Analytic Evaluation of Groupware Design

Pedro Antunes[1], Marcos R. S. Borges[2] , Jose A. Pino[3] , Luis Carriço[1]

[1]Department of Informatics – Universidade de Lisboa, Portugal
{paa,lmc}@di.fc.ul.pt
[2]Graduate Program in Informatics – Federal University of Rio de Janeiro, Brazil
mborges@nce.ufrj.br
[3]Department of Computer Science – Universidad de Chile, Chile
jpino@dcc.uchile.cl

Abstract. We propose an analytic method to evaluate groupware design. The method was inspired by GOMS, a well-known approach to analyze usability problems with single-user interfaces. GOMS has not yet been amply applied to evaluate groupware because of several fundamental distinctions between the single-user and multi-user contexts. The approach described in this paper overcomes such differences. We also illustrate the application of the model by applying it to the design of a collaborative tool for software engineering requirements negotiation.

1. Introduction

Groupware systems are becoming increasingly popular, yet many groupware applications still have usability problems [1]. In groupware, the computer system aims at supporting human-human interaction affected by variables such as group dynamics, social culture, and organizational structure [2]. These variables, whose values are sometimes unpredictable, make groupware difficult to design, especially when compared to traditional software [3].

The usability issue has long been recognized as an important aspect in the design of computer systems. In groupware it can have a strong impact both on the overall efficiency and effectiveness of the team, and on the quality of the work they do [3]. The design of groupware systems should consider the various aspects that affect their usability, but there are few proven methods to guide a successful design.

Many researchers believe that groupware can only be evaluated by studying collaborators in their real contexts, a process which tends to be expensive and time-consuming [4]. Ethnographic approaches can be utilized to evaluate groupware, but these techniques require fully functional prototypes, which are expensive to develop. Also important is the overwork generated when usability problems are detected at this stage. Redesigning the groupware requires work that could have been avoided if the problem had been detected during the initial design. We believe we can reduce these problems if we apply analytical methods to evaluate usability prior to the implementation.

In this paper we propose an analytic method based on validated engineering models of human cognition and performance to evaluate the usability of groupware systems. The proposed analytical method has been derived from GOMS [5, 6]. GOMS (Goals, Operations, Methods and Selection Rules) [5] and its family of models, such as GOMSL [6], offer an engineering solution to the analysis of human-computer interaction. GOMS has been successfully applied in several situations, to predict usability, to optimize user interaction, and to benchmark alternative user interfaces [7].

GOMS addresses singleware, i.e. one user interacting with one device. Although it is possible to conceive and model multiple user interactions with one device using GOMS, we realized that such an approach is not beneficial for groupware designers, in particular if they are concerned with shared space functionality.

GOMS is based on a cognitive architecture (user and physical interface) and a set of building blocks (goals, operators, methods and selections rules) describing human-computer interaction at a low level of detail. We investigated which modifications would have to be made to the cognitive architecture and to the building blocks to apply the same ideas to groupware. This is explained in Section 2. In Section 3 we describe the proposed method, derived from the groupware cognitive architecture. In parallel with the method description, and to demonstrate its applicability, we apply it to the design of a groupware tool for software engineering requirements negotiation. Finally, Section 4 concludes the paper.

2. GOMS and Groupware

In general, the GOMS family of models has been associated with the Model Human Processor [5], which represents human information processing capabilities using perceptual, motor and cognitive processors. However, significant architectural

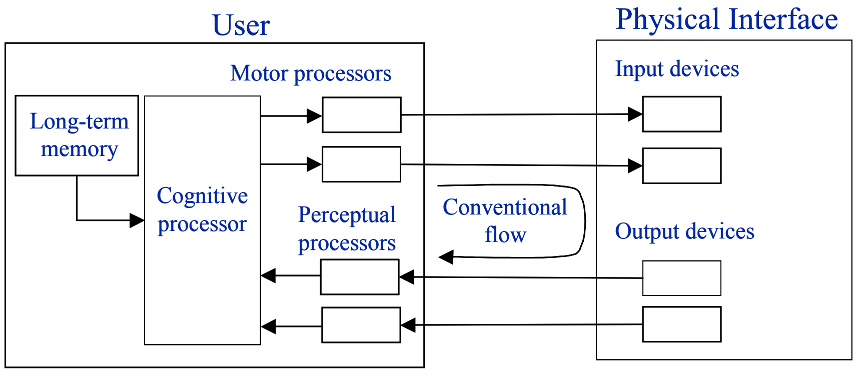


Fig. 1. Singleware architecture

differences are identified when considering individual models. For instance, KLM [8] uses a serial-stage architecture, while EPIC [9] addresses multimodal and parallel human activities. In spite of these differences, one characteristic common to the whole GOMS family of models is that it is *singleware* [10]: it assumes that one single user interacts with a physical interface comprising several input and output devices.

Figure 1 depicts this singleware architecture based on EPIC. According to some authors [11], this architecture applies to groupware in a very transparent way: in order to model a team of users, one can have several models, each one addressing the interaction between one user and the physical interface; and assume that (1) the physical interface is shared by multiple users and (2) the users will deploy procedures and strategies to communicate and coordinate their individual actions. Thus, groupware usage will be reflected in conventional flows of information, spanning several users, which still may be described using the conventional production rules and representations.

The problem however is that this approach does not reflect two fundamental issues with groupware: (1) the focus should move from the interactions between user and physical interface towards the more complex interactions between users, mediated by the physical interface; and (2) with groupware, the conventional flows of information are considerably changed to reflect multiple control centers and parallel activities. From our point of view, in order to address these groupware issues, we have to re-analyze how the physical interface handles communication flows and discuss its role in relation with multi-user interactions.

In the singleware context, we may characterize the conventional flow of information in two different categories: feedback and feedforward. The first category corresponds to a flow of information initiated by the user, for which the physical interface conveys *feedback* information to make the user aware of the executed operations [12]. The second category concerns the delivery of *feedforward* information, initiated by the physical interface, to make the user aware of the afforded action possibilities.

In groupware, however, some additional categories may have to be considered. We analyze three different categories: explicit communication, feedthrough and back-channel feedback. The *explicit communication*, as defined by [3], addresses information produced by one user and explicitly intended to be received by other users. This situation can be modeled as one physical interface capable of multiplexing information from one input device to several output devices [11]. The immediate impact on the model shown in Figure 1 is that we now have to explicitly consider additional users connected to the physical interface.

The *feedthrough* category concerns implicit information delivered to several users reporting actions executed by one user. This flow of information is initiated by the physical interface and it is directed towards the other users. A simple form of generating feedthrough consists of multiplexing feedback information to several users.

The notion of feedthrough has a significant impact on task modeling for several reasons. The first one is that feedthrough is essential to provide awareness about the other users and construct a context for collaboration. We can regard this type of information as being processed by the physical interface in specialized input and output devices, capable of processing sensory information about who, what, when, how, where are the other system users. The major purpose of this specialization is to make an analytic distinction between awareness and the other types of information

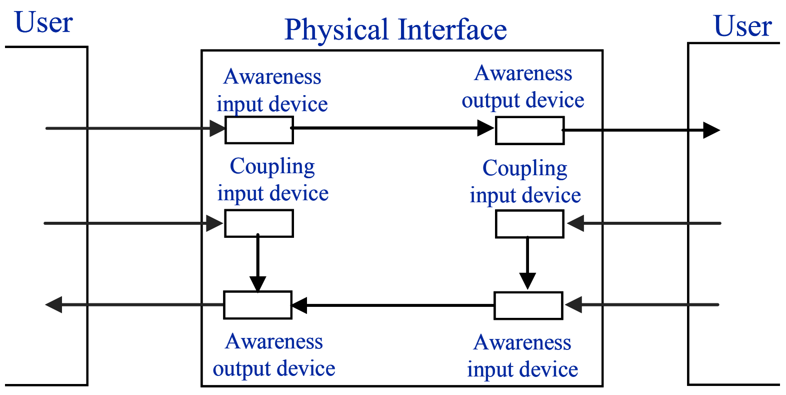


Fig. 2. Modifications to the physical interface required by the groupware architecture

mediated by the physical interface, so that we may focus on the former and avoid analyzing the later.

The proposed awareness output device also addresses one important groupware facet: not only it allows users to build a perceptual image of the collaborative context, but it also allows them to perceive the role and limitations of the physical interface as a mediator. This is particularly relevant when Internet is being used to convey feedthrough, causing delays which are significantly longer and less predictable than feedback delays [13].

The third reason for analyzing the impact of feedthrough is related to an important characteristic of groupware: it allows users to loose the link between executed operations and awareness – a situation called loosely coupled [14]. Two types of control are generally supported by groupware in a loosely coupled situation: (1) the user may get awareness information on a per-object demand basis, e.g. by moving the focus of interest; or (2) the user specifies filters that restrict awareness to some selected objects and types of events. In both cases this situation requires some cognitive activities from the user to discriminate and control awareness information, which can be modeled as a specialized input device devoted to control awareness information delivery.

Finally, the *back-channel feedback* category concerns unintentional information flows initiated by a user and directed towards another user to ease communication. No significant content is delivered through back-channel feedback, because it does not transmit user's reflection. Back-channel feedback may be automatically produced by the physical interface based on users' motor or vocal activities. We can model this type of activity in the physical interface as information flowing from a user's awareness input device to another user's awareness output device.

In Figure 2 we illustrate the modifications to the physical interface required by the groupware perspective. Our interpretation of the GOMS architecture, taking the groupware perspective in consideration, consists basically of modeling multiple users mediated by a shared physical interface; having awareness input and output devices, handling awareness information about the users operating in the system; and having coupling input devices, responsible for individually controlling the awareness information received by users.

Observe that this groupware architecture does not imply any modifications to GOMS, providing instead a contextualized framework adequate to a specialized application area, namely groupware. Also, the architecture does not address face-to-face situations where users exploit visual and body communication channels.

3. Method Description and Case Study

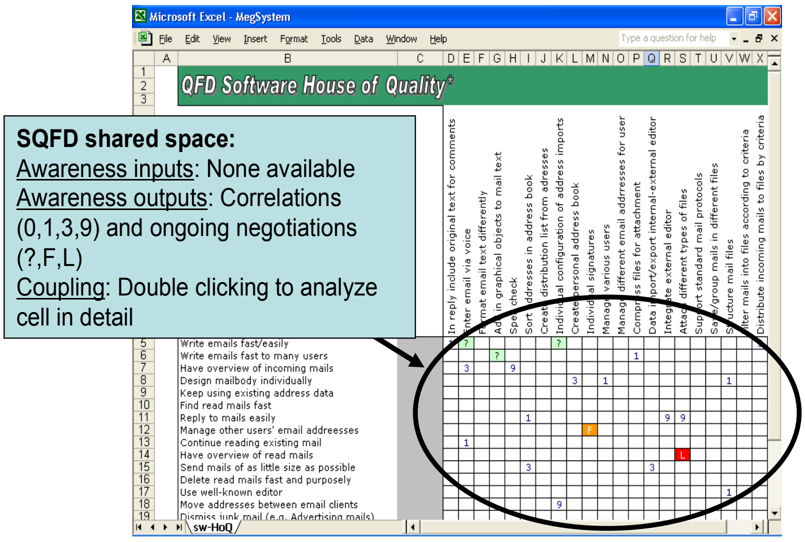
We will describe the proposed analytic method using a case study. The case study involved the development of a groupware tool for collaborative software quality assessment. The tool implements the Software Quality Function Deployment (SQFD [15]) methodology as the basic approach for evaluating software quality. The objective of this groupware tool is to facilitate the SQFD negotiation process, supporting mechanisms in a same-time, different-place mode. More details about this tool can be found in [16].

Step 1 – Defining the physical interface

Our first step consists in characterizing the physical interface of the groupware tool under analysis. Considering the complexity of many groupware tools, we divide this physical interface in several components, which we may designate shared spaces, defined as follows: a shared space is a distinctive combination of awareness input/output and coupling devices, capable of producing explicit communication, feedthrough and back-channel feedback.

In our case study, we find two shared spaces. The SQFD shared space, shown in Figure 3, 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 is provided in this space, but there is a coupling mechanism allowing users to analyze a cell in more detail. This coupling mechanism leads users to the "Current Situation" shared space shown in Figure 4.

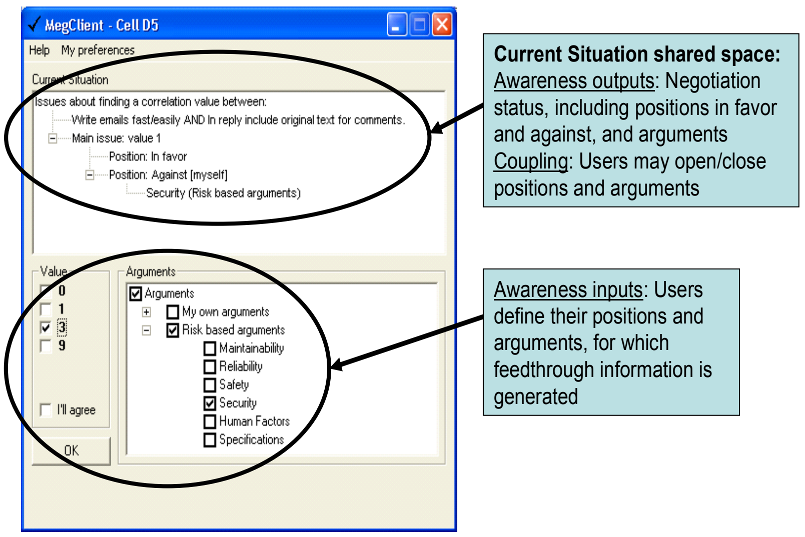
The "Current Situation" space displays the overall status of the negotiation process, reminding about the cell that is currently selected in the SQFD shared space and


Microsoft Excel - MegSystem
File Edit View Insert Format Tools Data Window Help
Type a question for help
QFD Software House of Quality*
In reply include original text for comments
Enter email via voice
Format email text differently
Add in graphical objects to mail text
Spell check
Sort addresses in address book
Create distribution list from adresses
Individual configuration of address imports
Create personal address book
Individual signatures
Manage various users
Manage different email addrresses for user
Compress files for attachment
Data import/export internal-external editor
Integrate external editor
Attach different types of files
Support standard mail protocols
Save/group mails in different files
Structure mail files
Filter mails into files according to criteria
Distribute incoming mails to files by criteria
sw-HoQ

SQFD shared space:
Awareness inputs: None available
Awareness outputs: Correlations (0,1,3,9) and ongoing negotiations (?,F,L)
Coupling: Double clicking to analyze cell in detail

5 Write emails fast/easily
6 Write emails fast to many users
7 Have overview of incoming mails
8 Design mailbody individually
9 Keep using existing address data
10 Find read mails fast
11 Reply to mails easily
12 Manage other users' email addreesses
13 Continue reading existing mail
14 Have overview of read mails
15 Send mails of as little size as possible
16 Delete read mails fast and purposely
17 Use well-known editor
18 Move addresses between email clients
19 Dismiss junk mail (e.g. Advertising mails)

Fig. 3. The SQFD shared space



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)

☐ 0
☐ 1
☑ 3
☐ 9
☐ I'll agree

☑ Arguments
☐ My own arguments
☑ Risk based arguments
☐ Maintainability
☐ Reliability
☐ Safety
☑ Security
☐ Human Factors
☐ Specifications

Current Situation shared space: Awareness outputs: Negotiation status, including positions in favor and against, and arguments
Coupling: Users may open/close positions and arguments

Awareness inputs: Users define their positions and arguments, for which feedthrough information is generated

Fig. 4. The Current Situation shared space

showing the various positions assumed by all negotiators. The bottom half of this space allows users to express their individual positions.

Note that the SQFD and "Current Situation" shared spaces provide several pieces of awareness information about the SQFD process. Considering the SQFD space, the users may perceive selected correlations (values 0,1,3,9) and negotiation status (?,F,L) for each cell. The top half of the "Current Situation" space displays the current correlation, positions in favor and against, and users' arguments supporting those positions. All this information is constantly updated according to feedthrough information obtained by the system.

Step 2 – Breakdown definition of collaborative activities

In this second step we describe the functionality associated to the identified shared spaces. Following the GOMS approach, the shared spaces functionality is successively decomposed from the more general to the more detailed.

Let us exemplify with our case study. The first method we illustrate in Figure 5 is the "Negotiate SQFD", describing how a user interacts with the SQFD shared space.

The method consists of mentally selecting a cell in the SQFD, analyzing its status, and deciding or not to negotiate the cell using the "Current Situation" space. The task is considered finished when the user accepts all values in the SQFD.

We show the additional details of two lower-level methods related with the SQFD space. The first one describes how users analyze the cell situation, which includes analyzing awareness information about the activities of others on the same cell in the SQFD space. The second method describes how a user accesses the "Current Situation" space.

```
Method: Negotiate SQFD
S1. Select cell.
S2. Analyze situation of cell.
S3. If want to negotiate value, then accomplish goal: Open Current Situation.
S4. If agreement on all cells, return with goal accomplished.
S5. Go to S1.

Method: Analyze situation of cell.
S1. Verify cell is empty or 0,1,3,9,?,F,L.
S2. Return with goal accomplished.

Method: Open Current Situation.
S1. Double click on cell.
S2. Return with goal accomplished.

Method: Negotiate value
S1. Analyze current situation.
S2. If do nothing, return with goal accomplished.
S3. If want other value, then accomplish goal: Propose alternative value.
S4. If insist on a value, then accomplish goal: Support proposed value.
S5. If agree with others, then accomplish goal: Withdraw proposed value.
S6. If change opinion, then accomplish goal: Change proposed values.
S7. If want to block, then accomplish goal: Block negotiation.
S8. If want to unblock, then accomplish goal: Unblock negotiation.
S9. If want firm position, then accomplish goal: Firm position.
S10. If remove firm position, then accomplish goal: Remove firm positions.
S11. If system is requesting confirmation, then accomplish Goal: Confirm value. Else go to S1.
S12. Return with goal accomplished.
```

Fig. 5. Methods describing shared spaces functionality (excerpt)

We also describe in Figure 5 the functionality associated to the “Current Situation” space, which is significantly more complex than the SQFD space. The “Negotiate value” method describes how users interact with this space at the highest level of detail. The users face several alternative actions while negotiating a value for the cell. Note in Step 11 that the system may request a confirmation from the user about the current correlation proposed for the cell. If all users agree, then the negotiation is considered finished for that cell.

Note also that the groupware tool gives the privilege to a user to block the interaction over a cell, a situation that is common in negotiations and used in various ways to increase individual gains. Another functionality supported by the tool is allowing a user to manifest a “firm” position about a cell value. In this situation, the tool asks the other users if they agree with the firm position. If everybody agrees, the negotiation of the cell is considered complete; otherwise, it is handled similarly to a blocking situation. The complete description of this functionality is very extensive and therefore omitted from the paper. We observe however that this step requires a significant amount of work to go down to methods describing the fine grained details of collaborative activities (e.g. consider that the “Current Situation” may require the user to scroll down to find the negotiation details, thus increasing the complexity of the “Analyze current situation” method).

Step 3 – Detailed analysis of collaborative activity definitions

In this step the method proceeds with a detailed analysis of the specification, as proposed by GOMS approach. The focus on collaborative activities derives from the application of the modified architecture perspectives on steps 1 and 2. We centre on a basic measure obtained from method definitions to benchmark design solutions: *cognitive workload*, defined as the number of steps specified in a method, including steps specified in lower-level methods (e.g. the "Negotiate SQFD" method has 9 steps).

Let us illustrate this analysis with our case study. An important goal that we had to accomplish when developing the SQFD tool was to make the negotiation of a cell a highly efficient task, since a SQFD matrix many times has hundreds of cells which may have to be individually negotiated. This optimization was mostly done by working on the cognitive workload measures of the "Negotiate value" and "Analyze current situation" methods. Both of them have high cognitive workloads for several reasons. The "Analyze current situation" method (not specified in the paper) has high cognitive workload because it examines how users perceive the current situation of a cell, which may already have been subject to a long negotiation process and requires the user to recall and go through several correlation values, issues, positions and arguments. This requires a significant number of verifications and decisions. A design solution to avoid this complexity, suggested by our analysis, consisted in structuring information in multiple levels, providing the most important information (positions in favor or against) in the first place, so that the user may "conserve" cognitive effort avoiding to go through the other information elements (so, the tradeoff was to have more methods with few steps).

The "Negotiate value" method (described in Figure 5) has high cognitive workload because of the number of decisions faced by the user: do nothing, propose, other value, change opinion, etc. Ten decisions were identified. However, analyzing the design implications, we preferred to concentrate all those decisions on one single method to optimize the time spent performing this necessary task (so, the tradeoff was to have few methods with many steps).

4. Discussion of Results and Conclusions

This paper presents an analytical method, derived from GOMS, to evaluated groupware usability during the design stage as a complement of traditional methods based on ethnography and real settings evaluation. The method is a revised version of a previous method discussed by the authors [17]. The example discussed in this paper illustrates well the proposed design method in which the whole collaboration process may be structured as a repetitive collection of smaller collaborative tasks orchestrated through a shared space. We believe the combination of this method with traditional usability evaluation is mostly adequate to successful groupware interface design.

The method has a strict focus on shared space functionality, allowing to benchmark different design solutions based on cognitive workload. The cognitive workload measures the presumed effort necessary to collaborate through shared spaces, based on the total number of steps defined in methods describing collaborative activities.

Unquestionably, one salient characteristic of this method is that it does not focus on collaboration as a process. For instance, the SQFD tool implements a negotiation protocol, where an initial bid is offered and other people negotiate their bids until everyone agrees (see [16] for a detailed explanation).

Although this process may be inferred by a detailed analysis of the method specifications, we argue the approach does not make it salient, giving importance to the mediating role of the shared spaces and the opportunities to optimize shared space usability. Therefore, this method should be regarded as complementary to other methods evaluating broader aspects of collaboration design. The fundamental implication for design raised by this method is that very fine-grained design decisions related with shared space usability might now be evaluated with an objective criterion: cognitive workload.

Acknowledgments

This work was partially supported by grant from CNPq Prosul. Professor Marcos R.S. Borges was partially supported by a grant from the "Secretaria de Estado de Educación y Universidades" of the Spanish Government. The work of Professor José A. Pino was partially supported by a grant from Fondecyt (Chile) No. 1040952.

References

1. Pinelle, D., Gutwin, C.: Groupware Walkthrough: Adding Context to Groupware Usability Evaluation. Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves. ACM Press, Minneapolis, Minnesota, USA (2002) 455-462
2. Grudin, J.: Groupware and Social Dynamics: Eight Challenges for Developers. Communications of the ACM 37 (1994) 92-105
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 (2003) 281-311
4. Steves, M., Morse, E., Gutwin, C., Greenberg, S.: A Comparison of Usage Evaluation and Inspection Methods for Assessing Groupware Usability. Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work, Boulder, Colorado, USA (2001) 125-134
5. Card, S., Moran, T., Newell, A.: The Psychology of Human-Computer Interaction. Lawrance Elrbaum, Hillsdale, NJ (1983)
6. Kieras, D.: A Guide to Goms Model Usability Evaluation Using Ngomsl. University of Michigan (1996)
7. John, B., Kieras, D.: Using Goms for User Interface Design and Evaluation: Which Technique? ACM Transactions on Computer-Human Interaction 3 (1996) 287-319
8. Card, S., Moran, T., Newell, A.: The Keystroke-Level Model for User Performance Time with Interactive Systems. Communications of the ACM 23 (1980) 396-410
9. Kieras, D., Wood, S., Meyer, D.: Predictive Engineering Models Based on the Epic Architecture for a Multimodal High-Performance Human-Computer Interaction Task. ACM Transactions on Computer-Human Interaction 4 (1997) 230-275
10. Ritter, F., Baxter, G., Jones, G., Young, R.: Supporting Cognitive Models as Users. ACM Transactions on Computer-Human Interaction 7 (2000) 141-173

11. Kieras, D., Santoro, T.: Computational Goms Modeling of a Complex Team Task: Lessons Learned. Proceedings of the SIGCHI conference on Human factors in computing systems. ACM Press, Vienna, Austria (2004) 97-104
12. Douglas, S., Kirkpatrick, A.: Model and Representation: The Effect of Visual Feedback on Human Performance in a Color Picker Interface. ACM Transactions on Graphics 18 (1999) 96-127
13. Gutwin, C., Benford, S., Dyck, J., Fraser, M., Vaghi, I., Greenhalgh, C.: Revealing Delay in Collaborative Environments. Proceedings of the SIGCHI conference on Human factors in computing systems. ACM Press, Vienna, Austria (2004) 503-510
14. Dewan, P., Choudhary, R.: Coupling the User Interfaces of a Multiuser Program. ACM Transactions on Computer-Human Interaction 2 (1995) 1-39
15. Haag, S., Raja, M., Schkade, L.: Quality Function Deployment Usage in Software Development. Communications of the ACM 39 (1996) 41-49
16. Ramires, J., Antunes, P., Respício, A.: Software Requirements Negotiation Using the Software Quality Function Deployment. In: Fuks, H., Lukosch, S., Salgado, A. (eds.): Groupware: Design, Implementation, and Use, Vol. 5807. Springer-Verlag, Heidelberg (2005)
17. Antunes, P., Borges, M.R.S., Pino, J.A., Carriço, L.: Analyzing Groupware Design by Means of Usability Results. In: Shen, W., James, A., Chao, K., Younas, M., Lin, Z., Barthès, J. (eds.): Proceedings of the Ninth International Conference on Computer Supported Cooperative Work in Design (CSCWiD '05). Coventry University UK, Coventry, UK (2005) 283-288