The Design and Building of the Graphic User Interface for The Collaborative Desktop

Konrad Tollmar and Yngve Sundblad
IPLab (Interaction and Presentation Laboratory)
NADA, KTH, S-100 44 Stockholm, Sweden
Email: konrad / yngve @nada.kth.se

Abstract
In this paper we describe design and implementation considerations for the graphic user interface for CoDesk (the Collaborative Desktop), which is an environment for CSCW (Computer Supported Cooperative Work). CoDesk is an attempt to make collaboration a natural part of the daily use of a computer. Our way to achieve this is to put the user in the centre of the computing in a similar way that applications and documents are defined and visualised in the desktop metaphor.

The Collaborative Desktop is aimed to be a generic environment with a GUI (Graphic User Interface) that can be used by many persons with different backgrounds. We have developed it as an extension of a computer environment we know works for many users: the desktop metaphor that has made daily computing a lot easier and error tolerant.

Here we give an overview of the design and implementation of the CoDesk user interface: design principles, design process, implementation techniques. In an iterative and interactive design process we have focused on creating a dialogue between the developers and potential users. The last chapter contains some conclusions and a vision how the traditional desktop metaphor could be extended using new ideas of the graphical design.
The Design and Building of the Graphic User Interface for The Collaborative Desktop

Konrad Tollmar and Yngve Sundblad
IPLab (Interaction and Presentation Laboratory)
NADA, KTH, S-100 44 Stockholm, Sweden
Email: konrad / yngve @nada.kth.se

Background
The effort within our research group IPLab to model, design and implement prototypes of an environment for CSCW (Computer Supported Cooperative Work) started within the Swedish MultiG research programme (1990-1993) on distributed multimedia applications in high-speed networks [33]. Since 1993 the effort is funded by the Swedish technical and work environmental research programme SAMT (Samarbete och Teknik, i.e. Cooperation and Technology). It is also one of our contributions as a partner in the ESPRIT Basic Research Action COMIC (Computer-based Mechanisms for Interaction in Cooperative work).

Our work is based on studies of cooperation in knowledge work, design work and software engineering. We are aware that this spans a limited range of work but this form of “free work” is becoming more and more common into several work settings, [20]. Another argument for using these forms of free work as a basis is to get rid of so structured models of cooperation that they lack in reality by assuming a “perfect world” that seldom occurs, [5].

In reports, videos and prototypes our research group IPLab has analysed collaboration in design and specified some general functional requirements and interface design principles on CSCW support in a distributed design environment and distributed software engineering: [27], [36], [1].

Our earlier results can be summarised in terms of a set of general requirements that a CSCW environment should support: informal collaboration, sharing and record keeping of information, presentation of ideas, sharing of background knowledge, strategies reducing the need for co-working and tools that support co-working.

Building CSCW applications and systems is difficult and special development tool support is needed. A number of CSCW tool kits and environments have emerged for development of multi-user interfaces, e.g. Rendezvous [32], shared editing systems, e.g. GroupKit [14] and conference systems, e.g. MMConf [12]. Criticism in e.g. [40] argues that unfortunately many of the earlier CSCW environments:

"... provide little or no support for representing the cooperation taking place.”

They and we argue that the most obvious drawback is that many earlier environments are closed applications rather than open platforms. Clearly the underlying technology is not mature enough and the architecture in distributed system and database management system (DBMS) maps badly on
the concepts needed in CSCW. Note that we do not mean that CSCW support should not be built from existing technology but that special attention must be put on supporting cooperation concepts.

Thus there is a need for basic CSCW environments that could be used for basic cooperative work and that support future specialisation and incorporation of new tools.

CoDesk design principles

Extended desktop metaphor

CoDesk is a basic environment for CSCW where we have extended the traditional desktop metaphor with a few new objects that enable cooperative work. Without limitation to a specific model of cooperation each user could tailor, form, her desktop to the individual need for cooperation and communication. In CoDesk it should be as easy to look for your colleagues as for shared or individual working material. Central in CoDesk is support for groups or teams to form cooperative settings.

Primarily CoDesk provides mechanisms that extend the network from a computer network to also be a user network by integrating the essence in communication and collaboration via different tools and media. Figure 1 gives a view of the CoDesk user interface.
Direct manipulation

Another key principles in CoDesk is to provide an easy-to-use environment. In general the CoDesk user interface can be described as a direct manipulative graphical environment where we have defined a set of graphical objects, represented as icons. The principal objects in CoDesk are Member, Group, Room, Document, Folder, Tool.
Objects can be manipulated with direct manipulative drag-and-drop actions, e.g. dropping a user icon on the telephone icon to initialise a phone call.

**FIGURE 2 - Drag and drop operations in the CoDesk.**

**Icon layout**
There are pairwise similarities in the form of use of the traditional desktop objects and the cooperation objects that we visualise using guidelines formulated in [26] for icon design based on the cognitive impact:
- A rectangle, vertical square root of 2, for basic entities: users and documents
- A horizontal golden rectangle for permanent containers: groups and folders.
- A diamond for movement and tension: tools and rooms

**Views**
When a user enters a Graphic User Interface (GUI) system, such as Macintosh, [2], Windows, Open Look or Motif, [31], she first meets the window system. Most, not all, of these systems also define a desktop layer, with some desktop tools. A system like CoDesk extends this with an additional layer, the network layer. That layer gives access to remote and shared data, communication and access control. This puts an extra cognitive load at the user that needs to be reduced by a simple and easy to learn desktop layer.

One typical problem encountered is that multi-user interfaces often provide unexpected access to material that should not be accessible. We try to avoid compromised security by making it clear what is shared and available.

Different views of the same data can personalise and increase effectiveness of multi-user interfaces but can lead to poor awareness of the state in the data and radically decrease gained benefits and reduce the understanding of the system. Most views in CoDesk have at least three different representations, as a window, as an icon in a standard browser and as an information dialogue, with a colour key indicating the current access state.

There is a tricky balance between the demand of reducing the complexity in the system and the need for efficient representations. Visual clues within the views take care of some of the problems. It is very important to make it clear for the users how accessible the data is, from free and unlocked to unavailable for any kind of change.
Awareness

Important in supporting cooperative work is to handle shared working resources. When using our prototype we found that users of shared resources needed to be aware of the presence of other users and their access to the shared objects. This cooperation awareness, or “social browsing”, [11], can be provided in the graphical user interface by different forms of highlighting objects. So far we have defined 3 generic forms of awareness: active, notify and passive. An active object indicates that it is used, e.g., a user that has logged in to the CoDesk. An object can also be notified, this provides a mechanism to trigger colleague’s attention to certain objects. A notified object expires after a certain time and become a passive object - that is the default awareness mode.

Support for casual interactions

Studies of work by our group IPLab and by others, [26], [29], have shown that casual interaction is of utmost importance for how work is done. Interaction through non-planned meetings etc., gives the rich possibilities for exchange of ideas and conversation that a good workplace can offer.

The awareness mechanisms described above also supports the user in finding informal situations for casual interactions.

A net of knowledge

One important need in modern work life is efficient handling of information in a society that produces so much information that traditional text-based and TV-based media are insufficient. The need to handle the “information overflow” has been characterised as a change in the social paradigm of society, [30], and different visionary computer based solutions have been suggested, [7] and [10]. These solutions are all focused on the management of published information in global and open but personalised libraries.
A common way to handle information overload is to use other people as references rather than excessive reading of documents, [18]. The KnowledgeNet, defined in [28], and integrated into CoDesk, is designed to support a social work situation in which collaboration among peers can take place by sharing and integration of knowledge. This process is supported by shared knowledge bases of experts accessible by CSCW tools and undocumented knowledge is made public in the same way as libraries make documented knowledge public.

There are many ways in which undocumented knowledge can be shared in a distributed environment. Knowledge can be broadcast in the form of lectures, announcements etc. Knowledge can be obtained by explicitly asking others for information and advice. Knowledge can be implicitly transferred during meetings. A database about “who-knows-what” can be made accessible and maintainable by the participants, figure 4. This information space is “peopled” with the originator of the information visible and accessible.

One of the main characteristics of The KnowledgeNet is to be user controlled, i.e. the members themselves decide if one of their documents should be made public and define and maintain their own expertise areas.

**A tool approach**

In CoDesk the KnowledgeNet vision is supported by providing all users with tools for a seamless integration of synchronous and asynchronous modes of interaction, for example enable social ad-hoc communication and let the user toggle between activities as in real life. We think that one can use CoDesk both to talk and work as a complement of more formal and planned work processes.

The tool-oriented approach, as in [9], aims at designing a user controlled environment that facilitates for the users to do what they want, without limitations and assumptions imposed by the system.

Usually the tool perspective focuses on individual use that one might find contradictory to cooperative work. But, as stated in [13], user control is a key factor for usability, certainly also for CSCW systems. With a tool-
oriented approach the users can apply and develop individual original skills that will form the core as the basic resource in cooperative work teams.

As in, e.g., [29] and [4], we thus propose and concentrate on design of generic collaborative tools rather than trying to model specific collaboration tasks.

**Extendibility of the information space**

CSCW systems based on “a shared information space” gain growing attention. Such CSCW systems are attractive as both time and location independent. From our studies we know that they need to be augmented by direct user communication.

To build a shared information space in an environment like CoDesk you need to use it. There is no feasible way to pre-fabricate such an information space. As in [17] we strongly believe that the initial design of the system will be re-designed during the use of it. This does not decrease the demand of the initial design. We claim that it is very essential both to be able to relate to and use “old” information and provide a more formal information space that reflects the organisation where it is used.

**Objects and room metaphor in the CoDesk GUI**

**Members, groups and rooms**

The most central type of object in CoDesk is the individual person, member, represented both as an icon and as forms (e.g. “cards”) with attributes, including name, communication lists and KnowledgeNet who-knows-what information. Groups are simple collections of members.

The room metaphor is used for different interactive settings, early used as an extended desktop for single user work arrangement at Xerox PARC, [16], and also for CSCW applications, e.g. in [6]. In our metaphor a room is used to represent a collaborating group or a specific action. The rooms are additions to the groups and should been seen as dynamic cooperative settings. Rooms are familiar environments for cooperation and work. Rooms are where you meet people, do your work, read a paper etc. For movement and navigation in the rooms the desktop metaphor is used through pictorial representation, the graphical user interface, and search-and-retrieve tools. Note that rooms are not only used for sharing but also for individual use like a private mail list.

We also explore the role of rooms in supporting “social browsing”, as in Cruiser, [37], by “group awareness” mechanisms. The user can set allowed “disturbance level” from group members, in the same room or making a “random walk” visiting a couple of rooms. The most common way to communicate with some members will be to install a common room with some tools and working material, e.g. documents, specific for that group. To support temporary connections with other group members a temporary room could (automatically) be installed by, for example, a direct phone call to another user.

**Tools and Documents**

As noted in [35] work behaviour is characterised by multitasking, and many activities and interactions are structured into communication chains that
criss-cross each other. This means that tools for collaboration should allow and support many collaborative activities at the same time. A user can jump from one activity to another, have “sleeping” activities that will be continued later on, etc. The ability to adopt different kinds of tools is considered, [15], to be a main feature in a successful CSCW system and has therefore also been one of our major goals. We believe that our architecture makes it possible to integrate and use a large amount of ordinary single user tools into the Collaborative Desktop.

Common tasks for which collaboration through computer is very suitable is writing text, designing graphics, sound or video together. Here the collaboration is mediated through the “material”, documents, we work with. As stated in [5], designing CSCW system from the viewpoint of a common information space could be very valuable and useful.

Duality between old and new basic objects
To simplify the use and understanding of these different kind of objects we have worked out a dual relationship between the new CoDesk objects and the traditional desktop objects. This means in our CoDesk system that the syntax and semantics for performing direct manipulative operations on a cooperation partner is similar to handling document, room is similar to tool and group to folder:

— Users represent knowledge and competence, like documents represent formal knowledge.
— Groups are used to form groups of user, as folders are used to organise documents.
— Rooms are means to get in contact with users, as tools are used to get a visual appearance of documents.

The folder object gives a simple and convenient container for sorting and organising documents. Folders are collections of documents and/or folders (recursively), which gives a hierarchy similar to groups.

Iterative Design of the Collaborative Desktop
Our design has not been directed towards a specific category of users but focused on presenting our ideas and conceptual model in different contexts to a rather large spectrum of plausible users.

Basically have we iteratively used a different techniques in the design of the Collaborative Desktop: HyperCard prototypes, functional prototypes, live demos, videos, paper mock-ups as well as, naturally, academic studies of earlier work. In this section we give some experience of using the different design techniques.

A strong focus has been to from the very beginning to show the visual appearance of the interface and the model in our prototypes. This has let us as designers to use our creativity rather than constrained us into structures of representations.

Hypercard Prototypes
An early phase of the design of an environment for cooperation via workstations was creation of several HyperCard prototypes. The first ones were rather traditional applications complemented with static views and tool panels.
Rather soon we started to experiment with a direct manipulative interface. It was rather easily to prototype in HyperCard, by writing a couple of drag-and-drop extensions. After some iterations it became a direct manipulated pictorial (graphical) user interface, see figure 5, in which it is possible to organise and view structures and the basic objects: members, groups, rooms, documents, folders and tools.

FIGURE 5 - First HyperCard prototype.

We have later on continued to use this prototyping techniques to examine some of specific tools, e.g. an authoring tool specially designed for cooperative writing and illustrated in figure 6.
In conclusion this form of prototyping has let us very quickly create a dynamic visual representation of some ideas. Some could for example be too complicated to realise while others work as test beds for new ideas. Basically we use two kinds of prototypes, horizontal and vertical. As we experiment with the general CoDesk environment we use horizontal prototypes to design the overall feel and look. HyperCard prototypes for specific tools are examples of vertical prototyping. Using both vertical and horizontal prototyping seems to be promising, but as demonstrated in [19] it might be far from sufficient. Attention needs also to be focused on other aspects in the system design.

Functional Prototype Demonstrations

For real testing with users and live demonstrations of the CoDesk ideas we have built several generations of “semi-working” prototypes for a multimedia based UNIX environment (Sun SPARC stations).

With live demonstrations at major CSCW and Human Computer Interaction conferences, CSCW’92, [3], ECSCW’93, CHI’94, [39], and at seminars and workshops, we have reached leading communities with CSCW knowledge and have got very valuable feedback and comments. In many aspects this has been very creative and has given us credibility and self-confidence.

The computer science challenges to realise parts of CoDesk environment has also given us a forum to reach the very living and rich “hacker” community.

Videos
A traditional design medium is video, which can be used as a replacement of live demonstrations although it lacks the dynamics of a demonstration.

Another drawback with video production is that it needs considerable planning especially if you do not have a video editing system in-house. We have also noticed a considerably increased demand on the video quality for conferences submissions, which is hard to meet. We have taken steps in the opposite direction by acquiring a simple to use video editing system that we hope can be used by almost every one in our lab. Our intention is that the quality should be found in turn-around time rather than the quality of the image and sound mixing.

A strong feature of video is that it enables us to show examples of use of a system in different environments. Scenarios that span over longer periods can be shown. With video a “polished” version of a prototype system that resembles as closely as possible a finalised product can be simulated.

We used videos, e.g. [38], for both purposes, as simplified demonstrations as well as polished product.

**Paper Mock-ups**

A complement to building software prototypes is to use paper mock-ups with the obvious advantage in the early design phases that with a paper model it is clear that the design is not fixed and could easily been changed. This tends to relax the discussion and makes paper mock-ups a good medium to communicate about design.

A very simple form of mock-up is overhead slides. Other forms are:

— Abstract cards that could be used as a game.
— Screen shots (with slips representing icons, menus etc) to perform walk through in the system.
— Paper and pen for sketching.

We have experimented with screen shots of CoDesk, see figure 7, to illustrate system walk through and testing of the metaphors.

**Figure 7 - Using paper mock-ups in the design.**
Some immediate qualitative results from just a few test subjects were that the basic understanding how direct manipulation in a desktop interface not as common as we expect it to be. There is a rich spectrum of different approaches and ways to perform a certain task. Even if the basic functionality in CoDesk is rather simple we need to provide a multidimensional and consistent orthogonal set of functions that gives a freedom for many alternative ways to perform a certain task.

Another result is that naming is extremely important to do properly. Functions are often given either too concrete or too abstract names. Names need to be consistent and nicely laid out to make sense and be readable. It happened that people corrected names with the pen on the paper models!

We plan to use paper mock-ups to experiment how the introduction of a system like CoDesk be done in a real work situation. Earlier work as with the "Organisational Kit", [8], shows that this form of situated design could have a strong impact how a system is incorporated into an organisation.

**Implementation aspects of the CoDesk environment**

Here we give a short overview of some conceptual and implementation aspects of the Collaborative Desktop. The features we highlight are the Graphical User Interface classes, event and sharing mechanisms.

**GUI classes**

The basic structure is based on an object oriented view where we develop the graphical user-interface through interactive objects. We have used InterViews, [24], for this. Like other modern user-interfaces we make a distinction between data objects and view objects.

The basic graphical components define the user interface to the subjects, which contain the abstract information in the application without any information of how it will be represented on the display. The information about how to display and interact with a subject is contained in the interactive object.

Our interaction model supports three basic direct manipulation operations:

- Select operation, e.g. mouse click.
- Action operation, e.g. double click.
- Select and drag operation, e.g. drag-and-drop or move operation.

These are the basic operation that every interface object understands. We can compose the basic operations into composite operations, e.g. multiple selection drag-and-drop.

Four different classes contain the graphical components: Browser, Editor, Info and Icon. Each of these represents different views of the abstract Subject class. A Browser is an abstract class for all kinds of graphical containers while an Editor could be used to derive a specific editor for an object type. Info is an abstract class for displaying properties and attributes of a particular object. Icon is an abstract class for icons containing the drag-and-drop functionality.

To make the interface consistent we need some restrictions on the basic operations. Those restrictions are mainly properties of the Subject. An example is a drop operation, which indicates different visual feedback of the drop target depending on the target state. Different states can be
visualised by a generic update mechanism embedded into all graphical
components.

**Events**
The CoDesk should enable users to be aware of the actions of other users.
The mechanism used to provide this form of awareness is event handling.
When any action takes place on an object, that is of possible interest for
other objects, events are generated. The implemented services provide
facilities to allow events to be defined, created, related to other users and
handled in appropriate ways. Users and objects can declare interest in
certain events or types of events by registering a subscription.

Within the CoDesk system events belong to a special class of system
objects. Based on the information held in the event objects, the service
client can calculate awareness factors between objects and thence between
users (of those objects).

The Subscription Handler is a kernel in the event services that supports
users’ (and objects’) awareness of each others’ actions.

Mechanisms are needed to handle a subscriber’s
- Immediate information about changes in an object
- Information about changes in an object when accessing it
- Immediate information about anyone taking interest in (access to, or even
  just pointing at) an object
- Information, when accessing an object, that someone else has accessed it
- Information about interests of other users

These services are also needed for sharing objects in order to provide
“live” awareness as well as supporting awareness of shared object.

**Sharing objects**
As stated above cooperation mediated through a group’s working material
needs to be supported in a CSCW environment. A considerable amount of
work in CSCW systems has been devoted towards coordination and
communication. CSCW support for time and place independent shared
objects also needs attention.

The feature which most distinguishes the sharing of objects in CoDesk
from other multiuser storage systems is the focus on sharing and the
provision of mechanisms which support the management of this sharing.
The explicit goal is to provide an awareness of the action of others on objects
within the shared object service. When an object is accessed or altered
CoDesk should inform the dependent users that the action has occurred in
order to promote awareness across the service.

We started our first implementation of sharing objects in the CoDesk
using the X.500 directory service. We use the X.500 directories as a
distributed database. The directory is intended to support human user
querying, allowing the user to find telephone and address information of
organisations and other users.

The X.500 directory is:
- a database.
- intended to be very large and highly distributed.
- hierarchically structured, with entries arranged in the form of a tree
We use X.500 as a general distributed database. It is used for saving data and information shared by members on an Internet domain. We have found the naming facilities very useful and rather close to our requirements. Currently we are in the process of replacing the X.500 directory services with a modern object oriented database (OODB) system. We have to extend the OODB with some functionality needed to implement some of the sharing mechanisms of CoDesk. As an example the lastModify attribute in X.500 is needed to provide a flexible and efficient awareness mechanism. For that we need to monitor (subscribe to) some conditions in the database. That is handled by definition of a subscription via a trigger, once-only or perpetual. Triggers are defined within the classes as

```cpp
class A {
    some-list
    void do_something();
    Public:
    Name name;
    Status mode;
    void update(Status x_mode);
    Trigger:
    Perpetual subscribe() : changed(mode)==>do_something();
}
```

**Some remarks on the Graphical User Interface**

As described earlier our idea is to enable sharing of your local desktop by sharing your and your colleagues working material and tools. Thus, the users needs to configure the "desktop" according to their needs of cooperation. We have built the CoDesk system on the traditional desktop metaphor to enable users to relate and use earlier experience when they use the CoDesk environment. But there are both problems using the traditional desktop environments and challenges in extending it to fit better into a CSCW situation.

**Drawbacks in the desktop GUI**

CoDesk extends the desktop GUI with an additional layer, the network layer. That layer gives access to remote and shared data, communication and access control. This puts a cognitive load at the user that needs to be reduced by a simple and easy to learn Desktop layer.

—We believe that the graphical user interface could never be to simple due to the complex environment the users need to handle.

Tools for computer supported cooperative work suffer from a scarce resources even on today’s workstations — the screen space. A common remark that we have heard is: "Why do those standard GUI that we see today have to be that static and dull?". Our standard answer to that is that due the complex environment and limiting screen an environment as CoDesk or Apple Finder needs to be rather transparent in order to leave "space" to the applications and tools. But as we will describe in the next sections there are several extension of the GUI that could emerge from considerations of CSCW.

**New Styles in CSCW desktops**

As argued earlier we need to visually represent cooperative awareness so that users of shared resources can be aware of the presence of other users and their access to shared objects. So far we have defined 3 generic forms of
awareness: active, notify and passive. This form of awareness on per object basis is only one dimension of several.

Cooperative awareness that spans multiple objects is also important. Group cohesion is normally gained using bulletin boards, or some other common area, to place notes and drawings. A generic multipurpose note service like post-it would probably be effective, see figure 8.

Also wanted / needed (from the paper mock-ups) is support for visually provided contextual clues — like in what environment this is. An example is the use by different TV-channels of their logotypes on screen. On the desktop that could be represented as different form of backgrounds and the use of personal icons and layout, see figure 8.

**Conclusions**

From the Collaborative Desktop effort we have learnt that
- Producing a working prototype of a generic CSCW environment is, even with the best generally available tools, a very considerable effort
- Tools and components for distributed systems, interface design and implementation and distributed databases are not well designed for integration into one system
- Awareness of other users, that is crucial in CSCW applications, puts specific demands on distribution, interface and database tools, including making access visible rather than “transparent”
- Sharing objects and information as a way of mediating cooperation is in real world applications a feature as important as the communication channels
- Specific collaboration services and shared object services are useful modularisations for building CSCW environments

We conclude that CSCW system development tools where the cooperation taking place can be represented through collaboration and shared object services are very useful and should be elaborated further according to the principles and features described here. For gaining better experience and knowledge of the human and social factors of CSCW in real world applications it is crucial to make production of working prototypes for field testing much easier and faster. The ideas, experience and services presented here contribute towards the goal of a development environment
for fast building of functional CSCW prototypes and efficient construction of full scale CSCW applications.

**Acknowledgements**

We are indebted to the IPLab and Esprit COMIC communities for numerous discussions on the issues presented here, especially to Hans Marmolin, our joint designer of CoDesk.

**References**


17 Henderson A. & Kyng M., There’s is no place like home: Continuing Design in Use. In Design at Work, Greenbaum & Kyng (Eds.), Lawrence Erlbaum Ass., Hillsdale N.J. (1991)


20 Kling R. & Iacono S., Desktop Computerization as the Product of Social Movement, University of California, Irvine (1985).


38 Sundblad Y., Marmolin H. & Tollmar K., The Collaborative Desktop, Video 12 min, IPLab, NADA, KTH (1992)

Figure texts

FIGURE 1 - CoDesk - An environment for CSCW

FIGURE 2 - Drag and drop operations in the CoDesk.

FIGURE 3 - Awareness in the GUI, welcoming (lit up) and logged out (dimmed) member, active (dimmed) and passive (white) document.

FIGURE 4 - Users informal knowledge forms competence groups.

FIGURE 5 - First HyperCard prototype.

FIGURE 6 - A HyperCard Prototype of Cooperative Authoring using versioning.

FIGURE 7 - Using paper mock-ups in the design.

FIGURE 8 - "Yellow Notes" + visual context on the desktop.