Real-Time Collaboration in Heterogeneous Computing Environments

Ivan Marsic
Department of Electrical and Computer Engineering
Rutgers — The State University of New Jersey
Piscataway, NJ 08854-8058

Abstract

The recent proliferation of computing devices and the contexts in which they are used demand diversity in collaborative applications as well. The objective of our research is to enable conferees to share applications that are adapted to their (heterogeneous) computing environments. This is achieved by using eXtensible Markup Language (XML) for the communication medium. The conferees share the same data or a subset of that data, represented in XML, but they may see it displayed in different ways as needed or desired. The framework adapts to the computing and network environment and transforms information so that it matches the client's local capabilities and resources.

Keywords: CSCW, groupware, group communication, heterogeneous computing, intelligent agents.

1. Introduction

Traditional groupware systems for synchronous or real-time collaboration require identical applications running on nearly identical hardware platforms. However, the recent proliferation of computing devices and contexts in which they are used demands diversity in collaborative applications as well. Kraut *et al*. [11], for example, show that field workers make repairs more quickly and accurately when they have a remote expert helping them. It is likely that the expert will be in a central office with a workstation, whereas the field worker will have a wearable computer. Thus, heterogeneity naturally occurs in practical tasks.

The heterogeneity of computing platforms manifests itself in CPU speed, memory, display capabilities, and network bandwidth, with the last two accounting for the most prominent differences. Of these, displays will likely take the most diverse forms as already visible in the latest developments. Examples are displays embedded in eyeglasses, heads-up displays, windshield-projected displays in cars, or micro displays. These displays are invariably more limited in size and quality than those on the desktop. Enlarging the screen real estate for mobile computers will remain impractical due more to inconvenience than to economic reasons. Moreover, we are dealing here with more than technology limitations. We are dealing with limitations in human information processing. A stationary person focusing on a monitor in an office allocates vastly different cognitive resources to the computer display than a roaming person who has to pay attention to other things in the environment [10].

The DISCIPLE framework provides for collaboration using Java components (Beans and Applets). DISCIPLE is an *application framework*, i.e., a semi-complete application that can be customized to produce custom applications. The completion and customization is performed by end users (conference participants) who at runtime select and import task-specific Java components. Collaborators import Beans by drag-and-drop manipulation into the workspace. The imported Bean becomes a part of a multi-user application and all conferees can interact with it. The application framework approach has advantages over the commonly used toolkit approaches in that with toolkit approaches the application designer makes decisions about the application functionality whereas in our approach the end user makes decisions. The main goal of the DISCIPLE project is supporting shared applications in heterogeneous computing environments.

2. The Framework Architecture

The architecture of the DISCIPLE framework is shown in Figure 1. As a first approximation, all wired clients are assumed to belong to one class and all wireless clients belong to another class (Figure 1b). The *Collaboration Bus* spans network fabrics and provides a virtual interconnect for geographically distributed clients [19]. It implements environment-aware protocols for object interactions and encapsulates mechanisms for coupling, coordination and concurrency management.

The *Intelligent Agents Plane* is responsible for ensuring that its client is an effective participant in the heterogeneous collaboration session. It uses a knowledge base to track and translate the client's interests,

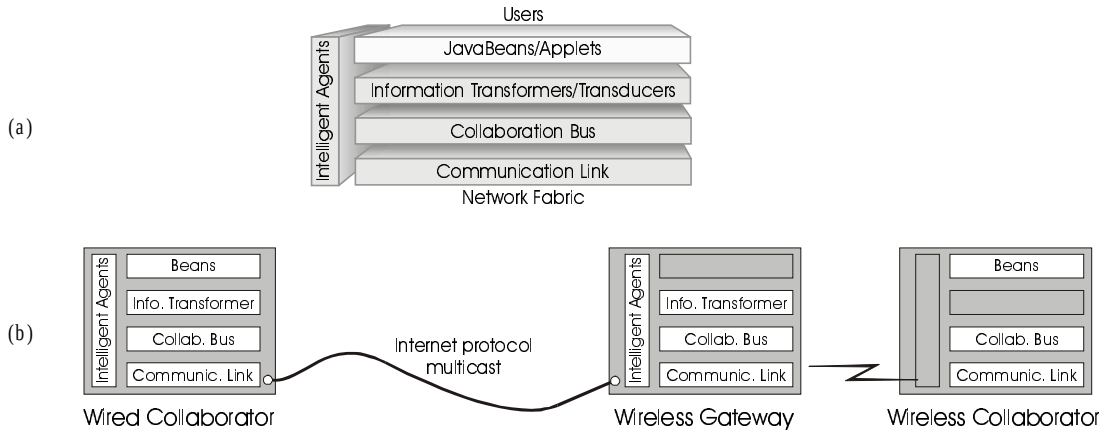

Figure 1: Architecture of the DISCIPLE collaboration framework. (a) Components of the framework. JavaBeans/Applets are not part of the framework since they are supplied by the application developer rather than by the framework developer. (b) Distribution of components in a heterogeneous environment.

computing capabilities and quality-of-service (QoS) requirements into a *client profile* that defines the type and level of information abstraction needed. The profile is then used to visualize the incoming data or to modify outgoing events using the services of the information transformer module.

The *Information Transformer/Transducer Module* maintains a suite of media-specific information abstraction algorithms. Information abstraction aims at intelligently reducing information content while maintaining semantics. Information-transforming processes assist the bandwidth- and display-disadvantaged mobile wireless clients. Examples for information abstraction include image-to-text, image-to-speech, text-to-speech, and speech-to-text conversions. Such a translation is critical for enabling collaboration across heterogeneous clients and interconnects with large discrepancy in computing capabilities.

DISCIPLE provides an infrastructure for application state sharing, and applets provide task-specific functionality. An *applet* is a small application program that provides a graphics user interface and accomplishes the task that the user is interested in, e.g., computing image features or computing and visualizing a spreadsheet table. A user would typically use multiple applets to collaborate with other users. DISCIPLE supports both third-party collaboration-unaware beans and our collaboration-aware beans [12].

3. Data-Centric Collaboration Using XML

We assume that interactive groupware applications deal with structured documents. Our approach to deal with heterogeneity focuses on documents or data and transforms and visualizes the data to fit the computing capabilities of different users, while preserving semantics of the data [13]. This division of the underlying data and the way it gets displayed mimics Model-View-Controller (MVC), a well known and frequently used design pattern to develop interactive applications with flexible human-computer interfaces [1]. MVC divides an interactive application into three components. The *model* contains the core functionality and data, *views* display information to the user, and *controllers* handle user input.

We chose XML as a medium to represent the data the conferees are collaborating on as well as the instructions for visualizing the data. XML is a markup language for documents containing structured information [8]. Once the data is written in XML, an associated XSL (eXtensible Style-sheet Language) document has to be written to define the way it gets displayed. The XML document is platform-independent and thus corresponds to the *model*, whereas the style sheet depends on the displaying device and corresponds to the *view*. By applying the rules defined for the various objects in the XSL file on the XML data the result tree is formed which can be unique for every user. The views may differ due to different user needs, different expertise/concerns, or due to different display capabilities.

The data-centric groupware paradigm separates model events (document modifications) and view events (view modifications), Figure 2. It offers the following advantages over conventional, application-centric groupware:

- Model-view separation at the document level via XML/XSL documents
- Simple communication protocol with a standard message format based on ASCII XML messages
- A single data structure (tree) and an associated limited set of operations

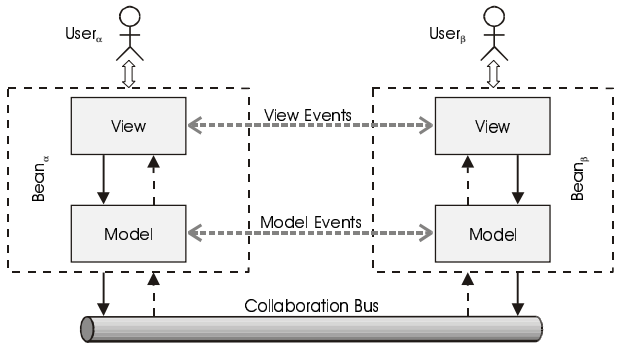

Figure 3: Change replication in the DISCIPLE system.

The last item is particularly important for advanced concurrency control techniques, such as transformational concurrency control algorithms [4].

3.1. Synchronization of Collaborative Clients

Synchronization of the clients has specific issues even when all replicated models are the same. For various reasons the models at collaborating peers may be different. One reason may be bandwidth conservation—we may not want or need to transfer entire XML documents to a collaborator operating from a low-bandwidth wireless link. Or, a collaborator may have a small-size display and does not need the high-resolution map used by other peers. Yet another reason may be security—certain data should not be accessible to all collaborators. Heterogeneous models that we account for here are similar in the sense that they preserve *semantics* but vary the *quantity* and *accuracy* of data. Since XML is a well-defined language, the heterogeneity can be quantified. The models could differ in the following ways:

D1. Different number attributes in the corresponding XML elements
D2. Different number of XML elements
D3. Different attribute values
D4. Different type of attributes in an element
D5. Different tag-type of elements in models

Combinations of these differences introduce additional complexity. Two major issues with collaborative systems with heterogeneous models are (*i*) consistency across the participants and (*ii*) system stability in the presence of automatic corrections of invalid states. The system is consistent and stable if the distance between the models does not grow unlimited as the operations get applied to the models. This is subject of our current research [13].

4. Intelligent Agents

To deploy the data-centric scheme, the system must build the client profiles. This is the task for intelligent agents, Figure 1a. Agents span all layers of the architecture since they need information from all layers to make intelligent decisions. This is similar to Minsky's K-lines [15].

Agents sense the networking environment and automatically transform the information exchanged between the conferees to match diverse user profiles (which include computing and network capabilities, and user's interests, expertise, and possibly physical disabilities). Agents also dynamically generate the view based on the capabilities of the client device and other components of the profile.

An example task for intelligent agents is determining the characteristics of the communication links. This information may include available throughput or confidence of the existence of wireless links. It is then used by the application to adjust to the computing environment. The application in our case is the DISCIPLE framework and it adapts in such a way to automatically apply different information transformations and XSL documents for data visualization.

4.1. Available Bandwidth Measurement

We have developed a throughput measurement tool based on the bottleneck link bandwidth method described in [2]. It is an active measurement method which employs the Packet Bunch Mode (PBM) of evaluation to estimate the link bandwidth. Figure 3 compares the performance of our tool to that of a popular tool called Pathchar [6]. The latter uses ICMP error packets in estimating the link

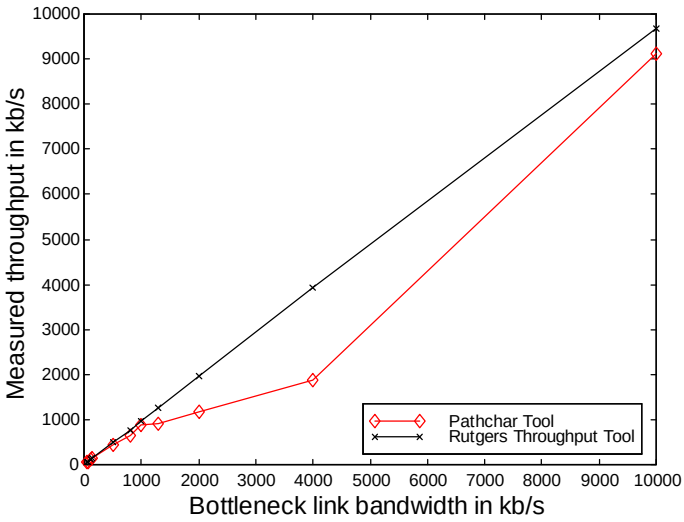

Figure 2: Comparison of tools for throughput estimation. The HSSI clock is only able to accurately set the link speeds up to 4000 Kbps, consequently there is a gap from 4000 Kbps to 10000 Kbps, which is the link speed for a 10Mbps Ethernet LAN.

characteristics and also relies on injecting UDP packets to measure the link characteristics. For experiment, the bottleneck link bandwidth could be altered in a laboratory testbed network by introducing a High-Speed Serial Interface (HSSI) between the two gateways. The endpoints for the test were located across the two ends of the HSSI. Figure 3 shows an almost linear curve for our tool whereas Pathchar shows significant deterioration starting at about 1 Kbps.

4.2. Using RTT to Detect Wireless Links

We have used round-trip time (RTT) values to identify the characteristics of the communication links [3]. Generally speaking a RTT is the interval between the sending of a datagram and receiving its acknowledgment. It includes both the network propagation delays, such as route-queue delay and link delay, as well as host-processing delay, such as the time spent at the sender and receiver processing the datagram and acknowledgment. In the normal case, the propagation delay is the most significant contributor to the round-trip time.

The experiments and analysis show that the statistical patterns of the RTTs are very different over wired and wireless links. The mean value of RTTs in the wired case is smaller than that in the wireless case. Moreover, the shape of the distribution of RTTs in the wired case is more pulse-like so that its RTT variance is smaller than that in the wireless case. Thus, if RTTs collected by the intelligent agent show an abnormal pattern such as large mean value and variance, then using fuzzy logic the agent can deduce the existence of wireless link(s). We have proposed to use *if conditionals* or *statements* to differentiate the wired and wireless cases. The fuzzy reasoning engine accepts the mean value and variance of RTTs as the fuzzy inputs, and outputs the confidence of the existence of wireless links.

5. Test Applications and Experiments

Two issues are pertinent for the presented framework:

- How well it generalizes to arbitrary applications?
- Can users collaborate meaningfully?

XML is a general markup language that can represent any structured document. Most office applications deal with such, and they are also the most likely candidates for collaborative work. XML documents get parsed into tree data structures, which may not be the most efficient data type for specific applications (e.g., spreadsheet). The coZmo Bean developer thus may decide to internally convert the tree to another data type (list, array, graph, etc.). Backward conversion to the result tree is then necessary for every event for remote distribution.

The set of applications implemented and tested to date includes text-based chat, whiteboard, 3D graphics editor, collaborative mapping application, speech signal acquisition and processing, image analysis tools, a military mission planning extension for the whiteboard, and a medical image-guided diagnosis system for the diagnosis of leukemia [4].

The second issue is more difficult to evaluate. Even though a common view is a basis for group work, the experience accumulated thus far shows that non-WYSIWIS systems do not impair collaboration and may even facilitate it [18]. But the heterogeneity presented here goes further than any prior work. Let us first consider model events (Figure 2), as in our current implementation. Each collaborator "lives" in their own world and he/she does not need to know that the other collaborators see anything different. Collaboration is not adversely affected as long as the views behave logically, i.e., the application does not enter invalid states. Let us assume that the XML element corresponds to a truck on a terrain map. If the terrain is not flat, the change in the (x,y) coordinates only may result in the truck being buried under the ground or floating above the ground. If the truck's z-coordinate is not automatically adjusted, the 3D user may wonder whether the truck is floating above the ground on purpose or by accidence. On the other hand, the constraints should be applied carefully, since the automatic correction would prevent the user from ever positioning the object above the ground even if desired so.

Although this should work in principle, it should be tested in practice by building complex applications and testing them on real-world problems. We defined a DTD language for representing 3D figures and scenes and developed two coZmo Beans: one that can render 2D graphics and the other that renders 3D graphics. We also created an XML document with a map and objects on it, as well as two XSL documents for the corresponding viewers. The hypothesis being tested is that conferees can collaborate on the (nearly) same document displayed with different user interfaces that best suit their needs. In our tests, a user with a high-end workstation has a 3D representation of a terrain map, whereas the user with a Xybernaut wearable computer uses a 2D-map representation with reduced resolution. coZmo should provide for meaningful collaboration. Figure 4 shows the two coZmo beans loaded in the DISCIPLE workspace along with the chat bean.

Our preliminary experiments indicate no major difficulties in collaboration. Conferees are able to refer to the "same" object and manipulate it, although they see it quite differently in their displays. A more rigorous study is planed when the software becomes stable.

The next step is to account for view events, in case the users require view coupling or telepointers. Views in general may be quite different and it may be difficult or impossible to establish correspondence. For example, one user may see object coordinates represented as points on a

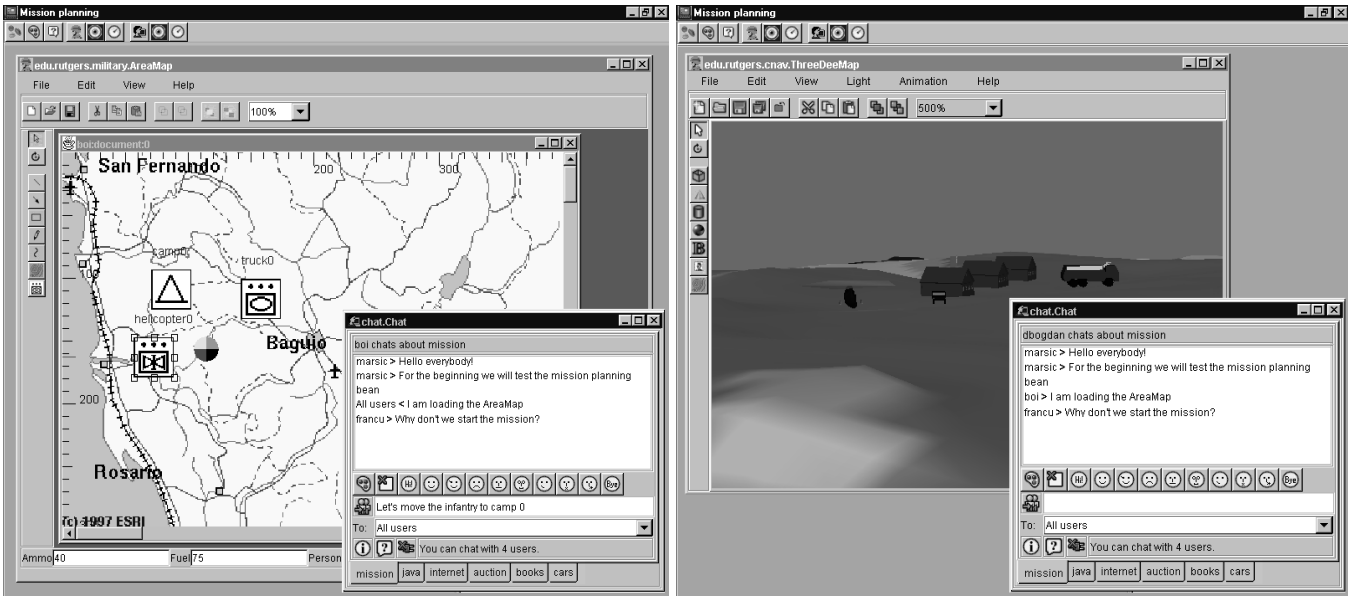

Figure 4: Two collaborating users share the same mission-planning workspace but with different versions of the area map application. The left user uses two-dimensional and right user uses three-dimensional rendering of the mission-planning document. They both use the same chat JavaBean.

map, whereas another user may see it as a bar chart. It is then difficult to determine how to couple zooming or display telepointers. One option for telepointers may be "semantic cursors" as hinted at in [16]. At present, our simplifying assumption is that heterogeneous resources lead mainly to quantitative differences in visualizations, such as with rendering 2D and 3D graphics. In the 2D/3D case, the correspondences can be easily established between 2D and the 3D's orthographic projection.

6. Related Work

Efficient reference to common objects depends on a common view of the work at hand and this is best expressed in the WYSIWIS (What You See Is What I See) idealization. However, strict WYSIWIS was found to be too limiting and relaxed versions were proposed to accommodate personalized screen layouts [18]. Many collaborative systems and toolkits address the issues of sharing across dissimilar terminals. For example, Garfinkel *et al*. [9] discus at length the adaptation of the shared display to various display devices. Even though this architecture features distributed (possibly different) views, it has a centralized model which makes it equivalent to the non-distributed case and thus much simpler to deal with.

Several researchers proposed MVC separation in synchronous groupware, with either centralized or distributed models and polymorphic views rendered on distributed hosts. GroupKit [16] and several groupware toolkits thereafter use model-view separation so that developers can create models and drive different views. However, no actual implementation is attempted, and in some cases the situation is greatly simplified by using centralized groupware architecture.

Our focus is on collaboration with heterogeneous computing platforms, which may result in sharing different data across the conferees. The Visage system from Maya Design [14] focuses on complementary issues. Visage is a powerful (single-user) visualization system for creating custom visualizations and direct manipulation of large and diverse datasets. Some of its unique features include dynamic data navigation through drill-down and roll-up (aggregation) techniques. While Visage addresses diverse data visualization, it is not explicitly intended for collaboration in heterogeneous computing environments. Recent work on Visage Link adds multiuser collaboration capabilities, but does not consider heterogeneous models or data.

Although some components of the framework presented here are seen to exist in other systems, to our best knowledge, there is currently no other research team that focuses on collaboration with heterogeneous platforms and addresses this broad range of issues.

7. Conclusions

We present a data-centric framework for synchronous collaboration of users with heterogeneous computing platforms. Our approach allows clients with different computing capabilities to share different subsets of data, e.g., due to compression in order to conserve bandwidth. XML, which serves as the communication medium, is a standard that has already gained wide acceptance, and

provides a powerful medium for both data exchange and visualization specification. Other unique features of the DISCIPLE framework include simultaneous support for collaboration-aware and collaboration-transparent applications and an interface that is open and customizable with respect to the context in which it is placed and particular user needs.

Ideally, we should be able to dynamically generate the view based on the capabilities of the client device. Our current efforts are in the direction where the device publishes its characteristics and automatically receives the XSL file adapted appropriately. The solution presented can be extended to a broader domain of data sharing in heterogeneous environments. Dearle [5] discusses the problem of how to maintain a persistent user's desktop state for mobile users. The problem is similar to asynchronous collaboration of multiple users, except that here the user "collaborates" with himself/herself at different times. The problem can be generalized, so that the user can resume work at a different platform, with a different display device. For example, a user may carry his/her XML documents (models) on a wearable computer. As the user connects the computer to different display devices, heterogeneous views may be created according to the device characteristics.

Further information about the DISCIPLE project and heterogeneous collaboration is available at:

http://www.caip.rutgers.edu/disciple/

Acknowledgments

This research is supported by DARPA Contract No. N66001-96-C-8510, NSF KDI Contract No. IIS-98-72995 and by the Rutgers Center for Advanced Information Processing (CAIP).

References

[1] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns.* John Wiley & Sons, Inc., New York, 1996.

[2] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. Technical report TR-96-006, Department of Computer Science, Boston University, March 15 1996.

[3] L. Cheng and I. Marsic. Fuzzy reasoning for wireless awareness. Submitted for publication.

[4] D. Comaniciu, P. Meer, D. Foran, and A. Medl. Bimodal system for interactive indexing and retrieval of pathology images. In *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV'98)*, Princeton, New Jersey, pp.76-81, 1998.

[5] A. Dearle. Toward ubiquitous environments for mobile users. *IEEE Internet Computing*, 2(1):22-32, January/February 1998.

[6] A. Downey. Using pathchar to estimate Internet link characteristics. In *Proceedings of ACM SIGCOMM '99 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Cambridge, MA, pp.241-250, August/September 1999.

[7] C.A. Ellis and S.J. Gibbs. Concurrency control in groupware systems. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, Seattle, WA, pp.399-407, 1989.

[8] Extensible Markup Language: http://www.w3.org/XML. See also http://www.xml.com/.

[9] D. Garfinkel, B. C. Welti, and T. W. Yip. HP SharedX: A tool for real-time collaboration. *Hewlett-Packard Journal*, 45(2):23-36, 1994. http://www.hp.com/hpj/94apr/toc-04-9.htm

[10] D. Kahneman. *Attention and Effort*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973.

[11] R. Kraut, M. D. Miller, and J. Siegel. Collaboration in performance of physical tasks: Effects on outcomes and communication. In *Proceedings of the ACM 1996 Conference on Computer-Supported Cooperative Work (CSCW'96)*, pp.57-66, 1996.

[12] W. Li, W. Wang, and I. Marsic. Collaboration transparency in the DISCIPLE framework. In *Proceedings of the ACM International Conference on Supporting Group Work (GROUP'99)*, Phoenix, AZ, pp.326-335, November 1999.

[13] I. Marsic. State synchronization in heterogeneous groupware. Submitted for publication.

[14] Maya Design Group, Inc. Visage Link. http://www.maya.com/visage/base/technical.html

[15] M. Minsky. *The Society of Mind*. Simon & Schuster, Inc., New York, 1985.

[16] M. Roseman and S. Greenberg. Building real-time groupware with GroupKit, a groupware toolkit. *ACM Transactions on Computer-Human Interaction*, 3(1):66-106, March 1996.

[17] Sun Microsystems, Inc. JavaBeans specification, 1999. http://java.sun.com/beans/docs/spec.html

[18] M. Stefik, G. Foster, D.G. Bobrow, K. Kahn, S. Lanning, and L. Suchman. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Communications of the ACM*, 30(1):32-47, January 1987.

[19] W. Wang, B. Dorohonceanu, and I. Marsic. Design of the DISCIPLE synchronous collaboration framework. In *Proceedings of the 3rd IASTED International Conference on Internet, Multimedia Systems and Applications (IMSA'99)*, Nassau, The Bahamas, pp.316-324, October 1999.