de Ven et al., 1976). *Implicit coordination* is tacit (unspoken and understood) coordination that occurs with increased familiarity with a task and a group, resulting in group knowledge (Espinosa et al., 2001; Weick and Roberts, 1993). Implicit coordination is the primary means of coordination for most groups (Gersick, 1988), and it has been shown to produce better results in software engineering than explicit coordination because it decreases the overhead and cost imposed by explicit coordination (Crowston and Kammerer, 1998; Espinosa et al., 2001). Given certain conditions, collectives will exhibit implicit coordination that can be described as "collectively intelligent": they perform in the best interest of the group (Weick and Roberts, 1993). Such implicit coordination has been shown to aid the development of software requirements (Crowston and Kammerer, 1998).

This literature on coordination causes us to raise the question of how explicit and implicit coordination can be used to help HE groups with CSW. We believe the answer comes from extending coordination theory to the HE context. For both CSW-supported and non-supported HE, a number of explicit coordination mechanisms can be employed before HE so that implicit coordination can make the collaboration more productive. Because high levels of implicit coordination require using shared mental models or shared cognition (Espinosa et al., 2001; Weick and Roberts, 1993), participants can be trained in HE, which would provide them with pre-task instructions that define the goal and the approach for achieving the goal. In addition, the proper configuration of CSW can also help facilitate implicit coordination (Crowston and Kammerer, 1998) by providing a structure and means by which the work can be accomplished in a way that supports these mental models and shared cognition. In the case of HE, CSW can reinforce shared mental models if the tool is configured to show HE violation categories and is designed to allow users to assign violations to these respective categories.

CSW provides communication capabilities that further support implicit coordination. Two key CSW software features that can be used to create shared cognition in CSW are *group memory* (Tyran and George, 2002; Wegner et al., 1991) and *group awareness* (Lowry and Nunamaker Jr., 2003). *Group memory* exists when the knowledge of a group is shared (Dennis and Garfield, 2003; Wegner et al., 1991). Group memory performance is superior to that of individual memory across a variety of conditions (Hartwick et al., 1982), especially when combined member resources are needed (Hinsz, 1990). Such combined resources are needed in tasks such as usability testing, requirements-gathering evaluation, and software inspections (Sauer et al., 2000; Yin and Miller, 2004). Group memory is fostered through CSW when all individual contributions can be seen by all group members through a shared interface (Nunamaker Jr. et al., 1991; Satzinger et al., 1999).

Implementing formalized group memory via a shared CSW interface also fosters group awareness. *Group awareness* is the ability to know what other group members are doing at a given time without

direct communication; this implicitly increases social pressure on group members to contribute more, coordinate work, and avoid duplicate work (Lowry and Nunamaker Jr., 2003). Trying to accomplish the same result with direct communication stops others from working or at least interrupts their work.

Two additional features of CSW help improve the development of group memory in a shared interface: (1) self-scribing ability and (2) parallelism. *Self-scribing ability* allows each individual to type comments directly into group memory (Rodgers et al., 2004). Self-scribed formalized group memories can be used to document work sessions (Nunamaker Jr. et al., 1991) so that information is not overlooked (Harari and Graham, 1975; Maier, 1970). *Parallelism* is the ability of group members to contribute information simultaneously (Dennis et al., 2001). In traditional FtF groups, *production blocking* is a major cause of poor group performance because while one person speaks, others must wait (Dennis, 1996b). Parallelism should also result in reduced cognitive interference because participants do not have to wait to contribute their ideas (Dennis and Valacich, 1993).

Traditional PCR tasks use nominal groups in Step 1 that do not build a formalized group memory until Step 2, at which time they combine their individual results from Step 1. The problem is that this group memory cannot be shared, accessed, or coordinated in Step 1; thus, it is only possible for group members to participate in the generating activity. This lack of coordination decreases the ability of such groups to avoid duplication of effort and FPs and delays the spread of knowledge throughout the group until Step 2. Individuals in traditional nominal groups in Step 1 have no ability to interrelate; thus, they may "act heedfully, but not with respect to others" (Weick and Roberts, 1993, p. 371). This lack of group memory and group awareness (resulting in no shared cognition or coordination) is depicted in Figure 2.

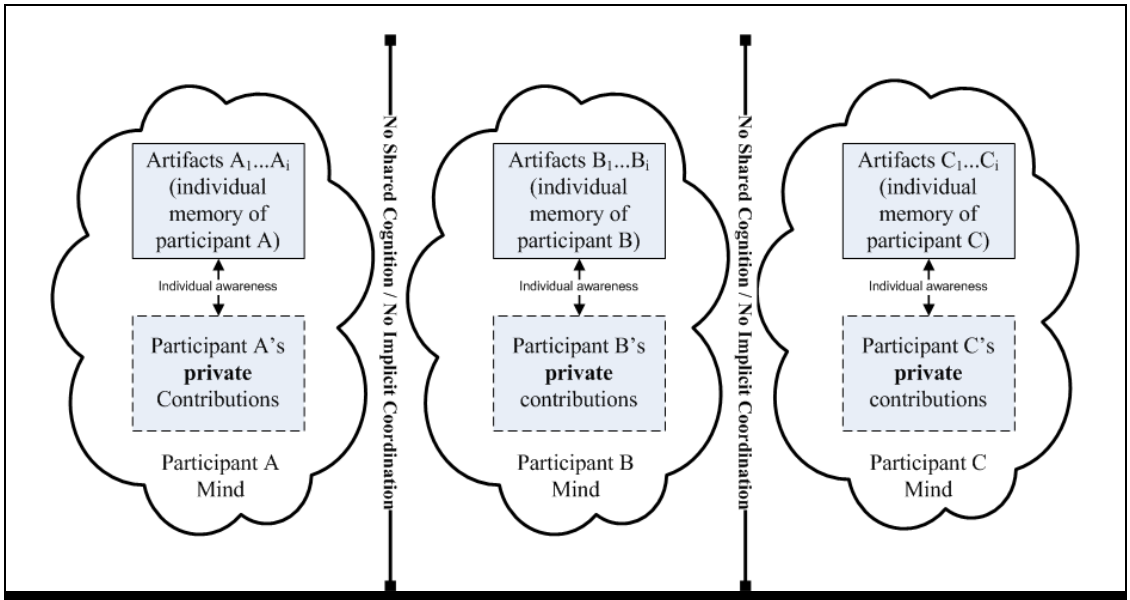

Figure 2. Lack of Shared Cognition and Implicit Coordination

In contrast, nominal CSW-supported groups in Step 1 will have a shared interface that fosters group memory, which improves implicit coordination. The information that individual group members create can be pooled synergistically to form group memory. Nominal CSW-supported participants (whether proximate or distributed) will be able to see the bugs other participants are reporting and implicitly coordinate to avoid duplicate work and FPs in Step 1—even though no mention of avoiding duplicates or FPs is directed in Step 1. Similar benefits have been suggested in comparable analogy tasks (Hender et al., 2002) and in software code inspections (Tyran and George, 2002). Thus, by introducing implicit coordination into Step 1, groups now generate, categorize, reduce bugs, and even start to build consensus from the outset of a task rather than waiting until Step 2. Figure 3 depicts this development of group memory and group awareness, which fosters shared cognition and subsequent implicit coordination.

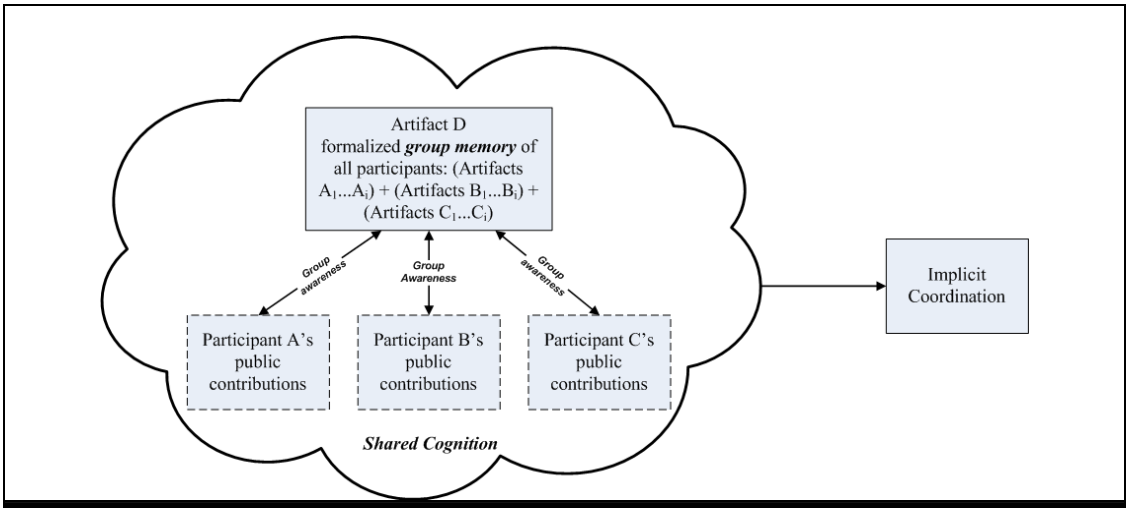

Figure 3. Shared Cognition and Implicit Coordination in CSW-Supported Groups

In the traditional unsupported process, the focus and objectives of PCR tasks change dramatically from Step 1 to Step 2. In Step 1, group members focus on finding bugs without direct discussion among group members. The focus in Step 2 is not on finding bugs but rather on cleaning up the list of total bugs derived from the independently-generated lists. In Step 2 of the traditional unsupported process, group members no longer operate as nominal groups; they fully interact and directly communicate with each other verbally to finalize their list of bugs.

Because HE is an integrative task that involves two key steps, predictions about Step 2 must take into account the work that took place in Step 1. Groups that lack computer support in Step 1 should have more residual problems (i.e., duplicates and FPs) that they must work through in Step 2 than groups that had computer support in Step 1. This means that traditional FtF groups starting Step 2 will have more work to accomplish due to a lack of implicit coordination in Step 1. These factors should result in non-CSW groups being more likely to make more additions, deletions, and changes in reported bugs, duplicates, and FPs than CSW-supported groups. In summary, we derive the following hypotheses relating to production and productivity:

H1. FtF unsupported groups will produce more total violations than supported groups do in Step 1 *(a)* and Step 2 *(b)*.

H2. FtF unsupported groups will produce more duplicates and false positives than supported groups do in Step 1 *(a)* and Step 2 *(b)*.

H3. FtF unsupported groups will produce fewer usable violations than supported groups do in Step 1 *(a)* and Step 2 *(b)*.

H4. Groups that lack CSW support in Step 1 will make more changes in Step 2 than CSW-supported groups.

3.3. The Effects of Group Size on Productivity in Step 1 and Step 2

The influence of group size on HE production and productivity has received little research attention, yet it has important implications. If larger groups find considerably more bugs, increased group size may be warranted. However, if adding more inspectors produces diminishing returns, then smaller groups would be more efficient. As group size increases, more people will inspect the interfaces, so there should be an increase in total violations detected, including duplicates and FPs, in both nominal and CSW groups.

Group process losses tend to increase with group size, especially in larger traditional FtF groups (Gallupe et al., 1992; Valacich et al., 1995). Furthermore, larger groups tend to inhibit individual

participation and create more communication difficulties than smaller groups (Steiner, 1972). These problems are generally caused by human limitations in communication bandwidth and attention; empirical research shows that simultaneous cognitive activities interfere with one another (Ball and Zuckerman, 1992). Thus, as group size increases, it becomes increasingly difficult to communicate, to pay attention to each group member, and to hear each group member—all of which contribute to increased group process losses. Researchers have asserted that because of these factors, maximum effective group size in unsupported groups is no more than five to six members for most group tasks (Hackman and Vidmar, 1970). As noted, however, for the unique task of HE with nominal groups, the maximum recommended size has been three to five members (Nielsen and Landauer, 1993).

Focus theory posits that the way group attention is allocated during collaborative work directly affects group productivity (Briggs and Nunamaker Jr., 1999). During collaborative teamwork, an individual's attention is divided across three activities: communication, deliberation, and information access. At any one time, an individual attends to only one of these three activities. To obtain the benefit of teamwork and to utilize the abilities and contributions of different individuals, groups must communicate and have access to information. Time spent communicating and accessing information reduces time spent toward the desired outcome, which in the context of HE means less time searching for additional bugs. During HE, reviewers must access both the interfaces and the defects found by others (information access). Inspectors also communicate by sharing problems found and discussing whether the problems are bugs. Although larger groups may find more bugs, increasing group size creates different challenges for nominal and CSW groups (aside from those mentioned above), but produces similar results.

Size and Nominal Groups. In nominal groups, inefficiency will occur because evaluators lack a convenient mechanism for sharing found defects with other evaluators who are inspecting the same application interfaces. This is true regardless of whether people are working synchronously or asynchronously. Consequently, an evaluator may waste time finding and recording defects that have already been found and recorded by other inspectors (Rodgers et al., 2004). Such inefficiency may be a problem, especially for nominal groups, because, as some research on inspection processes has shown, inspectors find a considerable degree of duplication (Myers, 1978). In effect, there will most certainly be considerable duplicates and FPs that inflate the overall total violations. This will greatly affect Step 2.

Size and CSW Groups. Although group memory and group awareness can help evaluators in CSW groups avoid wasting effort searching for bugs that others have already found, this benefit comes at a cost to attention. Consuming information via group memory or group awareness requires concentration that otherwise might be directed toward searching for bugs. As group size increases, more people can contribute more bugs; however, this increased volume of information may overwhelm a user's willingness or ability to keep track of the shared information. The process of reading and integrating information typed into the CSW into a person's memory may overload that person's limited cognitive resources (Dennis, 1996a). In summary, in CSW groups, total violations, duplicates, and FPs should increase with group size, although as stated in the previous hypotheses, CSW groups will still retain significant benefits over traditional nominal groups. Therefore,

H5. FtF supported, FtF unsupported, and distributed groups of three will have *(a)* fewer total violations, *(b)* fewer usable violations, *(c)* fewer FPs, *(d)* fewer duplicates, and *(e)* fewer changes than will their corresponding groups of six.

4. Methodology

4.1. Experimental Design

The overall experiment used a 2 x 3 factorial design. The independent variable (IV) of group size was small (three members) and medium (six members). The IV of process and tool choice had three levels, as shown in Table 3.

Table 3. Treatments and Experiment Conditions

	Step 1			Step 1 and 2	Step 2	
Treatment	**Proximity**	**Commun-ication**	**Group memory and group awareness**	**Tools used to store violations**	**Meeting type**	**Commun-ication**
$1_{3 \text{ and } 6}$	Same room but not proximate	No oral or textual discussion (anonymous)	No	Word	FtF	Oral only (non-anonymous)
$2_{3 \text{ and } 6}$	Same room but not proximate	No oral or textual discussion (anonymous)	Yes	CSW	FtF	Oral only (non-anonymous)
$3_{3 \text{ and } 6}$	Same room but not proximate	No oral or textual discussion (anonymous)	Yes	CSW	Synchronous distributed	Text messaging only via NetMeeting (non-anonymous)

4.2. Tools

In all three treatments, participants were given the same 10 heuristic categories mapped into a tool so that they could easily type violations into any of the 10 categories. For the unsupported groups, we chose Microsoft Word because it is representative of the non-collaborative tools often used in the traditional HE process. Participants also had sufficient experience with Word to be comfortable using it. We gave each participant a preformatted Word document that contained a table for each of the 10 heuristic violation categories.

In the supported treatments, the names of each of the 10 heuristic categories were listed as a node on a shared, CSW outline tool so that participants could type violations in any of the 10 categories listed on the outline. Participants could also see the violations typed into each category by other participants in their group. The CSW tool was the Collaboratus shared outline tool. Collaboratus is a Web-based collaborative tool (Lowry et al., 2002) that provides group memory, anonymity, self-scribing ability, parallelism, and group awareness (Lowry and Nunamaker Jr., 2003). Collaboratus supports both FtF and Internet-based, distributed group work, allowing effective support for two of the treatments of this experiment. Collaboratus permits experimental control of communication, allowing participants to see the contributions of others, but not allowing direct communication (e.g., notes, discussion boards, annotations, and so forth). During Step 2 for Treatment 3, distributed virtual groups used the textual chat features of Microsoft NetMeeting in addition to Collaboratus.

4.3. Process

Treatment 1 groups performed HE FtF using the traditional process. In Step 1, subjects worked individually, with each subject recording his or her findings in Word. In Step 2, groups used oral discussion and Word to create a combined, single document containing the categorized violations from the individual documents. Using a group to combine the individual findings is a different method from that used in some nominal group exercises, in which combination is carried out by one individual without input from other group members.

In Treatment 2 for Step 1, participants logged their findings anonymously into Collaboratus. This tool enabled group memory, but participants were not allowed to communicate beyond seeing each other's bug postings in Collaboratus. In Step 2, the groups orally discussed bugs face-to-face and made changes in Collaboratus to finalize their combined bug lists.

In Treatment 3 for Step 1, participants used Collaboratus as in Treatment 2 but worked from distributed locations. As in the first CSW treatment, group members were not allowed to communicate beyond seeing each other's bug postings. In Step 2, these distributed groups discussed the bugs using the textual chat features of NetMeeting There was no oral discussion. Changes to the bug list were made using Collaboratus.

4.4. Participants

The 439 participants were students enrolled in a sophomore-level introductory information systems course that was open to all business majors and taught during two sequential semesters over the course of a year at a large Midwestern university. Students participated in the study voluntarily for course credit and were randomly assigned to groups and treatments. Measurement of demographic variables across participants showed no significant differences in the following variables: age, GPA, years of education, years of work, and gender.

Because the experiment was conducted using course laboratory sessions, not all groups were formed with the right size (three or six); thus, data from 417 participants was used, and an imbalanced design resulted. Specifically, we had no problems forming three-person groups, but we did have some difficulty forming groups of six. Furthermore, each laboratory session could be dedicated to only one condition, and not all laboratory session enrollments were of equal size. In summary, 107 groups were used in the following conditions: 32 unsupported groups of three (96 participants); 11 unsupported groups of six (66 participants); 27 FtF CSW-supported groups of three (81 participants); 11 FtF CSW-supported groups of six (66 participants); 16 virtual groups of three (48 participants); and 10 virtual groups of six (60 participants).[2]

4.5. Task and Procedures

We provided an entire class session (one for each of the two major data collections, which took place over the course of two semesters) with consistent training for all 439 participants on how to properly conduct HE. We provided examples and screen shots showing usable and less-usable interfaces, and we explained them in terms of the 10 heuristics. Students were given take-home review sheets with examples of the heuristics to reinforce their training.

Within the next week, students attended out-of-class laboratory sessions during which their assigned conditions were executed. To avoid mixing experimental conditions in the same session, each lab session (20 to 30 students) executed one experimental treatment. Participants evaluated the same series of Internet-based interfaces, which were designed to have many heuristic violations of varying complexity and severity. We designed the interfaces so that the participants could recognize the violations without any business or content expertise. Participants were not asked to complete any functional scenarios with the interfaces. The screens were implemented in functional prototypes so that the participants clicked on hyperlinks and actively explored the interfaces in order to find violations.

Each laboratory session was led by a professor aided by two graduate laboratory assistants, all of whom ran every session. The professor provided a brief introduction to the purpose, rules, and required processes for each session. The participants were provided with brief, scripted training on the tools they were to use for their treatments. After students were trained on Step 1, they were given 30 minutes to complete this step. Students were then trained on Step 2, after which they had 10 minutes to complete this step. To ensure the ability to create implicit coordination among groups, no structure, rules, guidance, or help was provided once a step was being executed.

After the experiment, all data were exported into tables that were individually evaluated by three trained judges. Each reported heuristic violation was evaluated for FPs and duplicates in relation to a predetermined list of all violations in the interfaces. If the judges found that students had reported

[2] When testing the unsupported groups of six, we had two groups with only four members; these were dropped. When testing the CSW-supported groups of six, we had two groups of five and one group of four, which were also dropped.

violations not on the initial list, they discussed these violations and in some cases added them to the master list of violations.

5. Analysis and Results

Key to our theoretical model of the group-level predictions is the effect of interaction between social presence and group size, in which group size moderates the relationship between media richness and group-level productivity constructs. Thus, we first tested for this interaction using MANOVA to correct for multiple comparisons of the interaction effect. This interaction is significant for total Step 1 at $F_{(2,107)} = 12.47$, $p < 0.0001$; changes Step 2 at $F_{(2,107)} = 3.57$, $p = 0.032$; usable violations Step 2 at $F_{(2,107)} = 4.13$, $p = 0.019$; FPs Step 1 at $F_{(2,107)} = 10.27$, $p < 0.0001$; duplicates Step 1 at $F_{(2,107)} = 10.23$, $p < 0.0001$; and changes at $F_{(2,107)} = 12.45$, $p < 0.0001$. The interaction is not significant for usable violations Step 1 at $F_{(2,107)} = 2.79$, $p = 0.066$; FPs Step 2 at $F_{(2,107)} = 3.61$, $p = 0.307$; or duplicates Step 2 at $F_{(2,107)} = 2.61$, $p = 0.078$.

We then used SAS LS Means to examine the statistical difference between the interaction means of the productivity measures. To ensure overall protection, only the probabilities associated with the preplanned comparisons (from our hypotheses) were used. Table A1.1 summarizes the interaction means, and Table A1.2 summarizes the preplanned comparisons with their p-values. Table 4 summarizes the means for Step 1, and Table 5 summarizes the means for Step 2. Table 6 summarizes the results of H1–H4. Finally, Table 7 summarizes the results of H5.

Table 4. Step 1 Means

	Group Totals			Per Participant		
	#1 Nominal FtF	**#2 CSW FtF**	**#3 CSW Virtual**	**#1 Nominal FtF**	**#2 CSW FtF**	**#3 CSW Virtual**
Small Groups						
Total Violations	47.7	39.3	32.7	15.9	13.1	10.9
Duplicates	14.5	7.3	4.3	4.8	2.4	1.4
False Positives (FPs)	10.3	10.2	9.1	3.4	3.4	3.0
Duplicates + FPs	24.8	17.5	13.4	8.2	5.8	4.4
Usable Violations	22.9	21.8	19.3	7.7	7.3	6.5
Medium Groups						
Total Violations	97.6	55.4	69.6	16.3	9.2	11.6
Duplicates	38.0	12.6	17.3	6.3	2.1	2.9
False Positives (FPs)	33.0	19.1	22.4	5.5	3.2	3.7
Duplicates + FPs	71.0	31.7	39.7	11.8	5.3	6.6
Usable Violations	26.6	23.7	29.9	4.5	3.9	5.0

Table 5. Step 2 Means

	Group Totals			Per Participant		
	#1 Nominal FtF	**#2 CSW FtF**	**#3 CSW Virtual**	**#1 Nominal FtF**	**#2 CSW FtF**	**#3 CSW Virtual**
Small Groups						
Changes	35.6	2.4	5.4	11.9	0.8	1.8
Medium Groups						
Changes	146.6	2.7	0.0	24.4	0.5	0.0

Table 6. Summary of Results of Hypotheses H1–H4

Hyp.	Measure	Size	Prediction for H(a) and H(b)	Step 1 outcome, H(a)	H(a) support? (Step 1)	Step 2 outcome, H(b)	H(b) support? (Step 2)
H1(a) and (b)	Total Violations	3	U > (S and D)	U > (S and D)	Yes	U = S = D	No
		6					
H2(a) and (b)	FPs	3	U > (S and D)	U = S = D	No	U = S = D	No
		6		U > S	Partial (U > S)		
H2(a) and (b)	Duplicates	3	U > (S and D)	U > S	Partial (U > S)	U = S = D	No
		6				U > S	Partial (U > S)
H3(a) and (b)	Usable Violations	3	U < (S and D)	U = S = D U = S, S < D	No	U = S = D	No
		6					
H4	Changes	3	U > (S and D)	n/a	n/a	U > S	Partial (U > S)
		6				U > (S and D)	Yes

U = Unsupported, S = Supported, D = Distributed

Table 7. Results of H5 on Interactions

Hyp.	Measure	Prediction (both steps)	Step 1 outcome	H support Step 1?	Step 2 outcome	H support Step 2?
H5(a)	Total violations	$S_3 < S_6$, $U_3 < U_6$, $D_3 < D_6$	$S_3 < S_6$, $U_3 < U_6$, $D_3 < D_6$	Yes	$S_3 < S_6$, $U_3 < U_6$, $D_3 < D_6$	Yes
H5(b)	Usable violations	$S_3 < S_6$, $U_3 < U_6$, $D_3 < D_6$	$D_3 < D_6$	Partial	$D_3 < D_6$	Partial
H5(c)	FPs	$S_3 < S_6$, $U_3 < U_6$, $D_3 < D_6$	$S_3 < S_6$, $U_3 < U_6$, $D_3 < D_6$	Yes	$S_3 < S_6$, $U_3 < U_6$, $D_3 < D_6$	Yes
H5(d)	Duplicates	$S_3 < S_6$, $U_3 < U_6$, $D_3 < D_6$	$U_3 < U_6$, $D_3 < D_6$	Partial	$S_3 < S_6$, $U_3 < U_6$, $D_3 < D_6$	Yes
H5(e)	Changes	$S_3 < S_6$, $U_3 < U_6$, $D_3 < D_6$	n/a	n/a	$U_3 < U_6$	Partial

U_3 = Unsupported size 3, U_6 = Unsupported size 6, S_3 = Supported size 3, S_6 = Supported size 6, D_3 = Distributed size 3, D_6 = Distributed size 6

6. Discussion

6.1. Summary and Discussion of Results

Total violations (H1): While unsupported groups of three and six had more total violations in Step 1 than the corresponding CSW-supported and distributed groups—supporting H1(a)—no differences were found among the groups in Step 2; thus, H1(b) is not supported. We attribute the results in Step 1 to the lack of implicit coordination in unsupported groups; this lack resulted in more duplicates and FPs, which, in turn, increased the total violations. However, in Step 2, the unsupported groups made more changes than the other groups did—decreasing duplicates and FPs and, thus, decreasing the overall totals.

FPs and duplicates (H2) and usable violations (H3): Unsupported groups produced more FPs than did FtF-supported groups in Step 1; however, no other predictions with FPs were confirmed, partially supporting H2(a). Unsupported groups of three and six in Step 1 produced more duplicates than did FtF supported groups, partially supporting H2(a). However, this was not shown in groups of three in Step 2, while it was shown in groups of six, partially supporting H2(b). Finally, no treatment had any particular advantage in producing useful violations, rejecting H3(a) and H3(b).

Changes (H4): Unsupported groups of three had more changes between Step 1 and Step 2 than FtF-supported groups of three, partially supporting H4. For groups of six, unsupported groups had more changes than either of the supported groups (FtF and distributed), fully supporting H4. An important observation here is that the supported groups had to do very little work in Step 2, while the unsupported groups worked frenetically on this step. In fact, the supported groups could have skipped Step 2 and had virtually the same results.

Interactions (H5): The prediction that an interaction effect would exist between social presence and the productivity measures, as positively moderated by group size, was largely supported. In terms of interactions, all groups of three produced fewer violations in Step 1 and fewer changes in Step 2 than corresponding groups of six; this fully supports H5(a). Distributed groups of three produced fewer usable violations than distributed groups of six in both Step 1 and Step 2, partially supporting H5(b). All groups of three produced fewer FPs than corresponding groups of six did, fully supporting H5(c). In Step 1, unsupported and distributed groups of three produced fewer duplicates than corresponding groups of six did, partially supporting H5(d). In Step 2, all groups of three produced fewer duplicates than did corresponding groups of six in either Step 1 or Step 2, fully supporting H5(d). Only unsupported groups produced more changes in groups of six vs. groups of three. These results show a strong positive interaction effect between the treatments and group size for the majority of measures.

To summarize, researchers should not assume that there are no important practical differences between these conditions in terms of FPs, duplicates, and usable violations. In our experiment, unsupported groups were able to make up some of the differences in Step 2, but we observed that such groups had to work much harder. The key issue centers on whether unsupported groups can compete with the efficiency gained by CSW-supported groups, especially if more FPs, duplicates, etc., are produced in Step 1 because of larger group sizes or more problematic interfaces. Furthermore, we question whether using unsupported groups would be sustainable as a repeatable process. It seems that participants would grow weary of working frenetically in Step 2, as opposed to the CSW-supported conditions in which participants do little work in Step 2. The findings with duplicates could be particularly problematic because it appears that FtF supported groups have great advantages over unsupported groups, regardless of size. This could become more problematic as group size increases and it becomes harder and harder to explicitly coordinate in Step 2, especially when work is not coordinated in Step 1.

7. Contributions

An important contribution of this research is to help specify what is required to support implicit coordination. Training, process instructions, software configuration, group memory, and group awareness all contribute to implicit coordination. This research also highlights the important productivity improvements that can be achieved through implicit coordination. These insights can benefit a number of collaborative processes, including, but not limited to, other PCR tasks. The study also showed that novice HE inspectors were able to use implicit coordination when given sufficient training, process, and tools.

One of the interesting productivity findings of this research was that FtF CSW-supported novice groups of three and six produced fewer duplicates in both steps of HE than did unsupported groups (with the exception of Step 2 in small groups). This is an important finding because the number of duplicates was a key surrogate measure of coordination in our experiment. In accordance with coordination theory, increased duplicates in non-CSW-supported groups were most likely caused by the substantial overlap of effort by the group members because they had no opportunity for

coordination in Step 1. Because the unsupported, nominal groups did not see the contributions of other group members until Step 2, they had no opportunity to create shared cognition or implicit coordination until Step 2. Again, these groups focused only on the generating activity.

The results regarding the number of duplicates are particularly noteworthy for CSW-supported groups because they indicate that shared cognition and implicit coordination existed in Step 1, allowing these groups to be further ahead by the time they started Step 2. Specifically, the group members intuitively avoided duplicates in Step 1 without being asked to do so (no groups were told to avoid duplicates in Step 1; the purpose of Step 1 was not to find and remove duplicates but simply to identify bugs). Yet both FtF nominal CSW and distributed nominal CSW groups intuitively avoided more duplicates than non-CSW groups did in Step 1. Because of the strict controls of the experiment, the avoidance of duplicates can best be attributed to the presence of a shared interface that was designed to foster group memory and group awareness. Hence, we believe this is evidence that implicit coordination changed the nature of the activities performed in Step 1 to go beyond generating to include organizing, reducing, and even preliminary steps toward building consensus.

In congruence with coordination theory, with pre-task preparation that created a shared mental model, tool configuration, and collaborative tool capabilities, implicit coordination occurred during Step 1 because no direct communication was allowed between the participants (recall that none of the treatments in Step 1 allowed participants to communicate directly with each other: no text messaging, no notes, no e-mail, no verbal discussion, and so forth). As a result of the implicit coordination fostered by group memory and group awareness, the CSW-supported groups started acting like coordinated teams in Step 1 without being told to. Again, in accord with coordination theory, these results support the notion of the heedful interrelating that is possible through effective coordination (Weick and Roberts, 1993). These groups started to build shared cognition because they were individuals who acted as a team (Weick and Roberts, 1993).

The other key finding is that this lack of shared cognition and implicit coordination in unsupported groups necessitated far greater effort in Step 2 from the unsupported groups, whereas the CSW-supported groups had little to do in Step 2. While the FtF non-CSW groups made significant changes throughout Step 2, their efforts to intentionally remove duplicates in Step 2 did not add as much value as was added by the supported groups who developed tacit agreement earlier in Step 1 in order to avoid duplicates.

This finding may indicate that it is cognitively easier (at least for novices) to avoid duplicates in the first place in Step 1 (provided one has access to group memory) than to try to remove duplicates ex post facto. In removing duplicates after the fact, one has to compare and contrast bugs while also considering removing FPs and engaging in verbal interaction, all of which can slow down the process. The outcomes on duplicates are particularly important because HE is an evaluation technique designed for speed and quality of results. Because of the significantly higher number of duplicates in nominal groups, the trained judges had to spend much more time sorting through the legitimate bugs for the nominal groups (by a factor of several hours). Hence, in practice, traditional nominal HE causes much more follow-up work for design groups, which need to implement the results without duplicates. If an HE group were to hand over its results to a design team and the results were full of duplicates, FPs, miscategorizations, and so forth, the results would place a significant burden on the design team, regardless of the number of correct bugs that were found.

These findings have key theoretical implications far beyond HE. Our findings may extend to all PCR tasks (although this will necessitate further testing). Our findings challenge the conventional wisdom and current practice that advocates the use of unsupported, nominal groups in Step 1 (e.g., Hvannberg et al., 2007; Tang et al., 2006). We provide evidence that this practice might be suboptimal.

Furthermore, by changing Step 1 of PCR, the other steps of PCR may also need to be reexamined in future research. Simply having participants discuss their bugs and remove duplicates and FPs may not be the most effective use of time for groups that experience implicit coordination in Step 1. Because CSW-supported groups are more coordinated and have less to disagree about, simple

discussions in Step 2 may be a suboptimal use of time, as evidenced by the little work performed by the CSW-supported groups in Step 2 of our experiment.

Thus, our findings build a foundation for showing how structured and scripted thinkLets, which are a fundamental part of CE (as further explained in Briggs et al., 2003; de Vreede, Kolfschoten et al., 2006), can be combined with implicit coordination for even more powerful results. While some may be tempted to believe that thinkLets apply only to explicit coordination, this is not the case. If a thinkLet script focuses and invites the group to verbally discuss ideas, it promotes explicit coordination. Such a thinkLet could then be labeled as focusing on explicit coordination. However, if a thinkLet simply instructs participants to perform their tasks in a particular fashion at the beginning of the process (i.e., a form of training or pre-task instruction), then that thinkLet could be said to focus on implicit coordination. In fact, some groups that use a thinkLet-based process over time will need less and less verbal interaction and, hence, have fewer coordination costs as they switch from explicit coordination to implicit coordination. Recognizing these differences and fitting thinkLets to the experience of the participants can help thinkLet designers create thinkLets for specific processes and outcomes with the right mix of implicit and explicit coordination.

For example, in the case of HE, it may be that groups are currently required to engage in too many patterns of collaboration in Step 2 than can be reasonably accomplished without explicit coordination and/or breaking the step into additional steps. Thus, we believe that it could be ideal to follow the implicit coordination from Step 1 with the following scripted substeps that involve key CE patterns of collaboration with thinkLets that are publicly available: (1) Clarify I: The group systematically reads through each bug without discussing removal, but the opportunity is given for group members to explain why they considered a problem to be a bug (a good thinkLet to accomplish this would likely be FastFocus); (2) Clarify II: The group clarifies the wording of the bugs and makes sure all duplicates and redundancies are removed (a good thinkLet to accomplish this would likely be BucketBriefing); (3) Build consensus: Once all the bugs have been reviewed and clarified, the group members discuss the bugs about which they disagree and reevaluate them (a possible thinkLet for this would be Red-Light-Green-Light); (4) Organize and reduce: Once consensus is built as far as what is or is not a bug, the bugs are recategorized and reduced (a good thinkLet for this process would likely be Concentration). The specific thinkLets for each of these patterns of collaboration would need further investigation, and new thinkLets may need to be developed. Furthermore, because these explicit patterns of collaboration would further increase understanding and group memory, it may add value in the case of complex software to repeat Step 1 and Step 2 to try and find more complicated violations.

Another subtle but important contribution of this research is that we show a way to better utilize novice evaluators in PCR-based tasks. This has potential for changing practice because novice evaluators can reduce software-engineering costs. Even more importantly, being able to better use novice evaluators allows software engineers to more effectively involve target users of and stakeholders in the development process of the software they are building. Substantial research has shown that increased user involvement throughout the stages of software engineering not only decreases bugs but also increases the likelihood of software adoption and buy-in, and provides many other political and organizational benefits.

8. Limitations and Future Research

One limitation of our research context is that, for novices, duplicates may be cognitively easier to avoid than FPs because duplicates can be grasped by comparison regardless of one's level of expertise in HE and ability to process analogies. In contrast, submitting original bugs in HE is most similar to an analogies task (e.g., Hender et al., 2002). Determining an object to be a usability violation requires one to remember specific heuristics (similar to analogies), mentally compare each screen element against the list of heuristics, and then make a cognitive judgment as to the degree to which each screen element adheres to each heuristic. The latter is more complex and requires more experience and judgment, especially because multiple screens tend to be used in HE. Thus, novices seeing something reported as a usable bug might naturally avoid submitting duplicates of the same bug. However, it is much more difficult to read a textual description of an error and to apply the

description in submitting additional bugs: The error report is dissociated from the graphical depiction of the screen element that contains the violation (which can be on one of many screens), and such associations likely require higher levels of cognitive processing.

Further, assuming that one is able to learn and develop patterns from a given list of reported bugs, novices are more likely than experts to report incorrect bugs and, thus, are more likely to set incorrect patterns for each other. Thus, any gain from pattern matching is likely to be offset by instances of matching to the wrong pattern. Hence, we expect that there will be very different patterns between groups of novices and experts, and these differences may require different processes and technology support for developed sustainable HE teams. This is another reason why it would be useful to examine whether repeated iterations of Step 1 and Step 2 would be helpful in building group memory and work patterns in order to find more complicated violations.

Furthermore, a key limitation inherent in all laboratory experiments is their lack of generalizability and external validity. However, these inherent drawbacks of experiments are counterbalanced by the benefits of control and establishing theory-based causality. Additionally, we believe the controlled environment was appropriate at this stage of investigation to better understand and predict the effects of CSW across varying sets of conditions with novice participants. Our experiment offered increased process realism, because we executed the two key steps of HE with novice evaluators.

This stream of research could also likely benefit from longitudinal studies, because the nature of PCR and software engineering in general is longitudinal. Though such research would suffer from less control, longitudinal research could have direct applicability to highly complex systems that require many weeks of usability assessment. This could provide more practical insights and rich group measures to help understand the changing group dynamics and communication that likely occur in PCR-based tasks over time.

It is also important to note that our total work measure focused simply on the number of additions and changes in the recorded evaluations. This measure should not be taken as an exact surrogate of the overall workload. To examine workload, we would also need to account for the mental activity going on within each individual, which is best measured through a mental workload measure, such as the NASA task-load index (NASA-TLX). This would be a fascinating addition to future research, because it would likely show a higher mental workload in Step 1 for CSW-supported participants because of the extra mental effort of the implicit coordination in these groups. Meanwhile, the mental workload in Step 2 should be higher in non-supported groups. Finally, if implicit coordination is superior to explicit coordination in this specific context, then the mental workload of CSW-supported groups in Step 1 would be lower than unsupported groups in Step 2.

Another potential limitation is the use of student participants. Use of student participants is appropriate when they fit the task and objectives of a study: participants in studies should have characteristics representing the population of interest and be presented with tasks for which they have the requisite skills and knowledge (Gordon et al., 1986). In this study, student participants clearly represent a subset of the broader population of typical novice end-user evaluators. They also have the skills and knowledge to perform the tasks assigned. Thus, we believe they served well as participants for this study. Of course, this does not remove the need to extend empirical studies to other types of participants, especially expert evaluators.

Another limitation is that the results may have been partially affected by differing levels of expertise with the particular tools used. The control groups used Word, with which all participants indicated significant exposure and experience. However, as the training on and exposure to Collaboratus and NetMeeting lasted only approximately 30 minutes, control participants clearly had more experience with their tool than did the CSW and distributed participants. Despite this difference, the results obtained through the introduction of the collaborative tools suggest that a positive effect can be realized with a limited level of experience with the tool. Therefore, it is conceivable that greater experience and familiarity with Collaboratus and NetMeeting could result in even stronger shared cognition, implicit coordination, and better performance than witnessed in this study.

Further research also needs to explore why there were no differences in usable violations between the CSW and nominal groups. These results may be an artifact of our websites, where perhaps there were too few obvious bugs to find in the limited amount of work time allotted. It is important that future research adjust the amount of time allowed in conjunction with the depth and quality of bugs that are built into the HE websites.

Finally, future research should explore variations in tool choices to further improve our understanding of their effects on HE processes and outcomes. It may prove insightful to use novice groups that have various levels of exposure to the chosen CSW and compare their results to those of control groups. This could inform both the academic and applied communities regarding the effect of tool experience and self-efficacy on the various process steps and outcomes. It would also be useful to explore other forms of CSW to determine whether the presentation of certain key features changes the expected results. Likewise, for synchronous-distributed groups, other forms of communication, such as instant messaging, teleconferencing, and video conferencing, could be explored to determine which technologies provide the necessary richness in communication for effective HE practices.

9. Conclusion

Software development is increasingly complex and costly and involves distributed global teams. The increasingly high expense of traditional usability evaluation methods has often caused such methods to be left out of the software development process—to the detriment of software quality, end users, and ultimately the developing firm. Researchers have developed "discount" usability evaluation methods—typically as PCR-based tasks—to decrease time and cost through simplifying the process and involving less costly non-experts. However, these methods have yet to benefit from the principles of CE.

Using the PCR-based task of HE, we demonstrated that CSW can provide implicit coordination that changes the very nature of Step 1 of the HE task such that groups not only generate bugs, but even start the process of categorizing bugs, reducing bugs (through avoiding duplicates), and building consensus. This change leads to potential improvements in the HE task overall. It also demonstrates the potential of CE to help groups improve and become self sustaining through implicit means, not just explicit means such as process scripts.

Acknowledgments

We would like to thank Dr. Bill Lewis for providing access to his course at the University of Kansas, and Charlie Gruber and Sasi Maganti for their help with the experiment. We thank the University of Kansas School of Business Technology Resource group for helping to prepare the laboratory and the Information Systems Department, and the Kevin and Debra Rollins Center for e-Business at the Marriott School (Brigham Young University) for providing technological resources. We thank Dennis Eggett and Dave Stromberg at the Center for Statistical Consultation and Collaborative Research, Brigham Young University, who helped with portions of the statistical analysis. We also thank those who conducted reviews on earlier drafts: Paul Cheney, Aaron Curtis, Michelle René Lowry, Brian E. Mennecke, R. Kelly Rainer, Denton Romans, Russell Sperry, Steven Tedjamulia, James J. Andersen, Mark J. Keith, Daniel E. King, Isaac Brent Lee, Mel Thorne, Janelle Higbee, Marvin Gardner, Paul Rawlins, Sarah Cutler, Allan Bond, Ben Mitchell, Jason Malwitz, Jeremy Knudsen, Stephen Todd, Laura Rawlins, Jeffrey L. Jenkins, the anonymous reviewers from the SIG HCI workshop, the BYU Information Systems faculty who participated in a Fall 2002 workshop that reviewed portions of this paper, and the participants of the 2007 HICSS workshop on Collaboration Engineering.

References

Agarwal, R. and Venkatesh, V. (2002), "Assessing a Firm's Web Presence: A Heuristic Evaluation Procedure for the Measurement of Usability," *Information Systems Research* 13(2), pp. 168-186.

Baker, K., Greenberg, S., and Gutwin, C. (2001), "Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration," *Lecture Notes in Computer Science; Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction (*2254) pp. 123-140.

Ball, S. A. and Zuckerman, M. (1992), "Sensation Seeking and Selective Attention: Focused and Divided attention on a Dichotic Listening Task," *Journal of Personality and Social Psychology* 63(5), pp. 825-831.

Biffl, S. and Halling, M. (2003), "Investigating the Defect Detection Effectiveness and Cost Benefit of Nominal Inspection Teams," *IEEE Transactions on Software Engineering* 29(5), pp. 385-397.

Briggs, R. O., de Vreede, G. J., and Nunamaker Jr., J. F. (2003), "Collaboration Engineering with ThinkLets to Pursue Sustained Success with Group Support Systems," *Journal of Management Information Systems* 19(4), pp. 31-63.

Briggs, R. O., Kolfschoten, G. L., de Vreede, G. J., and Dean, D. L. (2006), "*Defining Key Concepts for Collaboration Engineering,"* Paper presented at the Americas Conference on Information Systems, Acapulco, Mexico, pp.

Briggs, R. O. and Nunamaker Jr., J. F. (1999). *Focus Theory of Team Productivity and its Application to Development and Testing of Group Support Systems*. Tucson, Arizona, USA: University of Arizona.

Crowston, K. and Kammerer, E. E. (1998), "Coordination and Collective Mind in Software Requirements Development," *IBM Systems Journal* 37(2), pp. 227-245.

Cusumano, M. A. (2004), "Who is Liable for Bugs and Security Flaws in Software?," *Communications of the ACM* 47(3), pp. 25-27.

de Vreede, G. J. and Briggs, R. O. (2005), "*Collaboration engineering: Designing repeatable processes for high-value collaborative tasks,"* Paper presented at the 38th Annual Hawaii International Conference on System Science, Kona, Hawaii, USA, pp. 1-10.

de Vreede, G. J., Kolfschoten, G. L., and Briggs, R. O. (2006), "Thinklets: A Collaboration Engineering Pattern Language," *International Journal of Computer Applications in Technology* 25(2/3), pp. 140-153.

de Vreede, G. J., Koneri, P. G., Dean, D. L., Fruhling, A. L., and Wolcott, P. (2006), "Collaborative Software Code Inspection: The Design and Evaluation of a Repeatable Collaboration Process in the Field," *International Journal of Cooperative Information Systems* 15(2), pp. 205-228.

Dean, D. L., Orwig, R. E., and Vogel, D. R. (2000), "Facilitation Methods for Collaborative Modeling Tools," *Group Decision and Negotiation* 9(2), pp. 109-128.

Dennis, A. R. (1996a), "Information Exchange and Use in Small Group Decision Making," *Small Group Research* 27(4), pp. 532-550.

Dennis, A. R. (1996b), "Information Processing in Group Decision Making: You Can Lead a Group to Information, but You Can't Make It Think," *MIS Quarterly* 20(4), pp. 433-457.

Dennis, A. R. and Garfield, M. (2003), "The Adoption and Use of GSS in Project Teams: Toward More participative Processes and Outcomes," *MIS Quarterly* 27(2), pp. 289-323.

Dennis, A. R. and Valacich, J. (1993), "Computer Brainstorms: More Heads Are Better Than One," *Journal of Applied Psychology* 78(4), pp. 531-537.

Dennis, A. R., Wixom, B., and Vandenberg, R. (2001), "Understanding Fit and Appropriation Effects in Group Support Systems via Meta-Analysis," *MIS Quarterly* 25(2), pp. 167-193.

Espinosa, A., Kraut, R., Lerch, J., Slaughter, S., Herbsleb, J., and Mockus, A. (2001, December 16-19), "*Shared Mental Models and Coordination in Large-Scale, Distributed Software Development,"* Paper presented at the Twenty-Second International Conference on Information Systems (ICIS'2001), Atlanta, Georgia, USA, pp. 513-517.

Gallupe, R. B., Dennis, A. R., Cooper, W. H., Valacich, J. S., Bastianutti, L. M., and Nunamaker Jr., J. F. (1992), "Electronic Brainstorming and Group Size," *Academy of Management Journal* 35(2), pp. 350-369.

Garzotto, F., Mainetti, L., and Paolini, P. (1995), "Hypermedia Design, Analysis, and Evaluation Issues," *Communications of the ACM* 38(8), pp. 74-86.

Genucthen, M. v., Cornelissin, W., and Dijk, C. v. (1998), "Supporting Inspections with an Electronic Meeting System," *Journal of Management Information Systems* 14(3), pp. 165-178.

Gersick, C. J. (1988), "Time and Transition in Work Teams: Toward a New Model of Group Development," *Academy of Management Journal* 31(1), pp. 9-41.

Gordon, M. E., Slade, L. A., and Schmitt, N. W. (1986), "The 'Science of the Sophomore' Revisited: From Conjecture to Empiricism," *Academy of Management Review* 11(1), pp. 191-207.

Grünbacher, P., Halling, M., and Biffl, S. (2003), "*An Empirical Study on Groupware Support for Software Inspection Meetings,"* Paper presented at the 18th IEEE International Conference on Automated Software Engineering (ASE'03), Montreal, Canada, pp. 4-11.

Hackman, J. R. and Vidmar, N. (1970), "Effects of Size and Task Type of Group Performance and Member Reactions," *Sociometry* 33(1), pp. 37-54.

Harari, O. and Graham, W. (1975), "Tasks and Task Consequences as Factors in Individual and Group Brainstorming," *Journal of Social Psychology* 95(pp. 61-65.

Hartwick, J., Sheppard, B. H., and Davis, J. H. (1982), "Group Remembering: Research and Implications," In R. A. Guzzo (Ed.), *Improving Group Decision Making in Organizations* (pp. 41-72). San Diego, California, USA: Academic Press.

Hender, J. M., Dean, D. L., Rodgers, T. L., and Nunamaker Jr., J. F. (2002), "An Examination of the Impact of Stimuli Type and GSS Structure on Creativity: Brainstorming Versus Non-Brainstorming Techniques in a GSS Environment," *Journal of Management Information Systems* 18(4), pp. 59-85.

Hertzum, M. and Jacobsen, N. E. (2001), "The Evaluator Effect: A Chilling Fact About Usability Evaluation Methods," *International Journal of Human-Computer Interaction* 13(4), pp. 421-443.

Hinsz, V. (1990), "Cognitive and Consensus Processes in Group Recognition Memory Performance," *Journal of Personality and Social Psychology* 59(4), pp. 705-718.

Hvannberg, E. T., Law, E. L., and Lárusdóttir, M. K. (2007), "Heuristic Evaluation: Comparing Ways of Finding and Reporting Usability Problems," *Interacting with Computers* 19(2), pp. 225-240.

IEEE. (1989). *IEEE Standard for Software Reviews and Audits* (No. IEEE Standard 1028-1988(R1993)): Software Engineering Technical Committee of the IEEE Computer Society.

Jeffries, R., Miller, J. R., Wharton, C., and Uyeda, K. M. (1991, April 27-May 2), "*User Interface Evaluation in the Real World: A Comparison of Four Techniques,"* Paper presented at the SIGCHI Conference on Human factors in Computing Systems: Reaching through technology (CHI), New Orleans, Louisiana, USA, pp. 119-124.

Levi, M. D. and Conrad, F. G. (1996), "A Heuristic Evaluation of a World Wide Web Prototype," *Interactions* 3(4), pp. 50-61.

Lowry, P. B., Albrecht, C. C., Nunamaker Jr., J. F., and Lee, J. D. (2002), "Evolutionary Development and Research on Internet-Based Collaborative Writing Tools and Processes to Enhance eWriting in an eGovernment Setting," *Decision Support Systems* 34(3), pp. 229-252.

Lowry, P. B. and Nunamaker Jr., J. F. (2003), "Using Internet-Based, Distributed Collaborative Writing Tools to Improve Coordination and Group Awareness in Writing Teams," *IEEE Transactions on Professional Communication* 46(4), pp. 277-297.

Lowry, P. B. and Roberts, T. L. (2003, August 4-5), "*Improving the Usability Evaluation Technique, Heuristic Evaluation, Through the Use of Collaborative Software,"* Paper presented at the 9th Annual Americas Conference on Information Systems (AMCIS), Tampa, Florida, USA, pp. 2203-2211.

Maier, N. (1970). *Problem Solving and Creativity*. Pacific Grove, California, USA: Brooks/Cole.

Malone, T. and Crowston, K. (1990, October 7-10), "*What is Coordination Theory and How Can it Help Design Cooperative Systems?,"* Paper presented at the 1990 ACM Conference on Computer-supported Cooperative Work '90, Los Angeles, California, USA, pp. 357-370.

Malone, T. and Crowston, K. (1994), "The Interdisciplinary Study of Coordination," *ACM Computing Surveys* 26(1), pp. 87-119.

Mayhew, D. J. (1999). *The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design* (1st ed.). San Francisco, California, USA: Morgan Kaufmann Publishers.

Muller, M. J., Matheson, L., Page, C., and Gallup, R. (1998), "Methods and Tools: Participatory Heuristic Evaluation," *Interactions* 5(5), pp. 13-18.

Myers, G. J. (1978), "A Controlled Experiment in Program Testing and Code Walkthroughs/Inspections," *Communications of the ACM* 21(9), pp. 760-768.

Nielsen, J. (1992, May 3-7), "*Finding Usability Problems through Heuristic Evaluation,"* Paper presented at the SIGCHI Conference on Human Factors in Computing Systems, Monterey, California, USA, pp. 373-379.

Nielsen, J. (1993). *Usability Engineering*: Academic Press.

Nielsen, J. (1994, April 24-28), "*Enhancing the Explanatory Power of Usability Heuristics,"* Paper presented at the Computer Human Interaction (CHI), Boston, Massachusetts, USA, pp. 152-158.

Nielsen, J. and Landauer, T. K. (1993, April 24-29), "*A Mathematical Model of the Finding of Usability Problems,"* Paper presented at the SIGCHI Conference on Human Factors in Computing

Systems (INTERCHI '93), Amsterdam, The Netherlands, pp. 206-213.
Nielsen, J. and Molich, R. (1990, April 1-5), "*Heuristic Evaluation of User Interfaces,"* Paper presented at the SIG-CHI Conference on Human Factors in Computing Systems: Empowering People, Seattle, Washington, USA, pp. 249-256.
Nunamaker Jr., J. F., Dennis, A., Valacich, J., Vogel, D., and George, J. (1991), "Electronic Meeting Systems to Support Group Work," *Communications of the ACM* 34(7), pp. 40-61.
Porter, A. A., Siy, H. P., Toman, C. A., and Votta, L. G. (1997), "An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development," *IEEE Transactions on Software Engineering* 23(6), pp. 329-346.
Rodgers, T. L., Dean, D. L., and Nunamaker Jr., J. F. (2004, January 5-8), "*Increasing Inspection Efficiency through Group Support Systems,"* Paper presented at the 37th Hawai'i International Conference on System Sciences, Waikaloa, Hawaii, USA, pp. 18-27.
Satzinger, J. W., Garfield, M. J., and Nagasundaram, M. (1999), "The Creative Process: The Effects of Group Memory on Individual Idea Generation," *Journal of Management Information Systems* 15(4), pp. 143-160.
Sauer, C., Jeffery, D. R., Land, L., and Yetton, P. (2000), "The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research," *IEEE Transactions on Software Engineering* 26(1), pp. 1-14.
Sears, A. (1997), "Heuristic walkthroughs: Finding the problems without the noise," *International Journal of Human-Computer Interaction* 9(3), pp. 213-234.
Shaw, D. (1993), "*CD-ROM Interfaces for Information Retrieval: Heuristic Evaluation and Observations of Intended Users,"* Paper presented at the 14th National Online Meeting, New York, New York, USA, pp. 371-377.
Steiner, I. (1972). *Group Process and Productivity*. New York, New York, USA: Academic Press.
Sutcliffe, A. (Ed.). (2001), "*Heuristic Evaluation of Website Attractiveness and Usability,"* (Vol. 2220). Berlin, Germany: Springer-Verlag.
Tang, Z., Johnson, T., Tindall, R., and Zhang, J. (2006), "Applying Heuristic Evaluation to Improve the Usability of a Telemedicine System," *Telemedecine Journal and E-Health* 12(1), pp. 24-34.
Taylor, D. W., Berry, P. C., and Block, C. H. (1958), "Does Group Participation When Using Brainstorming Facilitate or Inhibit Creative Thinking?," *Administrative Science Quarterly* 3(1), pp. 23-47.
Tyran, C. K. and George, J. F. (2002), "Improving Software Inspections with Group Process Support," *Communications of the ACM* 45(9), pp. 87-92.
Valacich, J. S., Wheeler, B., Mennecke, B., and Wachter, R. (1995), "The Effects of Numerical and Logical Size on Computer-Mediated Idea Generation," *Organizational Behavior and Human Decision Processes* 62(3), pp. 318-329.
Van de Ven, A. H., Delbecq, L. A., and Koenig, R. J. (1976), "Determinants of Coordination Modes Within Organizations," *American Sociological Review* 41(April), pp. 322-338.
van Genuchten, M., van Dijk, C., Scholten, H., and Vogel, D. (2001), "Using Group Support Systems for Software Inspections," *IEEE Software* 18(3), pp. 60-65.
Vredenburg, K. and Butler, M. B. (1996), "*Current Practice and Future Directions in User-Centered Design,"* Paper presented at the Usability Professionals' Association Fifth Annual Conference, Copper Mountain, Colorado, USA, pp.
Vredenburg, K., Mao, J.-Y., Smith, P. W., and Carey, T. (2002), "*A Survey of User-Centered Design Practice,"* Paper presented at the Conference on Human Factors in Computing Systems; SIGCHI Conference on Human Factors in Computing Systems: Changing Our World, Changing Ourselves, Minneapolis, Minnesota, USA, pp. 471-478
Wegner, D. M., Erber, R., and Raymond, P. (1991), "Transactive Memory in Close Relationships," *Journal of Personality and Social Psychology* 61(6), pp. 923-929.
Weick, K. and Roberts, K. (1993), "Collective Mind in Organizations: Heedful Interrelating on Flight Decks," *Administrative Science Quarterly* 38(3), pp. 357-381.
Yin, Z. and Miller, J. (2004), "A Cognitive-Based Mechanism for Constructing Software Inspection Teams," *IEEE Transactions on Software Engineering* 30(11), pp. 811-825.

Appendix 1. LS MEANS Statistics for Productivity Measures

Table A1.1. LS Means

Treat	total1 size 3	total1 size 6	total2 size 3	total2 size 6	usabl1 size 3	usabl1 size 6	usable2 size 3	usabl2 size 6	fp1 size 3
High	39.26 (#1)	55.43 (#4)	38.78 (#1)	54.86 (#4)	21.78 (#1)	23.71 (#4)	21.78 (#1)	23.93 (#4)	10.19 (#1)
Medium	47.66 (#2)	97.58 (#5)	40.41 (#2)	63.08 (#5)	22.91 (#2)	26.58 (#5)	22.63 (#2)	23.00 (#5)	10.25 (#2)
Low	32.65 (#3)	69.40 (#6)	31.18 (#3)	69.40 (#6)	19.24 (#3)	29.70 (#6)	18.53 (#3)	29.70 (#6)	9.12 (#3)

Table A1.1. LS Means (Continued)

Treat	fp1 size 6	fp2 size 3	fp2 size 6	dup1 size 3	dup1 size 6	dup2 size 3	dup2 size 6	add size 3	add size 6
High	19.07 (#4)	10.07 (#1)	19.00 (#4)	7.30 (#1)	12.64 (#1)	6.93 (#1)	11.93 (#4)	0.48 (#1)	1.00 (#4)
Medium	33.00 (#5)	8.97 (#2)	21.83 (#5)	14.50 (#2)	38.00 (#2)	8.81 (#2)	18.25 (#5)	6.69 (#2)	27.33 (#5)
Low	22.40 (#6)	8.65 (#3)	22.40 (#6)	4.29 (#3)	17.30 (#3)	4.00 (#3)	17.30 (#6)	0.05 (#3)	0.00 (#6

= LS means number used in preplanned LS Means comparisons (next table)

Table A1.2. LS Means Preplanned Comparisons

Measure	i/j	1	2	3	4	5
total1	2	0.028*				
total1	3	0.142 (ns)	0.001***			
total1	4	0.001***	0.096 (ns)	< .0001***		
total1	5	< .0001***	< .0001***	< .0001***	< .0001***	
total1	6	< .0001***	< .0001***	< .0001***	0.021*	< .0001***
total2	2	0.701 (ns)				
total2	3	0.132 (ns)	0.059 (ns)			
total2	4	0.003**	0.006**	<.0001***		
total2	5	< .0001***	< .0001***	< .0001***	0.199 (ns)	
total2	6	< .0001***	< .0001***	< .0001***	0.032 (ns)	0.364 (ns)
usable1	2	0.550 (ns)				
usable1	3	0.256 (ns)	0.092 (ns)			
usable1	4	0.416 (ns)	0.727 (ns)	0.087 (ns)		
usable1	5	0.057 (ns)	0.134 (ns)	0.008**	0.313 (ns)	
usable1	6	0.004**	0.012*	0.000***	0.047*	0.314 (ns)
usable2	2	0.668 (ns)				
usable2	3	0.167 (ns)	0.073 (ns)			
usable2	4	0.388 (ns)	0.591 (ns)	0.050*		
usable2	5	0.641 (ns)	0.883 (ns)	0.119 (ns)	0.755 (ns)	
usable2	6	0.005**	0.011*	0.000***	0.067 (ns)	0.040*
fp1	2	0.970 (ns)				
fp1	3	0.601 (ns)	0.567 (ns)			
fp1	4	< 0.0001***	< 0.0001***	< 0.0001***		
fp1	5	< 0.0001***	< 0.0001***	< 0.0001***	< 0.0001***	
fp1	6	< 0.0001***	< 0.0001***	< 0.0001***	0.224 (ns)	0.000***
fp2	2	0.532 (ns)				
fp2	3	0.497 (ns)	0.874 (ns)			
fp2	4	0.0001***	< 0.0001***	< 0.0001***		
fp2	5	< 0.0001***	< 0.0001***	< 0.0001***	0.289 (ns)	
fp2	6	< 0.0001***	< 0.0001***	< 0.0001***	0.227 (ns)	0.845 (ns)
duplicates1	2	0.002**				
duplicates1	3	0.257 (ns)	0.0001***			
duplicates1	4	0.059 (ns)	0.498 (ns)	0.008**		

duplicates1	5	< 0.0001***	< 0.0001***	< 0.0001***	< 0.0001***	
duplicates1	6	0.002**	0.366 (ns)	0.0002***	0.189 (ns)	< 0.0001***
duplicates2	2	0.312 (ns)				
duplicates2	3	0.187 (ns)	0.026*			
duplicates2	4	0.035*	0.174 (ns)	0.002**		
duplicates2	5	< 0.0001***	0.0002***	0.0001***	0.026*	
duplicates2	6	0.0001***	0.001***	0.0001***	0.071 (ns)	0.756
changes	2	0.047*				
changes	3	0.893 (ns)	0.061 (ns)			
changes	4	0.918 (ns)	0.128 (ns)	0.834 (ns)		
changes	5	< .0001***	< .0001***	< .0001***	< .0001***	
changes	6	0.876 (ns)	0.110 (ns)	0.968 (ns)	0.824 (ns)	<.0001***

*p < 0.05, ** p < 0.01, ***p < 0.001

About the authors

Paul Benjamin Lowry is an assistant professor of Information Systems at the Marriott School, Brigham Young University and a Kevin and Debra Rollins Faculty Fellow, where he also directs the IS Ph.D. Preparation Program. His interests include HCI (collaboration, communication, entertainment, interaction design, adoption), e-business (privacy, security, trust, branding, electronic markets), and scientometrics of IS research. He received his PhD in MIS from the University of Arizona. He has had articles published in *Journal of Management Information Systems*; *Journal of the Association for Information Systems*; *Communications of the ACM*; *Communications of the Association for Information Systems*; *Decision Support Systems*; *IEEE Transactions on Systems, Man, and Cybernetics*; *IEEE Transactions on Professional Communication*; *Information Sciences*; *Small Group Research*; *Expert Systems with Applications*; and others. He serves as an associate editor at *AIS Transactions on HCI* and *Communications of the AIS*.

Tom L. Roberts is the Clyde R. King Professor of Information Systems at the School of Business at the Louisiana Technical University. His current research interests include collaborative technology, IT work groups, project management, and the behavioral aspects of the information technology profession. He received his PhD in MIS from Auburn University. Dr. Roberts has had papers accepted for publication in *Journal of Management Information Systems*, *Information and Management*, *IEEE Transactions in Software Engineering*, and *IEEE Transactions in Engineering Management* among others.

Douglas L. Dean is an associate professor at the Marriott School, Brigham Young University. He received his PhD in MIS from the University of Arizona in 1995. Dr. Dean's research interests include electronic commerce standards and collaborative tools and methods. His work has been published in *Management Science*, *Journal of the Association for Information Systems*, *Journal of Management Information Systems*, *Data Base*, *Communications of the Association for Information Systems*, *Expert Systems with Applications*, *Group Decision and Negotiation*, and others.

George M. Marakas is a professor of Information Systems at the School of Business at the University of Kansas. His teaching expertise includes systems analysis and design, technology-assisted decision making, electronic commerce, management of IS resources, behavioral IS research methods, and data visualization and decision support. He has received numerous national teaching awards, and his research has appeared in several top journals, including *Information Systems Research*, *Information and Management*, *Management Science*, *International Journal of Human-Computer Studies*, and *European Journal of Information Systems*. He is also the author of five best-selling textbooks in information systems: *Decision Support Systems for the 21st Century*; *Systems Analysis and Design: An Active Approach*; *Data Warehousing, Mining, and Visualization: Core Concepts*; *Management Information Systems*; and *Introduction to Information Systems* with Professor James O'Brien.

Journal of the Association for Information Systems

ISSN: 1536-9323