Tycho: Facilitation Support for Groupware User Tests

Julián Grigera[1,2,3], Juan Cruz Gardey[1,2], Alejandro Fernandez[1,3], Alejandra Garrido[1,2]

1 LIFIA, Facultad de Informática, Univ. Nac. de La Plat, 50 y 120, La Plata, Argentina
2 CONICET, Argentina
3 CICPBA, Argentina
{julian.grigera, jcgardey, alejandro.fernandez, garrido}@lifia.info.unlp.edu.ar

Abstract. Running user tests for groupware requires tracking the progress of multiple users performing co-dependent tasks, while facilitating and observing their coordinated actions. This requirement negatively impacts the main objective of user testing, which is detecting usability flaws. User testing of groupware becomes more challenging when run remotely. Even if there are tools for remote user testing that can get the job done for a single user, they are not prepared for collaborative scenarios. In this paper we argue that tool support for the facilitation of user tests of groupware is missing. Consequently, we propose a method for user tests with tool assistance that makes it possible to automate task synchronization, especially for different workflows that must be run concurrently. We evaluated our proposal by comparing it to a manually facilitated approach during on-site coordinated user tests. The results indicate that, while the task of designing and running user tests with tool support takes more time, it allows the testers to better focus on detecting usability problems.

Keywords: Groupware Evaluation, User Testing, Automated Testing.

1 Introduction

User testing is the most popular usability evaluation method, since it allows to capture real usage data [1]. However, when the system under evaluation involves more than one user performing coordinated tasks, user testing turns increasingly complex.

In today's web, many applications include features to support collaboration among users - what is known as groupware. One of the challenges of running user tests of groupware systems is facilitating many users at once, which can be especially hard if the tasks they must fulfill are different from each other. Facilitation of synchronized tasks alone is already cumbersome and may lead to participants' confusion (*"should I go on, or wait for the others?"*), boredom or frustration.

In typical user tests, volunteers perform a series of tasks specifically designed to identify such issues. When the user test is run on-site, a usability expert takes care of observing and recording performance information that can later be interpreted in terms of efficiency, effectiveness and satisfaction. But in the context of groupware systems, the usability expert must also serve as facilitator, coordinating the tasks among test subjects. The more effort the facilitators spend in guiding volunteers, the less attention they can pay to usability.

To illustrate the complexities of groupware user tests, consider MediaWiki (Wikipedia's wiki engine). MediaWiki supports collaborative, in-place editing of web-pages. We could run a test for the event of edit conflicts during concurrent page editing, involving two users (U1 and U2), a test designer, and a facilitator, facing

The test designer devises the following plan for the tasks of participants:

U1 and U2 should first watch a short video that explains how to edit MediaWiki pages. Then, both should navigate to the same wiki page (the facilitator will indicate which one, let's call it P). U1 should edit the page (i.e. switch to edit mode). While U1 is in edit mode, U2 should edit the page and make a few changes. Following, U2 should save the page. Then, U1 should attempt to save the page. Instead of the updated page, U1 will be presented with a conflict resolution screen. U1 should watch a short video that explains how conflicts are resolved. Then UI should try to resolve the conflict, and save the page.

Fig. 1 provides an overview of the tasks each participant must complete. Vertical bars show important synchronization points.

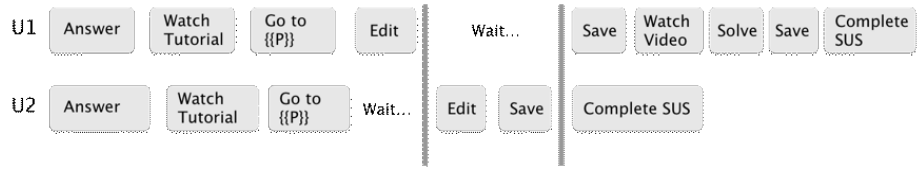

Fig. 1. Schematics for a coordinated user test with meeting points for synchronization.

During the test, the following information should be captured: demographic data for each user, time spent in each task, and a Software Usability Scale (SUS) score from each participant, regarding the tool.

The facilitator may repeat this test with pairs of remote users, randomly assigning them to take the role of U1 or U2, making sure that they all receive the same instructions (depending on the role), and that their tasks are synchronized as required by the scenario. Hence, the test protocol should be clearly documented (tasks, instructions, synchronization, etc.) to foster reproducibility. These complexities could be greatly reduced with the help of a specific method and a supporting toolset,

Groupware applications commonly address distributed scenarios, where users are remote, and running remote user tests requires tool assistance. There are already tools for remote user testing that provide instructions for the task to the test participants, while automatically logging task performance measures [2]. This way, two of the main duties involved, i.e., facilitating tasks and recruiting volunteers, are greatly streamlined, at the cost of losing face-to-face interaction between facilitators and participants. However, to the best of our knowledge, none of these tools can synchronize or facilitate a collaborative task that requires coordination among subjects. Also, in order to compare results of multiple test sessions for the same user interface (UI), reproducibility is key. Test plans and test results must be documented in a way that ensures that the test results of independent groups are comparable.

In summary, user tests of groupware pose challenges in terms of clearly directing the coordinated actions of volunteers (potentially distributed), in terms of observation

and data collection during tests, and regarding reproducibility of tests. We argue that tool support for the facilitation of user tests of groupware is missing, and if available, it needs to overcome those challenges. Our hypotheses are:

1. Tool driven test facilitation reduces the stress of facilitating groupware user tests.
2. Tool driven test facilitation improves effectiveness of participants following instructions, especially in groupware
3. Tool driven test facilitation improves precision in the measurement of task completion time
4. Tool driven test facilitation improves precision when capturing task completion results
5. Tool driven test facilitation fosters tests reproducibility

We developed a tool that helps researchers design and run remote tests, which includes synchronization capabilities with the use of semaphores. Using it, usability experts are able to create different task models for different roles, and remotely run the tests. The tool guides all test participants in a coordinated way and, at the same time, collects information on how the tasks are completed in terms of efficiency, effectiveness and satisfaction. This can also be useful for on-site tests, given that the automated facilitation can greatly relieve the involved staff who can concentrate more on the observation, as stated in our first hypothesis. Since the tool stores the tests designs and results, it is easy to reproduce the tests and compare their results for improvements. Moreover, it is possible to export and share test designs, so they can be reproduced in different settings.

2 Related Work

Our work relates to two existing and mature research and practice areas: user testing, and groupware evaluation. We first clarify our use of groupware related terminology.

2.1 Background on cooperation, collaboration, and groupware

The terms cooperation and collaboration are nowadays frequently used in literature, both in academic and non-academic writing. They are used flexibly, sometimes interchangeably, assuming a commonsense interpretation on the part of the reader. However, using these terms with a clear, concrete, and well bounded interpretation is of paramount importance when the discussion focuses on the interactions among persons, and the mediating role of the technology.

J. Bair provided one of the first detailed definitions of the terms. In his work on support of cooperative work with computers [3] (which was later reproduced by Borghoff [4]), Bair analyses cooperative work in terms of the needs and intensity of task-focused communication of a team or group. He defines four levels, in a continuum of increasing intensity of communication: information, coordination, collaboration, and cooperation. While Bair defines collaboration as “working together”, he defines cooperation as “working or acting together towards a common purpose” (*common*

purpose being the key difference). For Bair, collaboration is characterized as participation in the same process, with potentially unequal involvement. Although there may be a common output, each collaborating individual is evaluated independently. The frequency of interaction among participants can vary widely depending on the parts of the process they participate in. In contrast, Cooperation is characterized by a sublimation of the goals of the individuals in favor of the goals of the team. Decisions are made by consensus, with frequent interaction. The team is evaluated as a whole, and the results are owned by the team. Other authors agree with Bair's characterization of both classes/categories, but not on the terms used to name them. For example, recent work by Collazos et al. [5] defines collaboration in the way Bair defines cooperation, and vice-versa. Such usage of the terms is common in the CSCL community. In this article, we adhere to Bair's original usage of the terms, and we focus on the collaboration area of the continuum.

The term groupware has traditionally been used in academia to refer to the software component in CSCW. However, the original definition of the term is more comprehensive: "intentional group processes plus software to support them" [6]. In fact, the level to which the intentional processes are encoded in the software varies; software can act as a "medium" that enables the implementation of the processes, or as a "mechanism" that enforces the processes and makes groups work [7] The vision of groupware in connection to explicitly modeled processes (i.e., graphs of dependent tasks, users assigned to tasks, flows of information among tasks, etc.) was fostered by the early work of Winograd and Flores [8]. It is central in BPM related approaches, and also to collaboration engineering [9]. The work reported in this article is strongly influenced by this view of collaboration as a process. That is, in the context of this work, usability testing of groupware is understood as *collaboration (involving the volunteers, and the tester) that can be modelled as a process*.

2.2 Usability Testing

Out of the different methods for usability assessment, the most popular one is user testing. In user testing, a usability expert observes real end-users while they perform typical tasks on a system [1]. The expert records the usability problems that participants run into and may also interview them to get a subjective satisfaction score. While the tasks are usually stated in a loose way to prevent participants from being biased, experts make a considerable effort at guiding them through the tests. In fact, the tests guide (a.k.a. facilitator) and the observer are ideally different people, so the task of detecting usability problems is not interrupted [10].

Usability test facilitation refers to the interaction between the test participants and the facilitator of the test session. It involves preparing the session in advance, meeting and greeting the participant, briefing the participant, learning about the participant's skills and background, conducting the test, and finally debriefing the participant. Mollich and Wilson addressed some of the common mistakes researchers make during usability testing in a Special Interest Group they conducted at CHI 2008 [11]. They reported, for example, that too much talking between the facilitator and the participant might turn a test session into a directed interview. The facilitator can ruin the test session by introducing unnecessary cues to the participant, by discussing with the note

taker, or by letting one of the participants in a co-participation session become dominant.

Test facilitation is especially challenging in the context of remote user tests. Andreasen et al. [12] made an evaluation of three different remote user testing methods with respect to the traditional user testing method in a lab setting. They compared a remote synchronous condition (the test is in real time, but the user and the evaluator are separately spatially) and two remote asynchronous conditions where the evaluator and the subject are separated both spatially and temporally. The results show that the usability problems found in the lab setting are very similar to those found in the remote synchronous condition. On the other hand, there are substantial differences between the lab setting and the asynchronous conditions, where the majority of the usability issues discovered by the lab setting were not found in the asynchronous conditions and, in these cases, the time required to complete the task was higher. Our work can directly relate to these findings, since on-site, tool-assisted facilitation is very similar to remote synchronous testing.

2.3 Groupware Evaluation

Groupware evaluation is a challenging task, as it seeks to draw conclusions not only regarding properties of the groupware tool itself and the user's stance towards it (i.e., usability), but also about the tool's impact on the real work setting it attempts to support. In the year 2000, almost 20 years after the inception of the CSCW field, Pinelle and Gutwin conducted a review of CSCW articles published from 1990 to 1998 [13]. They observed that there was no clear consensus on which methods were appropriate to evaluate groupware. Almost one third of the articles they considered had no evaluation. Pinelle's work was later updated by Wainer and Barsottini to account for the changes that occurred in the period from 1998 till 2004 [14]. They reported a significant decrease in articles with no empirical evaluation, and a significant increase in Field experiments, and quantitative evaluation. The use of ethnographic methods (descriptive, qualitative) to understand work practices was found to be still dominant. A growth was observed for explanatory, hypothesis testing research. The most common types of evaluations during the observed period were lab experiments, followed in frequency by field experiments, and field/case studies. Later, in 2017, Wallace and colleagues [15] updated the review to account for the contributions made until 2015. They observed a continued decline in the frequency of non-empirical research; explanatory research peaked in 2008-2011 and then declined; descriptive research increased and now represents 50% of the articles. They also observed an increase of the combined use of quantitative and qualitative methods in the later years.

According to these reviewed articles, real work evaluations (organizational impact, or impact on work practices) are time consuming. Studies that conducted this type of evaluation reported to have invested from 4.5 to 36 months of time (observing the impact of the system). Early formative evaluations (with a prototype, in lab settings) can help get rid of larger problems. Later, longer summative evaluations (in case or field studies) can focus on impact (and more subtle aspects) on the basis of a more robust system. Early evaluations aim at supporting larger corrections in the approach (of problems that would be too obtrusive to make valid observations of impact),

whereas later evaluations focus on assessing impact in real work situations. Laboratory experiments are more frequent in the early stages, whereas field studies and case studies are more frequent in the later stages.

A review of evaluation methods for groupware systems was performed by Herzkovic et al. [16]. The study classifies the existing methods based on the stakeholder (developer, organization, etc.) who carry out them and the state of the system (under development or finished). In their work, the authors claim that very few groupware systems perform formal usability evaluations, although some efforts were made. A simpler alternative to multi-user tests consists in evaluating a single user at a time Khan [17], which is a much more basic setup than running with all users at once, but it requires extra resources (at least one tester per user) and it is more complex to coordinate if users are at different locations.

Another approach to assess groupware usability consists in inspection methods. However, standard inspection methods are not well suited for evaluating groupware systems because these methods are intended for single-user applications where an evaluator 'inspects' how a user performs a task individually. Usability evaluation methods for groupware must also consider teamwork, that is, the actions needed to complete a task as a group. Nielsen's heuristics were adapted to address the teamwork when assessing groupware system usability [18]. The heuristics are defined on the basis of the mechanics of collaboration [19], a set of core actions that happen in every shared workspace, such as communicating with other members of the group or negotiating access to shared resources.

Evaluating a system's usability through an inspection method requires the modeling of the tasks that the users will perform in order to discover true problems and successes in a user interface. Pinelle et al. [20] developed a task-modeling technique called CUA, specific for the evaluation of groupware systems. The technique allows the definition of collaborative tasks using the mechanics of collaboration, and individual tasks to specify the steps that group members carry out individually. Considering this framework, in a later work the authors proposed a usability inspection method for groupware called groupware walkthrough [21]. It is based on CUA for identifying and analyzing collaborative tasks and a walkthrough process to determine the system's support for the realization of those tasks. The walkthrough process consists of an expert analyzing how the tasks of a scenario are carried out by a group member. For each task, the evaluator identifies how it was done and the usability issues. At the end of the task, some questions are asked to the group member to identify the task's effectiveness, efficiency and member's satisfaction.

Articles that focus on usability evaluations of collaborative tools, both in the lab and in real work settings, frequently report having using screen recording, and screen sharing software as means to monitor and analyze interaction. For example, Geszten et al. [22] used OBS (Open Broadcast Software) to record on-screen activity and voice, while monitoring Skype communications among experiment participants located in different rooms. Instructions were provided at the start of the experiment, in the form of a single objective that participants had to complete (while self-organizing). Observers had to observe and record communication breakdowns and usability problems. While such an approach is adequate for summative evaluations, it does not cover the need of facilitating specific, detailed collaboration scenarios (such as a specific interaction, in a given state of the system).

A recent work by Bringas et al. [23] proposes a framework to identify sequences of tasks that have usability problems. To use the framework, evaluators first create a model of the available tools (or functions), and the actions and tasks that users can perform. Participants use the system under test and complete a usability survey. Then, the logs of the system are processed to match the model. A neural network is trained with the survey results and the processed logs to identify problematic sequences. This work focuses on post-mortem analysis of usage logs, while our work focuses on moderating and guiding user interaction during evaluation. Although the authors do not provide much detail regarding the changes one must make to a tool so it can be evaluated with the proposed framework, it is clear that it must provide usage logs that are rich enough. Our tool, on the contrary, poses no requirement or restriction on the system under test (other than being a web application). We believe that both approaches could be combined, our tool to moderate and to generate rich usage logs, and the framework proposed by Bringas et al. to analyze logs.

3 Approach Overview

To overcome the challenges of groupware user tests, we developed Tycho, a tool for digital facilitation of user tests. It consists of three core elements: a test protocol modelling language; a tool to support the creation of protocols for user tests; and a two-part tool to administer the tests protocols and collect results.

A test protocol describes the tasks that a test participant must perform. A protocol, for example, can present a video introduction that the participant must watch (e.g. a tutorial on how to edit a wiki page), ask the participant to navigate to a web-page and do something with it (e.g., edit the wiki page), and then ask the participant to complete and submit a SUS questionnaire. Protocol designers can freely combine tasks created from a palette of task prototypes. Tasks in a protocol are organized in a sequence. Only when a task has been completed, is the next one in the sequence made available to the participant.

User testing of groupware tools frequently requires coordinated actions. For such scenarios, the protocol modelling language includes the concept of **named semaphores**. One protocol (executed by one participant) waits on a semaphore that another protocol (executed by a different participant) signals.

Facilitators may need to introduce variable elements into their protocols. For example, to deal with generalization, a protocol might be conducted on various pages of an application (e.g., various wiki pages, various products in an e-commerce site, etc.). For those cases, the modelling language offers variables, which can also be used to share small pieces of information among participants (e.g., the URL of a page that one participant created, and another participant must edit)

The Tests Designer module eases the creation of **protocols** and **sessions**. A session represents a workspace where one or many different protocols are defined and used. Semaphores and variables belong in a session and are shared by all protocols within it. The Tests Designer ensures only valid configurations are used and assists facilitators in correctly configuring protocols. Protocols can be cloned and used as the basis for newer ones.

Managing tests involves two components: a browser add-on (the Facilitator), and a monitoring server (the Monitor). The Facilitator component runs in the participant's web browser. It obtains the protocols from the server, guides the subject, monitors events, and reports results and observations to the server. The Monitor is a frontend for testers that allows them to track the execution of tests, manually set semaphores and variables, and collect reports and observations. Test results and observations can be exported in JSON format for further analysis in other tools (such as R, Python, SSPS, etc.).

A distinctive aspect of Tycho is that it transparently binds itself to the web-applications used during tests in a web-augmentation style [24]. It can detect and record low level events such as clicks, form submits, and navigation. These events then reported on a task-by-task basis. Tycho does not require changes to the application / website under test. Moreover, Tycho takes control of all of the browser's document windows (i.e., the area of the web-browser where web-documents are rendered), including tabs, thus monitoring tasks that span windows. Fig. 2 summarizes the approach and supporting tools/components.

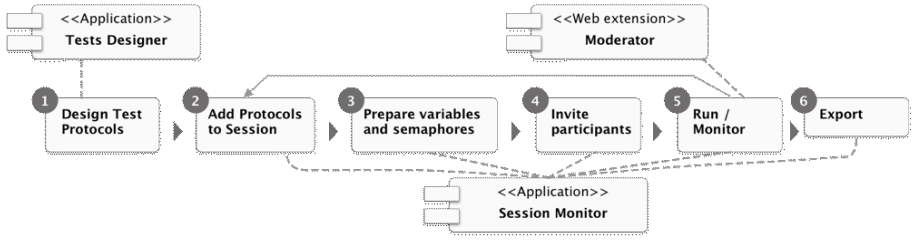

Fig. 2. Proposed user test definition process (in the middle), and supporting tools offered by Tycho.

Tycho's architecture is rather straightforward. The main component is a web application implemented in Pharo Smalltalk, which has an underlying business model layer (with MongoDB for persistence). On top of this layer there is, on one hand, the frontend where the Tests Designer module is implemented, and on the other a RESTful API that communicates with the facilitator browser add-on. This API, together with some underlying business logic constitutes the Monitor. The browser add-on is implemented as a web extension, with versions for Google Chrome and Mozilla Firefox browsers.

3.1 Protocol Models

The tests designer is available as a web application. It allows test designers (e.g., UX experts) to create user tests protocols. Each protocol can be composed of different tasks and defines a path for volunteers to follow. This path includes different activities, which range from simple messages to self-evaluating assignments. There are also special activities for task coordination. Currently, the following elements can be used as part of protocols models:

Message screen: Blocks all browser windows (i.e., web content is not visible, and interaction with web content is not possible). Presents a message. Offers a button labeled that marks the task as finished. Records the time elapsed since the message was presented until the task was marked as finished.

Basic task instructions: Provides instructions for a task. The participant uses buttons, labeled "start", "pause", "abandon" and "finish", to control the activity. When the task is paused (or still not started) all browser windows are blocked. Instructions are available at all times and do not interfere with the participants' experience. The designer can include a script to be evaluated when the participant marks the task as finished. Reports the time elapsed from start to finish (or abandon), not counting the time in pause. The result of the script (currently JavaScript) is reported. The designer can hide the and buttons.

YouTube video: Blocks all browser windows. It plays a YouTube video with video controls. Offers a button labeled that marks the task as finished. Records the time elapsed since the video was presented until the task was marked as finished.

Simple questionnaire: Blocks all browser windows. Presents the participant with a series of questions defined by the protocol designer. Offers a button labeled that submits the participant's responses, and marks the task as finished. Records responses. Records the time elapsed since the questionnaire was presented until the task was marked as finished.

SUS questionnaire: Blocks all browser windows. Presents the participant with a Software Usability Scale (SUS) questionnaire. Offers a button labeled that submits the participant's response and marks the task as finished. Records the response and the computed SUS score. Records the time elapsed since the questionnaire was presented until the task was marked as finished.

Wait for semaphore: Blocks all browser windows until a semaphore is signaled. Presents the participant with a message followed by a configurable description of the reason for the wait. When the corresponding semaphore is signaled, the window is unblocked, and the task is automatically marked as finished. The semaphore can either be signaled by a Signal semaphore task (see below) or manually via the Test Monitor. Records the time elapsed since the message was presented until the task was marked as finished.

Signal semaphore: Signals a named semaphore, and marks the task as finished. Nothing is presented to the user. Records the time when the task was marked as finished (i.e., when the semaphore was signaled).

3.2 Preparing a Session

Once the designer has completed the design of the required protocols, a facilitator (which might not be the same person) prepares a test session. The facilitator selects protocols from a list of available protocols and adds them to a new session. If any of the included protocols depends on variables or semaphores, they are automatically added to the session, and can be set before running the test.

The tool generates a unique access handle for each protocol in the session. The facilitator will then provide the handles to the participants that will take part in the session.

Once the facilitator starts the session, participants can use their handles to take part in the test. During the session, the facilitator can monitor progress, and interact with the variable elements in the session (for example, the facilitator can signal semaphores).

3.3 Facilitation Component

To join a test, participants use the Facilitation Component. This component is implemented as a web extension that can be installed in Google Chrome or Mozilla Firefox. Once installed, the participants are prompted to enter the access handle, which is generated by the UX experts in the tool previously described. This allows them to join the user test with a specific role (protocol), although they are not necessarily aware of the existence of other roles, depending on each particular test's needs.

Once a participant joins the test, the component starts guiding them through the activities, at times asking for data, blocking the screen to capture the attention by showing a message, or halting the execution if a semaphore must be waited upon. It is usual to end a session with a questionnaire to capture perceived satisfaction. After each task in the protocol is finished, the component sends all captured data to the server.

Fig. 3 shows a screenshot of Tycho's Protocol Designer and Facilitator add-on component, showing a sample protocol based on the motivating example.

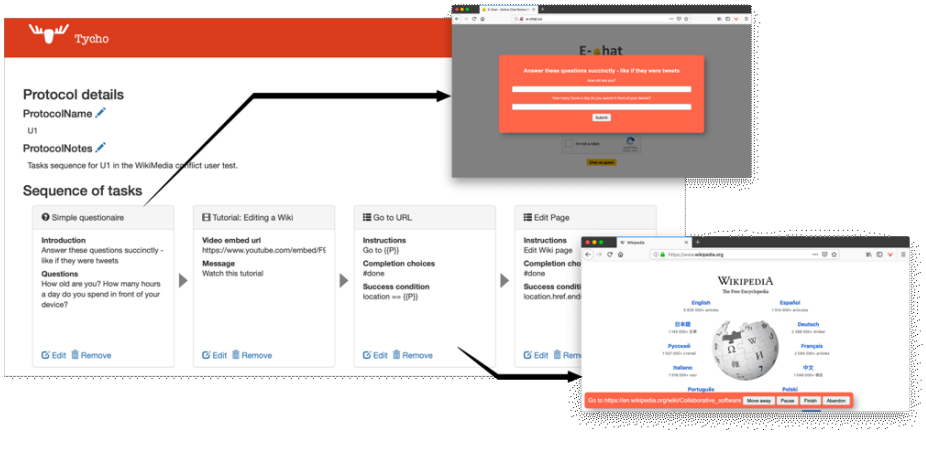

Fig. 3. Screenshot of Protocol Designer on the left. The displayed protocol corresponds to the MediaWiki example for user U1. The smaller screenshots on the right show how the add-on component guides the participants through the related steps – the top screenshot shows a Simple Questionnaire, and the bottom one shows a Basic Task Instructions component.

4 Evaluation

In order to evaluate the proposed approach and its alleged benefits, we ran a study with real participants. The main goal was to investigate whether the tool support for facilitation improves the overall experience of testing groupware activities, so we focused on comparing manual facilitation vs. tool-assisted.

4.1 Evaluation Design

The validation had 2 types of users: testers and volunteers. The testers were in charge of designing the protocols for the user tests, facilitating (i.e. guiding the volunteers through the steps, either with or without tool assistance) and detecting usability flaws during the tests. The volunteers, on the other hand, were the end-users participating in the user tests. The validation was organized as follows: all testers ran 2 rounds of user tests each. In one of the runs, they facilitated the test manually, and in the other with tool support (half the testers started with the manual test, and the other half started with the tool-assisted test). On each round they coordinated a collaborative task involving 2 volunteers as end-users, who had to fulfill tasks that depended on each other's actions, such as “wait for your partner to create a document” or “share with your partner the URL you just generated”.

In order to reduce potential biases, the application used by the volunteers was changed from round to round. The two applications were an online chat platform (called E-Chat) and Google Drive (aka GDrive). Fig. 4 illustrates the organization of a sample pair of rounds.

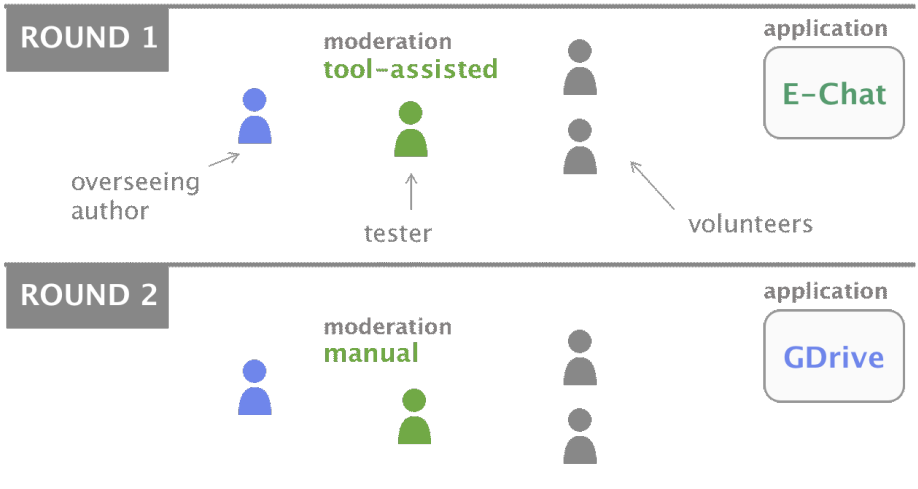

Fig. 4. Sample round of the study. A group of subjects (i.e. tester and volunteers) generates 2 samples, alternating the variables: order, facilitation mode, and application.

Each user test was additionally overseen by one of the authors, in order to capture different aspects of the user test, such as the way the testers delivered the instructions to the end-user volunteers or how much time was spent in experiment design and later facilitation, and how much in trying to detect usability flaws during the test.

The tasks were predefined and provided to the testers, but only vaguely specified so they had to design the specific steps by themselves. In the case of E-Chat, the volunteers had to create a chat channel and have a brief interaction, while in the case of GDrive, they had to create and share a document and then both make a small edition.

During each test, the overseeing authors captured the total time of the tests, including design time, effectiveness of the tests (i.e. whether they were run successfully, independently of the results on the subject applications themselves), and satisfaction of the experience of running the coordinated user test. This would allow us to compare the performance of the manual facilitation vs. the tool-assisted option.

The complete list of data to capture from the testers was:

- Time taken by the testers to design the protocols (both manual and assisted runs)
- Time taken by the testers to run the tests.
- Testers perception on attention load and satisfaction for the task of running the test. This was captured by a short survey of 3 questions with a 5-point likert scale (detailed next).
- Confidence in time tracking for volunteers' tasks completion times.
- Number of usability problems detected by the testers.

The data collected from the volunteers was:

- Effectiveness, efficiency and satisfaction, captured as number of finished tasks, time taken to complete them and SUS questionnaire, respectively. This data was captured by the testers.
- Volunteers' perception on how they felt during the test regarding the guidance they received. This was captured by a survey of 6 questions with a 5-point likert scale (detailed next).

Regarding the testers' view on the experience of single-handedly running a coordinated user test, the data was gathered by a survey with the following assertions, where the value 1 was "totally agree" and 5 was "totally disagree":

- TA1: "I had difficulties to get people to follow the instructions." (1 is worse)
- TA2: "I was able to pay attention to usability issues." (1 is better)
- TA3: "The task of facilitating the test made it difficult to take notes." (1 is worse)

Regarding the volunteers' view on how the tests were run, the survey included the following assertions. Again, the values ranged from "totally agree" (1) to "totally disagree" (5):

- VA1: "I knew what I was expected to do at all times." (1 is better)
- VA2: "Being observed made me feel uncomfortable." (1 is worse)
- VA3: "I was able to clear out any doubts I had." (1 is better)
- VA4: "I know what information was captured during the test." (1 is better)

- VA5: “Instructions were clear.” (1 is better)
- VA6: “The facilitator’s indications and questions interrupted me or made me lose my focus.” (1 is worse)

It should be noted that both surveys were designed in the style of SUS questionnaires, where positive assertions are intertwined with negative ones. This prevents users from mindlessly filling a uniform score for all questions, either high or low. In the results, however, this was normalized by reversing the scores of the negative assertions, in order to make comparisons possible.

4.2 Results

We ran a total 12 tests with 6 testers (all male, ages $\bar{x}$=29 =11.66) and 12 volunteers (3 female, ages $\bar{x}$=42 =8.62), 6 tests were run with manual facilitation and 6 with the assistance of Tycho. The time it took the testers to design their protocols was, on average, 7’30’’ for the manual tests and 18’40’’ for Tycho. The user tests themselves took, on average, 5’30’’ for the manually facilitated and 8’20’’ for the ones facilitated with tool assistance.

According to the results of the tester’s view surveys, the tool-assisted guide was rated better in all 3 assertions. In the volunteer’s perspective survey results, each technique got a higher score in half (3 out of 6) the assertions, although the averages were quite close in the comparison. The results for tester and volunteer perspectives can be seen in Fig. 5 and Fig. 6 respectively.

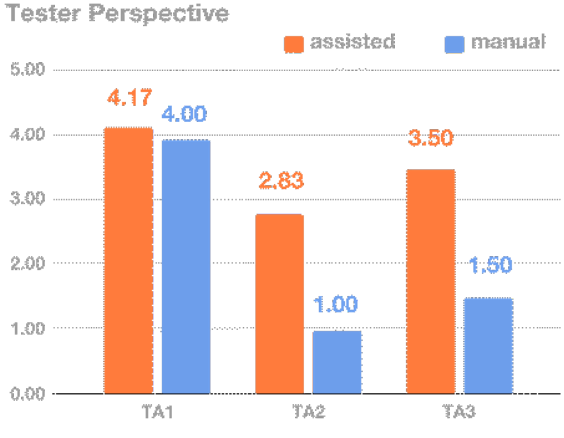

Fig. 5. Results of the survey comparing assisted vs. manual facilitation according to the testers’ perspective.

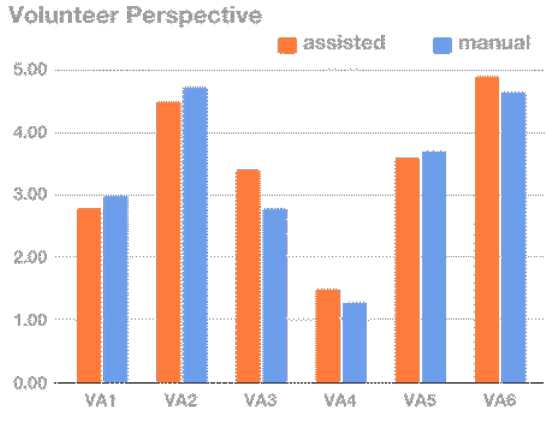

Fig. 6. Results of the survey comparing assisted vs. manual facilitation according to the volunteers' perspective.

In the tool-assisted cases, the times for all tasks that end-users performed were automatically captured. All 5 testers indicated they were 100% confident of the tracked times, while in the case of manual tests, some of the times were lost, and the average confidence in time tracking was 68.33%. This supports hypothesis 3.

4.3 Discussion

In the case of the volunteer perspective survey, even though the results are similar, this is a good sign for assisted facilitation. The reason for this consideration is that the expected performance for the tool-assisted facilitation alternative was lower, considering the potential intrusiveness of a tool that's unknown to the volunteers acting as a guide, forcing them to read instructions instead of listening, sometimes blocking their screen, and demanding additional interactions (such as clicking the "start" or "finished" button). In the study, however, these factors didn't deteriorate the experience. They did take more time in average, which does not support hypothesis 2.

On the other hand, the results on testers' perspective, especially in TA2 and TA3, suggest that they were much more alleviated by the use of the tool, which enabled them to focus more on the task of detecting usability issues, which supports hypothesis 1. They did require more time to specify the tests on Tycho, but we consider that to be a natural result of the level of detail demanded by an automation tool. Also, even if they did receive prior training, this was the first time they actually used Tycho, so the design times could be lowered in further designs.

Overall, the results of the study indicate that the assisted guidance in coordinated user tests takes more time than the manual one, both in the design and tests itself. Volunteers' satisfaction is equivalent, but the testers found it considerably easier to facilitate the sessions, as well as detecting usability problems. We consider this to be a positive first result for assisted guiding, since the main objective of the user tests is to detect usability problems, and not only does the tool improve that aspect, but it also

enables reproducibility. Even more, this means that, adding some recording feature, or simply running shared desktop software, creating a remote testing session could be possible for coordinated user tests.

4.4 Threats to Validity

Even if the end user volunteers are paired with the same tester in both rounds, we preferred that they repeated that aspect instead of the facilitation technique, since the learning effect would be larger, especially for those using the tool (Tycho) for a second time. Considering that the application under test changes as well, we decided that repeating teams was the lesser bias, and this way we were able to maximize the samples with the same team of volunteers.

5 Conclusions

In this paper we have explored how tool-assisted facilitation affects the experience of coordinated user tests on groupware web applications, which is a complex and time-consuming task. We also described a tool that we developed to study this effect, called Tycho. Tycho was designed to create and run remote or on-site user tests, and, by using a browser add-on, guides the volunteers through all the steps, including questionnaires, task instructions and messages, including special coordination tasks: semaphore wait and signal. Tycho aims at supporting early formative evaluations, with focus on short duration, task oriented, facilitated experiments. It provides information about the user's interaction through the system, efficiency in task performance, satisfaction, and tool support for specific tasks. It can be used for evaluations in controlled settings that could take the form of lab experiments or remotely facilitated tests, which is valuable in preparation for longer term, naturalistic evaluations such as field or case studies. It supports observation, by removing from the observer the burden of facilitating and coordinating the actions of the participants. It offers mechanisms to capture both quantitative and qualitative data, keeping also a detailed record of all UI interactions, along with the instructions the participants received at each point in time. Although video recording and archival of the participants' screen is a planned feature, it is still not available.

By running a study with real users, we found out that the tool-assisted guide in coordinated user tests makes it considerably easier for the testers to focus on detecting usability problems, instead of being overloaded with facilitating the test, especially when it’s run by a single person. Results have also shown that the tool-assisted tests take longer to prepare and run. Nevertheless, we consider it to be a small price to pay given all the benefits of tool-assisted testing like reproducibility, accurate measurements (such as time), and, most importantly, more focused testers.

During the studies, the tool itself has proven to be useful, but it requires improvement both in the server component and add-on. We captured feedback from both testers and end-user volunteers, and we are currently working on streamlining the protocol design process by adding new features and removing unnecessary burdens in the tool’s

navigation and data input. We are also planning on extending our validation with more tests, including longer tasks and more volunteers, i.e. more complex scenarios for groupware use. Even if the tests we ran in this work helped to confirm our conjectures about the benefits of tool support, we believe that in more challenging groupware contexts it would be even more beneficial, allowing a single tester to run synchronized tests (either on-site or remote) with multiple users and capture usability flaws at the same time, otherwise extremely difficult.

Acknowledgments. The authors acknowledge the support from the Argentinian National Agency for Scientific and Technical Promotion (ANPCyT), grant number PICT-2015-3000.

References

1. J. Rubin, D. Chisnell, and J. Spool, *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, 2nd edition. Indianapolis, IN: Wiley, 2008.
2. T. Carta, F. Paternò, and V. F. de Santana, 'Web Usability Probe: A Tool for Supporting Remote Usability Evaluation of Web Sites', in *Human-Computer Interaction – INTERACT 2011*, Berlin, Heidelberg, 2011, pp. 349–357. doi: 10.1007/978-3-642-23768-3_29.
3. J. H. Bair, 'Supporting cooperative work with computers: addressing meeting mania', in *Digest of Papers. COMPCON Spring 89. Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage*, Feb. 1989, pp. 208–217. doi: 10.1109/CMPCON.1989.301929.
4. U. M. Borghoff and J. H. Schlichter, *Computer-Supported Cooperative Work: Introduction to Distributed Applications*. Berlin Heidelberg: Springer-Verlag, 2000. doi: 10.1007/978-3-662-04232-8.
5. C. A. Collazos, F. L. Gutiérrez, J. Gallardo, M. Ortega, H. M. Fardoun, and A. I. Molina, 'Descriptive theory of awareness for groupware development', *J Ambient Intell Human Comput*, vol. 10, no. 12, pp. 4789–4818, Dec. 2019, doi: 10.1007/s12652-018-1165-9.
6. P. Johnson-Lenz and T. Johnson-Lenz, 'Groupware: Coining and Defining it', *ACM SIGGROUP Bulletin*, vol. 19, no. 2, p. 34, 1998, doi: 10.1145/290575.290585.
7. Peter Johnson-Lenz and Trudy Johnson-Lenz, 'Post-Mechanistic Groupware Primitives: Rhythms, Boundaries, and Containers', *The International Journal of Man Machine Studies*, vol. 34, pp. 395–417, 1991.
8. T. Winograd, 'A language/action perspective on the design of cooperative work', in *Proceedings of the 1986 ACM conference on Computer-supported cooperative work*, New York, NY, USA, Dec. 1986, pp. 203–220. doi: 10.1145/637069.637096.
9. G. L. Kolfschoten and G.-J. de Vreede, 'The Collaboration Engineering Approach for Designing Collaboration Processes', in *Groupware: Design, Implementation, and Use*, Berlin, Heidelberg, 2007, pp. 95–110. doi: 10.1007/978-3-540-74812-0_8.
10. S. Krug, *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems*, 1st edition. New Riders, 2009.
11. R. Molich and C. Wilson, 'Tips and tricks for avoiding common problems in

usability test facilitation', in *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, New York, NY, USA, Apr. 2008, pp. 2379–2382. doi: 10.1145/1358628.1358689.

12. M. S. Andreasen, H. V. Nielsen, S. O. Schrøder, and J. Stage, 'What happened to remote usability testing?: an empirical study of three methods', in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '07*, San Jose, California, USA, 2007, pp. 1405–1414. doi: 10.1145/1240624.1240838.
13. D. Pinelle and C. Gutwin, 'A review of groupware evaluations', in *Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2000)*, 2000, pp. 86–91.
14. J. Wainer and C. Barsottini, 'Empirical research in CSCW — a review of the ACM/CSCW conferences from 1998 to 2004', *J Braz Comp Soc*, vol. 13, no. 3, pp. 27–35, Sep. 2007, doi: 10.1007/BF03192543.
15. J. R. Wallace, S. Oji, and C. Anslow, 'Technologies, Methods, and Values: Changes in Empirical Research at CSCW 1990 - 2015', *Proc. ACM Hum.-Comput. Interact.*, vol. 1, no. CSCW, p. 106:1-106:18, Dec. 2017, doi: 10.1145/3134741.
16. V. Herskovic, J. A. Pino, S. F. Ochoa, and P. Antunes, 'Evaluation Methods for Groupware Systems', in *Groupware: Design, Implementation, and Use*, Berlin, Heidelberg, 2007, pp. 328–336. doi: 10.1007/978-3-540-74812-0_26.
17. M. A. Khan, N. Israr, and S. Hassan, 'Usability Evaluation of Web Office Applications in Collaborative Writing', in *2010 International Conference on Intelligent Systems, Modelling and Simulation*, Liverpool, United Kingdom, Jan. 2010, pp. 146–151. doi: 10.1109/ISMS.2010.37.
18. K. Baker, S. Greenberg, and C. Gutwin, 'Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration', in *Engineering for Human-Computer Interaction*, Berlin, Heidelberg, 2001, pp. 123–139. doi: 10.1007/3-540-45348-2_14.
19. C. Gutwin and S. Greenberg, 'The mechanics of collaboration: developing low cost usability evaluation methods for shared workspaces', in *Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000)*, Jun. 2000, pp. 98–103. doi: 10.1109/ENABL.2000.883711.
20. D. Pinelle, C. Gutwin, and S. Greenberg, 'Task analysis for groupware usability evaluation: Modeling shared-workspace tasks with the mechanics of collaboration', *ACM Trans. Comput.-Hum. Interact.*, vol. 10, no. 4, pp. 281–311, Dec. 2003, doi: 10.1145/966930.966932.
21. D. Pinelle and C. Gutwin, 'Groupware walkthrough: adding context to groupware usability evaluation', in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, Apr. 2002, pp. 455–462. doi: 10.1145/503376.503458.
22. D. Geszten, B. P. Hamornik, and K. Hercegfi, 'Exploring awareness related usability problems of collaborative software with a team usability testing approach', in *2018 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, Budapest, Hungary, Aug. 2018, pp. 000045–000050. doi: 10.1109/CogInfoCom.2018.8639865.
23. S. Bringas, R. Duque, A. Nieto-Reyes, C. Tîrnăucă, and J. L. Montaña, 'A Framework for Identifying Sequences of Interactions That Cause Usability

Problems in Collaborative Systems', *Electronics*, vol. 10, no. 4, p. 388, Feb. 2021, doi: 10.3390/electronics10040388.

24. O. Díaz, 'Understanding Web Augmentation', in *Current Trends in Web Engineering*, Berlin, Heidelberg, 2012, pp. 79–80. doi: 10.1007/978-3-642-35623-0_8.