

# *An Evaluation of Workspace Awareness in Collaborative, Gesture-Based Diagramming Tools*

**Christian Heide Damm and Klaus Marius Hansen**

*Department of Computer Science,  
University of Aarhus,  
Åbogade 34, 8200 Aarhus N, Denmark*

EMail: {*damm,klaus.m.hansen*}@*daimi.au.dk*

**Designing usable real-time, distributed collaboration tools is a complex but important task. *Workspace awareness* can potentially help in making real-time, distributed collaboration tools more usable through a communication of who is in the shared workspace and what they are doing. We present qualitative evaluations of the workspace awareness features of a gesture-based diagramming tool, *Distributed Knight*, that supports real-time, distributed collaboration. These studies suggest that using simple, non-intrusive awareness means results in fewer breakdowns, more symmetric collaboration patterns, better coordination, and higher perceived usability.**

**Keywords:** Real-time distributed collaboration, Diagramming tools, Gestures, Workspace awareness

## **1 Introduction**

Tools for real-time distributed collaboration are important (CNN 2001), but hard to implement well: Infrastructure, distributed communication, session management, privacy, presence, and awareness are examples of areas that pose design and implementation issues (Beaudouin-Lafon 1999). When working collaboratively at a distance, the awareness of who is working, where they are working on shared material, and what they are doing becomes particularly important. *Workspace awareness* encompasses means of making this information available in and through the shared workspace that collaborators are working on.

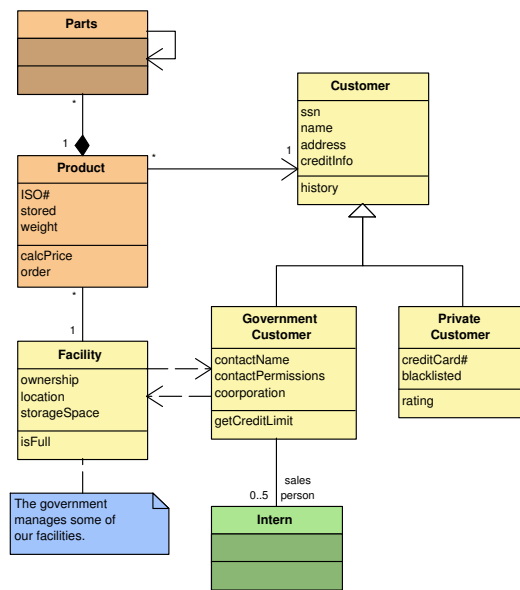


Figure 1: A Simple UML Diagram

This paper investigates workspace awareness in the context of a distributed version, *Distributed Knight*, of the *Knight* tool for collaborative object-oriented modelling (Damm, Hansen & Thomsen 2000). Object-oriented modelling is concerned with creating representations of concepts and phenomena in a problem domain as classes and objects in a solution domain for a software system being developed (Madsen, Møller-Pedersen & Nygaard 1993). The most common way to visualise object-oriented modelling is through the Unified Modeling Language (UML (OMG 2001)) notation. Figure 1 shows an example of a simple model in which, e.g., classes (representing concepts) are visualized as boxes, and associations (representing relationships) are visualized as solid lines between classes.

Whereas the *Knight* tool was designed to support *co-located* collaborative work (Figure 2), *Distributed Knight* adds support for *distributed* collaborative work. Tools for distributed work are becoming increasingly important (CNN 2001) as there is an increasing globalisation – also in the domain of software development. Different development groups within an organisation are distributed across geographical locations, and even different organisations may work closely together in joint or out-sourced projects. Consider as an example the following scenario:

Alice and Bob are heading a group working in Aarhus, Denmark, on a system for the US Department of Defence. They are collaborating closely with Chris, a domain expert who is located in the US. They often need to discuss specific details of the object-oriented model with Chris while they are in Aarhus. Also, on regular intervals, Alice or Bob

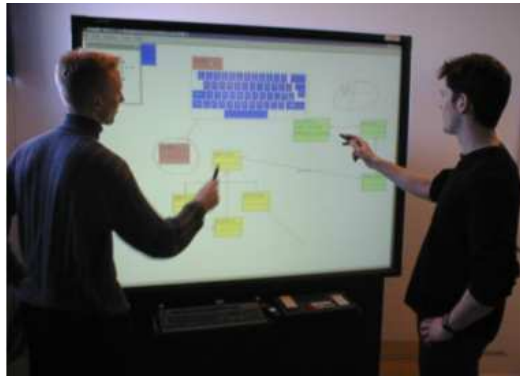


Figure 2: Using the Knight Tool on an Electronic Whiteboard

or both visit Chris in the US, and during their visits they need to be able to communicate with Aarhus regarding technical details of the project.

To fulfil this scenario a mixture of technology support for co-located and distributed collaboration in software development is needed. We have designed and implemented Distributed Knight to support scenarios such as these.

Although we do not claim that distributed collaboration can replace traditional face-to-face collaboration, we believe that distributed tools may make effective collaboration across geographical locations possible in situations where face-to-face collaboration is not feasible.

This paper is concerned with the usability of Distributed Knight and in particular with the specific use of workspace awareness and how it enhances the usability of the tool. The technical aspects of Distributed Knight have been reported elsewhere (Hansen & Damm 2004).

### ***1.1 Paper Structure***

The rest of this paper is structured as follows: Section 2 discusses the design of Distributed Knight for distributed, collaborative modelling with a particular focus on workspace awareness means. Section 3 then presents an evaluation of this design and discusses these results. Next, Section 4 discusses related and future work. Finally, Section 5 concludes.

## **2 Design of Distributed Knight**

The main interaction with the Knight tool is done through *gestural interaction*. In this context, gestures are marks drawn on the workspace and subsequently recognized by a computer tool (Buxton 1986). In Knight, gestures resemble what a user might have drawn on an ordinary whiteboard during modelling. As an example, Figure 3 shows how a user may create a UML class. This kind of interaction design enhances learnability and also makes it possible for, e.g., domain experts to participate in modelling sessions to a much



Figure 3: Left: A box stroke. Right: The class that is the result of a box stroke

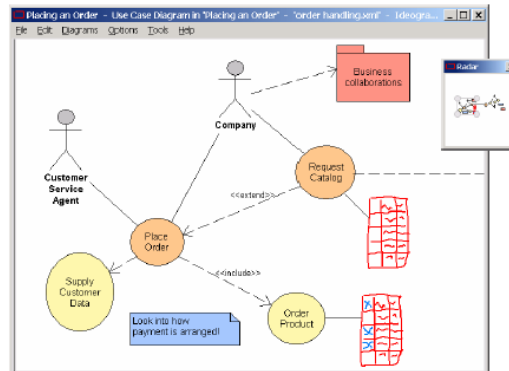


Figure 4: The user interface of the Knight tool

higher extent than what is possible with traditional UML tools. The Knight tool has been commercialized as *Ideogramic UML* by Ideogramic ApS (<http://www.ideogramic.com/products/uml/>)

The use of gestural interaction makes Knight usable on a range of input devices ranging from tablet PCs over desktop PCs to electronic whiteboards. The interaction style of Knight also makes an effective implementation of real-time distributed collaboration potentially usable on all these devices.

The Knight tool, and subsequently Distributed Knight, has been designed through an iterative, participatory design process. Following observations of object-oriented modelling, we iteratively designed and implemented the Knight tool (Damm et al. 2000). One of the major outcomes of the observations and participatory design was that there was a need for a tool that supported informal and incomplete models as well as formal. Figure 4 shows the user interface of Knight: It is basically a white surface, much like a whiteboard, on which users draw strokes that are interpreted as gestures.

The informality of the interface and the learnability of Knight were usability characteristics what we decided to pursue in a distributed version of Knight. In particular, we wanted to support ad-hoc collaboration in a fluent way. Take the *Cittera* tool (<http://www.canyon-blue.com>) as a counter-example. *Cittera* is a commercial tool that enables real-time distributed modelling in which semantic changes are immediately available to collaborators. To collaborate with others, a

user needs to

1. Coordinate a collaboration session with potential collaborators
2. Create a model and give it a meaningful name
3. Make the model a “shared model” so that multiple users may collaborate on it
4. Assign “write” privileges to collaborators

While reasonable when amortized over the life-time of a long-lived model, this is not very usable for spontaneous collaboration. Based on observations of, interviews with, and logs from instant messaging users, we design the session management component of Distributed Knight to use a context-aware instant messaging client, “Aware Messenger” (Hansen & Damm 2002) (Figure 5).

Using this client, users may see other users’ statuses and locations, may initiate and invite to sessions, and may see the active sessions and the participants of these.

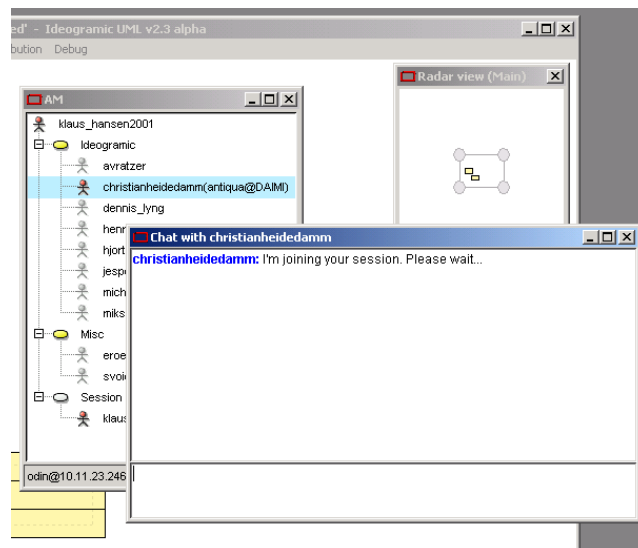


Figure 5: AwareMessenger

### 2.1 Workspace Awareness

In the evaluations described in this paper, the following workspace awareness facilities in Distributed Knight were relevant:

- *Intermediate updates*: If a user is drawing a gesture, moves an element, or text editing an element, a representation of these actions is shown in the user’s collaborators’ workspaces while the actions are being carried out. Figure 6 shows an example of this. This functionality basically aims to show what the user is working on.

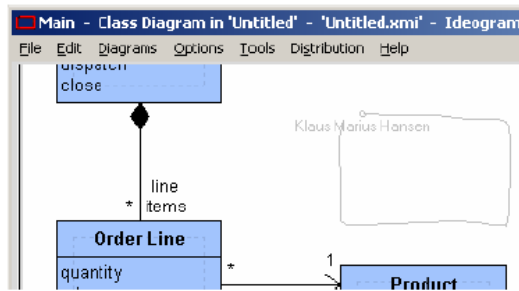


Figure 6: A collaborator is drawing a box gesture

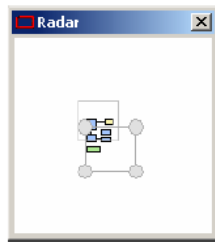


Figure 7: The radar shows the user's viewport (with handles for zooming) and a collaborator's viewport

- *Radar view:* A small radar view shows an overview of the workspace in Knight. In Distributed Knight, the radar also shows the viewports of collaborators (Figure 7). This widget resembles the radar widget of (Gutwin & Greenberg 1999) and aims at showing where the user is working.
- *Cursor representation:* The cursor of a collaborator is shown in a user's workspace if their viewports overlap. If their viewports do not overlap, a line indicating the direction of the collaborator's cursor and the distance to it is shown (Figure 8). This should help in locating where a collaborator is. The cursor line provides information that in some ways overlap with the information of the radar widget (see Section 3).

Other awareness facilities include information on who joins and leaves a session and various awareness information in Aware Messenger.

## 2.2 Implementation and Status

The software architecture of Distributed Knight is based on a peer-to-peer replication of data which eliminates the need for a central server (Oram 2001). The actual distribution of data is built using a variant of the publish-subscribe paradigm for distributed computing (Eugster, Guerraoui & Damm 2001). This allows for a decoupling of clients so that, e.g., Aware Messenger could be implemented without any changes to Distributed Knight.

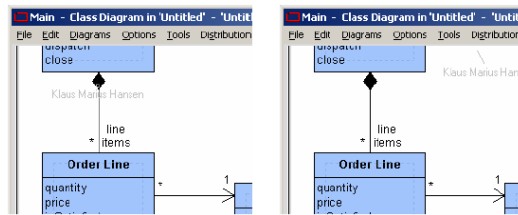


Figure 8: Cursor awareness in Distributed Knight

We have implemented distribution of class diagrams and this part of Distributed Knight is almost complete and has been in use by members of our research group and people at Ideogramic while being developed. One area in which little work has been done is that of concurrency control, i.e., making sure that actions by one user does not interfere with actions made concurrently by other users. Although this most certainly needs to be implemented, the evaluations showed that the workspace awareness features eliminated most concurrency problems (Section 3).

### 3 Evaluation

To evaluate the design of Distributed Knight, a usability study was made. The study had a particular focus on workspace awareness with the hypothesis that workspace awareness would increase and enhance collaboration. This evaluation is described in detail below.

#### 3.1 Participants

Following a pilot study with two participants, we arranged four sessions with two participants in each session. The participants were volunteers from the Computer Science Department, University of Aarhus and from a private company. None were students and most were experienced with software development. Since the goal was to evaluate Distributed Knight, the participants needed to be familiar with UML-based modelling using and with using stand-alone Knight. Thus, training was part of the study (see Section 3.2).

#### 3.2 Procedure

Each evaluation session consisted of five steps centered around a collaborative modelling task:

1. Participant consent. Introduction to UML and the Knight tool
2. Presentation of workspace awareness means in Distributed Knight
3. Introduction to task. Individual reading of requirements
4. Collaborative modelling task for 2\*25 minutes
5. Post-study questionnaire with a 1-to-5 Likert scale and an open-ended interview



Figure 9: Experimental setup

One experimental condition started with workspace awareness enabled in the modelling task and one started with workspace awareness disabled. After 25 minutes this was reversed. Participant pairs were assigned randomly to each condition. In the experimental condition that started without workspace awareness, step 2 was performed after the first half of the modelling session (step 4).

In each session, the participants were given a predefined assignment that they needed to work on collaboratively. The assignment was open-ended and complex: The participants had to model an administrative system for the University of Aarhus given a number of requirements that the final system had to fulfil such as handling salary, vacation, employment, students, administrative structure, buildings, and equipment. This assignment was purposely too large to finish in the given time frame meaning that the participants would have the freedom to choose to work in detail in areas or working more broadly on different areas of the model.

Both participants were given a broad description of the domain and requirements for a specific part of the domain. One participant was told to act as a domain expert of the area involving among others personnel and the other participant was asked to act as a domain expert for the area involving among others building administration. All participants reported the task to be realistic in the post-study interview.

The experimental setup is shown in Figure 9. Participants were located in the same room and physical distribution was simulated by placing a board between the workstations of the participants. Thus, no technology was used for audio transmission, and audio was near perfect quality and there was little network latency, eliminating a source of noise in the evaluation. Moreover, the setup enabled us to observe both participants at the same time. When using Distributed Knight in normal work, users are expected to use audio in combination with an instant messaging system or simply by using a phone.

During the evaluation, we observed and videotaped the participants. Observations were made with particular focus on focus shifts and breakdowns

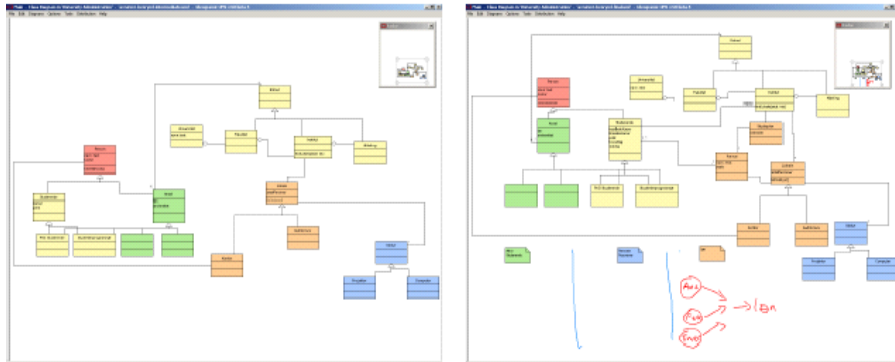


Figure 10: Examples from an evaluation session

(Bødker 1996). If the participants had trouble using the UML or the Knight tool, we would help them since the focus was on evaluating the usability of the distribution mechanisms in Distributed Knight. The model was saved after 25 and 50 minutes and logs of the users' actions were collected.

### 3.3 Discussion

In general, participants found Distributed Knight usable for distributed modelling (4,1 on a Likert scale) and all participants were able to produce quite elaborate models. For the type of scenario in the evaluation, Distributed Knight appears suitable for ad-hoc, real-time, distributed UML modelling.

Figure 10 shows an example of a model produced during the evaluation.

The left side shows the model as it looked after the first 25 minutes and the right side shows how the model looked af completion of the evaluation. The figure shows a typical pattern: In the first part of the evaluation, participants would work on the overall structure of the model and in the second part participants would go into detail with the model regardless of experimental condition.

#### 3.3.1 Suitability

Participants found workspace awareness important (3,9) and found the amount of workspace awareness facilities suitable in Distributed Knight (4,0).

The single most important awareness means was awareness through audio, however (5,0). We did not provide video for the evaluation and participants were neutral in whether or not the found video would have been useful (3,1). This evaluation thus adds to the ongoing debate on how important video is for awareness (Isaacs & Tang 1994, Mackay 1999, Veinott, Olson, Olson & Fu 1999)

In this evaluation, the most important workspace awareness features were awareness of movement of elements (4,0) and awareness of collaborators' cursors (3,8). Only the the direction line did not score above neutral in the questionnaire (2,8). This may be due to that the information could be deduced from the radar view and due to the fact that for large parts of the evaluation, participants worked with overlapping viewports. Interestingly, all awareness features except the direction line

were either judged very important or important by some participant. Conversely, all awareness features except awareness of movement of elements were either judged very unimportant or unimportant by some participant. This could indicate that all awareness features, except the direction line, were important and that none may be easily removed meaning that the choice of awareness features in Distributed Knight was reasonable and balanced.

It may be surprising that the rather simple awareness features in Distributed Knight provide sufficient support for real-time distributed collaboration. We believe that there are two reasons for this. First, the use of *workspace* awareness seems like a good choice in a diagramming tool such as Distributed Knight, where, by nature, a lot of the work and attention of the collaborators is centered on the shared workspace. Secondly, by using *gestural interaction*, the user can keep focus on the workspace, unlike in many traditional diagramming tools that make heavy use of toolbars, dialog boxes etc.

### 3.3.2 Collaboration Patterns

When working without workspace awareness, participants tended to work more in a turn-taking fashion than in parallel or tightly interleaved. This led to asymmetric collaboration patterns which may be problematic in cases such as the task of the evaluation. Figure 11 shows an example of this. The graphs show the number of semantic changes (creation, update, or deletion of an element) over time.

In the first half, with workspace awareness, the participants are equally active in creating the model, but in the second half, without workspace awareness, one participant does most of the modelling. This pattern was found in three of four evaluations.

### 3.3.3 Breakdowns and Focus Shifts

In general we experienced very few breakdowns in the workspace awareness condition. In this case most breakdowns were due to technical problems. In fact participants reported that the workspace awareness features of Distributed Knight were non-intrusive (4,1).

Two almost identical incidents in one of the sessions clearly show the importance of workspace awareness: During the first 25 minutes without workspace awareness, the two participants wanted to jointly move two groups of elements. There was some confusion as to which elements should be moved, and who should do it, and in the end, both participants moved a group, but in opposite directions, in effect counteracting each other by accident. During the last 25 minutes with workspace awareness enabled, the two participants wanted to swap two groups of elements, but this time they could see that the collaborator was moving the other group, and were thus able to coordinate their activities.

## 4 Related and Future Work

Awareness of the work of others is important in collaborative work (Dourish & Bellotti 1992). In cases such as Distributed Knight in which users can work in a synchronous, decoupled mode on a workspace that is larger than the viewport of a user, workspace awareness becomes particularly important. Gutwin and

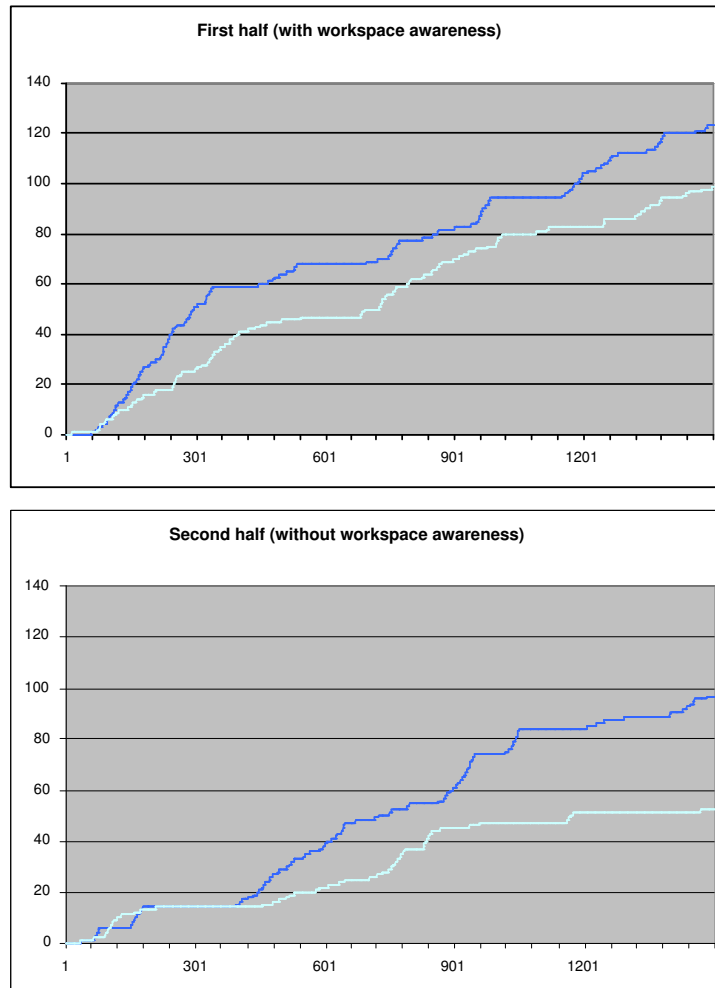


Figure 11: Example of activity levels in an evaluation session. The X axis shows time in seconds. The Y axis shows accumulated number of semantic changes

Greenberg have investigated workspace awareness in various settings (Gutwin & Greenberg 1999, Gutwin & Greenberg 2002). We extend this work by looking at workspace awareness in the case of distributed diagramming tools and pointing to how workspace awareness may enhance usability.

There are a number of UML tools for synchronous, collaborative modelling, but none employ workspace awareness features extensively: Workspace awareness in Cittera, e.g., is rather limited; the only indication of where in diagrams collaborators are currently working and what they are currently working on is made through a small lock symbol on UML elements. A textual activity log gives some sense of the history of collaboration and summarises the actions of all collaborators.

Embarcadero Describe (<http://www.embarcadero.com/products-/describe>) also provides real-time collaboration in which changes made are visible to other collaborators after an edited element has been deselected. Embarcadero Describe has even less workspace awareness than Cittera: There is no visual indication of what collaborators are currently working on, and only if a user actually tries to edit an element that is being edited by another user is a dialog box shown indicating this.

The COAST framework for synchronous groupware (Schuckmann, Kirchner, Schümmer & Haake 1996) has a real-time collaborative UML tool, *UMLEditor*, as a demo example with some workspace awareness facilities (<http://www.opencoast.org/download/>). This demo example was primarily developed as a way of investigating how to develop collaborative applications.

Also, to our knowledge, there have been no studies examining the usability aspects of a tool for distributed, collaborative UML modelling even though the formal and highly structured nature of UML models provide particular challenges. One aspect of this paper is then to report from such a usability study showing that Distributed Knight may be usable for distributed, collaborative modelling. Also, we hypothesize that use of workspace awareness enhances usability in this structured modelling case.

From the evaluation of Distributed Knight, a number of interesting redesign suggestions emerged. Some of these centred around making better use of the radar view: The radar could show actions (such as deletion) on a more abstract level than currently; having a separate radar view for each person would help when working on separate diagrams; and, finally, the radar view could be an anchor for choosing between levels of coupling between collaborators, allowing someone to couple his viewport with a collaborator's viewport.

Another set of redesigns suggested enhancing the semantics of the UML elements. Examples of such enhancements included marking who created an element, emphasizing the most recently used elements, and fading elements that were infrequently used.

An important future direction for Distributed Knight may be to support asynchronous, distributed collaboration. Technically, this involves integrating Distributed Knight with a configuration management system such as CVS

(<http://www.cvshome.org>). A configuration management system allows multiple users to share and version files, a necessary prerequisite for asynchronous collaboration in software development. From a usability perspective, asynchronous collaboration raises a range of issues on how to maintain awareness of others' work across time and place. Ideas from the Notification Collage (Greenberg & Rounding 2001) may help in doing this. Basically, the Notification Collage allows users to upload "media elements" such as sticky notes, video elements, and activity indicators to a server that will redistribute these elements to subscribers. In this way, the Notification Collage becomes a rich resource for awareness and collaboration. Integrating semantic events from Distributed Knight into the Notification Collage itself should be straightforward technically since both are built on a publish/subscribe model of distributed communication. Along these lines, techniques for creating spatial models of interaction and social visualization may also need to be employed to help social interaction (Lee, Danis, Miller & Jung 2001).

A simpler mechanism for sharing semantic events from using Distributed Knight may be to use mechanisms similar to the Tickertape of the Elvin Notification Server (Fitzpatrick, Mansfield, Kaplan, Arnold, Phelps & Segall 1999). The Tickertape client provides a single-line, horizontally scrolling text display. Users may publish and subscribe to groups, messages sent to such groups are shown to members in a Tickertape. A Tickertape might combine automatic messages from Distributed Knight clients with messages submitted by users. Having messages visible to group members in this way could give awareness of previous synchronous sessions.

A next major research direction for Distributed Knight could be a stronger focus on pervasive computing issues. Knight can already be used pervasively on a range of input devices, but in order for collaboration to be fluent, supporting for ad-hoc shifts between available devices should be implemented. The range of input devices, on which Distributed Knight is usable, also brings about the interesting issue of mixed single-display and multiple-display groupware (Stewart, Bederson & Druin 1999): Using Distributed Knight on an electronic whiteboard supports single-display collaboration while collaboration at the distance with another node then can bring about the mix of single- and multiple display groupware. This raises a number of interesting issues to investigate and evaluate such as, e.g., to what degree the physical awareness cues in the single-display location need to be translated into virtual awareness.

## 5 Summary

This paper has presented the Distributed Knight tool for synchronous, distributed collaboration in object-oriented modelling. We have looked in particular at the workspace awareness features of Distributed Knight and discussed a qualitative evaluation of the current design of these features. This evaluation indicates that Distributed Knight is indeed usable for distributed modelling, and that in particular, the workspace awareness features of Distributed Knight may

- lower the number of breakdowns during collaboration,

- lead to more symmetric collaboration patterns,
- help coordinate work, and
- result in greater perceived usability.

## 6 Acknowledgements

Thanks to the people who participated in the Distributed Knight evaluation and to our co-developers of the Knight tool. This work has been partly sponsored by ISIS Katrinebjerg (<http://www.isis.alexandra.dk>).

## References

- Bødker, S. (1996), Applying activity theory to video analysis: How to make sense of video data in HCI, in B. Nardi (ed.), *Context and Consciousness*, MIT Press, pp.147–174.
- Beaudouin-Lafon, M. (ed.) (1999), *Computer Supported Cooperative Work*, Wiley.
- Buxton, W. (1986), There's More to Interaction than Meets the Eye: Some Issues in Manual Input, in D. Norman & S. Draper (eds.), *User Centered System Design: New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates, pp.319–337.
- CNN (2001), Telework trend rooted in convenience, .
- Damm, C., Hansen, K. & Thomsen, M. (2000), Tool Support for Object-Oriented Cooperative Design: Gesture-Based Modeling on an Electronic Whiteboard, in *Proceedings of CHI 2000, ACM Conference on Human Factors in Computing Systems*, pp.518–525.
- Dourish, P. & Bellotti, V. (1992), Awareness and Coordination in Shared Workspaces, in *Proceedings of CSCW'1992, ACM Conference on Computer-Supported Cooperative Work*, pp.107–114.
- Eugster, P., Guerraoui, R. & Damm, C. (2001), On Objects and Events, in *Proceedings of ACM OOPSLA'2001*, pp.254–269.
- Fitzpatrick, G., Mansfield, T., Kaplan, S., Arnold, D., Phelps, T. & Segall, B. (1999), Augmenting the Workaday World with Elvin, in *Proceedings of ECSCW 1999, European Conference on Computer-Supported Cooperative Work*, pp.431–435.
- Greenberg, S. & Rounding, M. (2001), The Notification Collage: posting information to public and personal displays, in *Proceedings of CHI 2001, ACM Conference on Human Factors in Computing Systems*, pp.514–521.
- Gutwin, C. & Greenberg, S. (1999), “The Effects of Workspace Awareness Support on the Usability of Real-Time Distributed Groupware”, *ACM Transactions on Computer-Human Interaction (TOCHI)* **6**(3), 243–281.
- Gutwin, C. & Greenberg, S. (2002), “A Descriptive Framework of Workspace Awareness for Real-Time Groupware”, *Journal of Computer-Supported Cooperative Work* **11**(3), 411–446.
- Hansen, K. & Damm, C. (2004), Building Flexible, Distributed Collaboration Tools using Type-Based Publish/Subscribe — The Distributed Knight Case, in *Proceedings of the IASTED International Conference on SOFTWARE ENGINEERING*, pp.595–600.
- Hansen, K. M. & Damm, C. H. (2002), Instant Collaboration: Using Context-Aware Instant Messaging for Session Management in Distributed Collaboration Tools, in *Proceedings of NordiCHI 2002*, pp.279–282.

- Isaacs, E. & Tang, J. (1994), "What Video Can and Can't Do for Collaboration: A Case Study", *Multimedia Systems* 2, 63–73.
- Lee, A., Danis, C., Miller, T. & Jung, Y. (2001), Fostering Social Interaction in Online Spaces, in *Proceedings of Interact 2001*.
- Mackay, W. (1999), Media Spaces: Environments for Informal Multimedia Interaction, in M. Beaudouin-Lafon (ed.), *Computer Supported Cooperative Work*, Wiley, pp.55–82.
- Madsen, O., Møller-Pedersen, B. & Nygaard, K. (1993), *Object-Oriented Programming in the BETA Programming Language*, Addison Wesley/ACM Press.
- OMG (2001), Unified Modeling Language Specification 1.4, Technical Report formal/01-09-67, Object Management Group.
- Oram, A. (ed.) (2001), *Peer-to-Peer. Harnessing the Power of Disruptive Technologies*, O'Reilly.
- Schuckmann, C., Kirchner, L., Schümmer, J. & Haake, J. M. (1996), Designing Object-Oriented Synchronous Groupware with COAST, in *Proceedings of CSCW'1992, ACM Conference on Computer-Supported Cooperative Work*, pp.30–38.
- Stewart, J., Bederson, B. B. & Druin, A. (1999), Single display groupware: a model for co-present collaboration, in *Proceeding of the CHI 99 conference on Human factors in computing systems : the CHI is the limit*, ACM Press, pp.286–293.
- Veinott, E., Olson, J., Olson, G. & Fu, X. (1999), Video Helps Remote Work: Speakers Who Need to Negotiate Common Ground Benefit from Seeing Each Other, in *Proceedings of CHI 1999, ACM Conference on Human Factors in Computing Systems*, pp.302–309.