Analyzing Shared Workspaces Design with Human-Performance Models

Pedro Antunes[1], Antonio Ferreira[1], and Jose A. Pino[2]

[1] Department of Informatics, University of Lisbon, Portugal
paa@di.fc.ul.pt, asfe@di.fc.ul.pt
[2] Department of Computer Science, Universidad de Chile, Chile
jpino@dcc.uchile.cl

Abstract. We propose an analytic method to evaluate synchronous shared workspaces design. The method uses human-performance models, developed in the Human-Computer Interaction field, to make time predictions about collaborative actions performed in selected critical scenarios. We apply this method to two case studies: the design of a collaborative game and the redesign of a collaborative tool for software engineering requirements negotiation. The benefits and limitations of the method are discussed, as well as some implications for design.

1 Introduction

The design and evaluation of groupware usability is a challenging endeavor for practitioners and researchers because existing methods have considerable trade-offs and impose significant constraints:

- On the one hand, the required evaluation resources (time, users, experts, apparatus) may be hard to find or simply unavailable, a condition that worsens due to the iterative nature of formative usability evaluation. This applies especially, but not exclusively, to controlled laboratory experiments [1];
- On the other hand, several evaluation methods are either descriptive or prescriptive and therefore provide little support for comparing design options and predicting usability results. This applies to methods such as Groupware Task Analysis [2], Collaboration Usability Analysis [3], Groupware Walkthrough [4], and Groupware Heuristic Evaluation [5].

We argue that groupware designers should complement existing practice and knowledge with the ability to make quick measurements and calculations about key characteristics of computer-mediated collaboration. Our motivation is based on the century-old need to measure before improving as well as on the evidence that fast evaluation enables several design iterations. We introduce a method that can be applied without users or functional prototypes to quantitatively *predict* and *compare* the usability of synchronous shared workspaces (here referred to as shared workspaces).

Shared workspaces present an interesting challenge to usability evaluation because collaboration among group members features strong interdependencies wherein individual actions affect the choices and outcomes of the other users. Furthermore, the impact of small, low-level, design decisions requiring perceptual or motor activity is much higher than in other contexts, where the emphasis may be on more cognitive tasks, such as decision making. These characteristics of shared workspaces are usually not captured by existing methods and tend to be overlooked. Instead, these methods focus on generic, high-level, communication and coordination mechanisms that the groupware should provide to support collaboration (e.g. the mechanics of collaboration [3]). We approach these two aspects of shared workspaces—action interdependencies and attention to detail—by focusing on the analysis of *critical scenarios* and by applying existing human-performance models from the Human-Computer Interaction (HCI) field:

– The analysis of critical scenarios raises the designer's consciousness about collaborative actions that have a potentially important effect on individual and group performance;
– The human-performance models address the fine-grained details of the interaction with the shared workspace and provide performance estimates without the participation of users or the development of prototypes.

Human-performance models, such as the Keystroke-Level Model (KLM) [6], are based on a cognitive architecture that approximates single-user interaction at a low level of detail (e.g. perceptual, motor, and cognitive processors). We discuss the contextualization of this cognitive architecture to the specific characteristics of groupware in Sect. 3, and introduce some basic concepts necessary to model awareness and control of information about the users' collaborative actions.

Section 4 describes the proposed method for evaluating group performance of users working together in a shared workspace. Two case studies are presented in Sect. 5 involving shared workspaces design to demonstrate the value of this method. We conclude the paper in Sects. 6 and 7 with a discussion of the benefits and limitations of the method, as well as with some implications for design.

2 Related Work

The application of human-performance models to the groupware context is very rare in the literature and virtually inexistent for workspace collaboration. We start this section with an overview of Distributed GOMS (DGOMS) [7] and a recent study involving a complex group task [8]. In both cases the same family of techniques, called GOMS (Goals, Operators, Methods, and Selection rules) [9], is used to provide quantitative estimates of human performance.

DGOMS [7] applies hierarchical task analysis and human-performance models to represent group activity and to predict execution time, distribution of workload, and other performance variables. This method successively decomposes group work in group tasks until individual subtasks can be identified. At this level of detail the subtasks are defined in terms of perceptual, cognitive, and motor operators, as well as with a new communication operator that is used to coordinate individual tasks executed in parallel. The problem, however, is that such a coordination mechanism is more appropriate to groups where users react to predefined events, and not sufficiently rich to describe the type of interdependency established by users working through shared workspaces [10].

Another application of human-performance models to groupware considers "teams of models" to analyze a complex task executed by a group of users [8]. The task involved several users with individual roles monitoring a display and executing actions in a coordinated way, via a shared radio communication channel. While this approach assumes that several individual models are necessary to explain collaborative work, the study does not address workspace collaboration and focuses instead on coordinated work.

We now review some usability evaluation methods specifically developed for groupware. Groupware Task Analysis [2] is a method that combines high-level hierarchical task analysis and field observations for addressing all stages of groupware design. It is based on a conceptual framework including agents, group work, and situation, in a similar manner to the work models defined by the Contextual Design approach [11], well known in the HCI field.

The next three methods of groupware usability evaluation are based on a common descriptive framework called "mechanics of collaboration" [3], whereas each method applies a different evaluation perspective. The mechanics are formalizations of high-level group work primitives (e.g. communicating and coordinating) that helps the designer focus on how the shared workspace supports the required collaboration. Starting with Collaboration Usability Analysis [3], this method couples field observations and a version of hierarchical task analysis that allows variation, iteration, and parallel work, for representing group work. The Groupware Walkthrough [4] method uses step-by-step written narratives or task diagrams corresponding to collaborative scenarios, and it aims at gathering the opinions of expert inspectors while using the shared workspace. Finally, Groupware Heuristic Evaluation [5] is based on a number of experts evaluating the compliance of a shared workspace with a list of heuristics.

In summary, existing methods for groupware usability evaluation are of two types: the first type is targeted at predicting performance in coordinated work scenarios where users react to predefined events (not even requiring group awareness); the second type can be applied to shared workspaces but have a descriptive or prescriptive nature that allows for high-level task analysis or depends on inspections performed by multiple usability experts. Our proposed method complements these two types of methods by addressing detailed interdependencies in critical scenarios of collaboration using existing human-performance models.

3 Theoretical Background

In general, human-performance models have been associated with the Model Human Processor (MHP) [9], that represents human information processing capabilities using perceptual, motor, and cognitive processors. Nevertheless, several architectural differences are identified when considering individual models: for instance, KLM uses a serial-stage architecture, while CPM-GOMS addresses multi-modal and parallel human activities (e.g. recognizing an object on the display while moving the hand to the keyboard) [12]. In spite of these differences, a common characteristic of existing human-performance models is that they are singleware, that is, they assume that just one user interacts with a physical interface. Figure 1 depicts this singleware architecture based on the MHP. We also illustrate that there is a conventional information flow in this architecture, from the cognitive to the motor processors, from the input to the output devices, and from the perceptual to the cognitive processors.

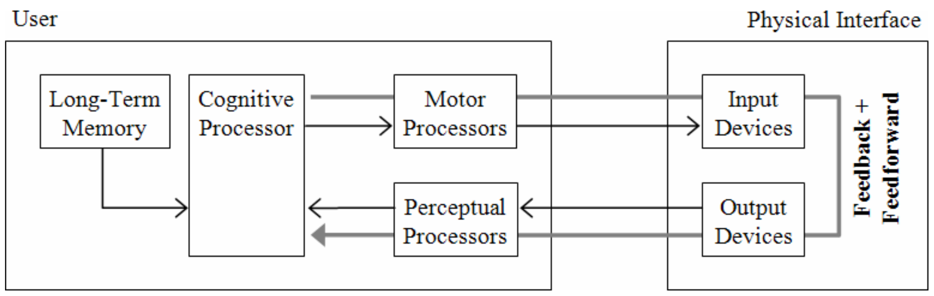


Fig. 1. Singleware architecture

According to some authors [8], the architecture depicted in Fig. 1 applies to groupware: to model a group of users, one can have individual models of the interaction between each user and the physical interface; one can also assume that the physical interface is shared by multiple users, and that the users will deploy procedures and strategies to communicate and coordinate individual actions. Thus, according to this view, groupware usage is reflected in some conventional information flows, spanning multiple users, which still may be described using the conventional production rules and representations.

The problem, however, is that this approach does not consider two fundamental groupware features: (1) the conventional information flows are considerably changed to reflect collaborative actions, mutual awareness, and interdependence; and (2) the focus and granularity should not remain on the interactions between the user and the physical interface but should significantly change to reflect the interactions between users, mediated by the physical interface. We address these two issues in the next section.

3.1 Groupware Conventional Information Flows

Let us start with the singleware architecture. In this context, we may characterize the conventional information flows in two categories: feedback and feedforward. The first category corresponds to information initiated by the user, for which the physical interface conveys *feedback* information to make the user aware of the executed operations [13,14]. The second category concerns the delivery of *feed-forward* information, initiated by the physical interface, to make the user aware of the afforded action possibilities [14]. Now, when we regard groupware, some additional categories have to be considered. In this paper we analyze explicit communication, feedthrough, and back-channel feedback.

Explicit communication addresses information produced by one user and explicitly intended to be received by other users [3]. For example, a user may express a request for an object to another user. This situation can be modeled as a physical interface capable of multiplexing information from input devices to several output devices. The immediate impact on the model in Fig. 1 is that we now have to explicitly consider additional users connected to the physical interface, as shown in Fig. 2.

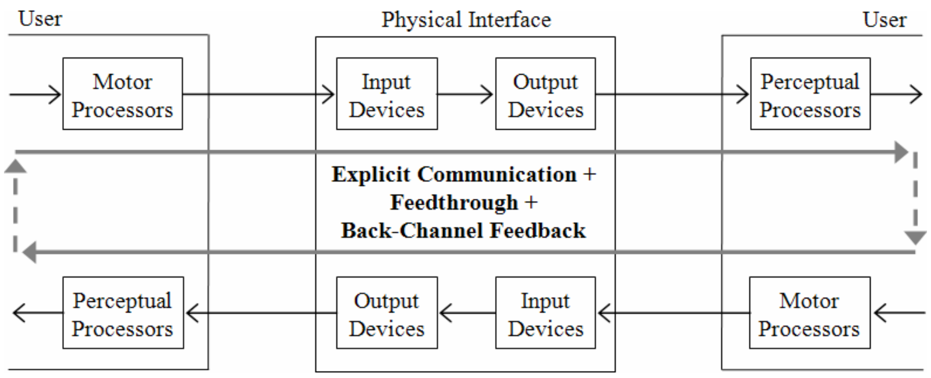


Fig. 2. Groupware architecture

Feedthrough concerns implicit information delivered to several users reporting actions executed by one user [15]. Feedthrough is essential to provide group awareness and to construct meaningful contexts for collaboration. For example, the shared workspace may show currently selected menus for each user that is manipulating objects. This information is automatically generated by the physical interface as a consequence of the user's inputs, and it is directed towards the other users. A very simple way to generate feedthrough consists of multiplexing feedback information to several users. Sophisticated schemes may consider delivering less information by manipulating the granularity and timing associated to the operations executed by the groupware [16].

Finally, *back-channel feedback* concerns unintentional information flows initiated by one user and directed towards another user to facilitate communication [17]. No significant content is delivered through back-channel feedback, because it does not reflect cogitation from the user. Back-channel feedback may be automatically captured and produced by the physical interface based on the users' body gestures and vocal activities.

3.2 Groupware Specializations of Physical Interface Devices

All information flows in the groupware architecture are naturally processed by the user's cognitive, perceptual, and motor processors, and the corresponding physical input and output devices. However, we regard the separate processing of explicit communication, feedthrough, and back-channel feedback in specialized input and output devices to show the distinction between collaborative and non-collaborative interactions. We define the *awareness input/output devices* as devices specialized in processing sensory information about who, what, when, how, and where are the other users operating in the shared workspace.

Another specific feature of the awareness input/output devices is that they not only afford users to construct a perceptual image of the collaborative context, but they also allow users to perceive the role and limitations of the physical interface as a mediator. This is particularly relevant when the Internet is being used to convey feedthrough information, where feedthrough delays are significantly longer and less predictable than feedback delays [18] and the available bandwidth and network availability may be limiting factors [19].

A further reason for proposing the awareness input/output devices is related to another particular characteristic of groupware: it lets users loose the link between executed operations and group awareness, a situation called "loosely coupled" [20]. Two types of coupling control may be considered: at the origin and at the destination. Users may control coupling at the origin to specify what and when private information should become public. But coupling can also be controlled at the destination: getting awareness information on a per-object demand basis, e.g. by specifying filters that restrict received awareness to some selected objects and types of events. In all cases this situation requires some cognitive activities from the user to discriminate and control awareness information delivery, and we model this situation with the *coupling input device.*

We illustrate the resulting groupware physical interface in Fig. 3. In summary, our interpretation of the MHP architecture, taking the groupware context in consideration, essentially emphasizes the cognitive activities related to the awareness and coupling features supported by the groupware physical interface.

4 Method to Evaluate Group Performance

Step 1: Defining the physical interface. The method starts by defining the physical interface of the groupware under analysis. We propose that the physical interface may be decomposed into several shared workspaces. Such decomposition simplifies the analysis of complex groupware tools, that often organize collaborative activities in multiple intertwined spaces, usually human recognizable, supporting various purposes, objects, and functionality.

Using the groupware physical interface in Fig. 3 as reference, we define a shared workspace as a distinctive combination of awareness and coupling devices. We exclude from the analysis any workspaces not having, at least, one awareness or coupling device, since they would not involve collaboration.

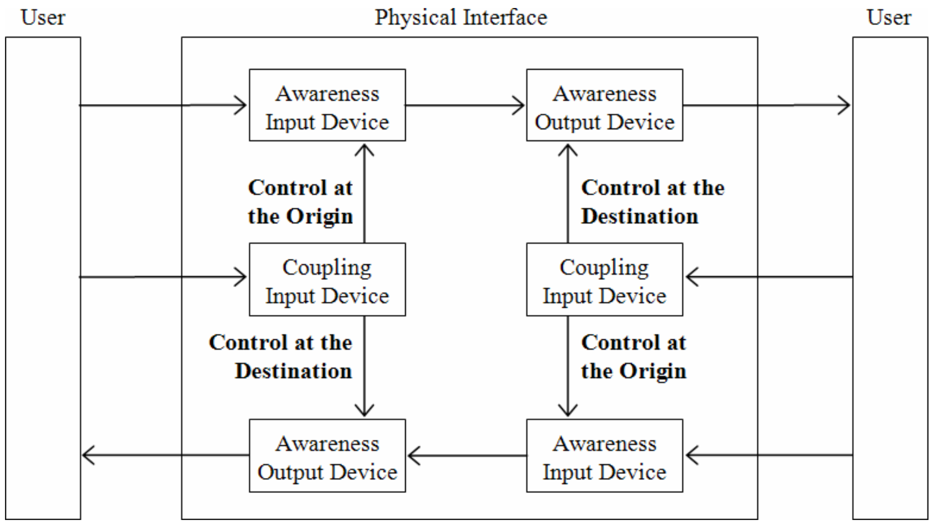


Fig. 3. Groupware physical interface

The outcome of this step is then: (1) a list of shared workspaces; (2) definition of supported explicit communication, feedthrough, and back-channel feedback information; and (3) characterization of supported coupling mechanisms. In this step alternative design scenarios may also be defined, considering different combinations of shared workspaces, awareness information, and coupling mechanisms.

Step 2: Breakdown definition of critical scenarios. The second step describes the functionality associated with the various shared workspaces with respect to critical scenarios, i.e. with a special focus on collaborative actions that have a potentially important effect on individual and group performance. This functionality is successively decomposed from the more general to the more detailed, using a top-down strategy, typical of hierarchical task analysis. Alternative design scenarios may be defined, considering several combinations of users' actions.

Step 3: Comparing group performance in critical scenarios. The final step is dedicated to compare the alternative design scenarios defined in the previous steps. These comparisons require a common criteria, for which we selected the *predicted execution time* in critical collaborative scenarios.

We utilize the Keystroke-Level Model (KLM) [6] to predict execution times because it is relatively simple to use and has been successfully applied to evaluate single-user designs [12]. In KLM, each user action is converted into a sequence of mental and motor operators, whose individual execution times have been empirically established and validated in psychological experiments: `M` is for mental preparation and takes 1.2 seconds; `P` is for pointing with the mouse to a target on a display, requiring 1.1 seconds; and `K` is for pressing or releasing a mouse button, taking 0.1 seconds [6,9]. Therefore, the designer may find out which sequence of operators minimizes the execution time of a particular user action.

Naturally, the application of KLM must be adapted to groupware, considering that the execution time we want to evaluate encompasses several users who work in parallel. Our approach consists of focusing the analyses on critical scenarios involving selected sequences of operations from more than one user. For instance, suppose we want to analyze several design alternatives for managing the access to shared workspace objects. A critical scenario occurs when a user accesses the object, immediately followed by another one trying to access the object but finding it locked. We may use KLM to estimate the execution times of these combined operations for each design option, and thus finding out which one minimizes the overall execution time. We discuss in detail the application of this method to groupware design in the next section.

5 Using the Method

5.1 Collaborative Game

We apply the method to a collaborative game in this case. The game explores a collaborative scenario where players have specific roles and act opportunistically according to the current state of the shared workspace. In particular, players can make either vertical or horizontal connections between points in a board. The objective of the game is to connect all points in the board as quickly as possible (Fig. 4). The points are connected in pairs, but this is only allowed if at least one of the to-be-connected points is already linked to a third point via a perpendicular connection. Initially, the board contains a single connection line.

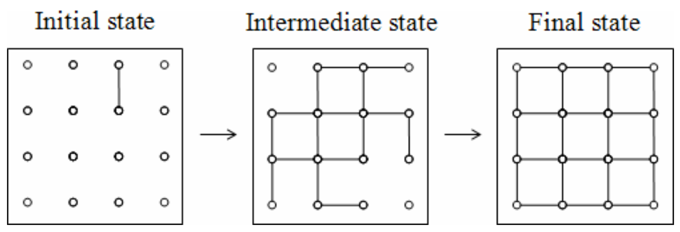


Fig. 4. Players act opportunistically to make connections between all points

Step 1: Defining the physical interface. The game provides a shared workspace displaying a public updated view of the board (Fig. 4). There exist several private workspaces also, one for each player, allowing them to actually connect the points with horizontal or vertical lines, depending on their specific role. However, the analysis of these private workspaces is out of scope, since we are only interested in collaborative actions. The player's moves are restricted to be done with a mouse having a single button.

Step 2: Breakdown definition of critical scenarios. The board operates in the following way. In order to connect two points, a player must first reserve them on the board. Multiple players may not simultaneously reserve the same points, but as this can happen and have a considerable performance penalty, we consider this a critical scenario. Reservation is done by selecting two adjacent points with the mouse and dragging them out of the board (to a private workspace). The connection is made public when the points are dragged back to the board, an action that also automatically releases the points. This is our design scenario A.

We also analyze an alternative scenario B, where, in order to increase awareness and minimize inadvertent selections of reserved points, the board displays a letter identifying the current owner next to the reserved points. This design provides awareness information only at the end of the reserve or release actions.

An additional alternative design scenario C features extra awareness while the points are being selected on the board. The main justification for this refinement is the production of more fine-grained and up-to-date awareness information.

We will next proceed with a detailed specification of the collaborative actions for the selected critical scenario. For now, we observe there are only two collaborative actions in this game, which we designate `RESERVE` and `RELEASE`.

Step 3: Comparing group performance in critical scenarios. We now focus on the fine-grained details of the `RESERVE` and `RELEASE` collaborative actions, to the point where they can be described with KLM operators. Starting with the `RESERVE` action, we assume the player begins by searching the shared workspace for a point that satisfies three conditions: (1) it must not be reserved; (2) it must allow a new connection; and (3) it must have a perpendicular connection to another point. This is converted into a single `M` operator because the verification of the three conditions is highly repetitive and players are trained in the game.

Once a point is located, the player moves the mouse pointer near it, `P`, presses the mouse button, `K`, and moves the pointer to an adjacent point, `P` (the connection between these two points will be drawn afterwards in the private workspace). The player then releases the mouse button, `K`, to complete the selection.

The last part of a reservation is done by dragging the selected points out of the board: the player adjusts the mouse pointer so that it rests on top of the adjacent point, `P`, presses the mouse button, `K`, drags the selected points out of the board, `P` (no `M` operator is required because the workspaces are always in the same place), and releases the mouse button, `K`. The complete sequence of KLM operators for the `RESERVE` collaborative action is `MPKPKPKPK`, which has a predicted execution time of 6 seconds. The `RELEASE` collaborative action is very similar to `RESERVE` in two ways: the predicted execution time is also 6 seconds, and the sequence of KLM operators is again `MPKPKPKPK`.

Now, having determined the sequences of operators for managing the board, we focus on the comparison of group performance in the critical scenario—when two players have the intention of reserving the same points—for the design alternatives A and B. We assume the first player will always succeed in order to simplify the analysis, and also that having more than 2 players reserving the same points is a rare event that does not deserve further attention.

Considering the design scenario A, the best case happens when two players start the reservation for the same points at the same time. In this case, after the 6 seconds needed for a complete reservation, the second player notices an error indication on the board (an `M` operator) and starts again with other points, which takes additional 6 seconds. The best execution time is then 13.2 seconds. The worst case happens when the second player begins just after the first player finishes a reservation; since no awareness information is provided, the total execution time increases to 19.2 seconds (see Scenario A in Fig. 5).

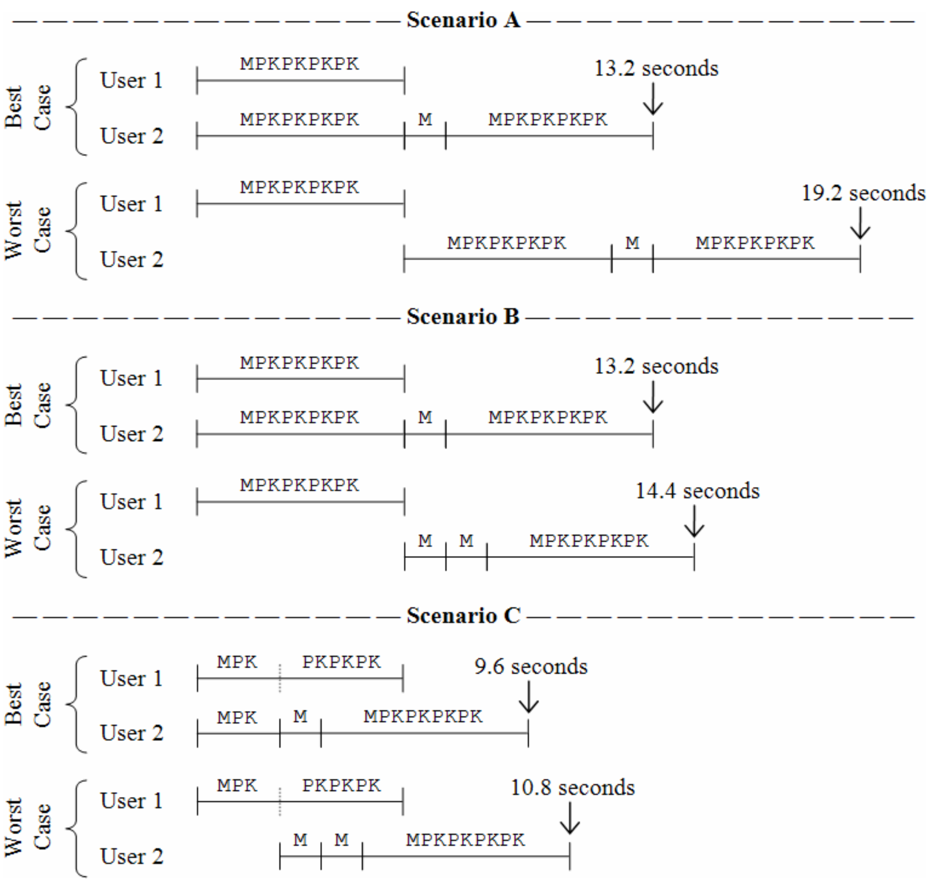


Fig. 5. Best and worst execution times for handling the critical scenario

For the scenario B, the best case is identical to that of scenario A. However, the execution time for the worst case is significantly reduced because the second player can interrupt an ongoing reservation as soon as the owner letter is displayed on the board. We represent this situation with two `M` operators: the first corresponds to the initial `M` of any reservation, while the second `M` is for interpreting the critical situation. The total execution time for the worst case is now 14.4 seconds (see scenario B in Fig. 5).

The optimization considered in scenario C provides awareness information upon the selection of the first point, i.e. right after a sequence of `MPK` (instead of the full `MPKPKPKPK`). In these circumstances both the best and worst cases benefit from reduced execution times (see scenario C in Fig. 5). If the two players start the reservation at the same time, then at about 2.4 seconds they both see their simultaneous selections on the board. Then, the second player (by our assumption) decides to stop the current selection and starts another one, an `M` followed by a new reservation, which takes 9.6 seconds. The worst case takes 10.8 seconds; its explanation is analogous to the worst case for scenario B, except the awareness supplied by the owner letter upon a full reservation is substituted by the awareness provided by the selection of the first point.

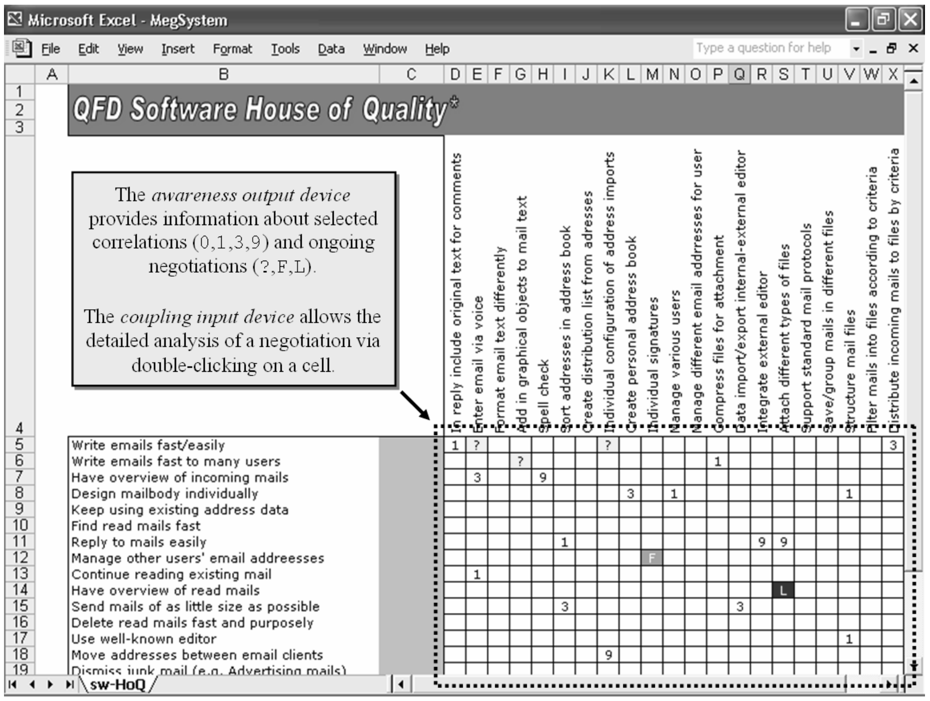
In summary, the method brought quantitative insights about the role of feedthrough information in group work support, predicting that for the selected critical scenario the design option C is faster than B by 3.6 seconds, and that B is faster than A by about 4.8 seconds, but only in the worst case scenario.

5.2 Software Requirements Negotiation Tool

We now demonstrate the application of the analytic method to an existing groupware tool that supports collaborative software quality assessment using the Software Quality Function Deployment (SQFD) [21] methodology. The objective of this tool is to facilitate the SQFD negotiation process by providing mechanisms in a same-time, different-place mode. Our starting point in this case was a previous experiment with the tool that gathered data via questionnaires, and that reported some usability problems, namely that it was considered difficult to use. Further details about this tool and about the previous usability evaluation can be found in [22].

Step 1: Defining the physical interface. The tool has two shared workspaces: SQFD matrix and "Current Situation." The SQFD matrix allows users to inspect a matrix of correlations between product specifications and customer requirements, as well as observing which correlations are under negotiation. Limited awareness information is provided by the matrix, but there is a coupling mechanism allowing users to analyze a cell in more detail. This coupling mechanism leads users to the "Current Situation," where they can observe the negotiation state in detail, including the proposed correlation, positions in favor or against, and supporting arguments. We briefly characterize the two shared workspaces in terms of input, output, and coupling devices in Figs. 6 and 7.

Step 2: Breakdown definition of critical scenarios. We focus our discussion on the "Current Situation" shared workspace to illustrate the method application. The user arrives to this space with the purpose of analyzing the negotiation state in detail. As currently implemented by the tool, the status information is hierarchically organized, showing: (1) the product specifications and customer


Microsoft Excel - MegSystem

QFD Software House of Quality*

The *awareness output device* provides information about selected correlations (`0`,`1`,`3`,`9`) and ongoing negotiations (`?`,`F`,`L`).

The *coupling input device* allows the detailed analysis of a negotiation via double-clicking on a cell.

	B	D: In reply include original text for comments	E: Enter email via voice	F: Format email text differently	G: Add in graphical objects to mail text	H: Spell check	I: Sort addresses in address book	J: Create distribution list from adresses	K: Individual configuration of address imports	L: Create personal address book	M: Individual signatures	N: Manage various users	O: Manage different email addrresses for user	P: Compress files for attachment	Q: Data import/export internal-external editor	R: Integrate external editor	S: Attach different types of files	T: Support standard mail protocols	U: Save/group mails in different files	V: Structure mail files	W: Filter mails into files according to criteria	X: Distribute incoming mails to files by criteria
5	Write emails fast/easily	1	?						?													3
6	Write emails fast to many users				?									1								
7	Have overview of incoming mails		3			9																
8	Design mailbody individually									3		1								1		
9	Keep using existing address data																					
10	Find read mails fast																					
11	Reply to mails easily						1									9	9					
12	Manage other users' email addreesses										F											
13	Continue reading existing mail		1																			
14	Have overview of read mails																L					
15	Send mails of as little size as possible						3								3							
16	Delete read mails fast and purposely																					
17	Use well-known editor																			1		
18	Move addresses between email clients								9													
19	Dismiss junk mail (e.g. Advertising mails)																					

sw-HoQ

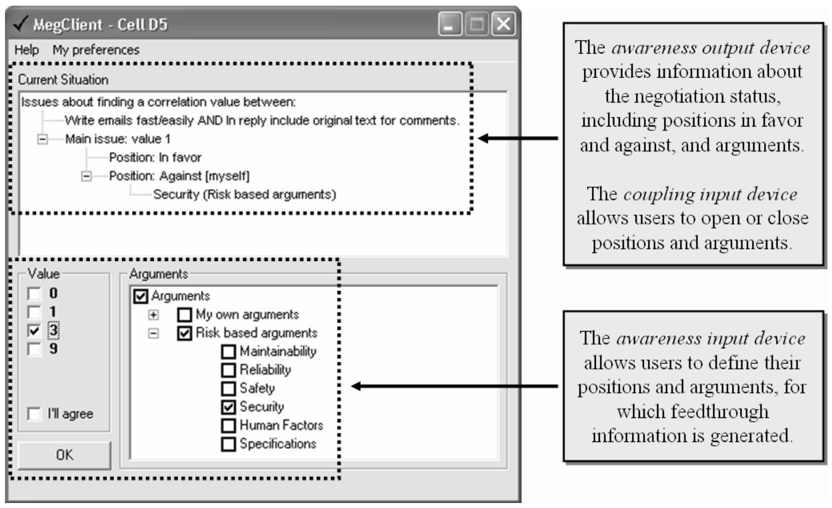
Fig. 6. The SQFD matrix

requirements under negotiation; (2) the currently proposed correlation; (3) positions in favor, followed by positions against the currently proposed correlation; (4) arguments supporting positions in favor or against. We designate this as design scenario A.

An alternative design scenario B considers a variation in the way information is shown to the user. We assume that users may give more importance to the aggregate information about the number of positions against/in favor, neglecting positions when there is a clear push towards one side or the other, and analyzing arguments in detail only when positions are balanced.

The selected critical scenario considers the proposal of an alternative correlation value in "Current Situation" after analyzing the negotiation status. We also consider a variation in the number of users involved in the negotiation process. The "Current Situation" may display the positions and arguments for up to 3 users (see Fig. 7). Beyond that number, a user has to scroll down the window to completely analyze the situation. Thus, we consider 3 and 6 users involved in the critical scenario. We assume that having more than 6 users negotiating the same cell is a rare event, which does not deserve further analysis.

Step 3: Comparing group performance in critical scenarios. For the design scenario A and 3 users, we have: the interpretation of the negotiation status, `M`,



Issues about finding a correlation value between:
Write emails fast/easily AND In reply include original text for comments.
Main issue: value 1
Position: In favor
Position: Against [myself]
Security (Risk based arguments)

The *awareness output device* provides information about the negotiation status, including positions in favor and against, and arguments.

The *coupling input device* allows users to open or close positions and arguments.

The *awareness input device* allows users to define their positions and arguments, for which feedthrough information is generated.

Fig. 7. The "Current Situation" shared workspace

followed by a decision, M, which is expressed via the selection of a check box, PKK, and a press in the "ok" button, PKK. This gives MMPKKPKK, which has a total execution time of 5.0 seconds. With 6 users, the execution time increases to 8.6 seconds, corresponding to MPKPK MMPKKPKK, in which the MPKPK operators are related to scrolling.

Considering the design option B, we have two situations: either the positions are balanced (a tie or simple majority) or unbalanced (i.e. absolute majority). In the unbalanced case, we assume the user will neglect arguments and thus we have MMPKKPKK (5.0 seconds to execute), similar to the previous scenario with 3 users. In the balanced case the user will analyze the positions in detail via the interpretation of the negotiation status, M, followed by the opening of the list of favorable arguments, PKK, and corresponding analysis, M, upon which the list is closed, PKK, to give room for the opening and interpretation of the against arguments, PKK M, so that, finally, the decision is made, MPKKPKK. The total execution time for the balanced case, MPKKMPKKPKKMMPKKPKK, is then 11.3 seconds. Note that these measures apply to the scenarios with 3 and 6 users. We also assume that the probability of having unbalanced positions is 25%[3]. Hence, in these circumstances, the average execution time for scenario B is about $0.75 \times 11.3 + 0.25 \times 5.0 \approx 9.8$ seconds, which is higher than scenario A for both 3 and 6 users. In other words, scenario B may be better or equal than scenario A, but there is a 75% probability that it is worse than scenario A, which severely penalizes the overall appreciation of the design in scenario B.

[3] This is the probability of having an absolute majority with 3 or 6 voters, assuming a uniform distribution. For 3 voters, the absolute majority requires having all in favor or against, i.e. 2 out of 8 combinations, or 25%.

6 Discussion and Implications for Design

Both the collaborative game and the requirements negotiation tool analyzed in this paper heavily depend on shared workspaces to orchestrate multiple users accomplishing a collaborative task. The design of these workspaces is thus critical to the overall task performance. Since we use a quantitative common criterion to evaluate group performance—the execution time predictions of collaborative actions in critical scenarios—we may benchmark various design solutions to estimate which shared workspace functionality offers the best performance.

It is important to note the two cases studied in this paper are quite distinct. The collaborative game is a fictitious tool intended to test preliminary design ideas in environments where players act opportunistically, while the requirements negotiation tool is a completely functional tool, that could nevertheless benefit from further optimizations. We analyzed several design solutions with the collaborative game related to the way users structure their actions according to awareness on other people. The requirements negotiation tool helped us to analyze how a coupling mechanism could be designed to conserve individual cognitive effort. We defined a critical scenario to evaluate the collaborative game highlighting coordination problems. By contrast, the critical scenario used to evaluate the requirements negotiation tool shows escalating problems with the number of users engaged in collaborative actions. Taken as a whole, the method always contributed to formative evaluation, offering clear indications about the potential performance of users working with shared workspaces.

The proposed method has two important limitations that we would like to discuss. First, it assumes a narrow-band view about collaboration, restricted to shared workspaces and their mediation roles. This contrasts with the other available groupware usability evaluation methods offering a wide-band view about collaboration, encompassing, for example, various communication channels, coordination policies, and broader issues such as group decision making or learning. However, the tradeoff to ponder is that the proposed method restricts the view in order to increase the detail about the mediating role of shared workspaces. This restricted view has ample justification in contexts where shared workspaces are heavily used, even when users perform intellective tasks (such as in the requirements negotiation case, where users apply their expertise to evaluate software quality, but are still requested to repetitively operate the tool).

Second, the method is somewhat limited by the selection of critical scenarios. As designers and evaluators, we have to ponder whether the selected critical scenarios are representative and have sufficient impact on the overall collaborative task to deserve detailed analysis. In our first case, the collaborative game, this question is delicate because the game was conceived to illustrate the method application with that critical scenario. However, the situation was quite different in the second case, because we started our analytic evaluation with a preliminary evaluation study indicating that the tool had usability problems [22]. Thus, some prior evaluation results allowed us to determine the critical scenarios for the subsequent evaluation task. We conjecture that this cyclic approach may reduce the bias introduced by critical scenarios. Furthermore, critical scenarios are commonly used as a sampling strategy in qualitative inquiry, allowing generalization [23]. The proposed method combines qualitative and quantitative approaches with the same purpose.

7 Conclusions

Confronting the obtained results with the driving forces mentioned in Sect. 1, we may conclude from this research that the proposed method can be used to quantitatively predict and compare the usability of shared workspaces, without requiring users or the development of functional prototypes. More specifically, available knowledge about human-performance models can be applied to predict execution times in critical scenarios involving intricate collaborative actions that have a potentially important effect on individual and group performance.

Research described in this paper is a preliminary step in the direction of exploring human-performance models to evaluate shared workspaces design. Our performance estimates were based on experimental measures of time spent by humans executing single user operations. Experimental research with groupware will be accomplished in the future, in an attempt to provide estimates for typical groupware interactions in critical scenarios.

Acknowledgments

This work was partially supported by the Portuguese Foundation for Science and Technology, Project POSI/EIA/57038/2004.

References

1. Fjermestad, J., Hiltz, S.: An assessment of group support systems experimental research: Methodology and results. Journal of Management Information Systems **15**(3) (1999) 7–149
2. van der Veer, G., van Welie, M.: Task based groupware design: Putting theory into practice. In: DIS'00: Proceedings of the conference on Designing interactive systems, New York City, New York, United States (2000) 326–337
3. Pinelle, D., Gutwin, C., Greenberg, S.: Task analysis for groupware usability evaluation: Modeling shared-workspace tasks with the mechanics of collaboration. ACM Transactions on Computer-Human Interaction **10**(4) (2003) 281–311
4. Pinelle, D., Gutwin, C.: Groupware walkthrough: Adding context to groupware usability evaluation. In: CHI'02: Proceedings of the SIGCHI conference on Human factors in computing systems, Minneapolis, Minnesota, USA (2002) 455–462
5. Baker, K., Greenberg, S., Gutwin, C.: Empirical development of a heuristic evaluation methodology for shared workspace groupware. In: CSCW'02: Proceedings of the 2002 ACM conference on Computer supported cooperative work, New Orleans, Louisiana, USA (2002) 96–105
6. Card, S.K., Moran, T.P., Newell, A.: The keystroke-level model for user performance time with interactive systems. Communications of the ACM **23**(7) (1980) 396–410

7. Min, D., Koo, S., Chung, Y.H., Kim, B.: Distributed GOMS: An extension of GOMS to group task. In: SMC'99: Proceedings of the IEEE international conference on Systems, man, and cybernetics, Tokyo, Japan (1999) 720–725
8. Kieras, D.E., Santoro, T.P.: Computational GOMS modeling of a complex team task: Lessons learned. In: CHI'04: Proceedings of the SIGCHI conference on Human factors in computing systems, Vienna, Austria (2004) 97–104
9. Card, S.K., Newell, A., Moran, T.P.: The psychology of human-computer interaction. Lawrence Erlbaum Associates, Mahwah, NJ, USA (1983)
10. Malone, T.W., Crowston, K.: The interdisciplinary study of coordination. ACM Computing Surveys **26**(1) (1994) 87–119
11. Beyer, H., Holtzblatt, K.: Contextual design: Defining customer-centered systems. Morgan Kaufmann Publishers, San Francisco, CA, USA (1998)
12. John, B.E., Kieras, D.E.: Using GOMS for user interface design and evaluation: Which technique? ACM Transactions on Computer-Human Interaction **3**(4) (1996) 287–319
13. Douglas, S.A., Kirkpatrick, A.E.: Model and representation: The effect of visual feedback on human performance in a color picker interface. ACM Transactions on Graphics **18**(2) (1999) 96–127
14. Wensveen, S.A.G., Djajadiningrat, J.P., Overbeeke, C.J.: Interaction frogger: A design framework to couple action and function through feedback and feedforward. In: DIS'04: Proceedings of the 2004 conference on Designing interactive systems, Cambridge, MA, USA (2004) 177–184
15. Hill, J., Gutwin, C.: Awareness support in a groupware widget toolkit. In: GROUP'03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work, Sanibel Island, Florida, USA (2003) 258–267
16. Gutwin, C., Greenberg, S.: The effects of workspace awareness support on the usability of real-time distributed groupware. ACM Transactions on Computer-Human Interaction **6**(3) (1999) 243–281
17. Rajan, S., Craig, S.D., Gholson, B., Person, N.K., Graesser, A.C.: AutoTutor: Incorporating back-channel feedback and other human-like conversational behaviors into an intelligent tutoring system. International Journal of Speech Technology **4**(2) (2001) 117–126
18. Gutwin, C., Benford, S., Dyck, J., Fraser, M., Vaghi, I., Greenhalgh, C.: Revealing delay in collaborative environments. In: CHI'04: Proceedings of the SIGCHI conference on Human factors in computing systems, Vienna, Austria (2004) 503–510
19. Cosquer, F.J.N., Antunes, P., Verissimo, P.: Enhancing dependability of cooperative applications in partitionable environments. Lecture Notes in Computer Science **1150** (1996) 335–352
20. Dewan, P., Choudhary, R.: Coupling the user interfaces of a multiuser program. ACM Transactions on Computer-Human Interaction **2**(1) (1995) 1–39
21. Haag, S., Raja, M.K., Schkade, L.L.: Quality function deployment usage in software development. Communications of the ACM **39**(1) (1996) 41–49
22. Antunes, P., Ramires, J., Respicio, A.: Addressing the conflicting dimension of groupware: a case study in software requirements validation (2006) To appear in Computing and Informatics.
23. Miles, M.B., Huberman, M.: Qualitative data analysis: An expanded sourcebook. Sage Publications, Thousand Oaks, CA, USA (1994)