Studying the Effect of Delay on Group Performance in Collaborative Editing

Claudia-Lavinia Ignat, Gérald Oster, Meagan Newman, Valerie Shalin, François Charoy

Studying the effect of delay on group performance in collaborative editing

Claudia-Lavinia Ignat[1], Gérald Oster[1], Meagan Newman[2], Valerie Shalin[2], and François Charoy[1]

[1] Inria, Université de Lorraine, CNRS
[2] Department of Psychology, Wright State University
claudia.ignat@inria.fr

Abstract. Real-time collaborative editing systems such as Google Drive are increasingly common. However, no prior work questioned the maximum acceptable delay for real-time collaboration or the efficacy of compensatory strategies. In this study we examine the performance consequences of simulated network delay on an artificial collaborative document editing task with a time constant and metrics for process and outcome suitable for experimental study. Results suggest that strategy influences task outcome at least as much as delay in the distribution of work in progress. However, a paradoxical interaction between delay and strategy emerged, in which the more generally effective, but highly coupled strategy was also more sensitive to delay.

Keywords: Collaborative editing, groupware, delay, usability measurement

1 Introduction

Real time collaborative editing [3] allows a group of people to modify a shared document simultaneously. One user's changes appear to other users almost immediately, with very small time intervals of inconsistent document status. Real-time collaborative editing has gained in popularity due to the wide availability of free services such as Google Drive. With several tools to support collaborative editing, the practice is increasingly common, e.g., group note taking during meetings and conferences, and brainstorming activities.

While collaborative editing tools meet technical goals, the requirements for group performance are unclear. One system property of general interest is network delay. Despite several years of continuous increase in network bandwidth, network delay has not decreased. In fact, network delay is presently considered as the constraining factor. This delay is due to the physical communication technology, be it copper wire, optical fiber or radio transmission. Additional delay in real-time collaborative editing can result from underlying architecture or merging algorithms associated with collaborative editing tools.

As in all collaborative work, in collaborative editing, users must "divide, allocate, coordinate, schedule, mesh, and interrelate their contributions" [6]. Unanticipated interactions between subtasks can emerge [8]. Both the avoidance and

resolution of such interactions requires communication among the team members [4]. However, not all tasks are sensitive to delay [2] and users might pursue strategies that are not sensitive to delay, or they might adjust their strategies if they are aware of delay [5].

To our knowledge no prior work questioned the maximum acceptable delay for real-time collaboration or the efficacy of compensatory strategies. In this paper we study the effect of delay on group performance on an artificial collaborative editing task where a group of four participants i) located the release dates for an alphabetized list of movies and ii) re-sorted the list in chronological order. The task is not unlike job shop scheduling [9]. We examine how strategy interacts with delay to affect task outcome and process in this task.

We start by describing the methods we used for designing our experiment. We then continue by presenting our results organised by measures. We next discuss our results and we end the paper by some concluding remarks.

2 Methods

Participants Eighty students affiliated with a European university participated in this experiment, in mixed gender groups of 4. The participants ranged in age from 21 – 27. All participants used French in their daily activities, although they had sufficient working knowledge of English to comprehend the movie titles in the task stimuli. An electronic announcement solicited participation. All participants received a 10 Euro gift certificate for their participation.

Apparatus The experiment was conducted using four desktop computers in a classroom setting. Participants were separated by partitions and could not directly observe other team members while they worked, although typing activity was audible. The server running the Etherpad application was hosted on an Amazon EC2 instance located in the US East (Northern Virginia) Region. Each desktop ran the Mozilla Firefox web browser executing the Etherpad web client application. Etherpad hosted the task stimuli and a Chat dialogue facility. User operations appeared color-coded in both the text and chat. Etherpad relies on a client-server architecture where each client/user edits a copy of the shared document. When a user performed a modification it was immediately displayed on the local copy of the document and then sent to the server. The server merged the change received from the user with other user changes and then transmitted the updates to the other users. When a user edited a sequence of characters, the first change on the character was immediately sent to the server, while the other changes were sent at once only upon reception of an acknowledgement from the server. With each change sent to the server, it created a new version of the document. Gstreamer software enabled the video recording of user activity. We also instrumented Etherpad to register all user keyboard inputs on the client side and to introduce delays on the server-side. The editor window displayed 50 lines of text. Users editing above the field of view of a collaborator could cause the lines

within the collaborators' view to "jump" inexplicably. Such a property is consistent with the inability to view an entire document as it undergoes modification from multiple team members.

Task & Stimuli Participants conducted a 10 minute search and sorting task, starting with an alphabetized list of movies. They used the internet to locate the release year for each movie and sorted the list in chronological order. The list contained 74 movies, extending beyond the window size of the editor.

Procedure The entire procedure was approved by a US University IRB. Participants began the session with informed consent. The present sorting task was second in a three-task series. Scripted instructions (translated into English) for the sorting task follow: *"We will provide you with the list of movies. Your task is to search for the release dates of the movies and assemble a single list of movies sorted and labeled with their release dates. You can use the browser for finding the year of release of a movie. The year of release of a movie should be placed before the movie title and the movies should be sorted in an ascending order, starting from the oldest to the newest movie. You will have 10 minutes for finishing the task. Please work as accurately as you can while still being efficient. You are free to coordinate your efforts with your teammates throughout the task using the chat interface at the bottom right side of the screen".*

Design The sorting task was conducted with four teams of 4 participants for each level of the continuous independent variable Delay, tested at 0, 4, 6, 8 and 10 seconds in addition to the 100 msec delay inherent to client-server communication. While participants viewed their own document changes in real-time, they viewed other participants' changes according to delay condition. Chat was implemented in real time for all conditions. Delay conditions were tested in random order, and all groups experienced a single level of delay across the three-task session.

Dependant Measures We examined sorting accuracy as an outcome measure. We also examined a set of process measures such as average time per entry and chat behavior, as well as strategies.

- *Sorting Accuracy* is measured by an insertion sort metric which is the most likely strategy for human sorting [7]. Insertion sort iterates over an input list of elements and generates an output sorted list. At each iteration, an element in the input list is removed and inserted in the proper location within the sorted list, terminating when no more input elements remain. The insertion sort metric quantifies the distance between the input list and the output sorted list. Here the group provides the input list and the output list is the target list of movies, ordered according to their release dates. The distance between an element in the input list and the corresponding element in the sorted list is measured in terms of the number of swaps between adjacent elements required to place the input element properly in the sorted list.

We normalized this distance with the distance in the worst case scenario, i.e. when the input list is sorted in reverse order. We additionally had to accommodate duplicated or missing movies, or movies with incorrect release dates. Therefore we eliminated the duplicated movies and the movies with an incorrect release date from the final list of movies generated by each group. We also eliminated from the output list the missing movies in the input list. The distance computed by the insertion sort metric was adjusted to be proportional to the number of movies that are not duplicated and for which users assigned the correct release dates. The formula that we used for each group score is : $\left(1 - \frac{\#Swaps}{\#SwapsWorstCase}\right) \times \#Movies$
$\#Swaps$ represents the total number of swaps between adjacent movies required using an insert sort method on the group's final list of movies. $\#SwapsWorstCase$ represents the total number of swaps between adjacent movies required by an insert method in the worst case, i.e. when the list of movies contains the movies in a descendant order according to the release dates. $\#Movies$ represents the number of movies in the final list of movies after a removal of duplicated movies or those with an incorrect release date. Two co-authors independently coded the insertion sort metric in different programming languages with identical results.

- *Average Time Per Entry* was computed as the period of activity in question divided by the number of characters input. Because the task characteristics potentially changed over the 10 minutes, with the first half corresponding to the identification of movie dates and the second potentially corresponding to the sorting, we also calculated separate average response times for the first and second halves of the session.
- *Chat behavior* was quantified as the number of turns, the number of words, agreement words (yes and OK), group oriented pronouns (You, your, one, us, who, each one, someone, no one, others) and ego oriented pronouns (I, my, me, mine). We examined agreement words, group-oriented pronouns and ego-oriented pronouns as a function of the number of words.
- *Strategies* emerged through detailed analysis and are described below.

3 Results

We provide results in three subsections, organized by measures. First we examine task strategies. Next we examine task outcome, followed by several measures of task process. For both outcome and process measures, we conduct regression modeling, using Delay condition and Strategy as predictors, and follow up with simple effect analyses by Strategy. We examine additional facets of process in the indicators of coordination as apparent in Chat.

3.1 Strategies

As we had no a priori hypotheses about how users would divide up the work, we developed a coding scheme based on a review of the videos, supplemented by the chat discussion. Two strategies emerged:

– Sort at the end (Strategy 0) where sorting starts after all years have been added for all movies. This strategy enables loose coupling among participants at the beginning of the task, but leaves a highly coupled sorting task for end, with no pre-established assignments.
– Continuous sorting sorting (Strategy 1) is done immediately after adding a year for a movie. This strategy begins with a highly coupled distribution of work among participants.

3.2 Outcome Measure

An insertion sort metric served as the outcome measure. Figure 1 displays the relationship between Delay, Strategy and insertion sort.

Strategy alone accounts for insertion sort score, $r(18) = .68$, adjusted $R^2 = .34$, $F(1, 18) = 10.89$, $p = .004$, $b_0 = 44.17$ $t(18) = 14.54$, $p < .001$, $b_1 = 4.53$, $t(18) = 3.30$, $p = .004$.

We examined the 9 groups who pursued Strategy 1 separately from the 11 groups who pursued Strategy 0. Among the 11 Strategy 0 groups, Delay does not predict insertion sort ($r(9) = .11$, adjusted $R^2 = -.10$, $F(1, 9) = .12$, $p = .742$). Among the 9 Strategy 1 groups, a linear model for Delay predicts insertion sort score ($r(7) = .69$, adjusted $R^2 = .39$, $F(1, 7) = 6.21$, $p = .042$). The intercept for the linear model is 66.46, $t(7) = 17.69$, $p = .000$ and the unstandardized slope is -1.38, $t(7) = -2.49$, $p = .042$. That is, each increment in Delay decrements the outcome measure by 1.38 Insertion Sort score units. A quadratic model does provide a better account for Strategy 1 groups ($r(7) = .75$, adjusted $R^2 = .50$, $F(1, 7) = 8.98$, $p = .020$). The intercept for the quadratic model is 65.99, $t(7) = 21.15$, $p = .000$ and the unstandardized Delay slope is $-.15$, $t(7) = -3.00$, $p = .020$. This model raises the possibility that Delay condition 10 results in qualitatively different behavior than the other conditions.

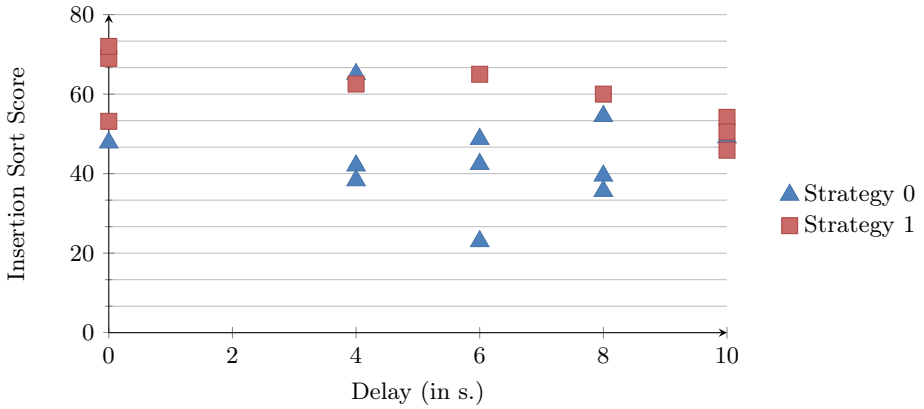

Fig. 1. Insert Sort Score by Delay, separated by strategy. Strategy 0 groups pursued sorting after finding movie years. Strategy 1 groups pursued continuous sorting.

3.3 Process Measures

We examined the average time between task inputs based on client recordings. Software error caused the loss of data for 4 groups. At the group level we used regression analysis to describe our results, treating Delay condition as a continuously valued independent variable. At the participant level, we used a nested ANOVA, using Delay condition as a categorical variable. The nested analysis allowed us to determine whether significant group effects precluded analysis of the Delay main effect. We examined Delay condition and Strategy as independent variables, with tests based on Type III Sums of Squares to account for the unbalanced design and an $\alpha = .10$ due to the small number of groups [1].

Significant group effects precluded statistical analysis of response time data across the entire experimental session. We proceeded with response time data from the session's first 5 minutes, where group effects were absent.

Group response time over the first 5 minutes accounts for insertion sort score, ($r(14) = .49$, adjusted $R^2 = .19$, $F(1, 14) = 4.48$, $p = .053$, $b_0 = 70.37$ ($t(14) = 7.20$, $b_1 = -.01$, $t(14) = -2.12$, $p = .053$). Slowing down decreases outcome.

Delay alone accounts for response time, $r(14) = .45$, adjusted $R^2 = .15$, $F(1, 14) = 3.56$, $p = .080$, $b_0 = 2504.77$ msec $t(14) = 4.81$, $p < .001$, $b_1 = 141.84$, $t(14) = 1.89$, $p = .080$.

As suggested in Figure 2, separate Strategy models suggest quadratic effects of Delay on response times. For Strategy 0 the best model missed overall significance $r(6) = .75$, adjusted $R^2 = .38$, $F(2, 5) = 3.12$, $p = .132$, $b_0 = -6331.84$ msec $t(5) = -1.54$, $p = .183$, Delay $b_1^* = 5.05$, $t(5) = 2.42$, $p = .060$, Delay^2 $b_2^* = -4.81$, $t(5) = -2.30$, $p = .070$. For Strategy 1, the best model had an $r(6) = .71$, adjusted $R^2 = .42$, $F(1, 6) = 6.12$, $p = .048$, $b_0 = 2281.23$, $t(6) = 7.02$, $p < .001$, Delay^2 $b_1^* = .71$, $t(6) = 2.47$, $p = .048$.

Chat We examined chat metrics as predictors of insertion sort score. Proportion of accord words predicted insertion sort score ($r(18) = .54$, adjusted $R^2 = .25$,

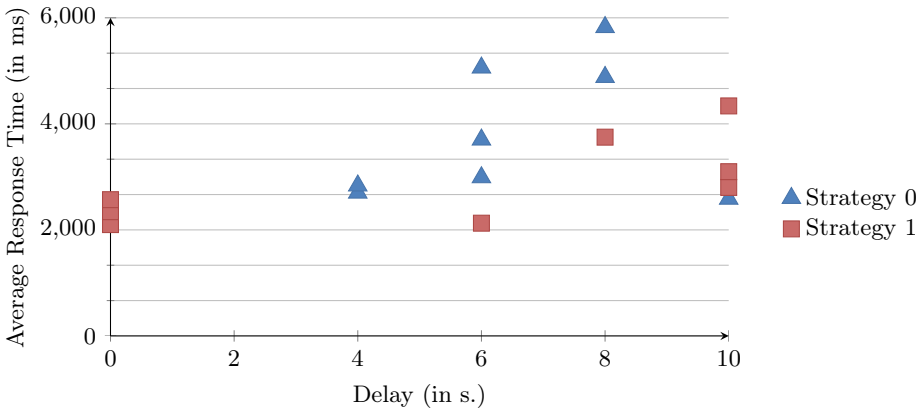

Fig. 2. Average response time by Delay from the first half of session.

$F(1, 18) = 7.24$, $p = .015$, $b_0 = -40.00$, $t(18) = 8.49$, $b_1 = 3.83$, $p = .000$, $t(18) = 2.69$, $p = .015$). We also examined Strategy and Delay as predictors of chat metrics. Of these analyses, only total words was sensitive to the independent variables. For Strategy 0, Delay predicts total words ($r(9) = .64$, adjusted $R^2 = .34$, $F(1, 9) = 6.22$, $p = .034$, $b_0 = 29.08$, $t(9) = .79$, $p = .448$, $b_1 = 14.32$, $t(9) = 2.49$, $p = .034$). For Strategy 1, Delay does not predict total words ($r(7) = .34$, adjusted $R^2 = -.01$, $F(1, 7) = .92$, $p = .368$, $b_0 = 123.36$, $t(7) = .5.39$, $p = .001$, $b_1 = -3.23$, $p = .034$, $t(7) = -.96$, $p = .368$).

4 Discussion

An artificial document editing task captures the upper limit of dependency and interactivity in collaborative editing, and permits the measurement of task outcome. Here we return to our original question regarding the relationship between delay and strategy on process and outcome.

The effect of delay depends on strategy. Such an interaction between strategy and experimental manipulation on outcome is consistent with prior studies in game-like motor control environments. However, somewhat counterintuitively, and unlike previous research, the overall superior strategy does not overcome the effect of delay. In fact, the insertion sort score declined with delay for continuous sort, but did not for sort-at-the-end. We suspect that continuous sort entails more coupling, because years must be in place prior to positioning, and because text position is changing frequently throughout the entire task as sorting proceeds. To manage the coupling in continuous sort, we see participants slow down with delay. However, the negative slope on the insertion sort metric for continuous sort relative to the flat slope for sort-at-the-end suggests that the continuous sort strategy is only adaptive within a range of delay. Untested levels of delay could actually result in worse performance for continuous sorting than a sort-at-the-end strategy.

The sort-at-the-end strategy did not encounter coupling until the later phases of task completion. However, the chat metric suggests that sort-at-the-end requires more local coordination as delay increases. Thus the coordination established by formal agreement at the outset in continuous-sort appears to favor efficient communication over the ability to respond to local perturbations. On the other hand, sort-at-the-end appears to favor the ability to respond to local perturbations at the expense of efficient communication.

Performance on a family of related tasks will help to address the relationship between delay and task properties. We have data for the effect of delay on two other artificial editing tasks that vary both the task time constant and the degree of subtask coupling. The analyses presented here suggest the need to add Delay levels, with 2 and 12 second delays and beyond. This will help determine whether the models that relate performance to delay are appropriately linear, or quadratic, with more rapid declines in task performance with delay.

5 Conclusions

The general effect of delay on an artificial document editing task is to slow the individual participant, which for the present task, decrements the outcome metric. However, similar to game-like tasks (e.g., [5]), the effect of delay on document editing, as measured by outcome, depends on strategy. A tightly coupled subtask decomposition that enhances outcome in the presence of minimal delay becomes detrimental at higher levels of delay, potentially less effective than a more loosely coupled task decomposition at the beginning of the task. Nevertheless, a loosely coupled strategy at the beginning of the task leaves a poorly coordinated, tightly coupled sorting task to the end of the task, increasing the need for communication and hampering overall performance. Given the time constant of the present task, strategy is at least as important as delay in the distribution of participant inputs to the team.

6 Acknowledgments

The authors are grateful for financial support of the USCoast Inria associated team and of the research program ANR-10-SEGI-010 and for sabbatical support from the Department of Psychology, Wright State University. Many thanks to Olivia Fox for her help on statistical analyses.

References

[1] Cooke, N.J., Gorman, J.C., Duran, J.L., Taylor, A.R.: Team cognition in experienced command-and-control teams. Journal of Experimental Psychology: Applied 13(3), 146–157 (September 2007)
[2] Dourish, P., Bly, S.: Portholes: supporting awareness in a distributed work group. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 541–547. Monterey, California, USA (1992)
[3] Ellis, C.A., Gibbs, S.J., Rein, G.: Groupware: Some Issues and Experiences. Communications of ACM 34(1), 39–58 (January 1991)
[4] Erkens, G., Jaspers, J., Prangsma, M., Kanselaar, G.: Coordination processes in computer supported collaborative writing. Computers in Human Behavior 21(3), 463–486 (May 2005)
[5] Gutwin, C., Benford, S., Dyck, J., Fraser, M., Vaghi, I., Greenhalgh, C.: Revealing delay in collaborative environments. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 503–510. Vienna, Austria (2004)
[6] Schmidt, K., Bannon, L.J.: Taking cscw seriously. Computer Supported Cooperative Work 1(1-2), 7–40 (1992)
[7] Sedgewick, R.: Algorithms. Addison-Wesley (1983)
[8] Simon, H.A.: The architecture of complexity. In: Proceedings of the American Philosophical Society. pp. 467–482 (1962)
[9] Tan, D.S., Gergle, D., Mandryk, R., Inkpen, K., Kellar, M., Hawkey, K., MaryCzerwinski: Using job-shop scheduling tasks for evaluating collocated collaboration. Personal and Ubiquitous Computing 12(3), 255–267 (January 2008)