Chapter #

Modeling Style of Work as an Aid to the Design and Evaluation of Interactive Systems

James Wu, T.C. Nicholas Graham, Katherine Everitt, Dorothea Blostein and Edward Lank
Department of Computing and Information Science
Queen's University
Kingston, Ontario, CANADA K7L 3N6
{graham,wuj}@cs.queensu.ca

Abstract: This paper presents the *workstyle model*, a novel technique for recording the working style of people using an interactive system. Workstyle complements task modeling by providing information on how people communicate and coordinate their activities, and by showing what style of artifact the work is to produce. We have applied the workstyle model to the evaluation of UML design tools and the design of a new tool, the Software Design Board. The design of the model itself was informed by studies of tools and designers both at Queen's University and at a large software development organization.

Key words: Model-based design, work style, software design tools, UML

1. INTRODUCTION

Task models [5] help in understanding the goals and activities of users of a software system. Through this understanding, the designer of a system can ensure that users' tasks are supported. Techniques such as cognitive walkthrough [25], task simulation and model checking [18] allow user interface designs to be checked versus task models, exposing areas where the system fails to adequately support user tasks, or where user tasks are supported in an inconvenient manner. Task models therefore help to ensure that software systems meet users' needs, and meet these needs in a usable manner.

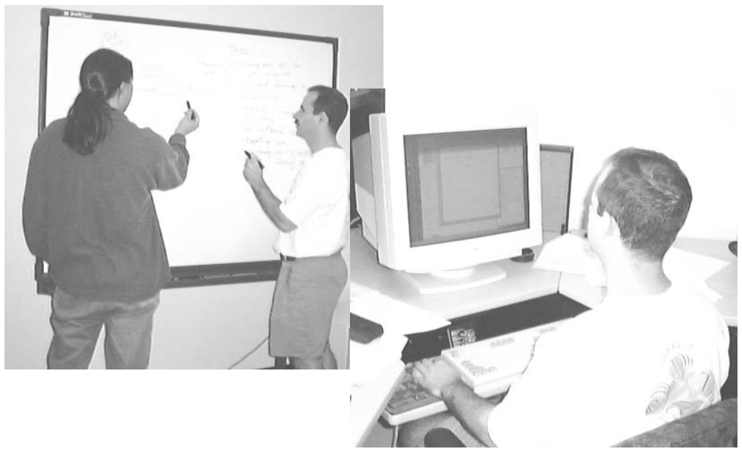

Figure 1. The workstyle model helps to identify task/tool mismatches. While a global task might be the creation of a UML design document, a whiteboard-based tool is more appropriate to brainstorming design, while a CASE tool is more appropriate to precise documentation.

Task models are less successful, however, in capturing what it means for a task to be supported in a usable manner. To address this problem, we present the *Workstyle Model*, a novel model capturing some aspects of users' preferred style of work. This allows analysis of whether an interactive system design supports users in performing their tasks, and of whether this support is consistent with the users' working style. The model concentrates on users' preferred collaboration styles and on the desired properties of artifacts developed through this collaboration.

The workstyle model was developed in the context of the Software Design Board project, a project aiming to provide better tools for software design. The model has been validated through evaluation of existing design tools, and has motivated the design of a new software design tool.

The paper is organized as follows. In sections 2 and 3, we motivate and introduce the model. In section 4, we present two case studies of the use of the model. The first is an electronic whiteboard-based tool supporting the creation of design diagrams in the Unified Modeling Language (UML) [20]. The second is a tool for collaborative design brainstorming.

2. MOTIVATION

In order to motivate the workstyle model, we consider the task of creating a design diagram in the UML notation. A task model can help us understand the activities involved in creating such a diagram: drawing and labeling nodes, connecting them with relations, editing and reformatting diagram elements, and so forth. Such a task model might lead us to develop a tool similar to Rational Rose, permitting mouse-based structural editing of design diagrams.

However, before committing to such a design, it is important to understand the users' preferred workstyle in addition to the tasks they need to perform. For example, designers may be working in a brainstorming style, or may be recording precise documentation from which the system is to be built. As illustrated by *Figure 1*, the brainstorming style is well supported by a whiteboard. A whiteboard provides sufficient space for small groups to work, and supports a fluid style of interaction where multiple designers may interact with the design artifact in parallel. The designers are also not restricted to following a precise design syntax. Alternatively, the precise design style is well supported by a traditional Computer-Aided Software Engineering (CASE) tool. Both tools support the tasks identified in the task model. However, they support the tasks in different ways, appropriate to either the brainstorming or precise design styles of work. As we shall see in the following section, the workstyle model supplements task information with information about users' preferred style of work.

3. THE WORKSTYLE MODEL

As motivated by the above example, we characterize the workstyle of a group in terms of how the group chooses to collaborate, as well as the kinds of artifacts the group chooses to create. As shown in *Figure 2*, the workstyle model characterizes working style as a space of eight dimensions. The first four describe collaboration style; the remaining four describe the properties of artifacts being created. A particular workstyle can then be represented as a point in this space. Interactive systems can also be plotted in this space, showing the workstyle(s) that they support. As will be shown in section 4, it is then possible to compare peoples' desired workstyles to those supported by tools, helping to identify potential tool mismatches.

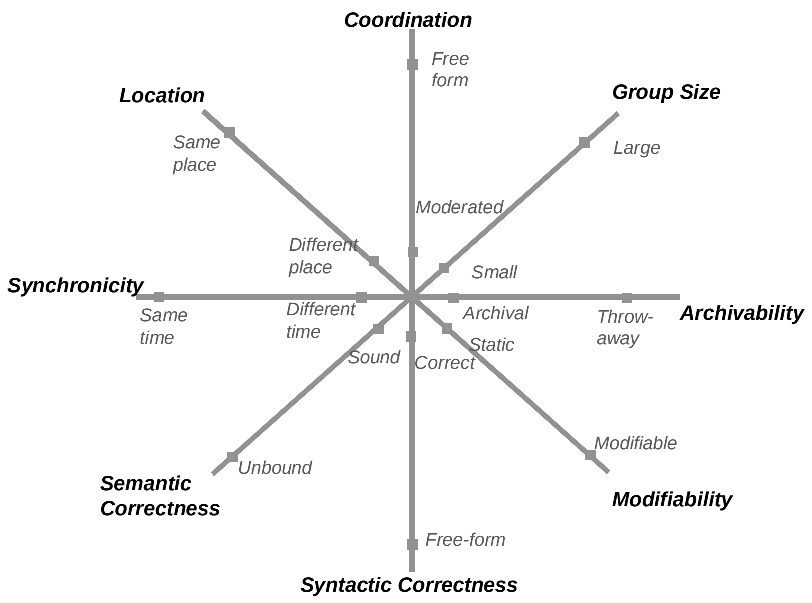

Figure 2. The workstyle model presents an eight-dimensional space on which working styles can be plotted. Workstyles are decomposed into collaboration style, and style of artifact being produced.

3.1 Dimensions Describing Collaboration Style

The first four dimensions describe how people choose to collaborate while performing their tasks. These are:

– **Location:** The location axis refers to the location of the people involved in the collaboration. People may be in the same place (*co-located*) or in different places (*distributed*.) In general, collaboration becomes more difficult at a distance [21], and requires more support from tools.

– **Synchronicity:** The synchronicity axis captures temporal aspects of the collaboration. People may work together at the same time (*synchronously*) or at different times (*asynchronously*). Face-to-face or telephone conversations are examples of same-time interaction, while email conversations or information sharing through a Lotus Notes database are examples of different-time interaction. Synchronicity is a continuous axis. E.g., a rapid exchange of emails can approach same-time interaction.

– **Group Size:** The group size axis captures the number of people involved in the collaboration. Group size influences what styles of interaction are practical. For example, brainstorming may work in small groups, whereas larger groups require support of tools or communication processes.

– **Coordination:** The style of interaction is influenced by the coordination model adopted [16]. For example, in a brainstorming session, free-form coordination is typical. Social protocols can determine the order of speaking or modifying shared artifacts. In meeting situations, more rigid coordination is typical, relying on a chair or even formal rules of order. Asynchronous tools typically rely on moderated coordination, such as check-in/check-out protocols.

3.2 Dimensions Describing Artifact Style

The remaining four dimensions describe the properties of artifacts that result from the users' tasks. The properties are:

– **Syntactic Correctness:** The artifact being produced may be required to follow a precise syntax. For example, a programmer must follow the rules of the programming language being used; a designer may choose to follow the precise rules of a notation such as UML. However, in the early stages of design, a requirement to adhere to precise syntax may hinder creativity by diverting designers from the global concepts of design [22,13].
– **Semantic Correctness:** An artifact is considered to be semantically *sound* if its meaning is clear: that is, unambiguous and free of contradiction. The production of semantically sound artifacts may be impractical and unnecessary. For example, design documents often contain contradictions, inconsistencies and missing information. According to Finkelstein [7], humans can work effectively in the presence of such contradiction. Semantic correctness is considered to be a continuum.
– **Archivability:** Archivability represents the difficulty of saving an artifact so that it can be used at a later time. For example, word processing documents have high archivability, as they can be saved to disk and retrieved later. A whiteboard has poor archivability, as once it is erased, its contents are lost. Archiving is considered more difficult if an archived version cannot be used in the same way as the original. For example, a whiteboard drawing can be archived by photographing it; however, the archived version (the photograph) can no longer be manipulated on the whiteboard.
– **Modifiability:** This axis represents the ease with which an artifact can be modified. Modifiability is closely tied to the modification task. For example, small modifications to a whiteboard drawing are simply performed by erasing and redrawing. A modification such as reformatting a complex diagram is, however, difficult. Modifications to a diagram produced using a drawing program are more completely supported, but may require complex editing operations.

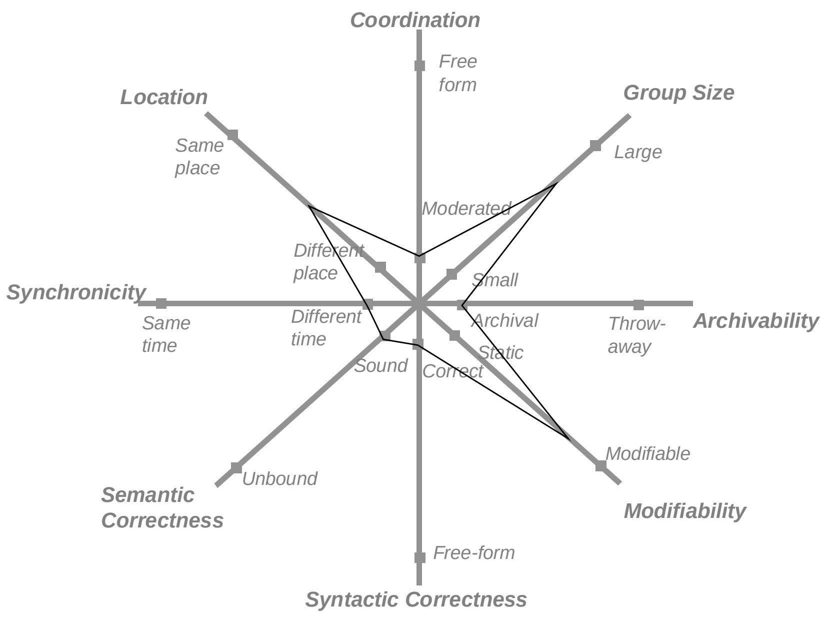

Figure 3. Workstyle supported by popular UML design tools, plotted using the workstyle model.

4. APPLICATION OF THE MODEL

The model presented in section 3 has been applied to the development of a UML design tool. We first used the model to show how existing UML design tools fail to match the work style of software developers. We then used the model to help motivate the design of a new tool.

It is useful to consider the workstyle supported by popular UML design tools such as Rational Rose, Aonix Software through Pictures (STP) and Together Control Center. As shown in *Figure 3*, these tools support small-to-medium groups. The activities of different designers are coordinated through a repository, using locking and merging to support concurrent work. Designers can work in different places. Communication is different-time, through the contents of the repository. The tools provide structure editing that guarantees that designs will be syntactically correct and semantically sound. Since the tools provide structure editing and the ability to save and restore designs, designs are highly modifiable and easily archived.

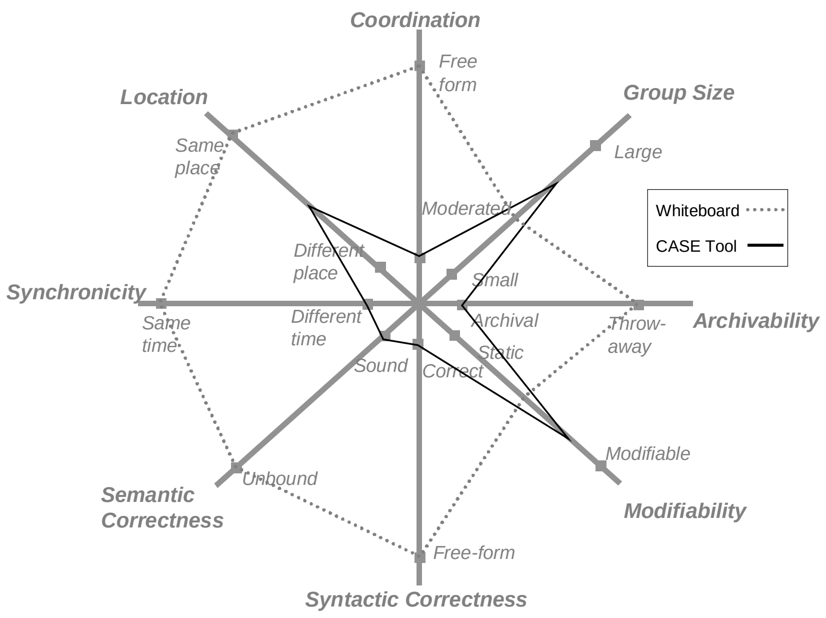

Figure 4. Ideal workstyle for brainstorming a software design overlaid with workstyle supported by existing UML CASE design tools

4.1 Tools for UML Design

UML design tools such as described above are a good fit with workstyles where relatively little communication with other designers is required, and where the goal is to create precise, archival designs. The tools are particularly helpful in enforcing UML syntactic and semantic rules, and provide good support for editing and archiving documents.

However, as discussed in section 2, traditional UML design tools provide poor support for the early stages of design, such as brainstorming. During these phases, designers spend large percentages of their time on communications tasks. For example, two studies show communication activities as requiring between 70% [4] and 85% [11] of software developers' time.

This communication is largely informal in nature: Kraut and Streeter [12] report that software developers report "discussion with peers" as their most important method of coordinating their activities. Following a large study of a software development project in the U.S. Navy, Norcio and Chmura report that informal discussion is correlated with progress in design, and that the more complex the design problem, the larger the percentage of designers' time is spent on discussion [2]. Studies of design tasks in general have shown that in brainstorming sessions, the majority of time is spent talking and gesturing at the artifact being produced [1,22]. In simple activity analysis of brainstorming tasks, we have observed that as little as 5% of brainstorming time is spent actually producing the design artifact, while the remaining 95% is spent discussing it.

At the same time, it has been observed that traditional design tools provide poor support for informal communication [24], and this lack of support has been linked to low adoption rates of these tools [9,10]. In studies of designers in a large software company, we observed that the use of UML design tools was low, and that much design was performed with informal media such as paper or whiteboards.

The inappropriateness of UML design tools for early stages of design can be clearly shown by examining the brainstorming workstyle. As shown in *Figure 4*, brainstorming is typically carried out by small groups working face to face. These groups typically use free-form coordination, using social protocols to determine who gets to speak or write next. In brainstorming, designers do not wish to be distracted by requirements to be syntactically correct, or even semantically sound [1,22]. Modifiability is important as early designs evolve rapidly. Archivability is important to allow early designs to be migrated to more formal designs.

Figure 4 shows that while UML design tools support the core tasks of the early stages of design, they do not support the workstyle of early design. The emphasis on asynchronous, moderated work with strong emphasis on syntactic correctness and semantic soundness is thoroughly incompatible with the free-form brainstorming workstyle. Ivarii's study shows that once designers move beyond this early design stage, they tend not to record the designs in a CASE tool unless management requires it. We therefore believe that *Figure 4* demonstrates a large part of the problem with current UML design tools.

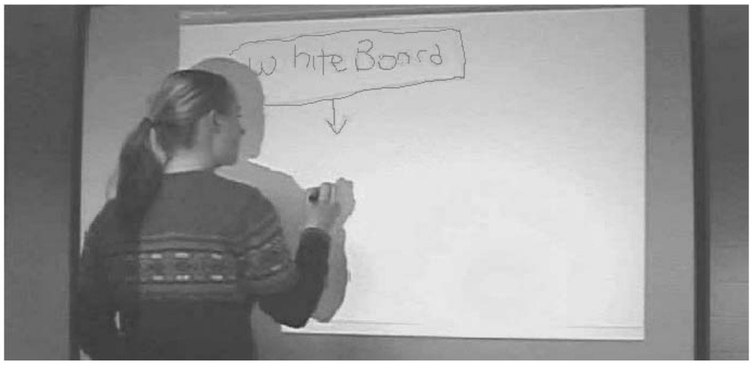

Figure 5. The Software Design Board: a tool for collaborative UML design

4.2 The Software Design Board

To address these problems, we have developed the software design board, a prototype UML design tool that supports a variety of workstyles appropriate to the early stages of design. As shown in *Figure 5*, the software design board is physically based on an electronic whiteboard. This whiteboard is a touch-sensitive membrane allowing drawings with a plastic stylus to be captured. The drawings are then projected onto the whiteboard display using a projector.

Using the software design board, designers can create UML designs simply by drawing them, similarly to the free-style drawing of the Tivoli system [19]. This gives the full flexibility of media such as paper or standard whiteboards. This contrasts with other whiteboard-based UML tools such as Knight [3], which is a structure-editor using a gesture language to specify editing operations. Once designs have been drawn, an experimental UML recognizer tool [14] can be used to turn them into structured drawings, reformat them, and export them in XML format suitable for use in a UML CASE tool.

The tool supports collaborative work in a variety of ways. Small groups can work together at the whiteboard, similarly to working with a standard whiteboard. Different-time work is supported (as with a traditional whiteboard), by leaving board contents available for others to look at. Different-place work is also supported, by allowing the whiteboard contents to be shared in real-time. I.e., shared drawing artifacts can be shared amongst two or more software design boards, so that changes to one board are immediately reflected on the others.

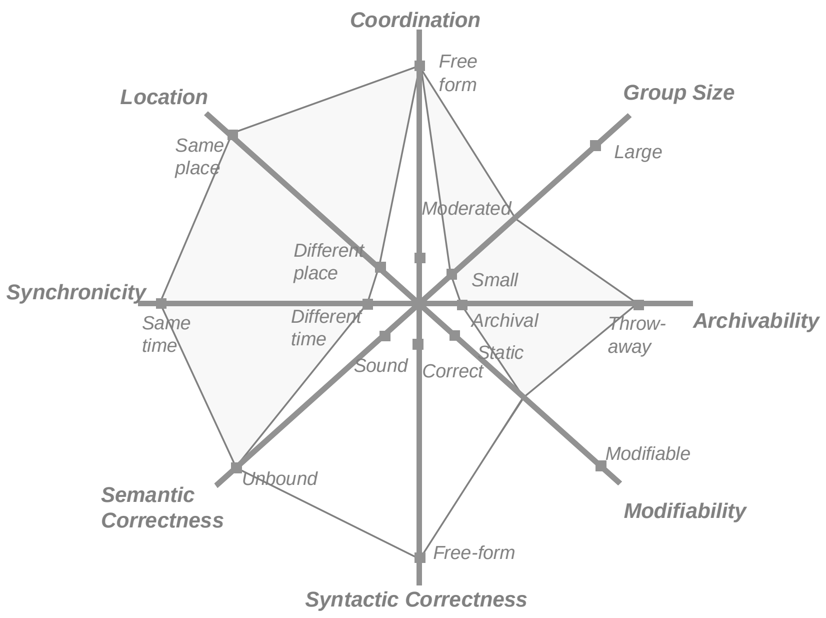

Figure 6. Workstyles supported by the software design board. These range over same-place and different-place, same-time and different time, small to medium groups. Consistently with early stages of design, coordination is free-form, with no restrictions on syntactic correctness or semantic soundness. Designs are somewhat modifiable through standard whiteboard erasing and redrawing. Designs can be archived by exporting them in XML to a standard UML CASE tool.

Since not all users may have electronic whiteboards available, the software design board can also be hosted on a standard PC, allowing whiteboard and PC users to share the same design. For same-time, different-place work, gestural information (pointing, circling, etc.) is also communicated (as motivated by the VideoWhiteboard [23].) On a PC, designers gesture using the mouse; on an electronic whiteboard, a camera is used to capture gestures.

4.2.1 Analysis

As shown in *Figure 6*, the software design board supports a broad range of working styles. These styles are a better fit with the early design workstyle (*Figure 4*) than the style supported by traditional UML design tools (*Figure 3*). Understanding work styles therefore helps us design tools that will better support the tasks to be performed.

The workstyle model complements a number other techniques for interactive system evaluation and design. For example, Lumsden has developed a sophisticated tool for matching CASE tools to their use [15]. Design notations such as OPAS [6] and our own dimension space [8] help in understanding the context of use of interactive systems, particularly how they fit within their physical environments. The Questions-Options-Criteria (QOC) method [17] helps evaluate design choices. While complementing these approaches, the workstyle model has the contribution of being simple to apply, and clearly showing where interactive system designs match or fail to match their intended work context.

5. CONCLUSIONS

In this paper, we have presented the *workstyle model*, a model capturing aspects of the style in which people carry out their tasks. The workstyle model complements task modeling by helping to develop deeper understanding of how tasks are to be performed. The model illustrates the ways in which people communicate during their work, and the properties of the artifacts that people produce.

We have shown that the workstyle model can be used to evaluate existing tools, and to help in the design of new tools. By examining the work styles supported by a tool, we can better understand how the tool can be effectively deployed, or identify mismatches between a tool and its intended use. By understanding the workstyles that a tool is to support, we can build tools that will be more effective in their deployment.

ACKNOWLEDGEMENTS

This work was partially supported by the Communications and Information Technology Ontario (CITO) Software Design Technology project and IBM's Centre for Advanced Studies. We gratefully acknowledge the donation by SMART Technologies Inc. of the SMART Board shown in *Figure 5*.

REFERENCES

[1] Bly, S.A. A Use of Drawing Surfaces in Different Collaborative Settings. In Proceedings of the *Conference on Computer-Supported Cooperative Work (CSCW'98),* Sept. 1988.
[2] Chmura, L. and Norcio, A. Design Activity in Developing Modules for Complex Software. In *Proceedings of Empirical Studies of Programmers,* pp. 99-116, 1986.

[3] Damm, C.H., Hansen, K.M., Thomsen, M., Tool Support for Object-Oriented Cooperative Design: Gesture-Based Modelling on an Electronic Whiteboard, in *Proc. CHI 2000*, pp 518-525, 2000.

[4] DeMarco, T., and Lister, T. *Peopleware*. Dorset House, New York, 1987.

[5] Diaper , D. Task analysis for human computer interaction, Ellis Horwood, 1989.

[6] Dubois, E., Nigay, L., Troccaz, J., Chavanon, O. and Carrat., L., Classification space for augmented surgery, an augmented reality case study, in Proc. INTERACT '99, 1999.

[7] Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., & Nuseibeh, B., Inconsistency Handling In Multi-Perspective Specifications, *IEEE Transactions on Software Engineering,* 20(8), pp. 569-578. 1994.

[8] Graham, T.C.N., Watts, L., Calvary, G., Coutaz, J., Dubois, E., Nigay, L., A Dimension Space for the Design of Interactive Systems within their Physical Environments, in *Proc. Designing Interactive Systems,* pp 406-416, 2000

[9] Iivari, J. Why are CASE Tools Not Used? *Communications of the ACM,* October 1996.

[10] Jarzabek, S. and Huang, R. The Case for User-Centered CASE Tools. *Communications of the ACM.* August 1998.

[11] Jones, T.C. *Programming Productivity.* McGraw-Hill, New York, 1986

[12] Kraut, R. E. and Streeter, L. Coordination in Software Development, *Communications of the ACM,* 38(3), pp. 69-81, 1995

[13] Landay, J.A. and Myers, B.A. Interactive sketching for the early stages of user interface design. In *Proceedings of CHI '95: Human Factors in Computing Systems,* Denver, CO, May 1995, pp. 43-50.

[14] Lank, E., Thorley, J.S., and Chen, S.J., An Interactive System for Recognizing Hand Drawn UML Diagrams, in *Proceedings of CASCON 2000*, pp. 1-15, 2000.

[15] Lumsden, J., Gray, P., SUIT - Context Sensitive Evaluation of User Interface Development Tools, in *Proc. DSV-IS'2000,* Springer LNCS, pp.79-95, 2000.

[16] Malone, T. W. and Crowston, K. What is coordination theory and how can it help design cooperative work systems?, in *Proceedings of the Conference on Computer-Supported Cooperative Work,* pp. 357-370, 1990

[17] McLean, A., Young, R.M., Bellotti, V.M.E., & Moran, T.P. Questions, options and criteria: Elements of design space analysis. *Human-Computer Interaction*, 6, pp. 201-250, 1991.

[18] Paternò, F. and Santoro, C., Integrating Model Checking and HCI Tools to Support Designers in Verification of User Interfaces Properties, In *Proceedings of DSV-IS'2000,* Springer LNCS, pp. 135-150, 2000.

[19] Pederson, E.R, McCall, K., Moran, T.P., Halasz, F.G., Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings, INTERCHI'93, pp 391-398, 1993.

[20] Rumbaugh, J., Booch, G., Jakobsen, I.: *The Unified Modeling Language Reference Manual*. Addison Wesley, 1999.

[21] Seaman, C.B. and Basili, V.R. Communication and Organization in Software Development: An Empirical Study. *IBM Systems Journal* 36(4), 1997.

[22] Tang, J.C. Findings from Observational Studies of Collaborative Work. *International Journal of Man-Machine Studies,* 1991.

[23] Tang, J.C and Minneman, S., VideoWhiteboard: video shadows to support remote collaboration, in *Proc CHI'91,* pp.391-398, 1991.

[24] Vessey, I. and Sravandapudi, A.P., CASE Tools as Collaborative Support Technologies. *Communications of the ACM,* 38(1), pp., 83-95, 1995

[25] Wharton, C., Rieman, J., Lewis, C., and Polson, P. The Cognitive Walkthrough Method: A Practitioner's Guide. In *Usability Inspection Methods,* J. Nielsen and R.L. Mack (Eds.), New York: John Wiley & Sons, pp.105-141, 1994.