# Groupware Concepts for Collaborative Mobile Groupware

**Vijay Gupta**

Research Scholar, Pacific university, Udaipur India

**ABSTRACT**: ***This paper has examined the implications for traditional groupware concepts and techniques when considered in a mobile context. In addition, a range of modern groupware toolkits have been described and their suitability for developing mobile groupware evaluated.***

----------♦----------

## OVERVIEW

Unfortunately, many of the design principles behind current groupware techniques assume the existence of a reliable and constant underlying communications infrastructure.

The following sections describe the potential impact of mobility on the following key groupware concepts:-

• Collaboration-aware and collaboration-transparent groupware

• Management of shared data

• Coupling

• Awareness

## COLLABORATION - AWARE AND COLLABORATION-TRANSPARENT GROUPWARE

A number of classifications exist for groupware systems. One important example is the categorisation of groupware systems that support multi-user interfaces as either collaboration- aware or collaboration-transparent [Lauwers,90].

**• Collaboration-transparent Groupware**

This class of groupware (also referred to as conference unaware [Riexinger,93]) has no facilities for handling cooperation embedded within the actual groupware application itself. These systems are generally derived from existing single-user based applications using some form of view-sharing software [Greenberg,90] to share the application's display across several workstations. One example of a distributed view-sharing system for the X windows system is XTV [Abdel,94] which works by intercepting and redistributing the X based protocol streams. This approach enforces strict WYSIWIS consistency across all displays and, consequently, offers little scope for end-user tailoring.

**• Collaboration-aware Groupware**

This class of system is explicitly designed for supporting cooperation between multiple users. Thus developers are required to decide how users should be presented with shared data and how they should be able to control and manipulate these representations. Because of this, the development effort involved with building collaboration-aware groupware is generally far greater than that involved in building collaboration-transparent groupware.

There are a number of problems associated with using collaboration-transparent groupware in a mobile environment. Firstly, the lack of WYSISIS flexibility afforded by such systems means that all conference members are required to receive each and every X event despite the fact that some members may have drastically different qualities of network connection. Secondly, the demands on network bandwidth are high owing to the very low level of granularity used, i.e. individual X events are distributed. A third, and associated, problem concerns the handling of latecomers (or members who have suffered a long period of network disconnection), i.e. if all X events are archived then the storage requirements imposed on servers would be prohibitive. However, because it is often important for users to observe the steps taken to create a shared artefact, such as a drawing, the simplistic approach of simply transmitting a screen shot might be inadequate. These problems are largely overcome by collaboration-aware groupware because the enforcement of WYSIWIS can be relaxed and the granularity of updates can be chosen to suite the semantics of the application itself.

## MANAGEMENT OF SHARED DATA INTRODUCTION

A number of researchers advocate the notion of separating an application's underlying data model from its associated graphical representation or view [Patterson,91], [Graham,92] and [Hill,92]. Examples of this approach include Smalltalk's Model-View-Controller (MVC) paradigm [Krasner,88] and the Abstraction-Link-View (ALV) model incorporated in the Rendezvous toolkit [Patterson,90]. The main argument for this approach is that it enables developers to build groupware in which different users can have different views of the same data model. Figure 3.1 shows an example where a user 'A' views some shared information as a pie chart whilst user 'B' views the same information as a bar chart One implication of this approach is that applications require shared access to the actual data model in a distributed environment. This raises the interesting issue of where exactly the data model should be located in the environment. Commonly, the groupware system either replicates data at each member node or locates the data model at some central repository. Groupware researchers have long argued the relative merits of centralised vs. replicated data architectures [Crowley,90], [Greenberg,90], [Patterson,94], [Hill,92], [Greenberg,94], [Wilson,95] and have also argued the need for dynamic data architectures [Greenberg,96], [Dourish,96a]. The following sections describe each of these three approaches in more detail and with particular focus on the potential impact of mobility.

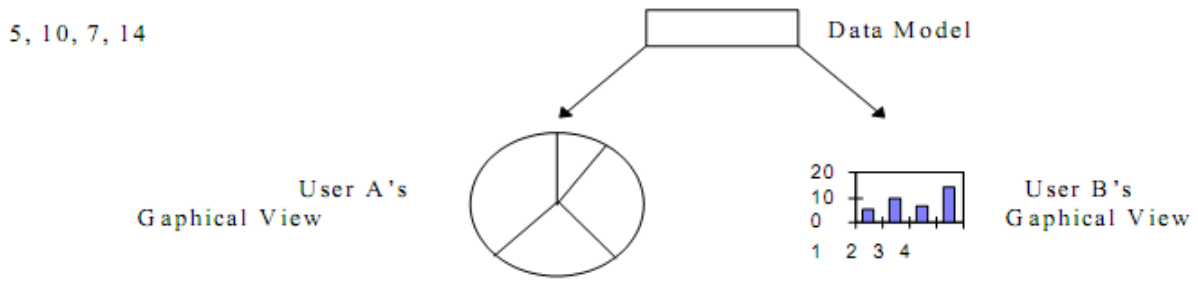


***Figure 1.1, The separation of the data model from its view.***

## CENTRALIZED DATA ARCHITECTURES

A centralized data architecture makes use of a centralised server to store the shared data. Client processes residing at remote sites are responsible for passing user input events to the central server and updating their display after receiving update broadcasts from the central server. The central server is responsible for processing these events and then broadcasting any changes to every remote site. Figure 3.2 illustrates a typical centralised architecture for a Tic- Tac-Toe application.

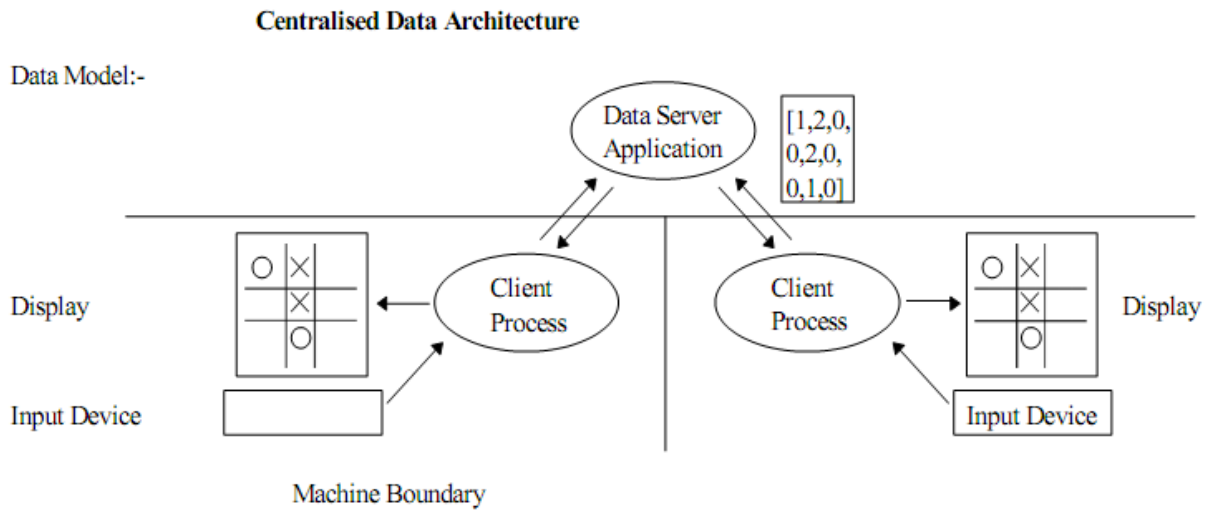


Figure 1.2, The centralised data architecture.

The advantage of a centralised scheme is that maintaining synchronisation between members is straightforward. The reason for this is twofold. Firstly, the shared data model is located in one place. Secondly, all user input events are serialised by the server, i.e. the server must process any single user input and then broadcast any required updates, before processing the next input request. For this reason the use of a centralised data architecture for groupware has had a number of advocates [Crowley,90], [Greenberg,90], [Hill,92]. However, the cost of this simplicity can be poor performance because the single data server application can become a processing bottleneck.

It is relatively straightforward to use locking techniques in order to support concurrency control in centralised data architectures. This is because the central data server can be responsible for managing the granting and denial of locks. One of the key issues when supporting locking is the choice of an appropriate object granularity on which locking can be performed. For example, in a co-authoring system, if the granularity of the object to be locked is set too large then the availability of the shared document will be unacceptably poor. This would be the case if the object being locked was the entire document and would result in no concurrent editing being permitted. Alternatively, if the locking granularity is set too small, e.g. locking single characters, then, although document availability would be very high, it is likely that the number of concurrent editing conflicts would also be high.

## REPLICATED DATA ARCHITECTURES

Replicated data architectures such as DistView [Prakash,94] maintain a replica of the data model at every site. Figure 3.3 shows an example of a replicated data architecture in which the application is also replicated at each site. Each replica is therefore required to coordinate both local and remote actions and also attend to the synchronisation of all copies of the shared data.

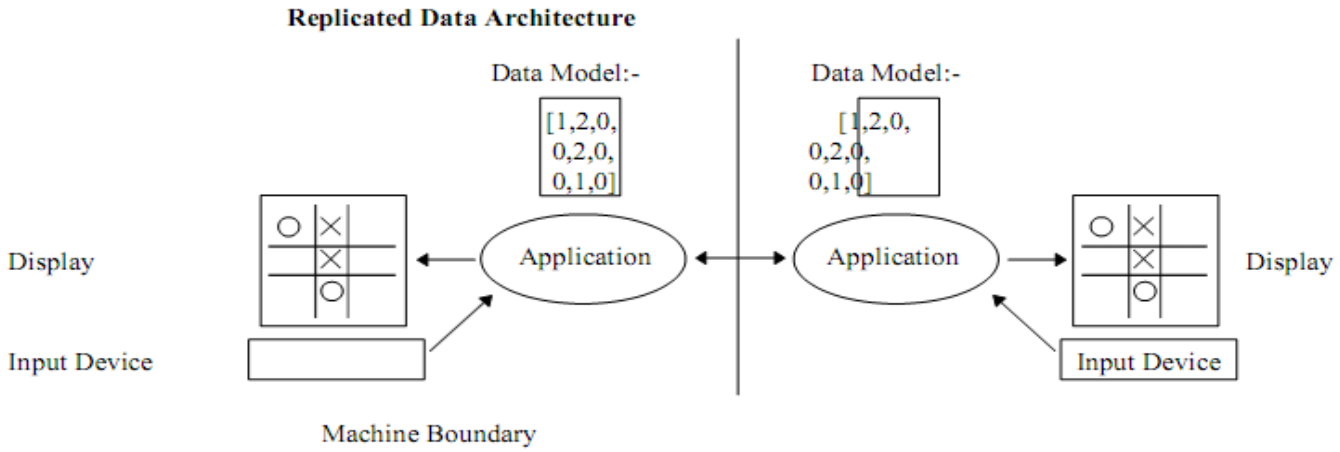


Figure 1.3, The replicated data architecture.

The main advantage of utilising this form of data architecture is the potential to perform parallel processing, i.e. the handling of user interactions and display updates can occur in parallel at each site. Process bottlenecks are thus less likely to occur.

The main disadvantage of the replicated approach is the increased complexity concerning issues such as concurrency control. Different systems manage this complexity in a variety of ways. For example, DistEdit uses the reliable atomic broadcast mechanism provided by ISIS to synchronise updates and thus achieve concurrency control. More specifically, DistEdit maintains a copy of the state of the buffer for each user and uses an atomic broadcast protocol to ensure mutual consistency between the buffers by guaranteeing that updates arrive in the same order at all group participants. However, the length of time that this may take (especially in the advent of communication delays) is non- deterministic.

DistEdit was designed so that only one user at any one time can have permission to make changes to the text. This user is termed the master whilst all other users are termed observers. The master may relinquish control at any point by issuing the appropriate command. The layering structure of a group editor using the DistEdit toolkit is shown below in figure 1.4.

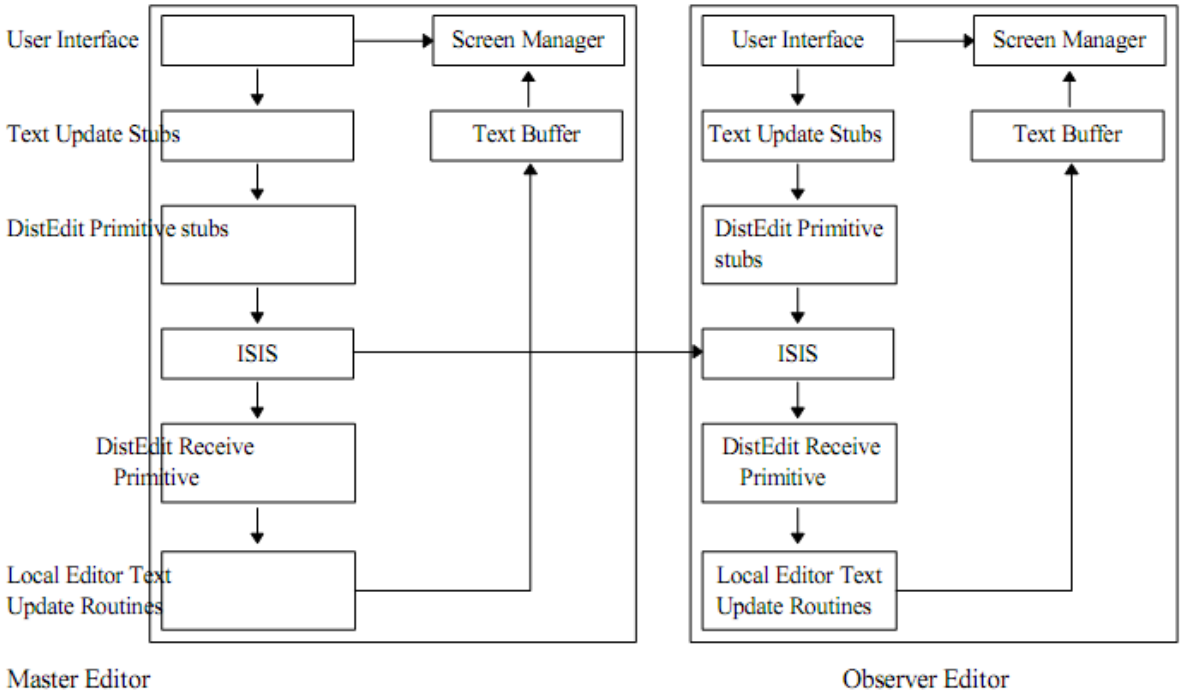


Figure 1.4, Structure of a group editor using the DistEdit toolkit.

As can be seen from the diagram above, the DistEdit primitives, called by the text update routine stubs, forward their argument, through ISIS, to all the members of the editing group, including the master. At the receiving end of the ISIS broadcasts are the DistEdit receive primitives, which after performing the appropriate cursor adjustments then call the local DistEdit primitives. In turn, the local DistEdit primitives layer maps the DistEdit primitives back to the editor's original update routines.

In replicated data architectures, there are two approaches that can be adopted when selecting when to update the local copy of shared data. Either the local copy can be updated first or the local copy can receive the update via the broadcast mechanism being used. If the former approach is used, then the local application will have a fast response time but will not be synchronised with the replicated copies of data until they too have been updated.

## ANALYSIS OF DATA ARCHITECTURES WITH RESPECT TO MOBILITY

Operation in a mobile context can cause problems for groupware based on either centralised or replicated data architectures. In particular, when utilising a centralised data architecture a group member that becomes disconnected from the server application is unable to access any shared data; however, other group members remain unaffected. Conversely, groupware based on a replicated data architecture allows data access to members even when they are disconnected from the rest of the group. The key problem with the replicated approach is that difficulties with group communications are likely to result in delays before the consistency of shared data across all group members can be achieved. However, such delays can be reduced by weakening the strength of consistency guarantees used by the system's broadcast mechanism.

To summarise, it is difficult to argue whether centralised or replicated data architectures are the most suitable when developing mobile groupware. One needs to carefully consider the set of trade-offs involved, including: programming complexity, synchronisation requirements, the number and location of participants expected and predicted network latency. It can be argued [Greenberg,96] that because parameters such as the number of group members and the quality of the network are dynamic, no single choice of static architecture solution will suffice. Indeed, what is required are dynamic and reactive data distribution architectures.

## COUPLING

Dewan and Choudhary [Dewan,91] define coupling as the means by which interface components share interaction state across different group members. Applications are

described as employing either tight or loose levels of coupling.

**• Tight coupling**

For certain highly-interactive tasks, e.g. group sketching, the process of viewing the creation of a shared artefact can be as important as viewing the final artefact itself. Such tasks require a tight coupling between the user's actions affecting the shared artefact and the updated representation of the shared artefact. For this reason, the granularity of updates tends to be very small, for example, a tightly coupled button widget would appear identical on all displays as it was being pressed, moved and released. In general, the tighter the level of coupling, the more sensitive the application will be to communication problems because of the higher volume of data traffic required by the fine granularity of group updates.

**• Loose coupling**

With loose coupling, one group member's actions are propagated to other group members only when a critical event is performed, i.e. the final state should be the same, but intermediate states are not observed. For example, a loosely coupled button widget might only reflect the release action, with intermediate feedthrough eliminated

[Bentley,94]. As a consequence of the fact that loosely coupled systems exchange state less frequently, compared to tightly coupled systems, the performance demands on the system are much reduced.

Dewan and Choudhary [Dewan,91] argue that flexible coupling is important for a variety of reasons. Firstly, groupware programs range from fully synchronous, through nearly synchronous, to asynchronous and coupling can be regarded as simply another way of controlling synchronicity. For example, one can argue that the only difference between a real- time text chat program that shows characters as they are being typed vs. complete messages (asynchronous email) is the level of coupling. Secondly, tightly coupled actions showing intermediary steps may be annoying to users in situations where they are pursuing their own individual work. Dewan implemented flexible coupling in the Suite toolkit [Dewan,92] by allowing programmers and users to set coupling attributes that indicate the level of coupling required for individual interaction. Suite also enables interaction entities to be considered as disjoint coupling sets. For example, the data state, view state and a format state can be coupled independently. This enables the view of the shared data to be formatted in different ways across displays. Furthermore, action coupling can be set to determine how the commands (or call-backs) attached to user actions are executed at other sites.

In his 'PREP' shared editor architecture, Neuwith [Neuwith,94] introduces the notion of deadlines for managing the timeliness of interactions. By defining an appropriate deadline parameter, users can be given feedback when the level of coupling achieved by the system falls below that level which has been specified as acceptable.

In a mobile context, it is important that flexible coupling is supported and that users can stipulate the acceptable minimum level of coupling. For example, when the level of group connectivity is good then a tightly coupled, highly synchronous level of interaction could be maintained by the system. However, if the level of group connectivity becomes poor then the system should have the capability of switching to a more loosely coupled, asynchronous, level of interaction requiring less data to be exchanged between the group.

Linked with the concept of coupling is the notion of response and notification times introduced in [Ellis,91]. The response time is defined as the time delay before a user's own interface is updated to reflect his/her actions and the notification time is defined as the time taken for the user's actions to be reflected across all group members. The principle of separating response and notification times is echoed by the work of Dix [Dix,95] who uses the terms feedback and feedthrough.

## AWARENESS

The concept of awareness could be described as the antithesis of transparency in that it is concerned with the supply of information to users as opposed to the masking of it. The main argument for supporting awareness in groupware is that people are adaptable and given the relevant information can solve most problems for themselves [Dix,95]. For example, consider the locking problems associated with enabling the concurrent editing of a shared document. By providing the authors with an awareness of the parts of the shared document that are currently being edited (i.e. workspace awareness) and also those parts for which permission to edit has been requested, authors can devise their own social protocols in order to avoid conflicting actions.

When considering the provision of awareness the following questions arise:-

• What information is relevant ?

• How best can this information be provided ?

• How can users be enabled to control the amount of awareness provided so that it does not interfere with the collaborative process ?

In a mobile context, applications which access remote data services require new forms of awareness or feedback for informing users of the constraints (such as communications delays) imposed by the mobile environment [Johnson,95][Johnson,97]. More specifically, mobile groupware applications need to give group members an awareness of the impact of mobile communications on the group's collaboration. For example, users could be made aware of the identity of those group members experiencing poor communications QoS. The provision of this type of information can be termed mobile awareness [Cheverst,98] and should help prevent group members from being forced to make (possibly false) assumptions regarding the current state of connectivity within the collaborating group.

## SUMMARY

To summarise, the implications for traditional groupware concepts and techniques when considered in a mobile context are as follows:-

• Problems associated with collaboration-transparent groupware When considered in the context of mobility, the development of collaboration- transparent groupware has a number of problems. For example, if using a windows-level splitter component, e.g. an X splitter, then network bandwidth requirements can be prohibitive. In addition, this approach provides little flexibility over the degree of coupling between displays and therefore an application might be unable to adjust its communication requirements, should network problems occur. A further problem with this approach is that the application (having been designed for single-user operation) is unlikely to provide users with sufficient feedback should inconsistency problems, caused by poor network quality, arise.

• Increased complexity of managing shared data

The increased potential for communication difficulties in a mobile environment places additional demands on the management of shared data. In particular, trade-offs between data consistency and availability need to be considered carefully and continually because, in a mobile environment, the quality of group communications is likely to fluctuate dramatically.

• The need for flexible coupling

When the quality of group communications is poor, it is difficult to maintain a close degree of coupling because of the high level of communications required. It is therefore important for mobile groupware to support flexible coupling, so that the degree of coupling used can match the level of group communications available.

• The need for increased (mobile) awareness

Depending on the application scenario, it may be useful to provide users with mobile awareness. Such awareness can reveal (where appropriate) the effect of the mobile communications environment on the group's collaboration, thus enabling users to make informed decisions in order to cope with possible communication difficulties.

## REFERENCES:-


- [Anker,97] Anker, T., V. Gregory, D. Dolev and I. Keidar. "The Caelum Toolkit for CSCW: The Sky is the Limit.", Proc. Third International Workshop on Next Generation Information Technologies and Systems (NGITS 97), June 30 - July 3, 1997, Neve Ilan, Israel.
- [APM,89] APM Ltd. "The ANSA Reference Manual Release 01.00.", Architecture Projects Management Ltd., Cambridge, U.K. 1989.
- [APM,92] APM Ltd. "An Introduction to ANSAware 4.0.", Architecture Projects Management Ltd., Cambridge, U.K. 1992.
- [Brinck,92] Brinck, T. and L.M. Gomez. "A Collaborative Medium for the Support of Conversational Props.", Proc. ACM CSCW'92 Conference on Computer Supported Cooperative Work, pages 171-178, Toronto, Canada, October 31-November 4 1992.
- [Burridge,98] Burridge, R. "Java Shared Data Toolkit User Guide.", User Guide, Version 1.4, Sun Microsystems Inc, June 1998.
- [Casio,99] Casio. "Casio Announces World's Smallest Multimedia Color Palm-Size PC.",Press Release, Casio Inc. http://www.casio.com/corporate/pressdetail.cfm?ID=60. January 1999.
- [Chang,84] Chang, J. and N. Maxemchuk. "Reliable Broadcast Protocols.", ACM Transactions on Computer Systems, Vol. 2, No. 3, pages 251-275, August 1984.

- [Dourish,96b] Dourish, P. "Consistency guarantees: Exploiting application semantics for consistency management in a collaboration toolkit.", Proc. ACM CSCW'96 Conference on Computer Supported Cooperative Work, pages 268-277, Boston, November 1996.
- [Edwards,96] Edwards, K., "Policies and Roles in Collaborative Applications.", Proc. ACM CSCW'96 Conference on Computer Supported Cooperative Work, Boston, November 16-20 1996.
- [Ege,87] Ege, A. and A. Ellis. "Design and Implementation of GORDION, an Object Base Management System.", Proc. 3rd International Conference on Data Engineering, pages 36-45, February 1987.
- [Greenberg,91] Greenberg, S. and R. Bohnet. "GroupSketch: A multi-user sketchpad for geographically-distributed small groups.", Proc. Graphics Interface '91, Calgary, Alberta, Canada. 1991.
- [Greenberg,94] Greenberg, S. and D. Marwood, "Real time groupware as a distributed system: Concurrency control and its effect on the interface.", Proc. ACM CSCW'94 Conference on Computer Supported Cooperative Work, pages 207-217, Chapel Hill, North Carolina, October 22-26 1994.
- [Greenberg,96] Greenberg, S. and M. Roseman. "Groupware Toolkits for Synchronous Work." Research Report 96/589/09, Department of Computer Science, University of Calgary, Calgary, Canada, November 1996.
- [HP,99a] Hewelet Packard, "HP Jornada handheld PCs.", http://www.hp.com/jornada/. 1999.
- [HP,99b] Hewelet Packard, "HP Omnibook 4100 Notebook PC - Data Sheet.", http://www.hp.com/omnibook/products/4100/datasheet.html. 1999.
- [IEEE,97] Institute of Electrical and Electronics Engineers. "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.", IEEE Standards document, 802.11-1997. ISBN 1-55937-935-9. June 1997.
- [IrDA,99] Infrared Data Association. "Technical Summary of IrDA DATA and IrDA CONTROL.", http://www.irda.org/standards/standards.asp. 1999.
- [ISO,92] International Standards Organisation. "Draft Recommendation X.901: Basic Reference Model of Open Distributed Processing - Part1: Overview and Guide to Use.", Draft Report, International Standards Organisation WG7 Commitee. November 1992.
- [Lauwers,90] Lauwers, J.C. and K.A. Lantz. "Collaboration awareness in support of collaboration transparency.", Proc. ACM SIGCHI'90 Conference on Human Factors in Computing Systems, pages 303-211, Seattle, Washington, April 1-5 1990.
- [Leopold,91] Leopold, R.J. "Low-earth orbit global cellular communications network.", Proc. IEEE International Conference on Communications - ICC '91, pages. 1108-1111. 1991.
- [Microsoft,98] Microsoft. "Distributed Component Object Model Protocol.", Internet Draft Specification, http://www.microsoft.com/oledev/olecom/draft-brown-dcom-v1-spec-02.txt. January 1998.
- [Microsoft,99] Microsoft, "Microsoft Windows CE", http://www.microsoft.com/windowsce/. 1999.