Analyzing Groupware Design by means of Usability Results

Pedro Antunes[1] *Marcos R. S. Borges*[2] *Jose A. Pino*[3] *Luis Carriço*[1]
[1]Department of Informatics, *Universidade de Lisboa, Portugal*
[2]*Graduate Program in Informatics – Federal University of Rio de Janeiro, Brazil*
[3]*Department of Computer Science – Universidad de Chile, Chile*
[1]*paa@di.fc.ul.pt, lmc@di.fc.ul.pt* [2] *mborges@nce.ufrj.br,* [3]*jpino@dcc.uchile.cl*

Abstract

GOMS is a well-known model that has been successfully used in predicting the performance of human-computer interaction, identifying usability problems and improving user-interface design. The focus of GOMS on the individual user, however, explains why it has not been applied in the groupware context. We were inspired by GOMS to define a model that describes collaborative tasks in a formal way. We 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 real 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 also expensive to develop.

We believe it is more productive to evaluate groupware through analytic engineering models for usability based on validated computational models of human cognition and performance. In this paper we propose the use of a method based on GOMS [5, 6] to evaluate the usability of groupware systems. Recognizing the strong theoretical and practical foundations of GOMS, we are interested on studying the applicability of some GOMS concepts to the collaborative context. Using this approach we intend to generate an effective way of finding usability problems early in the design of groupware.

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. This approach has been successfully applied in several situations, to predict usability, to optimize user interaction or to benchmark tools [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, since they are mainly interested on the collaborative context.

GOMS is based on a cognitive architecture (user and device) and a set of building blocks (goals, operators, methods and selections rules) that describe 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 order to demonstrate the applicability of the proposed approach, in Section 3 we use the model to describe a groupware tool; more specifically we describe a collaborative 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 [8], which represents human information processing capabilities using perceptual, motor and cognitive processors. However, significant architectural 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

9th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2005). Coventry, England: IEEE Computer Society Press, May, 2005.

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.

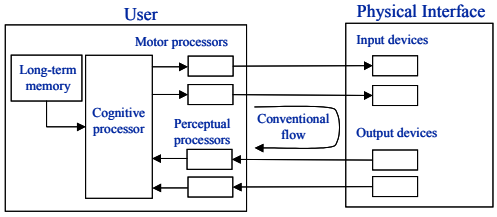

Figure 1 - Singleware architecture

Fig. 1 depicts this singleware architecture based on the EPIC architecture. 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 model each individual interaction between the 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 and granularity should not remain on the interactions between user and physical interface but should focus on the interactions between users, mediated by the physical interface; and (2) with groupware, the conventional flows of information are considerably changed to reflect e.g. parallel work. From our point of view, in order to address these groupware issues, we have to re-analyze the users' cognitive processing of the conventional flows of information and discuss them in relation with multi-user interactions.

In the singleware architecture 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 Pinelle et al. [13], addresses information produced by one user and explicitly intended to be received by other users. This situation can be modeled as one physical device capable to multiplex information from input devices 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 device.

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 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 the processing of awareness information in a specialized perceptual processor, called *awareness processor*, capable of processing sensory information about who, what, when, how, where are the other system users. We may also model the delivery of feedthrough to the awareness processor using a specialized output device, named *awareness output device*. Another characteristic of the awareness processor is that not only it affords 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 much higher and unpredictable than feedback delays [13].

The third reason for analyzing the impact of feedthrough is related to an important characteristic of groupware: it can afford 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 motor processor, called *coupling processor*. An input device will be devoted to control awareness information in the physical interface.

Finally, the *back-channel feedback* category concerns 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. That is the difference between explicit communication and back-channel feedback. We can model this type of activity as a motor activity executed by the coupling processor in response to some perceived inputs. The outputs produced by this motor activity will be delivered to other users through their awareness output device.

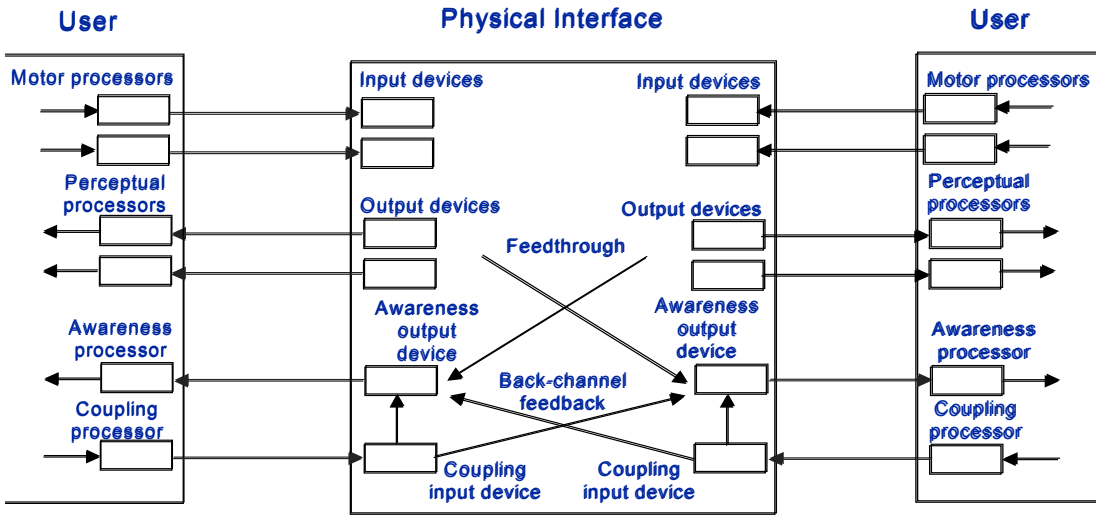

Figure 2 - Groupware architecture

In Figure 2 we illustrate the resulting groupware architecture. Our interpretation of the GOMS architecture, taking the groupware context in consideration, consists basically of introducing the awareness processor and output device, handling awareness information from the other users operating in the system; and the coupling processor and input device, responsible for controlling the amount of awareness information delivered to one user.

Observe that this groupware architecture does not imply any modifications to the GOMS architecture, 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. An Example

The following example is intended to briefly discuss the approach developed above. The example is centered on the design of a groupware tool for collective software quality assessment. Only an excerpt is shown. The full example can be found in [15]. Analyses of real situations using the proposed approach are also being carried out.

The tool implements the Software Quality Function Deployment (SQFD [16, 17]) methodology as the basic approach for evaluating software quality. According to this methodology, software quality is assessed by inspecting a matrix of correlations between a list of technical product specifications and a list of customer requirements. Each cell in this matrix indicates the strength of the relationship between a product specification and a customer requirement using the following numbers: 0, 1, 3 and 9 [16]. The final matrix should reflect a consensus of all evaluators.

The approach requires obtaining a consensus view from the customers about the software quality achieved along the software development process. However, considering they may have conflicting views, the matrix tends to be large and the cell values may have to be individually negotiated. The objective of this groupware tool is to facilitate this negotiation process, supporting mechanisms in a same-time, different-place mode. This is the main collaborative aspect of the tool.

Figure 3 shows our implementation of the SQFD, using a replicated MS Excel spreadsheet, where the rows represent the customers' requirements and the product specifications appear in the columns. The example shown in Figure 3 was taken from Herzwurm et al. [18]. Note that in Figure 3, besides the (0, 1, 3, 9) values, a cell may also be empty and have the following symbols: (?, F, L). These symbols mean respectively that the cell is being negotiated by several customers (?), one customer has a firm position about a correlation (F), and one customer locked the negotiation of a cell (L).

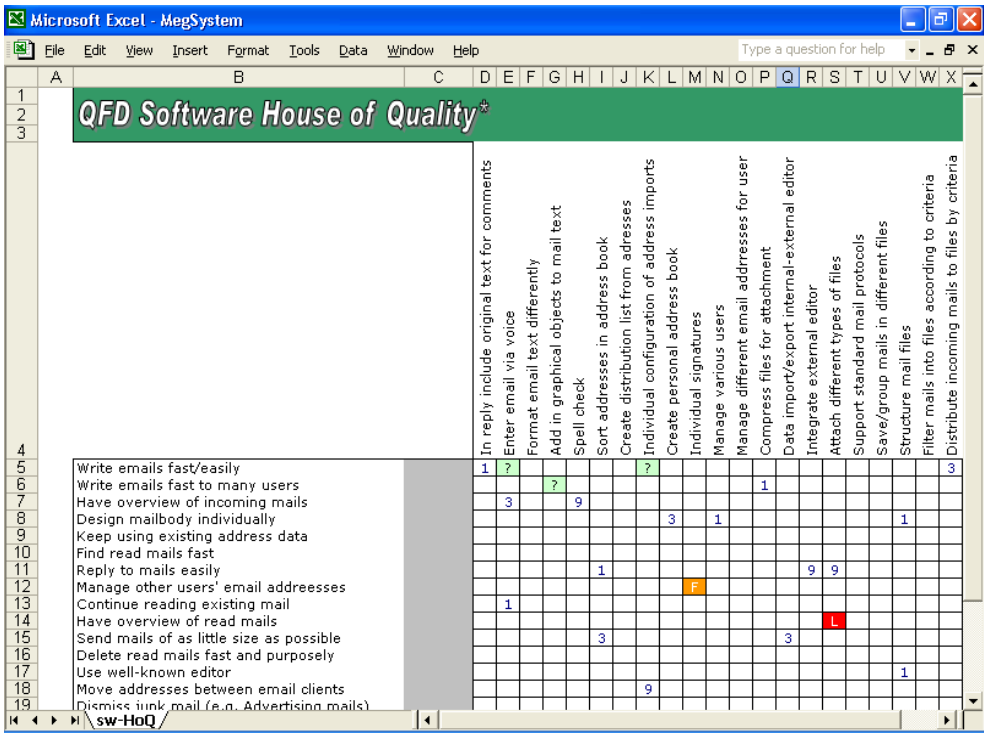

Figure 3 - The SQFD spreadsheet managed by the groupware tool (from [18]

Customers may have several attitudes towards the negotiation of a cell, revealed by suggestions of different values, compromising, and strong positions. At the limit, a customer may even decide to block or stall the negotiation process. The groupware tool must support these attitudes.

The negotiation process is supported by two main components: the MEG client and the MEG server. The MEG client and the MEG server use TCP/IP and RTD [19] technology to synchronize replicated MS Excel spreadsheets, residing in the users' personal computer. In this architecture, the users do not input cell values directly in the Excel spreadsheets, but in the MEG client. The MEG client interacts with the local replica of the MS Excel spreadsheets and with the MEG server. The MEG server is responsible for synchronizing the MEG clients, while RTD is used to synchronize the data on the repository with the Excel spreadsheets.

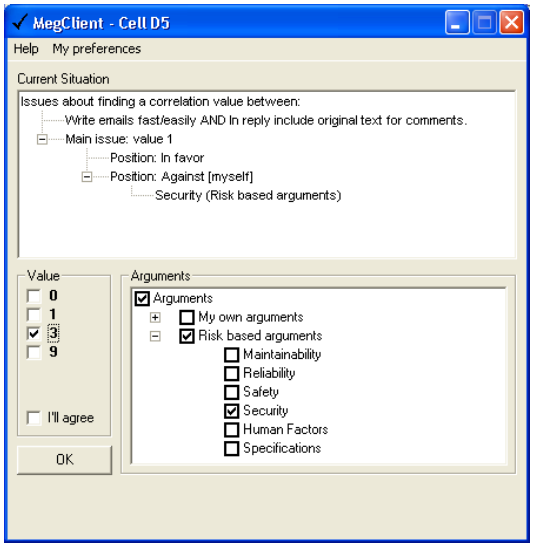

Figure 4 – The MEG user interface

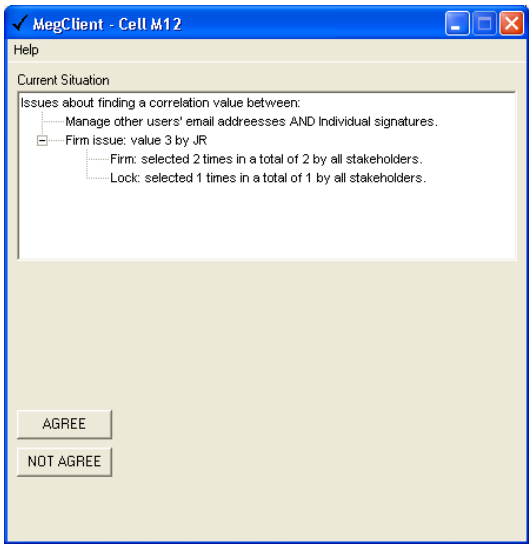

Figure 5 – The "firm" situation in MEG

The MEG client implements several user interfaces. Two of them are shown in Figures 4 and 5. The MEG user interface is divided in two major areas: the "current situation" area displays the overall status of the negotiation process, reminding about the cell that is currently selected in the SQFD spreadsheet and showing the different positions assumed by all negotiators; while the area bellow the "current situation" allows the user to express his/her individual position. Both the top and bottom areas change according to the status of the negotiation. The "current situation" area displays the following information:

- The elements correlated by the cell;
- The first value set by a user for the cell;
- The positions of other users in favor or against this value;
- The arguments supporting the positions of users.

Both the SQFD and MEG briefly explained above can be regarded as shared spaces. We will now describe part of the functionality of these shared spaces using the GOMS. First, let us specify the following objects that exist in the shared spaces:

cell: One cell of the SQFD spreadsheet for which a value must be agreed by the users
value: The correlation attributed to the cell
issues: The current status of the negotiation of one cell that is displayed to all negotiators
positions: The component of issues that lists the positions in favor or against the value currently in cell
arguments: The component of positions that lists the arguments supporting a position

The "Negotiate Spreadsheet" method describes how a user operates the SQFD. The method consists of analyzing the spreadsheet and deciding to propose or to negotiate the value of a cell using MEG. The proposal occurs when the cell is empty, while the negotiation occurs when the cell has already a value set. The task is considered finished when the user accepts all values in the spreadsheet.

Method: Negotiate spreadsheet
S1. Goal: Select cell.
S2. Goal: Analyze situation of cell.
S3. Decision: If want to propose value, then accomplish Goal: Propose initial value.
S4. Decision: If want to negotiate value, then accomplish Goal: Negotiate value.
S5. Decision: If agreement on all cells, return with goal accomplished.
S6. Go to S1.

There are two auxiliary methods to handle cell values. The first is intended to analyze the cell situation, which includes analyzing feedthrough information about activities of others on the same cell in the SQFD space. The second method describes the proposal of an initial value for an empty cell with MEG. This initial value has a special treatment by the users, because all of the subsequently proposed values will be presented by the system as being against or in favor of the first one.

The "Negotiate Value" method is dedicated to negotiate a cell value using MEG. The users have 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 value of the cell. If all users agree, then the negotiation is considered finished for that cell.

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

Method: Propose initial value.
S1. Accomplish goal: Select value from 0,1,3,9.
S2. Return with goal accomplished.

We may have several additions to the generic use of SQFD and MEG. One addresses the privilege given to a user to block the interaction over a cell, a privilege that is common in negotiations and used in various ways to increase individual gains.

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

Another functionality supported by MEG is allowing a user to manifest a "firm" position about a cell value. In this situation, MEG asks the other users if they agree with the firm position. If everybody agrees, the negotiation of the cell is considered completed; otherwise, it is handled similarly to a blocking situation.

4. Conclusions

The groupware tool analyzed provides a good example of what we called intensive collaboration, i.e., the whole collaborative task being a repetitive collection of smaller collaborative tasks, since users have to analyze and negotiate a large number of cell correlations to obtain a general consensus.

The results obtained from the example analysis do not focus on collaboration as a process. For instance, MEG implements a protocol for handling strong positions with the following steps: (1) a user defines a strong position on a value; (2) the other users are informed and questioned if they accept or not the proposed value; (3) the users respond; (4) MEG collects the responses and, if all agree, then the negotiation of the cell ends, otherwise the cell continues under negotiation, but blocked by a user. Although this process may be inferred by a detailed analysis of the described methods, we argue the approach does not make it salient, giving importance to the mediating role of the shared workspaces (spreadsheet and MEG) under the influence of such strong positions.

The implications for design raised by this model are twofold. The analysis of collaborative work using GOMS uncovers the mental conditions necessary to accomplish work, allowing the designers to specify shared artifacts that ease the users' grasping the design logic behind the tool. Designers may also compare different design options based on the analysis of the cognitive workload of a groupware tool.

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. and Gutwin, C., "Groupware Walkthrough: Adding Context to Groupware Usability Evaluation", Proc. SIGCHI Conference on Human factors in computing systems, Minneapolis, Minnesota: ACM Press, 2002, pp. 455-462.

[2] Grudin, J. "Groupware and Social Dynamics: Eight Challenges for Developers", Communications of the ACM 37 (1), January 1994, p. 92-105.

[3] Pinelle, D., Gutwin, C. and 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), pp. 281-311, 2003.

[4] Steves, M., Morse, E., Gutwin, C. and Greenberg, S. "A Comparison of Usage Evaluation and Inspection Methods for Assessing Groupware Usability", Proc. GROUP'01, Boulder, CO, September, 2001. ACM Press, pp. 125-134.

[5] Card, S., Moran, T. and Newell, A., The Psychology of Human-Computer Interaction. Hillsdale, NJ: Lawrance Elrbaum, 1983.

[6] Kieras, D., "A guide to GOMS model usability evaluation using NGOMSL". University of Michigan, 1996.

[7] John B. and Kieras, D., "Using GOMS for User Interface Design and Evaluation: Which Technique?" *ACM Transactions on Computer-Human Interaction* 3 (4), 1996.

[8] Baker, K., Greenberg, S. and Gutwin, C., "Empirical Development of a Heuristic Evaluation Methodology for Shared Workspace Groupware," in Proc. ACM Conference on Computer Supported Cooperative Work. New Orleans, 2002, pp. 96-105.

[9] Kieras, D., Wood, S. and 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*, vol. 4, no. 3, pp. 230-275, 1997.

[10] Ritter, F., Baxter, G., Jones, G. and Young, R., "Supporting Cognitive Models as Users," *ACM Transactions on Computer-Human Interaction*, vol. 7, no. 2, pp. 141-173, 2000.

[11] Kieras D. and Santoro, T., "Computational GOMS Modeling of a Complex Team Task: Lessons Learned." Proc. Conference on Human Factors in Computing Systems. Vienna, Austria: ACM Press, 2004, pp. 97-104.

[12] Douglas, S. and Kirkpatrick, A., "Model and Representation: The Effect of Visual Feedback on Human Performance in a Color Picker Interface," *ACM Transactions on Graphics*, 18 (2), pp. 96-127, 1999.

[13] C. Gutwin, C., Benford, S., Dyck, J., Fraser, M., Vaghi, I. and Greenhalgh, C., "Revealing Delay in Collaborative Environments." Proc. Conf. on Human Factors in Computing Systems. Vienna, Austria: ACM Press, 2004, pp. 503-510.

[14] Dewan P. and Choudhary, R., "Coupling the User Interfaces of a Multiuser Program," *ACM Transactions on Computer-Human Interaction*, vol. 2, no. 1, pp. 1-39, 1995.

[15] Antunes, P., Borges, M.R.S. and Pino, J.A., Carriço, L., "On the Analysis of Groupware Usability Using Annotated GOMS", Technical Report DI-FCUL-TR-04-18, Dept. of Informatics, University of Lisboa, Portugal, 2004.

[16] Haag, S., Raja, M. and L. Schkade, "Quality Function Deployment", *Communications of the ACM* 39 (1), pp. 41-49, 1996.

[17] Zultner, R., "TQM for Technical Teams", *Communications of the ACM* 38 (10), 1993.

[18] Herzwurm, G., Schockert, S., Dowie, U. and Breidung, M., "Requirements Engineering for Mobile Business Applications." Proc. First International Conference on Mobile Business. Athens, Greece, 2002.

[19] Cornell, P., Building a Real-Time Data Server in Excel 2002, Microsoft Corporation, 2001.