The Software Design Board: A Tool Supporting Workstyle Transitions in Collaborative Software Design

James Wu and T.C.N Graham

School of Computing, Queen's University
Kingston, Ontario, CANADA
{wuj,graham}@cs.queensu.ca

Abstract. Software design is a team activity, and designing effective tools to support collaborative software design is a challenging task. Designers work together in a variety of different styles, and move frequently between these styles throughout the course of their work. As a result, software design tools need to support a variety of collaborative styles, and support fluid movement between these styles. This paper presents the Software Design Board, a prototype collaborative design tool supporting a variety of styles of collaboration, and facilitating transitions between them. The design of Software Design Board was motivated by empirical research demonstrating the importance of such support in collaborative software design, as well as activity analysis identifying the lack of support in existing tools for different styles of collaboration and transitions between them.

1 Introduction

The design of large, complex software systems is a team activity. A study by DeMarco and Lister found that developers working on large projects spend up to 70% of their time collaborating with others [6], while Jones found that team activities account for 85% of costs in large scale development projects [18]. This degree of interactivity between team members has necessitated the development of tools that can support collaboration within the software design process.

Designing effective collaborative design tools is a challenging task. In addition to technical and implementation issues associated with concurrent and/or distributed work, designers are hampered by a lack of data on how groups work together in software design. Collaborative applications are too often developed based on the individual experience of the designer, rather than on detailed study of the target user group and target tasks. This can result in tools that are neither useful nor usable. Even when user-centred design techniques are applied, the results are often tailored to the needs of single users, without sufficient support for collaborative work [10].

To better support collaborative work, software design tools need to support a variety of *workstyles* for collaborative interaction, as well as support fluid transitions between these workstyles. A workstyle is a characterization of the style of interaction employed by a group of collaborators, or supported by an interactive tool [36]. For example, co-located collaborators working at a whiteboard are engaged in an entirely different workstyle than distributed collaborators asynchronously sharing a document stored in a repository. In earlier work, we have shown that members of collaborative groups interact with each other through a variety of workstyles, and move frequently between different workstyles throughout the course of their interactions [37].

In this paper, we present a prototype collaborative software design tool, the Software Design Board. Software Design Board supports a variety of workstyles appropriate to the early stages of software development, and facilitates transitions between them. The functional requirements of the tool are informed by studies of existing design tools and by results of empirical research into collaborative software design activities. In presenting Software Design Board, we begin with a brief examination of related tools in the domain. As Software Design Board is primarily intended for use with an electronic whiteboard, these related tools are those that support software design through the use of informal media. Next, we present the empirical research that motivated the importance of supporting transitions in workstyle in collaborative design. We then introduce a model for characterizing styles of collaborative work, and show how this model is used to identify mismatches between collaborative activities and existing tool support. Finally, we introduce the Software Design Board and show how it supports a variety of important workstyles and workstyle transitions.

2 Tools Supporting Collaborative Software Design Through Informal Media

People often carry out design work using informal media such as paper or whiteboards [20]. Particularly in the early stages of design, informal media are appropriate as they allow design diagrams to be quickly and fluidly sketched [34]. Computational analogues of such informal media include electronic whiteboards, data tablets and stylus input for computers. Tools supporting interaction with informal media attempt to extend the free form, fluid interaction afforded by physical informal media to these computational counterparts.

The main advantage of informal media tools is that they support a natural working style without imposing significant cognitive overhead on the user through heavyweight interaction mechanisms. They allow users to use the tool transparently, without having to think about the tool itself. The drawback of many of these tools is the limited, or non-existent, support for movement towards more formal, structured work. This lack of support may limit development as a design evolves and begins to require more formal treatment. Also, many of these tools are intended to be general-purpose, and lack features that may be useful in the early stages of software design.

We identify three subcategories of these tools. In each, we consider an archetype tool that is typical of the subcategory, and identify other similar tools.

- *Informal CASE Tools:* These are software design tools that support interaction through informal media. Ideogramic UML [15] is a commercial tool that evolved from the Knight research project [5]. IdeogramicUML is intended to support the “agile” use of UML [1], meaning effective and lightweight use of UML. It supports a wide variety of interaction devices, including PCs, tablets, Tablet PCs and electronic whiteboards. This tool supports gesture based modeling in UML, as well as free hand diagramming with no gestural interpretation. Furthermore, IdeogramicUML only supports co-located collaboration using electronic whiteboards, and requires additional tool support to be used by distributed teams. Other similar tools include UML Recognizer [21] and Tahuti [13].

- *Enhanced Electronic Whiteboards:* These are electronic whiteboard applications that attempt to replicate and extend the functionality of physical whiteboards using electronic whiteboards such as a Smartboard [28]. Flatland [24] is an augmented whiteboard application designed to support informal office work. Flatland provides various stylus-appropriate techniques for interaction and space management on an electronic whiteboard. Furthermore, it provides the ability to apply different behaviors to define application semantics. Flatland allows different segments on the board to respond differently to stylus input based on the applied semantics. However, it does not specifically support design tasks, but is intended to support for informal work in an office environment and as such can be appropriate in early software design tasks. Furthermore, Flatland does not support distributed collaboration, but only facilitates teamwork in a co-located setting. Other similar tools include Tivoli [25], Dolphin [30], and MagicBoard [4].
- *Shared Drawing Tools:* These tools support collaborative sketching or drawing tasks such as often found in early design work [31, 16] without providing support for any specific notation. ClearBoard [16] is a shared drawing program that allows two remote users to simultaneously draw in a shared space while providing awareness information such as hand gestures and gaze. It is based on the metaphor of 'talking through, and drawing on, a big transparent glass board' [16]. Clearboard also provides additional functionality such as simple stroke manipulations, recording of working results, as well as the ability to integrate generic files into the drawing space. Other similar tools include Commune [3], GroupSketch [11], and VideoWhiteboard [32].

Tools supporting interaction through informal media support collaboration in software design by facilitating unstructured interaction in a way appropriate to the early, creative design stages. They support an informal style of work that allows users to interact naturally and to use the tool transparently without imposing unnecessary overhead. Informal media tools support a small group of designers, and rely on social protocol to mediate group interaction. They typically produce informal artifacts of unbound semantics and free-form syntax. Most importantly for our purposes, informal media tools are typically independent of synchronicity or location, i.e. they support synchronous and asynchronous, as well as distributed and co-located interactions. This means they can support transitions in workstyle between synchronous/asynchronous and co-located and distributed styles of interaction.

3 Importance of Workstyles in Collaborative Software Design

We now present the empirical research that motivated the importance of supporting transitions in workstyle in collaborative design. We have performed extensive empirical studies into the nature of collaboration in software design [37]. We followed 5 development groups at a large software company over a 6-week period. Our research illustrated that not only is significant time spent collaborating within the design process, but also significant time and effort is spent in transitions between different collaborative styles of work. For example, team members may move frequently between asynchronous and synchronous workstyles, or between co-located and distributed workstyles, throughout the course of a single workday. These observations highlight the need for collaborative design tools that provide support for performing transitions between the various activities and working styles in which designers engage. Although some existing tools facilitate transitions in software designers' workstyles [7, 21, 12], most provide only basic communication facilities. More importantly, existing support for workstyle transitions is not commensurate with the frequency with which designers change between collaborative work styles [37, 38].

During our study, team members were observed to be highly interactive, spending on average more than two hours per day on communication tasks. Communication was predominantly face to face or via telephone or email. Also, team members often changed various aspects of their interaction such as location, synchronicity or modality of communication. These results provide evidence regarding the importance of collaboration and communication in software design, and motivate the need to support these activities in software design tools.

We also found that developers change locations frequently in order to collaborate, showing that on average, developers collaborated in more than 6 locations per day. According to interviews, this was due to a strong preference to work face-to-face. Many designers felt it was simpler, quicker and generally more efficient to use standard communication, including meeting face-to-face, than to establish remote interaction though tools. This often meant that people would walk up and down multiple flights of stairs numerous times each day to meet in person rather than use a telephone or another collaboration tool. These changes in location further indicate the frequency of workstyle transitions in collaborative software design.

Designers were also observed to frequently change the way in which they communicate, and to carry on multiple, simultaneous threads of collaboration. We found that it is typical for designers to attend a face-to-face meeting on a topic, then follow up with email, ask a supplementary question by telephone, follow up with more email, and so forth. Within individual threads of collaboration, we observed that designers change the mechanism by which they communicate more than once per day on average. These changes often involve a change in synchronicity (e.g. a change from telephone to email involves a change from synchronous to asynchronous interaction). Moreover, developers on average carried out more than three simultaneous threaded interactions in the course of a single day. All of these changes, between communication modalities, synchronicity and collaboration groups, reflect transitions in workstyle.

The results of this study have clear implications for the design of tools supporting team-based software design in large companies. These results show the importance of flexibility with respect to how a tool supports collaboration. Changes in physical location, synchronicity and communication modality are frequent, and tools should be designed to support such changes. Current tools do not sufficiently support such changes, if at all. In most existing tools, changes in synchronicity and location require a change in modality (e.g. from face-to-face to telephone) as well, imposing additional overhead on designers that choose to use them. More information on these empirical results can be found in the full study [37].

4 Understanding Workstyles

The *Workstyle Model* [36] allows us to characterize styles of collaborative work, either those employed by a group or supported by a tool. We can use these characterizations to identify mismatches between common activities and available tool support. These mismatches highlight areas where additional tool support is needed within a domain. Workstyle modeling complements task modeling [8] with supplemental information about how people communicate and coordinate their activities, and about the nature of the artifact to be produced. We have applied this model to the evaluation of how software designers collaborate, the forms of collaboration a wide variety of software design tools support, and to the design of the Software Design Board application itself. The development of the model itself was informed by the empirical study, presented in the previous section, as well as by informal laboratory studies of tools and designers.

In order to understand the relevance of workstyle analysis, consider the task of creating a design in some formal diagrammatic notation. A task model can identify the activities involved in creating such a design: drawing and labeling nodes, connecting them with relations, editing and reformatting diagram elements, and so forth. This model of design activities might lead to the development of a tool similar to Rational Rose [26] permitting mouse-based structural editing of design diagrams. However, in addition to the tasks that need to be performed, it is important to understand the users' preferred workstyle before committing to a design. Designers may be working in a brainstorming style, or may be recording precise documentation from which a system is to be built. A brainstorming workstyle is well supported by a whiteboard, which 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. Alternatively, recording of precise documentation is well supported by a traditional Computer-Aided Software Engineering (CASE) tool. It is important to note that, though both tools support the activities identified in the task model, they do so in different ways that are appropriate to entirely different styles of work. The workstyle model helps in the analysis of peoples' goals and tasks by helping to understand their preferred style of work.

The *Workstyle* model characterizes a working style as an eight dimensional space that addresses the style of collaboration and communication between designers and the properties of the artifacts that are created during the collaboration. The functionality of collaborative design tools can be plotted in this space to specify the set of workstyles that they can support. It then becomes possible to compare designers' preferred workstyles to those supported by available tools and to identify potential task/tool mismatches. These mismatches can be used to guide the design of new tools that are more appropriate to particular design activities. Figure 1 depicts a graphical representation of the axes of Workstyle Model on which workstyle analyses are plotted

4.1 Dimensions Describing Collaboration Style

The first four dimensions of the model describe the nature of the collaboration in which a group is engaged, or that can be supported by a tool. They are defined as follows:

- *Location:* The location axis refers to the distribution of the people involved in the collaboration. As people become more geographically distributed, supporting some collaborative workstyles becomes increasingly difficult [27].
- *Synchronicity:* The synchronicity axis describes the temporal nature of the collaboration. People may work together at the same time (*synchronously*) or at different times (*asynchronously*).
- *Group Size:* The group size axis captures the number of people involved in the collaboration. Support for larger groups typically comes at the expense of intimacy in the interaction between collaborators.
- *Coordination:* This axis describes how users' activities are coordinated, whether by the choice of tools they are using or through the adoption of some coordination model [22].

4.2 Dimensions Describing Artifact Style

The remaining four dimensions describe the nature of the artifacts produced by the group, or able to be produced by a tool. They are defined as follows:

- *Syntactic Correctness*: The artifact being produced may be required to follow a precise syntax. This requirement may inhibit progress in early stages of design by forcing initially abstract designs to conform to a predetermined syntax [20, 35].
- *Semantic Correctness:* An artifact is considered to be semantically *sound* if its meaning is unambiguous and free of contradiction. The production of semantically sound artifacts facilitates automated analysis and evolution.
- *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.
- *Modifiability:* This axis represents the ease with which an artifact can be modified. For example, small modifications to a whiteboard drawing are simply performed by erasing and redrawing.

4.3 Applying the Workstyle Model

The Workstyle Model can be applied to assess tools and/or the interaction style of users. The model can be used to evaluate the support provided by individual tools for various working styles, or applied to users to evaluate their working styles while accomplishing various tasks with preferred tools. To do so, values for each property are plotted on a two-dimensional representation of the model, as seen in Figure 1. A single workstyle is represented as a point in an eight dimensional space, while a range of workstyles is represented as a region in this space. Support for a single value in a particular property is indicated by a line intersecting the related axis, while a region over the axis represents support for a range of values in that property. So a plot that consists of a single line with no expanded areas can represent a tool or set of tools that supports a single, rigid workstyle. Similarly, if applied to users, the plot may represent a particular style of work used to accomplish some particular task. Conversely, a plot that covers an area of the graph may represent a tool or set of tools that supports a range of workstyles and transitions between them. Similarly, if applied to users, it may represent a change in the style of interaction that has occurred over a period of time. Once plotted, differences in the workstyles supported by various tools become visually apparent. These plots can be compared to workstyle plots of users accomplishing the tasks supported by those tools in their preferred manner. Mismatches between these plots identify tools that are not providing sufficient usability for their supported tasks. More detail and examples of applying the Workstyle Model can be found in [36, 38].

4.3.1 Workstyle Example – UML Design Tools

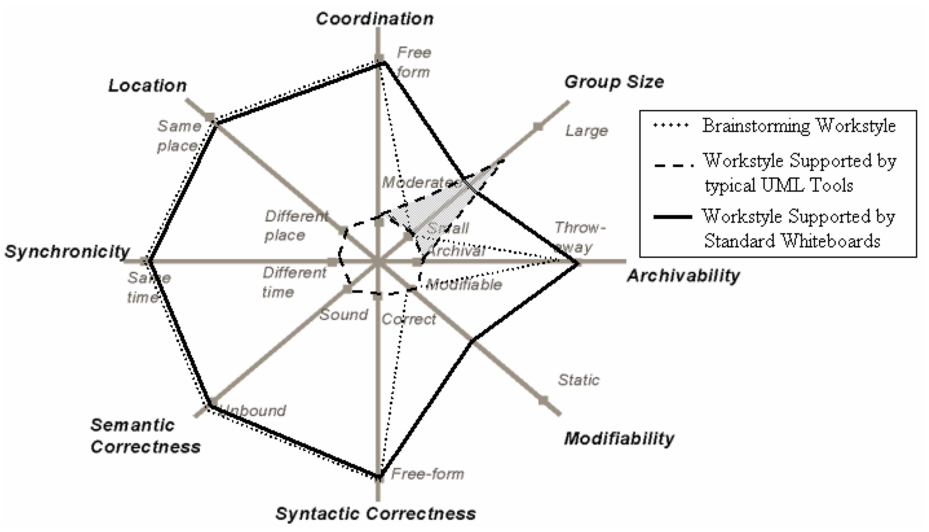

Fig. 1. A Workstyle comparison between UML tools and standard whiteboards in support for typical brainstorming activities.

It is useful to consider the workstyle supported by popular UML design tools such as Rational Rose [26]. Design tools such as these are a good fit with workstyles where little real-time communication with other designers is required, and where the goal is to create precise, archival designs. However, these design tools provide poor support for the early stages of design, such as brainstorming. During these phases, designers spend significant time on communications tasks.

The inappropriateness of UML design tools for early stages of design can be clearly shown by examining the brainstorming workstyle. As shown in Figure 1, *brainstorming is typically carried out by small groups working face to face, using free-form coordination and 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 [2, 31]. Modifiability is important as early designs evolve rapidly, and archivability is important to allow early designs to be migrated to more formal designs.*

Figure 1 clearly shows that while UML design tools may support the core tasks of the early stages of design, they do not support the workstyle of early design (brainstorming). The emphasis on asynchronous, moderated work with strong emphasis on syntactic correctness and semantic soundness is incompatible with the free-form brainstorming workstyle. A better match to the workstyle of early design is the workstyle supported by standard whiteboards. These tools support small groups of co-located users working synchronously, and rely upon social protocol to mediate user interaction. They impose no requirements on syntax, nor do they interpret any semantic meaning from the input.. The main incompatibility of these tools to the brainstorming workstyle is the limited ability to easily archive artifacts created on the board.

In this example, we have seen how workstyle analysis can be applied to a tool and compared to the workstyle of the collaborative activities in which it may be used. Such comparisons can highlight incompatibilities between a tool and the way in which it will be used within a particular context. Through this mechanism, tools can be selected for use in particular contexts to provide better usability to users carrying out their tasks.

5 Software Design Board: Supporting Workstyle Transitions in Software Design

Based on the findings from our empirical study into collaboration in software design, as well as workstyle analyses revealing inadequacies of existing design tools, we developed the Software Design Board to facilitate transitions between some common working styles as described by the Workstyle Model. This is achieved through the integration of informal media and flexible collaboration mechanisms, as well as support for migration between different software tools, devices and collaborative contexts. These facilities are intended to support fluid transitions between the some of the different styles of work in which designers are frequently engaged, specifically synchronous/asynchronous and/or co-located/distributed collaboration, and more generally, formal/informal interactions.

5.1 Functional Requirements

The functional requirements for the Software Design Board evolved from workstyle analyses of existing tools and of developers in the early stages of software design. For example, workstyle analyses of existing tools for collaborative software design revealed that each support only a single or limited set of collaborative workstyles. Furthermore, the empirical studies described in Section 3 revealed a variety of behavioral patterns in which developers frequently engage. Most importantly, the study found that team members regularly changed the nature of their interactions with each other in terms of synchronicity, location and modality. The results have specific implications on tool design; tools should be designed to support these frequent changes in workstyle.

All of these findings reveal some open problems in the area of tool support for collaborative software design, and motivated the functional requirements driving the design of Software Design Board. Specifically, the following are aspects of collaborative design that are poorly supported in existing tools:

- *Unsupported Workstyles:* Workstyle analyses of existing tools revealed that some workstyles are not supported by any individual class of tools. For example, large groups of synchronous collaborators, whether distributed or co-located, are not well supported by any available tool. This may be a result of hardware restrictions, or the limited applicability of such workstyles in practice. Additionally, no existing tools allow free-form interaction while supporting the creation of syntactically and semantically refined artifacts. Even informal CASE tools such as IdeogramicUML [15] employ a gesture-based syntax that places restrictions on free-form interaction.
    - **Functional Requirement 1**: Support the freehand creation of syntactically correct UML diagrams.
- *Lack of Support for Workstyle Evolution:* Workstyle analysis of existing tools revealed that individual tools only support a single or limited set of workstyles, and provide little or no support for movement between workstyles. However, our empirical investigations found that designers frequently move between synchronous/asynchronous and collocated/distributed styles of interaction. Additionally, transitions between workstyles often involve changes between interaction devices. For example, moving from an informal to a more formal workstyle may involve switching from an electronic whiteboard to a PC. Available tools do not sufficiently support migration between devices.
    - **Functional Requirement 2**: Support transitions between synchronous and asynchronous styles of collaboration.
    - **Functional Requirement 3:** Support transitions between collocated and distributed styles of collaboration.
    - **Functional Requirement 4**: Support transitions between physical devices.
- *Lack of Support for Multiple Collaborative Contexts:* In addition to frequently changing their collaborative workstyle, the results of the study presented in Section 3 show that individual designers also switch amongst a number of concurrent collaborative contexts. This means that they frequently move between multiple interactions with different groups. For example, a given designer may be participating in a number of concurrent projects or tasks, and may frequently switch their focus from one project to another. Furthermore, designers may participate concurrently in multiple collaborative contexts.
    - **Functional Requirement 5**: Support transitions between collaborative contexts.
- *Limited of Support for Integration of Existing Applications:* Current meta-tools that support sharing of existing applications, such as Netmeeting [23], impose

significant restrictions on collaboration that can be inappropriate to many of the important workstyles found identified during the empirical study. Mechanisms for integrating existing tools into a variety of collaborative workstyles would allow designers to collaborate on wide variety of tasks without giving up their preferred tools for accomplishing those tasks.

- **Functional Requirement 6**: Support integration of existing applications into all supported workstyles.

5.2 Overview of the Software Design Board

The Software Design Board (SDB) is a shared whiteboard application with additional functionality that supports collaborative software design. As seen in Figure 2, user interaction with this tool is similar to a typical interaction with a standard whiteboard.

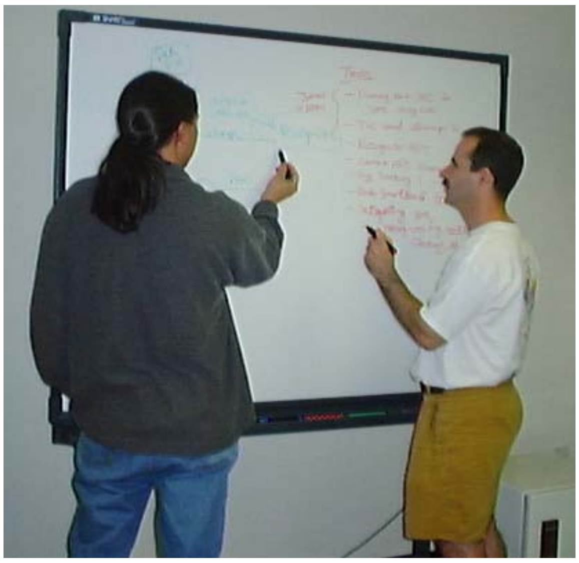

Fig. 2. Using Software Design Board.

Typical sessions using the tool via different devices are depicted in Figure 3. When used on a PC, the interface supports drawing using a typical structured drawing tool. Functionality is accessed through typical drop-down menus. When used on an electronic whiteboard or tablet PC, the user interface supports unstructured pen input of stroke information for freehand data such as diagrams, annotations, notes and lists.

This feature is in partial support of Functional Requirement 1 (*Support the freehand creation of syntactically correct UML diagrams*). Unstructured stylus-based input also provides the basis for lightweight user interaction with the tool. Furthermore, an integrated structure recognizer [9] supports automated translation of freehand diagrams into a more structured format appropriate for interpretation as UML or any other box-and-arrow notation. This functionality is similar to other tools [5, 21]. An example of this recognition functionality applied to a simple diagram is depicted in Figure 4.

In addition, objects can be placed on the board in and amongst the free hand data. These objects can include design documents or diagrams that may be browsed and annotated, or external programs that can execute other functionality. For example, a design document may be embedded into some area of the board allowing it to be communally browsed and annotated within the context of the other data on the board. This document is opened and displayed within the tool with which it was created, and all of that tool's functionality is accessible through the SDB's interface. This functionality supports Functional Requirement 6 (*Support integration of existing applications into all supported workstyles*). A typical session with an embedded design artifact is depicted in Figure 5.

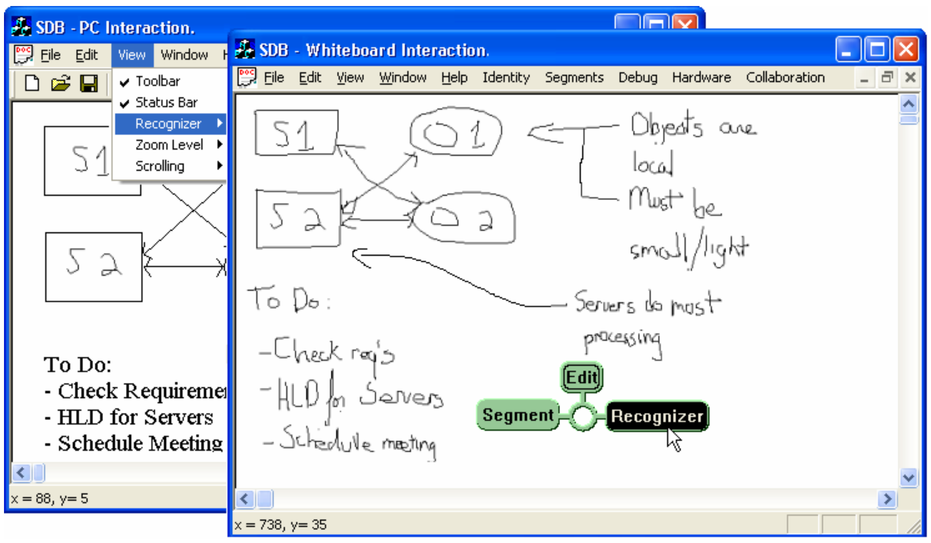

Fig. 3. Typical single-user sessions in Software Design Board. A PC user manipulates structured drawing elements and text, and interacts through drop-down menus. A whiteboard user draws free hand, and interacts through pie menus and gesture-based commands.

In order to support collaboration, the tool integrates communication and sharing mechanisms. For example, gesture transmission is supported within the context of synchronously shared whiteboard space. Voice communication mechanisms are planned, but not yet implemented. Additionally, any OLE-based communication tool can be integrated into the whiteboard space.

These communication objects are embedded and manipulated directly within the context of the board, and are maintained with the rest of the data on the board. For example, external applications such as web browsers or media streams may be embedded in the board space and used for communication. These communication mechanisms support Functional Requirement 2 (*Support transitions between synchronous and asynchronous styles of collaboration*) and Functional Requirement 3 (*Support transitions between co-located and distributed styles of collaboration*) by allowing the simultaneous use of functionality supporting all of these styles of interaction within a single application.

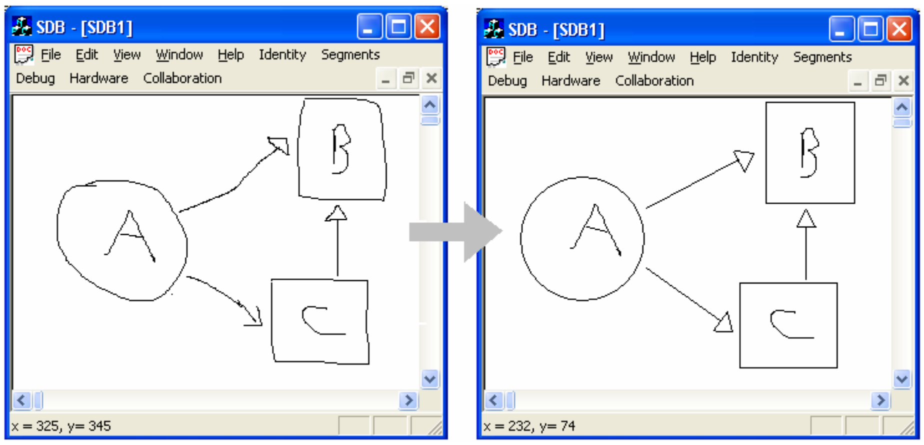
SDB - [SDB1]
File Edit View Window Help Identity Segments
Debug Hardware Collaboration
A
B
C
x = 325, y= 345
SDB - [SDB1]
File Edit View Window Help Identity Segments
Debug Hardware Collaboration
A
B
C
x = 232, y= 74

Fig. 4: Applying the syntax recognizer to a freehand diagram. Hand drawn elements such as circles, squares and arrows are recognized and converted into structured drawing elements.

The whiteboard space can be divided into any number of *segments*. These segments allow data to be shared in different ways. Generally, a segment is an area in the board containing contextually related data. As with a regular whiteboard, a user explicitly specifies the segmentation of data in the board through delineating strokes, e.g. a surrounding box or circle. Segments can be shared with others to allow users of other SDB clients to connect and synchronously interact with each other and share data. To share segments asynchronously, another client connects and copies the content of the segment to his/her local client. This data can then be manipulated without affecting the data in the original segment. Diverging copies of segments may be manually or automatically reconciled, if possible. When shared synchronously, data in a shared segment is viewed in decoupled WYSIWIS [29] fashion. Furthermore, at any time a user can change the way in which segments are shared. Synchronously shared segments can be easily detached and shared asynchronously, and vice versa. Gesture information is automatically transmitted between synchronously shared segments via telepointers. This functionality also supports Functional Requirement 2 (*Support transitions between synchronous and asynchronous styles of collaboration*) and Functional Requirement 3 (*Support transitions between co-located and distributed styles of collaboration*), by providing the mechanism by which users can freely and fluidly move between (synchronously or asynchronously) shared and private data.

Furthermore, on any SDB client, different segments may be shared concurrently and in different ways, between different groups. This functionality supports Functional Requirement 5 (*Support transitions between collaborative contexts*), by allowing users to move freely between different collaborative interactions contained within each segment. A typical session involving segment sharing is depicted in Figure 6.

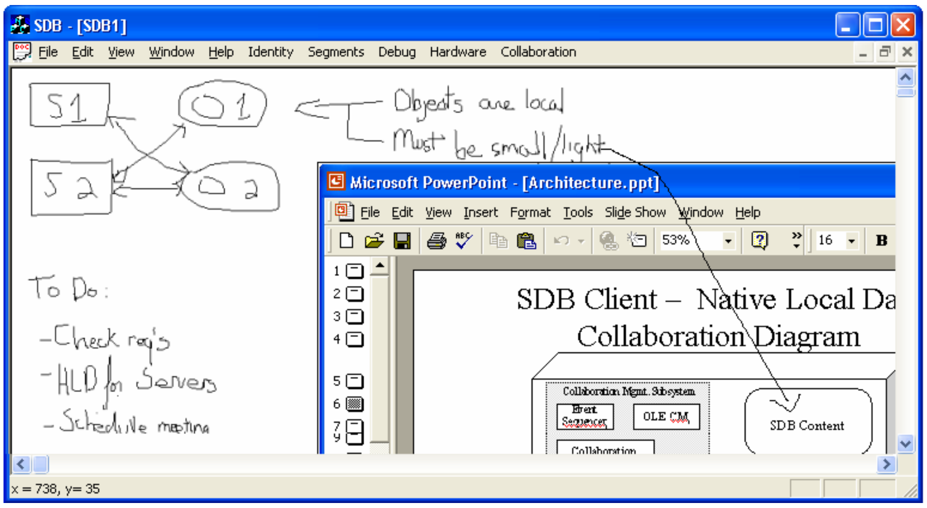
SDB - [SDB1]
File Edit View Window Help Identity Segments Debug Hardware Collaboration
S1
O1
S2
O2
Objects are local
Must be small/light
To Do:
-Check req's
-HLD for Servers
-Schedule meeting
Microsoft PowerPoint - [Architecture.ppt]
File Edit View Insert Format Tools Slide Show Window Help
SDB Client – Native Local Da
Collaboration Diagram
Collaboration Mgmt. Subsystem
Event Sequencer
OLE CM
SDB Content
x = 738, y= 35

Fig. 5. A design document embedded in a Software Design Board session.

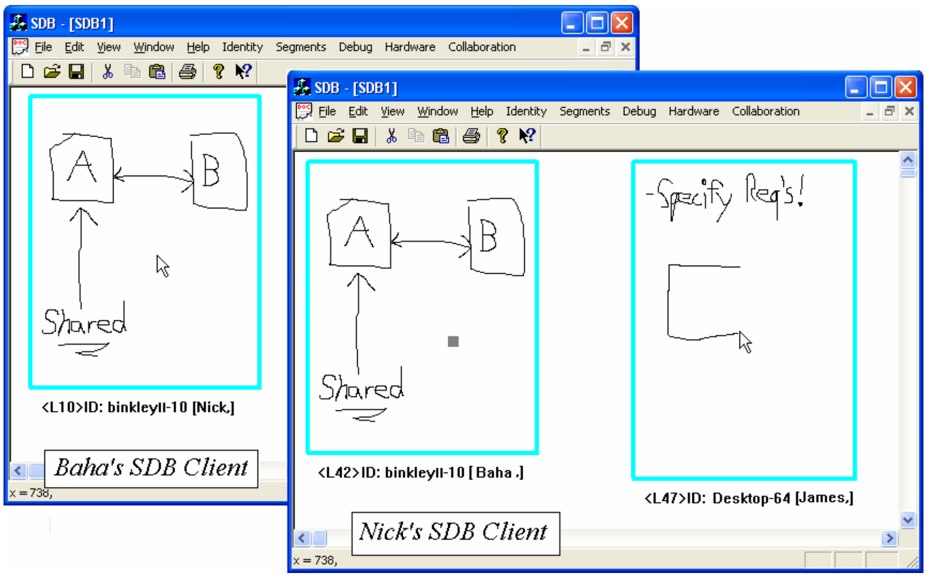
SDB - [SDB1]
File Edit View Window Help Identity Segments Debug Hardware Collaboration
A
B
Shared
<L10>ID: binkleyII-10 [Nick,]
Baha's SDB Client
A
B
Shared
<L42>ID: binkleyII-10 [Baha ,]
-Specify Req's!
<L47>ID: Desktop-64 [James,]
Nick's SDB Client
x = 738,

Fig. 6.The segment with ID binkley||-10 is shared between Baha and Nick. Baha's mouse pointer appears as a telepointer on Nick's client. Nick is concurrently sharing a different segment, with ID Desktop-64, with James.

Software Design Board implements a plastic interface [33] that can be used on different hardware devices. While the main platform for this application is an electronic whiteboard, it can also be accessed from a PC with or without an associated tablet. Widget-level plasticity supports appropriate interaction through each type of device [17]. For example, whiteboard users can use pie-menus and gesture based commands that are more appropriate to their stylus-based interfaces, while PC clients can use traditionally structured pull-down menus systems. There is also the potential to develop clients that facilitate access from a PDA or any other appropriate device. The interaction allowed by each interface is appropriate to the specific device. For example, interaction through a PDA would be greatly limited as compared to an interaction at a SmartBoard, and drawing facilities on a mouse-based PC client may be more structured than those on the SmartBoard, in order to accommodate the associated input mechanism. This functionality is in support of Functional Requirement 4 (*Support transitions between physical devices*).

Device appropriate interfaces allow users to interact with the application through any available or preferred hardware, and freely migrate between device types, as long as the limitations of the hardware are accepted. Migration between tools and devices is further supported by the segmentation of data. Segmentation facilitates data plasticity, wherein types of data within a segment can be manipulated appropriately in the context of a given device or application. If a segment is known to contain data of a particular type, then it can be interpreted or formatted appropriately for any specific device or tool. For example, if a segment is known to contain a UML diagram, then it can be interpreted and migrated via XML into an appropriate UML-based CASE tool.

In addition to the functionality described above, a variety of additional features are integrated into the user interface to facilitate interaction with the Software Design Board. Unlike a regular whiteboard, a session in the SDB can be essentially unbounded in size. To facilitate navigation, the interface to the workspace is scrollable and zoomable. If a more structured input mechanism is desired at the whiteboard, a floating keyboard and/or structured drawing palette can be made available through menu options. These options can be accessed from context sensitive and device appropriate menu systems. Finally, all functionality is available through both context sensitive pull-down menus and pie-menus that facilitate gesture-based commands. This allows advanced users to use the tool more effectively by bypassing the menu structure.

5.3 Workstyle Transitions in Software Design Board

We now consider some simple scenarios that illustrate how Software Design Board can be used to perform some common transitions between workstyles. This is not intended as a set of instructions for performing the indicated transition, but rather as examples of how such transitions are supported within the tool. Additionally, it is intended to demonstrate the ease with these transitions can be performed within the tool.

- ***Distribution Transitions***: A group of co-located collaborators works together around an electronic whiteboard (a co-located workstyle). They want to share their work with a remotely located group. They draw a box around their current work in order to define a segment, and use a simple gesture command to share that segment with the remote group. The availability of the remote group is indicated via the context-sensitive pie menus [14, 19] that structure the gesture. At the remote site, a change in the entry structure of the menu system indicates the availability of a newly shared segment. The remote group creates a local segment in their workspace, and uses a similar gesture to attach their segment to that which was newly shared with them. Synchronized copies of the original data now appear in both group's segments, and telepointers appear to provide a sense of awareness of the actions of each group to the other. The two groups now collaborate in this distributed workstyle.

- ***Synchronicity Transitions:*** A group of users interacts synchronously with data contained in a shared segment (a synchronous workstyle). Each user performs updates that are immediately reflected in every other user's view of the data. They decide to work separately so that each user may concentrate on a particular aspect of the data. Each user detaches his/her segment from the shared session, and is left with a local copy of the data to which asynchronous updates can be performed. Now each user interacts with the data in their local copy (an asynchronous workstyle).
- ***Device Transitions***: A user is drawing a design on an electronic whiteboard. Using the piemenu structure and gesture commands, he invokes the recognizer and converts the freehand design to a structured drawing. He then creates a shared segment containing the diagram on the whiteboard. He moves to his PC and starts the Software Design Board client. Using the traditional pull-down menu structure, he creates a segment, attaches it to the shared segment he previously created at the whiteboard. He continues to work on that diagram from the PC, manipulating the structured elements in a manner appropriate for mouse-based interaction.
- ***Context Transitions***: A user maintains two different shared segments in his Software Design Board workspace. Each segment is shared between a different group of colleagues with whom he collaborates, and therefore each segment maintains completely different data (each maintains a different work context). Through the course of the day he scrolls the workspace back and forth between those segments in order to interact with the different groups as required.
- ***Syntax Transitions***: A group of co-located users are brainstorming and free hand drawing a design on a whiteboard. Eventually, the drawing becomes too large and convoluted to easily manipulate in this manner. Some elements consume a disproportionate amount of board space; others overlap due to the freeform development of the diagram. The designers want to move the work into a structured drawing editor to clean up the drawing and continue work. They use a gesture command to select all relevant drawing elements, then another gesture to invoke the syntax recognizer. The drawing is automatically converted to discrete, structured drawing elements such as boxes, circles and arrows. A third gesture is used to invoke a 'Send To…' command, which causes the newly structured elements to be opened within a structured drawing editor. The group now restructures their drawing, and continues to work.
- ***Semantic Transitions***: A group of users has completed a freehand design diagram on a whiteboard. The users invoke the syntax recognizer to structure their drawing, as described above. Next, they use a gesture command to reselect all drawing elements, and another gesture to invoke the UML semantic interpreter. The structured drawing is automatically interpreted as a simplified UML class diagram– boxes are converted to classes, open arrows as generalizations, closed arrows as aggregations. A third gesture is used to invoke a 'Send To…' command,

which causes the newly structured elements to be opened within a UML editor for further manipulation.

5.4 Current Status of the Implementation

The Software Design Board application is currently a functional research prototype. Most of the functionality described in the previous sections exists, either wholly or partially, though some core functionality remains to be implemented. Functionality for moving, resizing and copying freehand elements still remains to be implemented, and structured drawing functionality and other PC-based interaction techniques are less developed. Distributed, synchronous sharing is currently limited to drawing data; synchronous application sharing functionality is only partially implemented and not yet functional. The functionality for implementing syntax transitions is not fully implemented. An XML DTD has been developed to describe these recognized free-hand diagrams, and standalone code for writing and reading these XML documents exists. However, this code has not yet been integrated with the Software Design Board application. Finally, only limited work has been done toward supporting semantic transitions, i.e. applying a semantic interpretation to the syntactic structure of the drawing described by the XML document. This work has been limited by the limited implementation supporting syntax transitions. As the functionality evolves to more completely support the syntax transition, so too will the functionality supporting the semantic transition.

6 Conclusions

In this paper, we have introduced a prototype collaborative software design tool, the Software Design Board. Software Design Board supports a variety of workstyles important in the early stages of software development, and facilitates transitions between them. The functional requirements for the tool evolved from workstyle analysis of existing design tools and from results of empirical research into collaborative software design activities.

The need to support workstyle transitions in tools for collaborative software design stems from the fact that designers switch amongst numerous collaborative styles throughout the course of the their work. Many factors influence the style in which they may choose to work (their *workstyle*), including the task at hand, availability of tools, distribution of collaborators, and personal preferences. These influences change frequently, thus designers often migrate between workstyles in response to such changes. Unfortunately, there are obstacles to such transitions. These may include having to recreate work artifacts in the format of a new tool, interruption of the flow of work, or physical relocation. Such obstacles may prove sufficiently burdensome that designers choose to continue to work in a style that is inappropriate for their current context. These obstacles exist because the variety of workstyles and workstyle transitions in which designers engage are not well supported by most existing design tools. Most of these tools are designed to support a single or limited set of workstyles, and their architectures are generally not capable of handling the dynamic changes in workstyle that are typical of collaborative design.

Software Design Board was developed to address some of these shortcomings and to support designers in some of the common workstyles and transitions in workstyle in which they frequently engage. Specifically, Software Design Board supports designers working synchronously/asynchronously, distributed/collocated and more generally, formally/informally. It supports the creation of syntactically bound or free-from artifacts, can be used through a variety of physical devices, and facilitates collaboration in multiple, concurrent contexts.

References

1. AgileAlliance, http://www.agilealliance.org
2. Bly, S., A. (1988). "A Use of Drawing Surfaces in Different Collaborative Settings". Conference on Computer-Supported Cooperative Work, Portland, OR.
3. Bly, S.,A. and S. Minneman (1990). "Commune: A Shared Drawing Surface." SIGOIS Bulletin: 184-192.
4. Crowley, J., Coutaz, J., Berard, F. (2000). "Things that See." Communications of the ACM **43**(3): 54-64.
5. Damm, C. H., Hansen, K. M., Thomsen, M. (2000). "Tool Support for Object-Oriented Cooperative Design: Gesture-Based Modelling on an Electronic Whiteboard". Proceedings of Conference on Human Factors and Computing Systems. The Hague, Netherlands.
6. DeMarco, T. and T. Lister (1987). Peopleware. New York, Dorset House.
7. Dewan, P. Choudary, R. (1991). "Flexible user interface coupling in collaborative systems". CHI ' 91, New Orleans, LA, ACM.
8. Diaper, D. (1989) Task analysis for human computer interaction, Ellis Horwood,.
9. Fonseca, M.,J., Pimentel, C., and Jorge, J., A. (2002). "CALI: An Online Scribble Recognizer for Calligraphic Interfaces**"**, Proceedings of the 2002 AAAI Spring Symposium - Sketch Understanding. Palo Alto, USA. pp51-58
10. Francik, E., Rudman, S. E., Cooper, D., and Levine, S. (1991). Putting innovation to work: adoption strategies for multimedia communication systems. *Communications of the ACM*, 34(12), pp. 52-64.
11. Greenberg, S. and R. Bohnet (1991). "GroupSketch: A Multi-user Sketchpad for Geographically Distributed Small Groups". Proceedings of Graphics Interface, pp 207-215.
12. Grundy, J. C., Mugridge, W.B, Hosking, J.G., Apperley, M. (1998). "Tool Integration, Collaboration and User Interaction Issues in Component-based Software Architectures". TOOLS '98, Melbourne, Australia, IEEE.
13. Hammond, T. and R. C. Davis (2002). "Tahuiti: A Geometrical Sketch Recognition System for UML Class Diagrams". Sketch Symposium, Stanford University, Palo Alto, CA.
14. Hopkins, D. (1991) "The Design and Implementation of Pie Menus", Dr. Dobb's Journal, CMP Media. December 1991.
15. Ideogramic – IdeogramicUML, http://www.ideogramic.com
16. Ishii, H. and M. Kobayashi (1992). "ClearBoard: A seamless medium for shared drawing and conversation with eye contact". Conference on Human Factors in Computing Systems, Monterey, CA, ACM.
17. Jabarin, B., and Graham, T.C.N. (2003) "Architectures for Widget-Level Plasticity", Proceedings of DSV-IS 2003 Portugal, June 11-13. pp. 124-238
18. Jones, T. C. (1986). Programming Productivity. New York, McGraw-Hill.

19. Kurtenbach, G. and Buxton, W. (1991) "Issues in Combining Marking and Direct Manipulation Techniques" In Proceedings of ACM UIST'91. pp. 137--144.
20. Landay, J. A. and B. A. Myers (1995). "Interactive Sketching for Early Stages of Design". CHI '95, Denver, CO, ACM Press.
21. Lank, E., Thorley, J.S., Chen, S.J. (2000). "An Interactive System for Recognizing Hand Drawn UML Diagrams". CASCON2000, Toronto, ON.
22. Malone, T. W. and K. Crowston (1990). "What is coordination theory and how can it help design cooperative work systems?". Proceedings of Conference on Computer-Supported Cooperative Work. ACM Press. pp. 357-370
23. Microsoft Corp. – Netmeeting, http://www.microsoft.com
24. Mynatt, E. D., Igarashi, T., Edwards, W.K. LaMarca, A. (1999). "Flatland : New Dimensions in Office Whiteboards". CHI '99, Pittsburgh, PA, ACM.
25. Pederson, E. R., McCall, K., Moran, T.P., Halasz, F. G. (1993). "Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings". INTERCHI '93. Amsterdam, Netherlands. April.
26. Rational Corp. – Rose, http://www.rational.com
27. Seaman, C.B. and Basili, V.R. (1997) "Communication and Organization in Software Development: An Empirical Study". IBM Systems Journal 36(4).
28. SMART Technologies, Inc. – SMARTBoard, http://www.smarttech.com
29. Stefik, M., Bobrow, D.G., Foster, G., Lanning, S., and Tatar, D. (1987) "WYSIWIS revised: early experiences with multiuser interfaces", ACM Transactions on Office Information Systems, 5(2), pp.147-167
30. Streitz, N. A., J. Geißler, Haake, J. M., Hol, J. (1994). "DOLPHIN: integrated meeting support across local and remote desktop environments and LiveBoards". Conference on Computer Supported Cooperative Work, Chapel Hill. NC.
31. Tang, J., C. (1991). "Findings from Observational Studies of Collaborative Work." International Journal of Man-Machine Studies. 34(2), pp. 143-160
32. Tang, J. C. and S. Minneman (1991). "VideoWhiteboard: Video Shadows to Support Remote Collaboration". Conference on Human Factors and Computing Systems, New Orleans, LA.
33. Thevenin, D., and Coutaz, J., (1999). "Plasticity of User Interfaces: Framework and Research Agenda" Proceedings of Interact '99 Edinburgh, Scotland. pp 110-117.
34. Wang, W., Dorohonceanu, B., Marsic, I. (1999). "Design of the DISCIPLE Synchronous Collaboration Framework". Internet, Multimedia Systems and Applications, Nassau, Bahamas, IASTED Press.
35. Wong, Y.Y. (1992) "Rough and ready prototypes: Lessons from graphic design". Short Talks Proceedings of CHI '92: Human Factors in Computing Systems, pp. 83-84, Monterey, CA,
36. Wu, J., Graham, T.C.N, Everitt, K., Blostein, D. and Lank, E. (2002) "Modeling Style of Work as an Aid to the Design and Evaluation of Interactive Systems". Proceedings of CADUI'02. Valenciennes, France.
37. Wu, J., Graham, T.C.N., Smith, P. (2003) "A Study of Collaboration in Software Design" ISESE 2003, Rome, IT. Sept 29-Oct 1.
38. Wu, J. (2003) "Tools for Collaborative Software Design" Queen's University, School of Computing. Technical Report 2003-462, Queen's University, Kingston, Ontario, Canada, January 2003.

Discussion

[Philippe Palanque] As you use the work style axes as a mean for evaluating the adequacy between tool and a work style do you not need more detailed information for each axes?

> [Nick Graham] All the axes are continuous and we use them more as an informational tool - we worked on making the axes more precise but we did not find it to be more useful.

[Jürgen Ziegler?] Are the dimensions independent or are there interrelationships between eg. modifiability and degree of semantic correctness?

> [Nick Graham] I think we can come up with examples for each pair of axes where you could be at either extreme and if you think of each pair of axes that the extremes are presented as cross products of all four possible positions, then we can come up with examples of all four positions for all the axis pairs, so we are quite confident that axes are orthogonal.

[Grigori Evreinov] Did you think of using parallel coordinate systems?

> [Nick Graham] No, that would be interesting; do you think that would be better?

[Grigori Evreinov] Yes, we have Information Visualization Research Group in our Department (http://www.cs.uta.fi/~hs/iv/) and the parallel coordinates system is presented on their site, so you can try it! or ask about the author Harry Siirtola

> [Nick Graham] That would be interesting!

[Jörg Roth] Your work style model reminds me of the Denver model from 1996 (they have 2 diagrams with 5 axes each instead of your 8)?

> [Nick Graham] There are similar in the sense that they are both related to groupware and presented as "quiviant diagrams". Beyond that the axes are actually very different to my recollection! I have compared to the Denver model, but to give you a proper answer I would have to look at the Denver model again, because I cannot remember the axes exactly!

[Michael Harrison] One of the interesting things about collaborative work is that, just like we have had this conference I will go away to a room and do some work and maybe have some ideas and produce some notes. Next time we have a collaborative meeting I may want to fold that back in to the collaboration and I was not sure how that kind of continuity could be achieved. This characterises different collaborative models whereas that is not essentially a collaboration model, but it is essential to the process of collaboration.

> [Nick Graham] That would be considered a tool transition, so one tool is pen and paper and the other your designed word software. We are very interested in that, so one approach is to say it would be wonderful if you had electronic paper that you had been scrip ling on and that could be imported right in to the tool, a poor mans approach to that would be to scan it, a really poor mans approach would be to sit and type it in. So those are examples of how

transitions can be easy or hard. The whole goal is certainly to find ways of making the transition easier so that people are more likely to do them.

[Hong-Mei Chen, University of Hawaii] The Work style model you presented here seems to be domain-specific to software design in your empirical case studies and not applicable to other kind of collaborative work. For instance, some brain storming tasks (as studied in Group Decision Support Systems - GDSS) consider important factors such as social cues and anonymity to be important.

[Nick Graham] I agree with you that there are many other axes that we could put in and we have actually studied it in IFIP WG 2.7/13.4 and discussed the kind of transitions that would come up, e.g. with respect to privacy. An example could be a situation where you start out in a context where privacy is not important to you and the all of a sudden you are asked to enter your credit card information and privacy becomes very important to you. This just to say, that these are also important issues and we do not claim to have solved every issue in the world. We have used this model in other domain, but will not make any claims that this is applicable to any domain and maybe we will come back next year with the 40 dimensions version!

[Rick Kazman] How do you deal with multiple updates to a single document when people work asynchronously but they want to merge their work?

[Nick Graham] We do not support merging in general since it is a difficult problem, but we do support merging of the whiteboard freehand drawings. Merging MS Word documents alone is big problem in it self!

[Rick Kazman] Are you aware of any general solution to the multiple merge problems?

[Nick Graham] No, all the solutions I have seen are point solutions often commercial, such as for MS Word, but no good general solutions.