Facilitator-in-a-Box: Process Support Applications to Help Practitioners Realize the Potential of Collaboration Technology

ROBERT O. BRIGGS, GWENDOLYN L. KOLFSCHOTEN, GERT-JAN DE VREEDE, STEPHAN LUKOSCH, AND CONAN C. ALBRECHT

ROBERT O. BRIGGS is a professor of information systems at San Diego State University. He earned his Ph.D. in management information systems at the University of Arizona in 1994. He researches the cognitive foundations of collaboration and uses his findings to design new collaborative work practices and technologies. He is cofounder of the collaboration engineering field and coinventor of the thinkLets design pattern language for collaborative work processes. He has made exploratory, theoretical, experimental, and technological contributions to the areas of team productivity, technology-supported learning, ideation, creativity, consensus, satisfaction, willingness to change, and technology transition. He has designed collaboration systems and collaborative workspaces for industry, academia, government, and the military. He cochairs the Collaboration Systems and Technology track for the Hawaii International Conference on System Sciences. He lectures worldwide on collaboration theory and practice, and on the philosophy of science.

GWENDOLYN L. KOLFSCHOTEN is an assistant professor at the Delft University of Technology in the Netherlands. She received her Ph.D. in collaboration engineering from Delft University of Technology in 2007. Her research focuses on how teams, experts, or stakeholders can effectively collaborate, and use collaboration support tools, in problem-solving tasks. She is an experienced facilitator of Group Support System workshops and sessions. Her current research focuses on cognitive perspectives on collaboration effectiveness, and implications for the design of collaboration support systems. She also works on the design of intelligent collaboration support systems, on creating shared situational awareness, and worked on collaboration engineering and the thinkLets concept to capture patterns in effective collaborative effort. She was a visiting scholar at the Massachusetts Institute of Technology in 2011. She was a program chair for Collaboration Researchers International Working Group 2010. Her research has been presented at the Hawaii International Conference on System Sciences, Collaboration Researchers International Working Group, Americas Conference on Information Systems, Group Decision and Negotiation, and other conferences and has been published in the Journal of Management Information Systems, International Journal of Human–Computer Studies, Journal of the Association for Information Systems, Computers and Education, and Group Decision and Negotiation.
GERT-JAN DE VREEDE is the Charles & Margre Durham Distinguished Professor and the managing director of the Center for Collaboration Science at the University of Nebraska at Omaha. He received his Ph.D. in systems engineering from the Delft University of Technology, the Netherlands, in 1995. He has been a visiting professor at the University of Arizona and the University of Pretoria. His research focuses on field applications of collaboration technologies, the theoretical foundations of collaboration, collaboration engineering, the facilitation of group work, and crowdsourcing. He is cofounder of the collaboration engineering field and coinventor of the thinkLets concept. He has published over 200 refereed journal articles, conference papers, and book chapters and was named the most productive Group Support Systems researcher worldwide from 2000 to 2005. His research has appeared in journals such as Journal of Management Information Systems, Journal of the Association for Information Systems, Communications of the Association for Information Systems, Small Group Research, and Communications of the ACM.

STEPHAN LUKOSCH is an associate professor at the Delft University of Technology in the Netherlands. He received his Ph.D. in computer science with distinction from the FernUniversität. His current research focuses on collaborative design and engineering in emerging interaction spaces such as mediated and augmented reality. From 2003 to 2008, he was assistant professor at the FernUniversität in Hagen, where he led a research group that focused on environments for cooperative working and learning. In 2007 he published a foundational book on patterns for computer-mediated interaction. His articles appeared in various journals, including the International Journal of Cooperative Information Systems, International Journal of Human Computer Studies, and Journal of Universal Computer Science. Currently, Dr. Lukosch is a steering committee member of the special interest group on Computer-Supported Cooperative Work (CSCW) of the German computer science society and Collaboration Researchers International Working Group conference on Collaboration and Technology. He serves on the editorial boards of Journal of Universal Computer Science and International Journal of Cooperative Information Systems.

CONAN C. ALBRECHT is a professor of information systems at Brigham Young University. He received a Ph.D. in information systems from the University of Arizona in 2000. He researches computer-based fraud detection techniques and online group technology. He is one of the primary authors of Fraud Examination, a textbook used at many universities worldwide. He has published articles on fraud detection and group theory in MIS Quarterly, Journal of Forensic Accounting, Journal of Accountancy, Communications of the ACM, Decision Support Systems, Information and Management, and other publications. Dr. Albrecht is currently working on a platform to provide free and pay-for-upgrade university-level textbooks to students.

ABSTRACT: The potential benefits of collaboration technologies are typically realized only in groups led by collaboration experts. This raises the facilitator-in-the-box challenge: Can collaboration expertise be packaged with collaboration technology in a form that nonexperts can reuse with no training on either tools or techniques? We address that challenge with process support applications (PSAs). We describe a collaboration support system (CSS) that combines a computer-assisted collaboration engineering platform for creating PSAs with a process support system runtime platform for executing PSAs. We show that the CSS meets its design goals: (1) to reduce development cycles for collaboration systems, (2) to allow nonprogrammers to design and develop...
PSAs, and (3) to package enough expertise in the tools that nonexperts could execute a well-designed collaborative work process without training.

KEY WORDS AND PHRASES: collaboration, collaboration engineering, collaboration support system, collaboration technology, computer-assisted collaboration engineering, process support application.

GROUPS FACE A COMPLEX SET OF COMMUNICATION, COORDINATION, AND COPRODUCTION CHALLENGES [6]. DIFFERENCES OF GOALS AND VALUES [24, 44], ORGANIZATIONAL CULTURE [43], FUNCTIONAL BACKGROUND [42], AND LEADERSHIP STYLE [38], FOR EXAMPLE, CAN AFFECT TEAM PERFORMANCE. THESE ISSUES MAY BE WORSE WHEN TEAMS SPAN GEOGRAPHY [16], TIME ZONES [13], AND NATIONAL CULTURES [5, 21, 45, 49]. FIELD STUDIES DEMONSTRATE, HOWEVER, THAT UNDER CERTAIN CONDITIONS, GROUPS LED BY EXPERT FACILITATORS CAN USE COLLABORATION TECHNOLOGIES SUCH AS GROUP SUPPORT SYSTEMS (GSS) TO REDUCE THE EFFORT REQUIRED TO ACHIEVE THEIR GOALS WHILE IMPROVING THE VALUE THEY CREATE (E.G., [1, 3, 12, 14, 18, 20, 23, 40, 41, 48, 53]). AN EXPERT FACILITATOR WORKS WITH A GROUP TO DEFINE ITS GOALS AND WORK PRODUCTS AND TO DESIGN A PROCESS FOR CREATING THOSE PRODUCTS. THE FACILITATOR CONFIGURES TECHNOLOGIES TO SUPPORT THE PROCESS AND LEADS THE GROUP THROUGH THE PROCESS. DURING EXECUTION OF THE PROCESS, THE FACILITATOR MONITORS AND INTERVENES TO IMPROVE EMERGING ISSUES OF COMMUNICATION, REASONING, INFORMATION ACCESS, DISTRACTION, AND GOAL CONGRUENCE [55].

SKILLED FACILITATORS, HOWEVER, ARE NOT FEASIBLE FOR SOME GROUPS. FACILITATORS TEND TO BE SCARCE AND EXPENSIVE, AND SO ARE NOT ECONOMICALLY FEASIBLE FOR SOME GROUPS. FURTHER, THEIR SPECIAL SKILLS GIVE FACILITATORS HIGH PROFESSIONAL MOBILITY, MAKING IT DIFFICULT FOR ORGANIZATIONS TO MAINTAIN A STABLE IN-HOUSE FACILITATOR-SUPPORTED COLLABORATION TECHNOLOGY CAPABILITY (E.G., [2, 39]).

THIS DILEMMA CREATED A CHALLENGING RESEARCH QUESTION: CAN NONEXPERTS REALIZE THE POTENTIAL BENEFITS OF COLLABORATION TECHNOLOGY WITHOUT THE EXPERT HELP? THIS QUESTION GAVE RISE TO COLLABORATION ENGINEERING (CE) [8, 54, 55], AN APPROACH TO DESIGNING COLLABORATIVE WORK PRACTICES FOR HIGH-VALUE TASKS AND TRANSFERRING THEM TO PRACTITIONERS TO EXECUTE FOR THEMSELVES WITHOUT ONGOING SUPPORT FROM AN EXPERT FACILITATOR [55]. CE RESEARCHERS HAVE SHOWN THAT NONEXPERTS CAN QUICKLY BE TRAINED TO EXECUTE WELL-ENGINEERED WORK PRACTICES WITH RESULTS COMPARABLE TO THOSE OF GROUPS LED BY COLLABORATION EXPERTS [28, 29, 30]. THIS, HOWEVER, WAS NOT SUFFICIENT BECAUSE GROUPS STILL REQUIRED SUPPORT FROM A TRAINED LEADER. WHEN A TRAINED LEADER LEFT, THE GROUP COULD NOT REALIZE THE BENEFITS OF THE TECHNOLOGY. THIS GAVE RISE TO THE FACILITATOR-IN-A-BOX CHALLENGE [11, 27]:

CAN COLLABORATION EXPERTISE BE PACKAGED WITH COLLABORATION TECHNOLOGY IN A FORM THAT NONEXPERTS CAN REUSE WITH NO TRAINING ON EITHER THE TOOLS OR TECHNIQUES?
Some researchers have explored ways to make computers exhibit expert facilitator behaviors, for example, by displaying facilitative textual prompts to stimulate critical thinking [17], by providing feed-forward and feedback guidance to support decision-making tasks [31], by automatically reconfiguring technologies when the system detects a collaboration issue [51], and by flexibly adapting the structure of a work process to support emergent needs [25]. Despite promising results, however, computers are not yet sufficiently advanced that they can fulfill the complete role of an expert facilitator. It is therefore still difficult for some groups to realize the potential benefits of collaboration technology.

In this paper, we report a design science research (DSR) initiative [22] to address this challenge. We initiate the DSR rigor cycle by drawing on the collaboration literature to propose process support applications (PSAs) as a conceptual approach toward packaging collaboration expertise with collaboration technology. We propose collaboration support systems (CSS) as an approach for creating and deploying PSAs, and present an exemplar instantiation of a CSS. We report qualitative and quantitative studies to validate the utility and generalizability of the CSS approach, completing the DSR design and relevance cycles. We discuss the implications for research and practice, completing the DSR rigor cycle.

**PSAs: Packaging Expertise with Tools**

This section draws on the scientific literature to propose a solution toward packaging collaboration expertise with collaboration technology. While a computer cannot yet function as a facilitator, groups may not need all of an expert’s expertise to succeed. They may need only the end product of an expert’s work—designed artifacts that provide the group with carefully designed process restrictiveness.

**Process Restrictiveness**

Process restrictiveness means limiting a group’s interactions to a subset of the possible, a specific sequence of activities executed under a specific set of constraints [56]. Much of the value groups gain from expert facilitators, from collaboration technology, and from training derives from process restrictiveness [15, 56]. An expert facilitator designs a work practice to restrict the group from actions that would limit their effectiveness, and to restrict the group to actions that would increase their likelihood of success. A facilitator invokes process restrictions with (1) a prescribed sequence of activities, (2) specified procedures for each activity, (3) guidance to invoke social norms, and (4) guidance to explain how people should use the technology. Thus, one way a group benefits from an expert facilitator is by letting the facilitator restrict their processes.

An expert facilitator also restricts the group to particular technologies, in particular, configurations. These are chosen to support actions that would increase group effectiveness while restricting actions that would be detrimental to their success. For example, the expert might configure a brainstorming tool to support simultaneous contributions from all participants while restricting their ability to delete one another’s contributions.
Thus, one way a group benefits from collaboration technology is by letting it invoke useful process restrictions.

When experts train a group to run a well-designed process for themselves, group members learn a set of process and technology restrictions—what activities they should execute, what tools they should use and how to configure them, and what constraints they should follow as they work. Thus, much of the value a group gains from expert facilitators, from collaboration tools, and from training derives from process restrictiveness.

Embedding Process Restrictiveness in Collaborative Software Applications

If there was a way for a collaboration expert to enforce process restrictions for technology users without resorting to facilitation or training, then it might be possible for nonexpert users to execute a work practice without a facilitator and without process training. If supporting technology could be made so simple as to be self-evident, then it might be possible for nonexperts to execute the work practice for themselves without technology training. We propose process support applications as a generalizable solution to achieving those goals. A PSA is a collaborative software application designed by a collaboration expert and optimized for one specific task. A PSA (1) presents a group with procedures for a well-designed sequence of activities, (2) provides the group with optimized technology configurations to support the procedures for each activity, and (3) communicates guidance to the group about the actions they should take and the constraints under which they should act during each activity.

A PSA would present the group with a sequence of activities. For each activity, it would present the group with just the right tools, configured in just the right way, linked to just the right data, with communication channels connected to just the right people, and with just the right collaboration guidance to execute the task. A PSA approach might make it possible to package sufficient collaboration expertise with the technology to move nonexperts successfully through an engineered work practice without a facilitator and without training. Consider the following:

- Current general purpose GSSs are complex because they must be configurable to meet a wide variety of needs [37]. Task-specific PSA tools could be made simpler by providing exactly and only the capabilities required for the specific actions of the specific activity.
- Task-specific tools in a PSA could be configured at design time to block unproductive actions, reducing the need for facilitator guidance about those actions.
- A task-specific PSA could be configured to move and transform data between activities, obviating the need to train users how to move contributions between activities.
- Task-specific tools in a PSA could provide guidance through, for example, tutorials, moderator scripts, audio and video clips, and activity-specific labels on its functional features. A brainstorming tool for a risk assessment PSA might say,
for example, “Type a risk here,” rather than “Type an idea here,” reducing the need for process and technology training. Thus, a PSA approach might make it possible for a group of nonexperts to approach a technology-supported collaborative work practice with no prior training and execute it successfully. If our arguments about process restrictiveness hold, then users should not need to know the various patterns of collaboration or why the collaboration engineer selected a particular sequence of patterns for a given activity. They should not need to know techniques for invoking variations of those patterns or why a particular technique was selected for a given activity. They should not need to know the variety of technologies and configurations that could be used to support the chosen technique or why the expert chose a particular technology and configuration for their work practice. They should only need a simple screen that instantiates and restricts them to those choices.

General Requirements for the PSA Approach

The PSA approach presents some interrelated operational, technical, and economic challenges. This section derives some general requirements pertaining to those challenges.

A collaboration engineer cannot unilaterally impose a new work process on a group because the group members can simply decline to adopt and use the process [52]. Collaboration engineers must therefore work in concert with practitioners to design PSAs that the users find useful and acceptable [19]. Therefore,

\[ R1: \text{A PSA solution should support joint design, co-creation, and validation of PSAs by experts and practitioners.} \]

Research shows that a structured design methodology can improve the consistency and quality of the collaboration process [26]. Therefore,

\[ R2: \text{A PSA solution should support the activities of a structured design methodology.} \]

Collaboration engineers tend not to be software development experts. Therefore,

\[ R3: \text{Collaboration engineers should be able to create PSAs without having to write software code.} \]

Data models for collaborative work practices are sometimes complex, but collaboration engineers and the practitioners with whom they work are not typically data modeling experts. Therefore,

\[ R4: \text{A PSA solution should provide a simple way to create a complex data model for a collaborative work practice.} \]

Collaboration needs can evolve rapidly, even over a period of days or weeks. Therefore,
R5: It must be possible to create and modify a PSA quickly.

The expense of building a new PSA from scratch for each collaborative task would be prohibitive. Synchronous collaborative applications typically require an order of magnitude more programming effort than single-user applications because multiple synchronous users can execute potentially conflicting actions on the same shared objects [34]. Therefore,

R6: A PSA solution should use a middleware environment that packages the hardest-to-build aspects of synchronous systems in a reusable environment.

Analysis of the collaboration technology marketplace suggests that most collaboration systems are comprised of varying configurations of a small set of core capabilities [37]. The expense of creating task-specific PSAs could therefore be further reduced if they could be snapped together from configurable components rather than built from scratch for each PSA. Therefore,

R7: A PSA solution should include a rapid development studio where loosely coupled components can be configured to behave as if they were a tightly integrated application.

In this paper, we call a rapid development studio for PSAs a computer-assisted collaboration engineering (CACE) environment. It would not be possible at CACE design time to predict the full range of data types, relationships, and data structures that might be required for the variety of group processes it may eventually support. Postimplementation modifications to data structures, however, can require expensive rework of the code. Therefore,

R8: The data server for the PSA solution should use a universal data model that accepts any data of any type in any set of relationships in real time.

At CACE design time, it will not be possible to predict the full range of collaborative capabilities a collaboration engineer will need to bring to bear to support groups. Therefore,

R9: The architecture for a PSA solution should be open, allowing programmers to easily plug in new capabilities as needed.

Mametjanov et al. [33] proposed an ontology of classes and relationships pertaining to the logical and physical design of collaborative work practices, software applications to support them, and development platforms for creating the software. Table 1 abstracts from that ontology a hierarchy of classes pertaining to PSAs. Each class pertains to different design goals for a work practice. Therefore,

R10: A solution for packaging collaboration expertise with collaboration technology should support logical and physical design and implementation for the classes abstracted from the ontological hierarchy.
Table 1. A Hierarchy of Classes Pertaining to Logical Design and Physical Design of Collaborative Work Practices and PSAs to Support Them

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace</td>
<td>A software environment that moves people through the activities of a designed work practice, providing them with the digital resources and guidance they will need to succeed.</td>
</tr>
<tr>
<td>Role</td>
<td>A collection of capabilities and guidance that supports and constrains the communications and actions of some subset of group members to improve their likelihood of success.</td>
</tr>
<tr>
<td>Agenda</td>
<td>A collection of activities for creating work products, and the conditional logic for deciding which activities should be executed by which roles in what order.</td>
</tr>
<tr>
<td>Activity</td>
<td>One step in the process for executing the work practice the workspace is designed to support.</td>
</tr>
<tr>
<td>Screen</td>
<td>The collection of all tools, controls, communication channels, data, and guidance that people in one role will require to execute one activity.</td>
</tr>
<tr>
<td>Tool</td>
<td>An arrangement of loosely coupled collaborative software components configured to behave as if they were a purpose-built, tightly coupled software object designed to support a specific group procedure.</td>
</tr>
<tr>
<td>Component</td>
<td>A configurable stand-alone software module that delivers a specific collaborative capability, e.g., a shared outliner, a voice channel, a shared document repository.</td>
</tr>
<tr>
<td>Control</td>
<td>Objects that appear on a screen to inform participants or to grant them control over their process, e.g., previous/next buttons, roster, or presence indicators.</td>
</tr>
<tr>
<td>Population rule</td>
<td>Specifies what kind of contributions a component will accept from users, will send to the server, and will request back from the server.</td>
</tr>
<tr>
<td>Contribution</td>
<td>An object contributed to the group by a participant, e.g., a block of text, an image, a vote, an audio/video stream, a document.</td>
</tr>
</tbody>
</table>

Source: Adapted from [33].

The ActionCenters Prototype CSS

The DSR design cycle requires that researchers develop an exemplar instance of a generalizable solution to the problem the research addresses. In this section, we describe the ActionCenters prototype, an expository instance of a PSA solution called a collaboration support system. It combines a CACE environment for designing and building PSAs with a process support system (PSS), a runtime environment for acquiring, managing, and running PSAs. We produced the prototype through a series of 34 DSR design cycles and relevance cycles [22] over a three-year period. In each cycle, stakeholders negotiated with the development team to design or redesign functionality, user interfaces, and user experiences (UI/UX) to meet stakeholder goals. DSR design cycles ran concurrently with DSR relevance cycles; at the end of each cycle, the development team delivered usable code to users for operational testing (relevance cycle), and user responses informed the next design cycle. In this section, we sum-
marize key aspects of the ActionCenters CSS and describe how a PSA is realized as a designed object. The primary design goals for this project were (1) to reduce development cycles for collaboration systems, (2) to make it possible for nonprogrammers to design and develop PSAs, and (3) to package sufficient collaboration expertise with the collaboration technology so nonexperts could execute a well-designed work practice without training on the techniques or the technologies.

ActionCenters PSA: The Designed Object

In the ActionCenters CSS, a PSA has three states. During design, development, and testing, a PSA exists in the CACE as a project (Figure 1). When a PSA is published for use in the field, it exists as an application. When users instantiate and use a PSA, it exits as a workspace.

The ActionCenters CACE realizes a PSA as a hierarchy of objects consistent with the ontology proposed by Mametjanov et al. [33] (Table 1). The objects in the hierarchy have multiple configurable properties, for example, a name, a description, and an identifying icon. Some objects have only a handful of properties, some have more than a hundred. The ActionCenters CACE provides one or more editors for configuring the various properties of each kind of object in the hierarchy (Figure 1).

PSA Object: Project, Application, Workspace

The top-level object in the hierarchy is the PSA object. In the ActionCenters prototype, a PSA object is called a project at design time, an application when it is published to a library for users to access, and as a workspace when it is instantiated online by a particular group to support execution of a particular task.

PSA Agenda: Activity and Phase Objects

A project has one and only one agenda, which is a collection of one or more activities and the conditional logic for their order of execution. Most properties of an activity pertain to its logical design, for example, an overview, goal, technique, work products, and duration. Some properties pertain to physical design, for example, instructions for participants and a tutorial script for moderators. Activities may be optionally organized into phases on the agenda.

PSA Role Object

A project has one or more roles for users. At activity design time, a collaboration engineer can design different capabilities and guidance for people in different roles.

PSA Screen Object

For each role for each activity, a project has one and only one screen object. A screen is a physical design element for a PSA. It represents a visual layout of the capabilities
that will be provided to users in a specific role for a specific activity. Its properties specify the identity, for example, size and position, of each object that will appear on the screen to the people in that role during that activity.

PSA Control Object

A screen may present a user with zero-to-many controls. A control is a software module that informs participants about their work process (e.g., a screen prompt, video clip, or a leader script) or that grants them control over some aspect of their process (e.g., to move to another activity, to invite others into the workspace, or to produce a report).

PSA Tool Object

A screen may also display zero-to-many tools. A tool provides users with capabilities to add, share, and manipulate their contributions. The properties of a tool object will, for example, identify the tool (e.g., name, version, description, icon), configure its display mode, identify its components, and establish its data set. A data set is a named
collection of user contributions. Every tool has one and only one data set. Multiple tools may use the same data set and so may share the set of contributions.

PSA Component Objects

Tools are composed of one or more loosely coupled, highly configurable collaborative software components, such as shared editors, shared document repositories, and polling capabilities. Components may be configured to behave as if they were a tightly integrated software application tailored specifically to support the task at hand. The components that comprise a tool are organized as a tree structure with one and only one component at the top level. Components may be configured such that mouse or keyboard events on contribution in a higher-level component cause a subcomponent to pop up. A tool could be assembled from a polling component and an outliner component, for example, so that voters in the polling component could pop up an outliner component to explain their votes. A different tool might be configured from the same components such that people collaborating on an outline could pop up a polling component for any outline heading to evaluate it on one or more criteria.

All the contributions added to the top level of a subordinate component are automatically assigned a subordinate relationship to the parent contribution in the superior component. For example, a diagram editor might contain three symbols representing three different business units: marketing, production, and engineering. It could be configured, for example, so that a double-click on the production symbol would pop up a comment component for the production unit. All the comments added to that comment window would automatically have a subordinate relationship to the production contribution in the diagram editor.

The properties of a component may specify, for example, the look and feel of contributions (e.g., fonts, formats, colors of contributions, tags, and icons), user rights (e.g., view, add, edit, move, judge, or delete contributions), drop events that specify what should happen when a user drags and drops one contribution onto another (e.g., move copy), or trigger events (e.g., mouse clicks and key presses that should launch a subordinate component).

Shells are a special subset of components for controlling the presentation of the other components that comprise a tool. A tab shell, for example, would present a set of sibling components on a series of tabs. A pane shell would arrange a set of sibling components in contiguous panes.

Every PSA component has one or more population rules. Population rules assign semantically meaningful contribution types to contributions and semantically meaningful relationships among contributions. In a PSA for software requirements negotiation, for example, a shared outliner component might be configured with a population rule to specify that all of the items at the top level of the outline might be of type requirement with a member-of relationship to the preliminary requirements data set. Its next population rule might specify that all of the items subordinate to a requirement in the outliner will be assigned the contribution type issue with a constrains relationship to its parent requirement. At runtime, the outliner component would receive its data set
and population rules from the server. It would then query the server to send it all the contributions of type requirement that have a member-of relationship to the preliminary requirements data set and all of the contributions of type issue with a constrains relationship to parent requirements. It would then display the requirements and issues according to its population rules.

Population rules provide a way to address R4, that there be an easy way for collaboration engineers to develop complex data models for a work practice. Population rules are also a solution to R8, that a PSA server should be able to accept any data in any set of relationships in real time without having to modify the server’s metadata. Population rules provide just-in-time metadata, because contributions arrive at the server with their metadata—a data set and one or more named relationships to one or more parent contributions.

Population rules also make it easy for a collaboration engineer to configure a PSA to move contributions automatically from one activity to the next. Tools that share the same data set and the same population rules will request and receive the same contributions from the server. To move data from one activity to the next, therefore, the collaboration engineer needs only to assure that the population rules match across activities. New contributions made through one tool will automatically appear in any other tools with the same data set and population rules.

PSA Contributions

In the ActionCenters CSS, any content a user creates and sends to the group is instantiated as a contribution object. Each contribution has a globally unique identifier. All other aspects of the contribution are stored as properties of the contribution. A user-generated heading in a shared outliner, for example, would be stored in a contribution’s text property. This makes contributions portable among components with different presentation modes, regardless of their digital content. If the outline heading was later opened in a diagramming component, the diagramming component might, for example, add properties to the contribution for default shape, size, orientation, position, and color, but reuse its text property as a shape label. Each component uses only the contribution properties about which it knows, ignoring all others.

ActionCenters CACE: The Design-Build Environment

The ActionCenters CSS includes a collaborative CACE platform where experts and nonexperts can work together to design, develop, and test PSAs. This section summarizes key aspects of the CACE and its architecture.

In the CACE, a user may enter an existing PSA project or create a new one. A user may add objects to the object hierarchy of a PSA project. Users may, for example, add roles for different kinds of users, and may add activities to the agenda. They may add controls and tools to a screen, components to a tool, and population rules to a component.
Any object in the PSA hierarchy may be opened so that the user may edit its properties. The CACE provides at least one kind of editor for each kind of object. For some of the more complex objects, it provides multiple editors, each optimized for configuring different aspects of the object. Figure 2, for example, shows the ActionCenters tool editor. In this editor, the user sees and configures the components that comprise the tool and the population rule for each component. The user may add components to a tool by dragging and dropping them from the palette on the left. The user may also drag-and-drop configured components from the tool or the whole tool itself from the tool editor back to the palette to save them for later reuse. Notice that the tool editor presents the users with 11 different tabs above and below the components and population rules it displays. Each of these tabs provides the user with different ways to edit the properties of a different aspect of the tool and its components.

The output of the CACE tool is a PSA object that is published as an “application.” The PSA object, however, does not actually contain executable collaborative software components and controls. Rather, the software components and controls actually exist in the PSS, the runtime aspect of the CSS. The PSA object is actually an XML...
Although the screen appears simple, it supports a carefully selected ideation technique that invokes a chosen pattern of collaboration to move a group quickly to a set of high-quality ideas.

(Extensible Markup Language) bundle that specifies all the properties of all the objects at all levels of the PSA object hierarchy. At runtime, the PSS reads the properties in the PSA object and configures its components and controls to behave as if they were a purpose-built application tailored to support the specific work practice for which the PSA was designed.

Figure 3 shows the runtime screen for users in one role for one activity in PSA to support military decision makers in a course-of-action (COA) development process. Despite the complexity of the collaboration process design and the underlying technology, the screen presents practitioners with a very simple user experience. The banner at the top of the screen and the logo at the bottom of the screen are instances of a screen control that may be configured to display HTML (Hypertext Markup Language) content. The HTML displayed there was bundled as a property of the control in the PSA object. The shared list in the middle of the screen is an instance of an outliner component. Its colors, fonts, formats, numbering scheme, and icons were bundled as properties of the outliner component in the PSA object. The “Next” button at the bottom of the screen is a navigation control whose properties were likewise bundled in the PSA object. At runtime, the screen looks to the user like an integrated application built specifically to support their task.

Validating the PSA Solution

This section examines the degree to which the ActionCenters CSS solution achieves its primary design goals: (1) to reduce development cycles for collaboration systems, (2) to make it possible for nonprogrammers to design and develop PSAs, and (3)
package sufficient collaboration expertise in the collaboration technology so nonexperts could execute a well-designed work practice without training on the techniques or the technologies.

Reduced Development Cycles for PSAs

Because some collaborative work practices evolve quickly as conditions change, it was required that it be possible to develop PSAs quickly. To demonstrate that a CSS approach could reduce development times for PSAs, we designed a PSA and then compared the development time for the PSA to a benchmark estimate provided by professional software developers of the effort required to build the same PSA as a stand-alone application.

We designed the PSA to support a military COA development process. Military decision makers at the operational echelon execute variations of this process in response to significant changes in the battle space. The COA PSA moved a group through five activities: (1) infer the adversary’s intentions, (2) evaluate the adversary’s actions, (3) propose courses of action, (4) evaluate viability of COAs, and (5) elaborate the pros and cons for the most viable COAs. Without an expert facilitator, the process typically runs 90 minutes and produces one viable COA; with a GSS and an expert facilitator, the process runs 30 minutes and produces three to four viable COAs [12].

We selected that task for several reasons: (1) it was a recurring, mission-critical, time-sensitive collaborative task where small gains in execution time and work-product quality yielded substantial benefits in terms of saved lives and property [12]; (2) we had facilitated the process a number of times with military decision makers in the field and so were familiar with its practitioners’ procedures and preferences for the process; and (3) its goals, work products, procedures, and indicators of excellence were well defined and understood by its practitioners.

A collaboration engineer mapped the patterns of collaboration for each activity, selected a collaboration technique for invoking each pattern of collaboration, and designed a screen to support each activity. The screen in Figure 3 supports the first activity in the COA PSA. It is based on a OnePage idea generation technique [7]. The collaboration engineer selected that technique because it is useful for small groups who need to generate a small set of ideas in a short period of time. The screen design presents users with several levels of guidance. The title bar sets the overall context for the activity. The list title cues users about the significance of the list. A screen prompt provides an explicit indication of what kind of contributions people should make and where they should make them. At runtime, each activity makes available to users a guidance video and a tutorial script to explain the details of the activity—its purpose, its products, and what actions users should take under what constraints using what capabilities. The guidance for the screen in Figure 3, for example, says:

You just heard an intelligence briefing indicating that something important has changed in the battle space. The commander has charged us to develop three possible courses of action and a recommendation for dealing with this change.
As a first step, consider the adversary’s intentions. With the facts you heard, and with your professional experience, what could the adversary be preparing to do? What could their intentions be?

For the next five minutes we will brainstorm on the adversary’s possible intentions. Please type an adversary’s intention in the edit box at the bottom of the screen.

Click the Add button to send your contribution to the group. It will appear on the list. Read the ideas contributed by others for inspiration. Build on one another’s ideas.

Add as many possible intentions as you can during the short time we have available. We will evaluate these possibilities in our next activity.

When time runs out or when the group runs out of ideas, click the Next button to move to the next activity.

Professional software developers for a commercial vendor of GSSs drew on archival records of prior projects to provide a benchmark estimate that it would require three to three-and-a-half person-years to develop the COA PSA from scratch as a stand-alone collaborative application. We implemented the same design into the CACE by configuring snap-together components and controls. That project required 21 person-hours.

We next developed a seven-activity PSA for creative problem solving, which required approximately 32 person-hours to develop and refine, and a four-activity PSA for conducting marketing focus groups that required about 20 person-hours. For the validation study reported below, we spent approximately 23 person-hours creating and refining a six-activity work practice for stakeholders in a real estate management project. Thus, we have demonstrated that the CASE solution fulfills general requirement R5, that it be possible to create and modify PSAs quickly.

Development of PSAs by Nonprogrammers

All the PSAs listed above were created in the CACE by nonprogrammers without writing new code. Thus, we have demonstrated that the CACE solution fulfills general requirement R3, that it should be possible for nonprogrammers to create PSAs. The creators of these PSAs were able to create the data models for the PSA without training as data modelers by configuring population rules for components. This demonstrates that the PSA meets R4, that it be possible for people who are not trained data modelers to develop a PSA.

Execution of Engineered Work Practices by Untrained Groups

This section reports the results of our first attempts to formally validate that a generalizable PSA solution can package enough collaboration expertise with the collaboration technologies that nonexperts could use a PSA to successfully execute an engineered work practice with no training on the techniques or the technology. We conducted two pilot tests with practitioners in the field, followed by a formal validation study among
university students. The formal validation study contrasts the ability of untrained participants to execute an engineered, PSA-supported work practice with the many untrained groups we and other researchers had observed in the lab and field over the past 20 years who were not able to realize the potential benefits of collaboration technologies without the assistance of an expert facilitator (e.g., [12, 32, 35, 36]).

The COA Pilot Test

In the first pilot test, we invited three, three-person groups who worked in a competitive industry if they would like to try out a new piece of collaborative military software to analyze their competitor’s strategies and devise new ways to compete. Before each group arrived, we set up three computers running a fresh workspace based on the COA development PSA workspace described above. None of the participants had prior experience with group support technology and none had military experience. Each group worked on a real problem from their professional setting.

When the groups arrived, a researcher welcomed them and invited them to sit at the computers. The groups were told, to paraphrase, “This is the application we were telling you about. The military will use it to develop courses of action when something important happens in the battle space. We would like to see how it works for civilians. It’s fairly self-explanatory. Why don’t you try it out and see how it goes? With your permission, I’ll sit here and observe.” The three groups were able to execute the five activities of the PSA without further intervention. For each activity, each group executed the collaboration techniques designed by the collaboration engineer.

The Creative Problem-Solving Pilot Test

In this pilot test, eight groups of four or five people at two universities in the United States were invited to join creative problem-solving workshops to address an array of challenges faced by the university. None of the participants had prior experience with group support technology, or with the specific work practice the PSA supported. The activities in the PSA were (1) welcome, (2) identify problems, (3) prioritize problems, (4) propose solutions, (5) develop pros and cons, (6) choose best solutions, and (7) thank you. Before the participants arrived, we set up computers and launched a fresh university problem-solving PSA workspace. Each group worked on real problems of its university.

When the participants arrived, they were invited to sit at a computer and to use the PSA for themselves while a researcher observed their activities. All the groups successfully completed the activities. Three groups executed all the collaboration techniques as designed by the collaboration engineer. For the propose solutions activity, five groups ignored a constraint in the guidance and so instantiated a more conventional brainstorming technique than that designed by the collaboration engineer. This suggests that the design did not incorporate sufficient process restrictiveness on that activity to assure it would be executed as intended. The participants were nonetheless able to complete the task successfully. We therefore designed a more formal validation study.
Formal Validation of the PSA Solution

To validate that untrained users could succeed with a PSA approach, we conducted a larger, more formal study of PSA use among undergraduate systems engineering students at a university in the Netherlands. The students used the PSA to support a course project where they were to analyze and reengineer the university’s existing business process for managing its real estate holdings and to design a new technology to support it. The students learned the basics of business process reengineering in their course but had no prior experience with the PSA.

Participants

Twenty-seven groups of five or six students from an undergraduate business process reengineering class at a university in the Netherlands were invited to use the PSA to support a preliminary phase in their work on a class business process reengineering project. The student groups had the option to use the PSA or to complete the project using their own means. Those who started with the PSA had the option to switch and execute their work using conventional processes, although they were encouraged to try the PSA again if they experienced technical difficulties. Twenty-six of the 27 groups used the PSA to support their project work; the group that chose not to use the PSA was eliminated from the study.

Individual students had the option to opt out of the measurement portions of the study. A total of 128 students responded to a survey at the end of the study. Of those, 66 percent were male and 34 percent were female. The participants ranged in age from 18 to 27 years, with an average age of 19.5 years. There was a statistically significant correlation between age and sex ($r = 0.193$, $p < 0.05$); male participants, who averaged 19.6 years, were, on average, older than the female participants, who averaged 19.1 years. Eighty-three percent of the participants were first-year students, 11 percent were in their second year, and 7 percent were in their third year.

The PSA: Business Process Reengineering

We developed a seven-activity collaborative PSA to support a preliminary phase of the business processes reengineering project the students learned in class. The PSA screens presented the participants with a simple user experience for each activity. On-screen guidance in and around the tools told the participants what kinds of ideas they should contribute and where they should contribute them. The tools were configured to support only these actions.

As with the COA development PSA, a banner on each screen set the context for the activity. A textual screen prompt just below the banner provided a one-sentence instruction that guided the users about what they should do. The bottom of each screen contained a set of buttons to pop up controls: (1) navigation buttons for moving from activity to activity; (2) a roster of team members; (3) an agenda for viewing the order of activities, for seeing which participant is on which activity, and for moving participants among activities; (4) a pop-up help panel containing a detailed script for the activity;
and (5) a recycle bin for recovering deleted contributions. The PSA was designed to support a small, egalitarian, potentially leaderless team structure. The PSA therefore had only one role, so all the participants had the same capabilities. Any participant could move the whole group to a different activity with the navigation button or the agenda control.

The activities of the process were (1) welcome, (2) brainstorm requirements for creative solutions, (3) develop creative solutions, (4) evaluate solutions, (5) elaborate the most-innovative solutions, (6) generate a report, and (7) respond to survey. The PSA had two administrative activities: welcome and report. It had four operational activities: brainstorming requirements, brainstorming innovative solutions, evaluating solutions, and elaborating the solutions deemed most innovative. Steps 2–5 required the participants to add and manipulate textual contributions. The seventh activity linked students to an online feedback survey that asked them about their experiences using the PSA. The process restrictions designed into the PSA made it unlikely (but not impossible) that users could use techniques other than those designed by the collaboration engineer.

The “welcome” activity was based on the overture collaboration technique [7]. It presented guidance to set the task context, explained the goals and deliverables of the project, and summarized the process. Participants could contribute no content during this activity.

The “brainstorm requirements for creative solutions” activity, which was based on the LeafHopper technique [7], encouraged the participants to deepen their understanding of their problem before they begin to generate solutions. The LeafHopper technique was selected because it encourages people to think in depth and detail by restricting their comments to a defined set of topics while allowing them to hop from topic to topic as dictated by their interest, expertise, and inspiration. In this activity, the PSA presented the users with five categories of requirements: (1) general, (2) actors, (3) boundary conditions, (4) technologies, and (5) required information. Clicking any category opened a comment page where participants could add requirements.

The “develop creative solutions” activity, based on the OnePage idea generation technique [7], encouraged participants to generate innovative solutions that can address their requirements. The OnePage technique was selected because (1) it encourages participants to contribute a broad range of ideas; (2) the groups were small, not likely to propose a large number of solutions to the problem, and so were unlikely to encounter a cognitive load limitation inherent in the technique; and (3) group members see one another’s ideas as they are entered, which can reduce cognitive inertia and increase creativity [47]. On the lefthand side of the screen, the participants could see the requirements they had generated in the previous activity. On the righthand side, they could see a list to which all the participants could contribute solutions simultaneously. The guidance encouraged the group to submit at least 30 ideas.

The “evaluate solutions” activity, based on the BucketVote evaluation technique and the BucketWalk clarification and reduction technique [7], encouraged a rapid, holistic evaluation of the solutions created in the previous activities. The BucketVote technique was selected because a fast approximation was sufficient for this phase
of the business process reengineering project. At this stage, the solutions were not expected to be sufficiently developed to warrant a point-by-point multicriteria analysis. The tool restricted the participants to one action—drag-and-drop ideas from a list on one side of the screen into a category (indicated with bucket icon) on the other side of the screen. The categories were (1) not innovative, (2) moderately innovative, and (3) highly innovative. The activity script provided the participants with a rubric for classifying proposed solutions by the degree to which they are innovative. The participants worked in parallel to evaluate the solutions by dropping them into buckets. The first participant who dropped a solution into a bucket determined its placement. The BucketWalk technique followed the BucketVote because participants are frequently not confident that their teammates have correctly placed ballot items in the buckets. When the solution list is empty, participants double-click a bucket to pop up a list of its contents. Participants discuss whether any items should be moved and occasionally moved one to a different bucket.

The “elaborate the most-innovative solutions” activity was a different instantiation of the LeafHopper technique [7]. It encouraged participants to add key implementation details to their most promising solutions; for example, which stakeholders should be involved, the processes they should follow, and the technologies they would require to implement the solution. The LeafHopper technique was selected because the team seeks to elaborate on existing ideas and because LeafHopper allows participants to start by adding details to the solutions where they have the most interest and expertise, and then circle back to review and fill in the other topics.

Measures

Successful Use by Nonexperts. The primary goal of this study was to test whether nonexperts could successfully execute an engineered work practice without training. To determine the extent to which the participants successfully executed the work practice, we examined their contributions to each of the four activities in the PSA that required their input. If the contents of the activity conformed to the terms of the assignment and with the guidance provided by the activity, the team was deemed to have completed the activity successfully. For example, the first activity required that students brainstorm requirements for a solution in each of five categories. The activity was deemed completed if (1) all the categories contained requirements, (2) the requirements were consistent with the category, and (3) the requirements were a credible response to the class assignment.

Work Product Evaluation. The course instructor provided feedback to the students on their project deliverables, which was another measure of success. The course instructor used a holistic evaluation method to assess the quality of deliverables for all the groups on a scale from 1 to 5, with 1 signifying “performance far below expectations” and 5 signifying “performance far above expectations.”

The participants who used the PSA were also asked to complete a 25-item exploratory survey about their experiences. Satisfaction is an important consideration when
fielding a new system because research shows that people who feel dissatisfied with
a new system, even for nontechnical reasons, tend to stop using it (e.g., [4, 50]), even
if they derive value from its use.

Satisfaction with Process. To measure satisfaction with the process, we used a five-item,
five-point Likert scale previously validated in two international studies of satisfaction
with technology-supported work practices [9, 46]. Cronbach’s alpha for the five-item
scale was 0.85. The items were as follows:

1. I feel satisfied with the way in which today’s meeting was conducted.
2. I feel good about today’s meeting process.
3. I liked the way the meeting progressed today.
4. I feel satisfied with the procedures used in today’s meeting.
5. I feel satisfied about the way we carried out the activities in today’s meeting.

Satisfaction with Outcome. To measure satisfaction with the outcome, we used a five-
item, five-point Likert scale validated by the same studies. Cronbach’s alpha for the
two-item scale was 0.92. The questions were as follows:

1. I liked the outcome of today’s meeting,
2. I feel satisfied with the things we achieved in today’s meeting,
3. When the meeting was over, I felt satisfied with the results,
4. Our accomplishments today give me a feeling of satisfaction, and
5. I am happy with the results of today’s meeting.

Tool Difficulty. We asked the participants about how difficult they perceived the
tools to be with a five-item scale of five-point semantic anchor questions. Interitem
reliability, as measured by Cronbach’s alpha for the five-item scale, was 0.84. The questions were as follows:

1. I was (uncomfortable/comfortable) with the tool we used today. (very uncom-
fortable, uncomfortable, neutral, comfortable, very comfortable)
2. The tool we used today were (difficult/easy) to use. (very difficult, difficult,
neutral, easy, very easy)
3. How difficult was it to figure out how the tool worked? (very difficult, difficult,
neutral, easy, very easy)
4. How difficult was it to figure out what we were supposed to do with the tool?
(very difficult, difficult, neutral, easy, and very easy)
5. How difficult was it to understand how the tools were supposed to support our
task? (very difficult, difficult, neutral, easy, and very easy)

Process Difficulty. We used five, five-point semantic anchor items to investigate the
participants’ perceptions of the degree to which aspects of the work process were easy
or difficult. The semantic anchors were very difficult, difficult, neutral, easy, and very
easy. Interitem reliability, as measured by Cronbach’s alpha for the scale, was 0.79.
The questions were as follows:
1. How difficult or easy was it to understand the group goals?
2. How difficult or easy was it to understand what deliverable we were supposed to create?
3. How difficult or easy was it to understand the process we were supposed to follow?
4. How difficult or easy was it to follow the process?
5. How difficult or easy was it to stay focused on the task?

**Adequacy of Guidance.** We used two exploratory semantic anchor questions to measure the degree to which the participants found the guidance provided to them by the PSA to be adequate. Interitem reliability, as measured by Cronbach’s alpha for the scale, was 0.81. The questions were as follows:

1. The instructions in the tool were (clear/unclear). (very unclear, unclear, neutral, clear, very clear)
2. The tool provided (insufficient/sufficient) guidance about what we were supposed to do. (very insufficient, insufficient, neutral, sufficient, very sufficient)

**Other Exploratory Questions.** We asked two other exploratory semantic anchor questions about participant’s experience of their collaborative work practice:

1. Our group (did not consider/considered) the multiple perspectives held by our members. (did not consider, considered few of, considered some of, considered most of, considered all of)
2. How difficult or easy was it to communicate with your teammates? (very difficult, difficult, neutral, easy, very easy)

**Open-Ended Questions.** We asked two open-ended questions about the PSA experience. The first question asked, “In what ways were the results of your brainstorm innovative or not innovative?” The goal of the question was not to measure whether the results were, in fact, innovative, but rather to explore the participants’ attitudes toward the work products they created using the PSA. The second question asked, “In what ways did the tool help or hinder you to come up with new ideas?” Again, the goal was not to measure whether the tools had actually helped or hindered, but to explore participant attitudes toward the PSA.

**Procedures**

The students formed teams of five or six people though self-enrollment. They were briefed in class on the problem and on the deliverables they were to create for the project. We created a new separate workspace with its own unique URL for each team. The participants received an e-mail with login instructions. The instructions explained how to log in, how to hand in the assignment, and the deadline. They did not describe the work process, the technology, or the collaboration techniques embed-
ded in the PSA. The students used their own computers and set their own times and
places to complete the work. At the end of their work in the PSA, they were asked to
respond to a survey to report their experience. One hundred twenty-eight PSA users
completed the survey.

Results

This section reports the results of the study to test whether the CSS solution
achieved its primary design goals and to explore the users’ experiences of their first
use of a PSA.

Nonexpert Execution of an Engineered Work Practice Without
Training

The primary design goal of the CSS solution was to package sufficient collabora-
tion expertise with collaboration technology so nonexpert users could successfully
execute the work practice for themselves with no training on either the collaboration
techniques or the technology. After all the students submitted their project deliverables
to their instructor, the researchers examined the contents of each team’s workspace
to determine the degree to which the groups had successfully completed each of the
four operational activities.

The PSA was composed of four operational activities that required participants to
think and contribute, and three administrative activities (welcome, generate a report,
and respond to survey). Each of the 26 groups was asked to attempt to execute the four
operational activities, for a total of 104 attempts. The groups successfully completed 97
of the 104 attempts, for a 93 percent completion rate. The 97 successful completions
were distributed among the groups as follows: 20 groups complete all four operational
activities in the PSA, five groups completed three of the operational activities, and
one group experienced technical difficulties after completing two of the operational
activities, and so completed the other two activities by other means.

Users’ Responses to Their First PSA Experience

There were no statistically significant relationships between any of the exploratory
measures and the sex or class level of the respondents. Furthermore, there were no
statistically significant relationships between any of the exploratory measures and
the number of activities a group completed in the PSA or the grades they received on
their project deliverables.

Table 2 presents the means and standard deviations for the participant responses
to each of the exploratory measures of their perceptions of their first exposure to a
PSA, and for the grades they received on their projects. We tested the hypotheses
that the measures were statistically significantly different than neutral. The resulting
t-statistics also appear in the last column of Table 2. Figure 4 illustrates the distribu-
tions of those measures.
The participants’ satisfaction with the process embedded in the PSA ranged from a score of 1 (very dissatisfied) to a score of 5 (very satisfied), but on average, the participants’ reports of satisfaction-with-process and satisfaction-with-outcome were not statistically significantly different than neutral. Likewise, their satisfaction-with-process ranged from 1 to 5, but was not statistically significantly different than neutral.

The participant reports of the degree to which their group considered the multiple perspectives of group members ranged from 2 to 5, with a mean of 3.32, which was statistically significantly higher than neutral (\(t = 4.37, \text{df} = 123, p < 0.001\)). Reports of the difficulty or ease of communicating with teammates averaged 3.55, which was also statistically significantly higher than neutral (\(t = 5.67, \text{df} = 124, p < 0.001\)).

The participants’ perceptions of the ease or difficulty of the PSA tools ranged from 1 to 5, with an average of 2.81, which was statistically significantly lower than neutral (\(t = 2.38, \text{df} = 125, p < 0.01\)). Participant perceptions of the difficulty of the work process likewise ranged from 1 to 5, but were not statistically significantly different
Figure 4. Distributions of Participants’ Responses to Their First PSA Experience
than neutral ($t = 0.28$, df = 126, $p > 0.16$). The participants rated the adequacy of the guidance they received in the PSA at an average of 2.59, also statistically significantly lower than neutral ($t = 4.65$, df = 125, $p < 0.001$).

The grades the participants received for their project deliverables ranged from 1 (far below expectations) to 5 (far exceeds expectations), with a mean of 3.11, which was not statistically significantly different than neutral (meets expectations).

Pearson’s correlation analysis revealed statistically significant relationships among some of the exploratory measures, as shown in Table 3. The analysis shows that, under the conditions of this study, satisfaction-with-process and satisfaction-with-outcome were strongly associated. Tool difficulty and process difficulty were strong predictors of the measures of satisfaction, and were strongly related to each other (tool difficulty and process difficulty scales ranged from 1 = “very difficult” to 5 = “very easy”). The easier people reported the tools and process to be, the more likely they were to feel satisfied with product and process. Guidance adequacy and communication difficulty were also predictors of the satisfaction measures and of the tool difficulty and process difficulty measures, but were not statistically significantly related to each other.

Our analysis also revealed a modest relationship between age and guidance adequacy. The higher one’s age, the more likely one was to give the guidance positive ratings.

Qualitative measures showed that some participants were positive about aspects of their experience, while others were negative. In response to the open-ended question about the degree to which the process gave rise to innovative or noninnovative ideas, the participants gave 50 positive responses, for example:

The results were innovative in the sense that they will lead to system optimization.

Some ideas seem to be very innovative and usable.

Different, sometimes surprising ideas.

However, the participants also made 20 neutral or negative comments, for example:

They were not very innovative.

They are not very innovative, but they can help with the rest of the process.

Not really innovative. It did help structure some existing ideas.

In response to the open-ended question about how the PSA helped or hindered the group to create more-innovative ideas, 46 gave positive responses, for example:

I think it gave everybody an outlet to give their opinion, quite helpful.

We had to produce a certain amount of ideas so it made us think out of the box.

The “drop-it-in-a-bucket” bit where we had to choose the most-innovative ideas was good.

Personally it helped me quite a bit, because I had the time to think about ideas.
Table 3. Pearson’s Correlations Among Exploratory Measures of Participants’ First PSA Experiences

<table>
<thead>
<tr>
<th></th>
<th>SP</th>
<th>SO</th>
<th>TOOLDIF</th>
<th>PROCDIF</th>
<th>GUIDADQ</th>
<th>COMMDIF</th>
<th>ALLPERSP</th>
<th>GRADE</th>
<th>AGE</th>
<th>SEX</th>
<th>GRPNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SO</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOOLDIF</td>
<td>0.551**</td>
<td>0.465**</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROCDIF</td>
<td>0.468**</td>
<td>0.366**</td>
<td>0.634**</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUIDADQ</td>
<td>0.473**</td>
<td>0.433**</td>
<td>0.738**</td>
<td>0.524**</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMDIF</td>
<td>0.302**</td>
<td>0.247**</td>
<td>0.224*</td>
<td>0.377**</td>
<td>0.083</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALLPERSP</td>
<td>0.100</td>
<td>-0.107</td>
<td>0.071</td>
<td>0.164</td>
<td>0.002</td>
<td>0.161</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRADE</td>
<td>-0.056</td>
<td>-0.047</td>
<td>-0.115</td>
<td>-0.135</td>
<td>0.003</td>
<td>-0.116</td>
<td>-0.008</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AGE</td>
<td>0.079</td>
<td>-0.094</td>
<td>0.175</td>
<td>0.146</td>
<td>0.205*</td>
<td>-0.053</td>
<td>0.064</td>
<td>0.025</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEX</td>
<td>-0.118</td>
<td>-0.089</td>
<td>-0.085</td>
<td>-0.159</td>
<td>-0.041</td>
<td>-0.093</td>
<td>-0.117</td>
<td>0.077</td>
<td>0.193*</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>GRPNUM</td>
<td>-0.182</td>
<td>-0.165</td>
<td>-0.081</td>
<td>-0.175</td>
<td>-0.055</td>
<td>-0.135</td>
<td>0.005</td>
<td>0.367**</td>
<td>0.062</td>
<td>0.150</td>
<td>1</td>
</tr>
</tbody>
</table>

Notes: The boldface items are those where a statistically significant correlation was observed. * p < 0.05; ** p < 0.01.
However, 26 participants made neutral or negative remarks, for example:

The tools did not help nor hinder us to come up with new ideas.

It didn’t help that much, it was a bit difficult to understand the whole thing.

It didn’t help me at all.

Not really helpful.

**Discussion**

In this section, we discuss the findings from our pilot studies and formal validation, discuss the limitations of this study, and outline directions for future research.

**On the Transferability of Engineered Work Practices to Nonexperts**

The primary design goal of the PSA solution was achieved. In contrast to the experiences GSS users reported over the past two decades, untrained PSA users were able to execute an engineered collaborative work practice for themselves without training on either the collaboration techniques or technologies. In this study, we achieved that outcome by (1) substantially simplifying the technology UI/UX to obviate the need for technology training, and (2) using technology configuration choices and on-screen guidance to invoke process restrictiveness—restricting users to actions and communications that would increase their likelihood of success and restricting users from actions and communications that could interfere with their chances of success.

It is important to note, however, that, in practice, process restrictiveness is neither a panacea nor poison. Restrictiveness creates value only when it (1) limits participants to useful behaviors that they need and want to perform, (2) restricts them from ineffective behaviors they do not want to perform, and (3) affords sufficient flexibility to accommodate the variety of conditions that typically emerge during execution of the work practice it supports. Restrictiveness would not be useful if it blocked participants from actions they needed to take. It would be possible to design PSAs that interfere with, rather than support, a group’s effectiveness. The skill of the collaboration engineer will therefore be a critical success factor for a PSA approach. Just as a facilitator works with a group to assure that the process restrictions embedded in a workshop design are useful to and welcomed by the group, the collaboration engineer will be required to attend carefully to the needs and interests of the prospective users of a PSA. The collaboration engineer’s task is made more difficult because, unlike the facilitator, the collaboration engineer will not be present to intervene if a group gets off track.

Process restrictiveness would likewise not be useful if it restricted participants from actions they wanted to take, whether or not the desired actions were productive, because users who wanted to take those actions could simply abandon the system in favor of other means. Thus, part of the design challenge for the engineer is not only to restrict users from ineffective behaviors but also to motivate the users to believe
that it is in their interest to avoid the proscribed actions. Much research remains to be done on how collaboration engineers can discover the needs and interests of a practitioner population as well as on how PSAs designs can be made sufficiently flexible to accommodate requisite variety, yet sufficiently structured to keep the group involved in effective and efficient thought and work, and sufficiently motivating that participants prefer them to other modes of work.

On the Creation of PSAs by Nonprogrammers

The CACE platform achieved its primary design goal. This study demonstrated that it is technically and operationally feasible to create an environment where collaboration experts who were not software programmers could snap together and configure collaborative capabilities to create new, task-specific collaborative software applications without having to write new code.

We demonstrated the generalizability of the solution by using the CACE to create applications for several different work practices, and by repurposing instances of the same collaborative components to manipulate different kinds of contributions (1) in the same tool, (2) in different tools on the same screen, (3) in different activities in the PSA, and (4) in different PSAs. We likewise used the same components and controls to (1) deliver different guidance in different contexts, (2) support or restrict the same actions differently in different parts of the same PSA, and (3) support different work practices for different collaborative tasks in different PSAs.

In this study, we also demonstrate that it is feasible to create a PSA using a CACE in hours or days rather than months or years without having to write new code. The approach may therefore be useful for supporting teams working in volatile, rapidly changing environments without invoking undue development costs.

If a collaborative work practice were to be used by one group on one occasion, then a professional facilitator might be a less expensive option than a PSA, because the design and development of a PSA requires more time and effort than facilitating the same work practice one time. If, however, a collaborative work practice were to be used on many occasions, possibly by many groups, then the PSA solution might be less expensive per session than the professional facilitator, because the collaboration expert would be required only at PSA design and development time, but the benefit of that effort would be derived repeatedly.

Satisfaction Responses

Average reports of satisfaction did not differ significantly from neutral in this study. The yield shift theory of satisfaction [10] suggests an explanation. It proposes that satisfaction responses are not a function of goal attainment but, rather, of shifts in the utility one ascribes to attaining a goal and shifts in the likelihood one ascribes to attaining a goal [10]. A neutral response would signify that a process or outcome did not alter people’s perceptions of the likelihood or utility of goal attainment—participants, on
average, got what they expected and so, on average, did not experience a satisfaction response. The average of the satisfaction measures, however, disguise the fact that some of the participants reported themselves to be deeply satisfied with the experience, while others reported themselves to be deeply dissatisfied (see Figure 4a and 4b). Several groups reported technical difficulties with the prototype system, which would only function properly in one kind of browser. Under some conditions, the system had slow response times. Such incidents may account for some of the negative comments and dissatisfaction reported by some of the participants.

The strong associations between satisfaction and the measures of difficulty also provide clues as to why some were satisfied and some were not. Those who found the tools and process more difficult also tended to report lower satisfaction that those who found them easy. Likewise, those who found the guidance adequate were more likely to report higher satisfaction responses than those who found it inadequate. More detailed studies would be useful to discover and improve the aspects of the tools and process that led some users to report that the tools were difficult to use.

The observed differences by age in perceptions of guidance adequacy suggest that it may be necessary to provide different user experiences to different stakeholder groups. More detailed research is warranted to discover what aspects of the process and tools the users find difficult and to derive design guidelines and best practices for addressing such difficulties.

Under the conditions of this study, we observed a strong association between satisfaction-with-process and satisfaction-with-outcome. Reinig [44] reported a similar effect and suggested that such a relationship might exist because people may feel satisfied with a process to the extent that it produces a good outcome. There are, however, plausible conditions under which people could be dissatisfied with the process but satisfied with the outcome and vice versa. A painfully inefficient legislative process, for example, may nonetheless produce an equitable law, and a sound medical diagnostic session may nonetheless not produce a useful diagnosis. Further exploration of this effect under other conditions may therefore be useful.

It is interesting to note that participants’ perceptions of the degree to which the group considered all the multiple perspectives of its participants was observed to be unrelated to all the other exploratory measures. One could conjecture a ceiling effect, given that this was the most positive of the measures. However, as Figure 4 shows, there was a wide range of responses to this measure, yet these variations did not correlate with the satisfaction measures. One possible explanation is that the participants were able get a satisfactory result even when they did not perceive that the group took their perspective into account. More research will be required, however, to better understand the dynamics of this construct in collaborative work.

Limitations and Future Directions

It was not the goal of the validation study to demonstrate that users of PSAs work more efficiently or effectively than unsupported groups, nonexpert GSS users, or facilitator-supported groups. Rather, the goal was to demonstrate that with careful UI/UX design
and carefully implemented technology-based process restrictiveness, nonexperts could successfully execute a well-designed technology-supported collaborative work practice without training on either the techniques or the technologies. That goal was achieved. Having demonstrated that point, however, it would be valuable for future exploratory and experimental research to pursue the follow-on questions:

- Can PSA users outperform unsupported groups, and if so, under what conditions?
- Can PSA groups outperform untrained GSS users, and if so, under what conditions?
- Can PSA groups obtain results comparable to those of groups led by professional facilitators, and if so, under what conditions?

It would be useful to focus such studies on a variety of phenomena of interest to teams, among them, project cycle time, labor hours, quality of work products, satisfaction with processes, and satisfaction with outcomes.

This study reported the successful transfer of relatively simple work practices to small groups of stakeholders with relatively low stakes under conditions of low risk. Follow-on research would be useful to explore the PSA approach for more-complex, higher-stakes work practices in conditions where higher risks prevail. The formal validation study was conducted among stakeholders from a single domain. It will also be useful to develop and explore the use of PSAs with other kinds of users in other kinds of organizations. This research also explored the use of a single medium to instantiate user guidance—text. Further research would be useful to explore whether other media or mixes of media could be used in ways that enhance the user experience and improve user effectiveness.

The proof-of-value prototype presented here does not yet have an extensive set of collaborative components. Polling, graphical, and streaming capabilities (e.g., voice, video, desktop sharing), and data access tools (e.g., shared document repositories, social tagging, syndication), for example, are not yet available. The array of work practices that can be supported by the present system is therefore limited. It will be useful, therefore, to create additional components to add to the system as researchers encounter new needs among the practitioners with whom they work.

Conclusions

This research completed multiple instances of the DSR relevance, rigor, and design cycles over a number of years. It completes its final rigor cycle by contributing back to the knowledge base a description of an important class of unsolved practical problems, a collection of general requirements for solutions to the problem, a generalizable approach to solving the problem (PSAs), an expository instance of a solution for the problem (CACE and PSS packaged in a CSS), and the first evidence that nonexperts could successfully execute an engineered collaborative work practice with no support from an expert facilitator and no training on the collaboration technology or techniques. The general class of problems that motivated this research, however, is not yet fully
solved. Process restrictions are not the only value that an expert facilitator brings to a group. An expert facilitator also monitors the interactions of a group continuously to diagnose emergent issues of communication, reasoning, information access, goal congruence, and distraction, and intervenes to change the process to address these issues. Thus, more valuable research remains to be done. That said, however, the research reported here shows that the facilitator is now at least standing in the box, and with further research, may eventually find a home there.

REFERENCES


21. Hardin, A.M.; Fuller, M.A.; and Davison, R.M. I know I can, but can we? Culture and efficacy beliefs in global virtual teams. Small Group Research, 38, 1 (2007), 130–155.


