Model-Based Virtual Environments for Collaboration

Gary J. Nutt

CU-CS-799-95

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Model-Based Virtual Environments for Collaboration

Gary J. Nutt

CU-CS-799-95 December 1995

University of Colorado at Boulder

Technical Report CU-CS-799-95
Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309

Model-Based Virtual Environments for Collaboration

Gary J. Nutt

December 1995

Abstract

The simultaneous evolution of personal computing tools and networks has focused attention on the notion of harnessing computer technology to assist in human collaboration on group work. While personal productivity tool technology and use have reached a high level of sophistication, the most basic ideas for how computer technology should assist in collaboration across the network have not converged. The approaches range from ones where coordination of work is uniquely human-controlled, to those where the computer schedules all significant work in the group. On the one hand, proponents argue the system should provide its users with the most effective form of coordination, while others argue the system absolutely must not appear to be dictatorial. Secondly, group work is usually based on trust and understanding among the team members – the more familiar the team members are with one another, the more effectively they collaborate. If a new system removes or changes this underlying social model, the system will probably only be effective if the users are willing to adapt their fundamental social behavior to fit the system's implied model. In this paper, our premise is that it is risky to build computer systems depending on such social change for their success. Instead, we consider collaboration systems based on formal coordination models, yet which must be adaptable to existing social models in the organization. Our approach is to assume a class of distributed systems to support group work where part of the work is represented formally in the system environment, and the remainder is stored as informal knowledge. The system environment must go to great lengths to support extant social models by incorporating significant technology to accommodate informal communication within a context established by the formal model of the work.

1 Introduction

The impact of computers on the way people manage information has been profound. The IBM PC was introduced as a personal information processing tool in about 1980, and in the subsequent 15 years personal computers have become deeply entrenched in the way information workers conduct their work. They use word processors and desktop publishing systems to prepare documents of a quality not attainable at any reasonable cost before 1980. Spreadsheets allow workers to quickly perform elaborate predictions of behavior using simple numerical models. Networks and terminal emulators enable users to query and retrieve global information from the corporation, and now with the growing use of the Internet, from the entire economy. Personal computing has grown so powerful it has fundamentally changed the way organizations conduct their business.

However, this growth has focused on *personal computing* as opposed to *distributed computing* among a group of individuals. Interactions among individuals using computers is generally limited to sharing a common information base (e.g, a database, bulletin board, news group, or web page), and the exchange of information using the telephone, electronic mail, and FAX.

In general, *groupware* is software that attempts to use computers and networks to provide services to assist a group of workers in conducting their work – hence the almost synonymous term *computer-supported cooperative work* (CSCW) [10, 22, 23]. While there is little argument concerning the need for computers to provide some kind of support for collaborative work, there is considerable difference of opinion about *how* this might be best accomplished.

Electronic meeting room advocates use computers and networks to enable humans located at physically different places to "meet" with one another at the same time. The electronic meeting room provide digital audio and video streams to each participant, and may also provide shared whiteboards, windows onto shared information, etc.

Virtual environment researchers extend the electronic meeting room idea by creating a human-computer interface to simulate a virtual meeting room/space/environment in which a participant can logically immerse oneself. Interactions among the electronic surrogates are then conducted within the virtual space. In particular, objects in the virtual environment are then directly shared by the participants.

A radically different approach is to focus on providing domain-specific assistance. Some evolutionary systems rely on the computer to provide increasingly specific personal productivity tools, with little explicit attempt to make the system stage the work to be performed. The computer environment is a logical environment containing various tools in which work arrives and human participants decide which tasks should be accomplished to perform the work – we call this the *situated work* approach.

At the opposite end of this spectrum are *workflow systems* where the organization's work has been analyzed and cast as a model composed of discrete steps. The workflow model also specifies how any "job" (or work) should flow among the set of steps. Humans become part of the underlying engine to execute various steps whenever work flows into the step.

Since none of these approaches, in and of themselves, appear to have established the basis for general CSCW, we conjecture a new class of systems combining various aspects of each of the approaches mentioned above. *Model-based virtual environments* (MBVEs) are distributed systems establishing a shared virtual environment with an embedded formal model to capture some aspects of the work of the group, but with the ability to annotate the formal model with informal

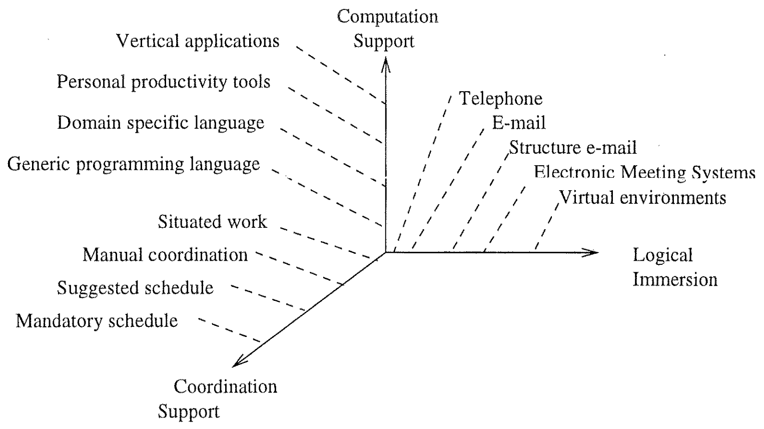

Figure 1: CSCW Characterizations

information.[1] The approach uses the notion of system support for unstructured communications, including informal communication, through electronic meeting room and virtual environment technology. The formal model provides a framework for workflow-like processing in cases where that is appropriate, but also provides a robust informal communication mechanism to stage and troubleshoot work when it is treated as situated work.

It is helpful to consider different dimensions for collaboration approaches, since it is apparent that different researchers have used different criteria and assumptions in attaching the CSCW problem. We propose the informal model in Figure 1 as a summary of pertinent characteristics of MBVEs.[2]

1.1 Logical Immersion into the Environment

Our first dimension is intended to describe the amount of logical immersion the human has in a collaboration session. At the low end of the dimension is support of the collaboration using the telephone; the environment among the collaborators is loose and informal, with communication occurring the telephone.

Electronic mail system logically immerse the collaborators deeper into the system. Whereas telephones are usually a distinct device from the computer and network, electronic mail is a part of the computer facilities, e.g., always available to the worker in an alternate window.

Some systems add a layer to electronic mail so it can be used to capture some of the context of a domain. For example, the mail system delivers work to a person, and routes work onto others once one person has finished with it (see Chapter 8 in [2] as well as various commercial products from Lotus and Microsoft).

Electronic meeting rooms represent a deeper commitment by the computer system to the group. There have been a number of studies and implementations of electronic meeting rooms; one set,

[1]In the context of this paper, the formal model can be thought of as a workflow-like model, though most of the thrust of the research does not depend on the interpretation of the model. For example, elsewhere we have proposed to study a specific model-based virtual environment in which the formal model is an architectural description language for complex software [32].

[2]This model is only intuitive, rather than a precise taxonometric model. For example, the axis are not necessarily orthogonal. Our use of the model is to focus on three different driving forces in CSCW systems.

including the Bellcore Cruiser [18], Xerox Etherphone [39], TeamWorkstation and ClearBoard-1 [26], and DEC Argo [20] can be characterized as "desktop videoconferencing systems." These systems are a natural extension of conventional desktop window systems to embrace windows for images, audio streams, video streams, etc. The fundamental extension is the incorporation of a more diverse set of media into the desktop environment than existed in first generation window systems.

Another class of electronic meeting rooms employ a collection of geographically distributed meeting rooms augmented with computer-based tools, e.g., Nunamaker et al.'s EMS [30]. Colab [37], the Xerox media space [6], and Xerox Liveboard [15]. Participants physically gather in the individual physical meeting rooms, and are logically gathered across the physical meeting rooms by the computer and network facilities. The emphasis in these projects is to provide effective means for extending normal "meeting tools" across the physically distributed rooms.

Virtual environment systems such as Dive [16], Rapport, VMM, and Archways [4], and RING [19] create another level of immersion into the work by virtualizing a shared room and its artifacts, *including its participants.* A participant has a virtualization that is placed in the environment where it interacts directly with shared objects, including other participants. Boman provides a recent survey article on virtual environments describing several systems with characteristics similar to the VPR described in this proposal [7]; the July, 1995 issue of IEEE Computer focusing on virtual environments [24]; the May, 1995 issue of IEEE Computer focusing on multimedia [25]; the AT&T Technical Journal on multimedia [1]; and the ACM Multimedia proceedings [17].

1.2 Computation Support

The level of domain-specific computation support is an important criterion in characterizing a CSCW system. Our model orders the amount of computation in terms of the degree to which the system "automates" computation; this ordering is admittedly intuitive.

General purpose computer hardware is intended to be used in all information processing applications — from accounting to missile tracking. The software specializes the system's behavior to focus on a particular domain or problem. Thus the first level of computation support is the presence of generic, high level programming languages for the system. This allows professional programmers to define the specific behavior of the hardware so it provides a useful function to the end user.

Domain-specific programming languages are intended to provide higher level of computational support by enabling experts in a domain to apply their knowledge to programming without becoming programming specialists. Thus, in our characterization, these languages provide a higher level of domain-specific computation support than do generic programming languages.

Personal productivity tools are typically horizontal applications that can be applied to many different domains. These tools include spreadsheets, web page editors, desktop publishing software, etc. We place these tools are higher on the computation support axis since they seem to enable larger groups of users to use the computer for their work than is possible with domain-specific ("end user") programming languages.

The next level of specialization is vertical application software, or software written to solve a specific task. This includes a program to generate the monthly accounts receivable and payable, to compute the orbit of a satellite, or to simulate war games. This level of support is the highest, since it allows the user to perform some set of commands to exercise the program's behavior.

1.3 Work Coordination Support

A third dimension in CSCW systems is the degree with which they attempt to incorporate domain semantics or work specification into the environment. CSCW systems embracing the situated work philosophy attempt to design the system so it creates a electronic environment that is perceived to be similar to (or at least only incrementally different from) an extant manual environment. Leverage is gained from computers by computation support and incremental "micro level automation" by which various tasks in the environment are accomplished using software tools [38]. This fits well with the trend in personal productivity tool development.

CSCW systems must support the interaction of different people using distinct computing devices, e.g., a network of workstations and server machines. This hardware autonomy implies the need for the system to provide mechanisms to manage information sharing, resource sharing, and general concurrency control.

Parallel programming language specialists have observed that parallel and distributed programs can be decomposed into two logical parts: a computation model and a coordination model [21]. They argue that the computation model is represented by conventional "sequential programming languages" executed by a single-threaded computation. Further, if the computation is to support more than one thread of computation, it must include (implicitly or explicitly) a means of specifying the interaction among the various threads — this is the coordination model. Within the area of parallel and distributed languages, there is no common agreement for whether or not the coordination model should be implicit or explicit, and if it is explicit how it should be defined and represented.

In general, the z-axis of Figure 1 is intended to represent the nature of automation of the coordination model in a CSCW system. Situated work tends to minimize the importance of the model, so we place them near the origin. Manual coordination refers to systems providing tools to enable the group to coordinate their activities, including shared memory, shared resources, and synchronization mechanism such as locks.

In contrast to the situated work proponents, Malone and Crowston argue computers should explicitly be used to coordinate tasks [28]. Workflow models explicitly provide a coordination model, e.g., as a precedence graph among activity steps in a procedure. Malone and his colleagues use workflow models – also called *process models* – to categorize and publish procedures according to their coordination strategies [29]. In our characterization, we differentiate between workflow system that provide *suggestions* of work to be performed by human users versus systems that automatically schedule work for the human worker (such as FlowPATH [8]).

1.4 Our Approach

We believe it is necessary to take a broader perspective on CSCW than is typical of past studies. In particular, there have been strong arguments for supporting various points on each of the three axes of the characterization model, but we are unaware of any argument for attempting to support more than a limited part of the space. Our approach is to *experiment* with models and systems that address many points in the characterization space. In some cases we have built experimental systems to address one aspect of the problem, but in this paper we advocate MBVEs to address all three dimensions of the problem.

In the next section of the paper we provide additional discussion about how the MBVE addresses each dimension of the space. Next we consider how an MBVE prototype can be used to support collaboration during design activity. Then we consider how the MBVE prototype can be used to support collaborative work based on a process model. We conclude with brief remarks regarding the prototype system we are building, based on more detailed comments in the companion paper [33].

2 Supporting Group Work

MBVEs are intended to provide virtual meeting environment in which there is a requirement for:

1. A high degree of logical immersion in the application domain, thus we advocate the use of virtual environments.
2. Significant computation support, e.g. at the level of model-specific application software support for analysis, design, and perhaps enactment.
3. A moderate amount of coordination support, controlled by the system's end users.
4. A model to establish the foundation for communication and informal coordination of work among the collaborators.

The virtual environment provides the meeting room facilities to support collaborative management of a common formal model *and* the requisite informal communication forum used by the team members to establish the underlying social model of the organization. The model provides a precise language for addressing the framework and specifics of the work, and for focusing the team's discussions during meetings, negotiation, and evolution of the "product" defined by the model. Just as telephones and electronic mail are installed in an organization to conduct it's business, the virtual environment is installed to conduct business related to the formal model, yet it explicitly provides the same opportunity for informal communication existing in the office, business telephone calls, and electronic mail communications.

Supporting Design. We believe there are two distinct stages to accomplishing the collaborative work: first, the group needs to reach consensus regarding the *design* of the formal model to be used to describe the work. Our approach is to provide an explicit, domain-sensitive mechanism to formalize some of the intuitive ideas of "how the group intends to work together." For example, in office work, the formal model might be a process model identifying steps in the process and the suggested order in which they might be executed for any individual parcel of work — a workflow system. Or, the formal model might describe the software development process, including how the subject software will be decomposed, designed, implemented, tested, and released. There are many other examples of model-based group work.

Our experience with workflow models and systems suggests this type of model design can be a crucial part of a group's building consensus about how they will conduct their work. Traditionally, the organization provides some guidelines and restrictions about how the work will be done, but the team members develop their own detailed model based on the exact nature of the work parcels, personalities, experiences, social organization, etc. When a new procedure is installed or a new member joins the group, there is a phase of learning and evolution revolving around the development and refinement of the model.

By making the model explicit, and by combining it with the virtual environment, there is a unique opportunity to add substantial value to meetings through the capture of the model design rationale derived in informal meetings. Classically, people meet to air their views, to listen to other people's views, to learn about other team member's behavior, and to reach consensus regarding the team's position. Minutes for the meeting are rarely kept, and in the cases they are formally kept they tend to be inaccurate regarding exactly the nature and outcome of the meeting. Since MBVEs can capture the audio and video stream for the meeting so it can be distributed across sites, there is an obvious opportunity to save critical meeting *snippets* where critical arguments and consensus occur in the meeting.

How should the snippets be organized? The formal model is the framework for the agreed-upon work of the group; meeting snippets can be viewed as informal annotations of the formal model. The MBVE creates a unique situation in which it is possible to capture the team's rationale and to save it according to the part of the project framework that has been captured by the model.

Note that in a design organization, such as a marketing department, strategic planning department, engineering department, architect firm, or construction planning group, the design model *is the product* of the work (rather than being the documentation of the group process). We see this specific class of applications of MBVEs as forming sufficient rationale for this development, though in the next paragraph we argue how it is also important to support computation and coordination of the work.

Supporting Enactment. Suppose the organization creates a product or provides a service; the model is essentially a process model for the group. Once the model has been initially designed, it can be used as the basis of computation and coordination support for the group's work. Once the group begins to operate using the model, exceptions and change to the process are inevitable e.g., see [14, 35]. Change in the model may be caused by familiarity with the actual work, by the changing character of the work, by differences in group composition, by changing global regulations, etc. By embedding the model in the work environment, the MBVE provides easy access to the model during enactment; this means the collaborators can *easily* customize, tailor, change, or annotate the model as changes are observed. That is, the goal is to make the model be an effective support tool rather than an auditing tool that introduces more overhead to the work.

Exceptions pose a different problem. Saastamoinen describes three class of exceptions relating to cases that are known but not handled, cases that were not conceived of during the model design, and cases that cannot even be addressed by the general approach at all. When an instance of any of these classes of exceptions occurs, the model and system will fail; this, in turn, means the humans must *troubleshoot* the workcase/model. If the workcase is indeed one-of-a-kind, it must be handled essentially outside the formal model coordination facilities. If the exception is one that recurs more often than was believed would happen at design time, then the model can be revised to accommodate the "exception." The MBVE provides explicit support for exception handling by providing an electronic meeting room in which humans can informally troubleshoot the workcase in the context of the "intended procedure."

Adding Value. The MBVE can be thought of as a modeling system explicitly supplemented with mechanisms to support informal communication within the context of the formal modeling environment, and to provide facilities for saving, organizing, and retrieving conclusions from such informal communication. In particular, we advocate the augmentation of the formal modeling environment with mechanisms to assist a group of humans to collaboratively and informally study, design, and evolve the formal model. The MBVE is a domain-specific, multiperson, virtual meeting room to support collaborative browsing, navigating, querying, and updating of the information within a specific design environment. A basic modeling system establishes a well-defined design domain, but the overall system must provide additional facilities to enable a group of designers to immerse themselves in the model *and* the informal information to discover problems in the design, to negotiate different approaches to resolution, and to review the design rationale for various model components. Normally this is done through conversation and meetings. The MBVE creates a CSCW environment to support informal conversation and more formal meetings among a group of geographically dispersed designers where the context of the meeting is set by the design model and information base.

We observe that if any new system removes or changes an organization's underlying social model, it will probably only be effective if the users are willing to adapt their fundamental social behavior to fit the system's implied model. In this study, our premise is that it is risky to build computer systems depending on social change for their success. Instead, we advocate collaboration systems based on formal coordination models, yet which explicitly support existing social models in the organization through informal broadband communication. We believe this is accomplished by having a high degree of logical immersion, a relatively high level of computation support, and a moderate and end-user-controlled amount of automatic coordination. In the remainder of this paper we explain how the MBVE provides support along these dimensions. We then elaborate on design applications then on enactment applications. We conclude with general remarks regarding the general feasibility on developing an effective MBVE.

2.1 Logical Immersion

CSCW systems implicitly provide some degree of immersion into a processing domain by virtue of the functionality and computing paradigm provided by the application software used by the group members. The aspect of logical immersion on which we focus relates to the supplementary tools to support informal communication among the group members independent of the immersion provided by the common computation tools.

Electronic mail has played an important role in providing a supplementary mechanism to enable group members to correspond with one another. Bulletin boards and news groups have also been used to help establish a group environment, e.g, we always provide students with a news group (or equivalent) for every course we offer. Web pages are quickly establishing themselves as a superior alternative to bulletin boards, since they are much richer in the range and nature of information that can be incorporated into a page.

The MBVE is envisioned as providing a *same time, different place* or *synchronous* virtual meeting room. While physical immersion in a meeting room is not possible for people in different places, the psychological effect of attending a meeting is the intent of logical immersion facilities. The point on the logical immersion axis where the MBVE focuses is analogous to desktop virtual reality rather than going to fully immersed VR. We have made this choice since we believe the desktop presence is sufficient for willing meeting attendees, i.e., it is neither necessary nor desirable to provide full immersion into the virtual space. Prototyping the MBVE is intended to provide us with concrete feedback regarding this conjecture.

A second level goal of the MBVE is to provide facilities in which we can begin to experiment with *different time, different place* or *asynchronous* meetings. Synchronous meetings are traditionally an annoyance to the individual since they require the person's presence and attention at a time chosen by the group (or its leader). Alternatively, the individual is likely to be far more effective in the meeting if s/he can choose the time at which they participate in the meeting. The MBVE is an enabling technology to experiment with asynchronous meetings: can the group operate effectively with surrogate attendees? How can meeting versions be used to accomplish some aspects of collaborative work? There are many other radical and innovative ideas to explore once the MBVE prototype is usable.

2.2 Computation Tools

This dimension of computing has flourished over the last decade. We have evolved from a society where commerce was conducted using typewriters and telephones to one based on corporate databases, decision support systems, desktop databases, spreadsheets, desktop publishing, and internet browsers. End user programming, per se, has not been particularly successful unless one considers spreadsheets to be a form of programming (it is often disqualified on the basis of its limited I/O functionality).

The critical aspect of this axis with respect to the MBVE study is that the environment include applications with significant domain support. In this case, we represent domain knowledge with a specific formal model. In our prototype the formal models are colored Petri nets (CPNs) information control nets (ICNs). CPNs and the bilogic precedence graph (BPG) variant of ICNs have been used as design and analysis languages to study computer systems, distributed computations, network protocols, office procedures, etc. ICNs are the base model of the Bull FlowPATH workflow product.

We plan three types of computational support for these studies: first is the use of the models to represent, design, and analyze the work — a typical application of CPNs and ICNs/BPGs [11, 34, 31]. Second, is to use ICNs as the basis of goal-based workflow enactment system [12]. Third the computation support within the context of a model of the work, e.g., an ICN model. We defer discussion of goal-based workflow enactment to the subsection on coordination models, and summarize our position on the other two aspects of computation support here:

Modeling Work. There are two levels at which a system can offer support: first it can be used to identify parts of the general work that might be automated. Second, as long as a model is involved in the work analysis, the system can provide support for creating, managing, and analyzing the work represented by the model.

A process model is intended to capture the *behavior* of a process by specifying the computational steps ("activities") and the coordination among their execution. We advocate supporting collaborative work by having the work analysts identify computational tasks so they can be supported through explicit programming support, while modeling "unstructured" work where a human must decide the nature of the work, possibly on a case-by-case basis (see [11, 12]).

Modeling support is provided by providing a computer tool to create models of the work, and other tools to analyze the models for various properties — performance and other behaviors. The Olympus modeling system is one example of a modeling system that represents the type of modeling support we are embedding in our MBVE prototype [34].

Supporting Computation with a Contextual Environment. Past experience with workflow models and systems led us to relax the rigidity with which one represents group work (see [13]). Blumenthal has derived the Bramble prototype system to provide computation support to the execution of unstructured work in a goal-based ICN model [5]. The idea behind Bramble is that unstructured work can be modeled as a part of structured work, and that when an instance of the unstructured work should occur, the system ought to be able to provide broad types of assistance to the human user responsible for conducting the work. Bramble uses a variety of techniques to provide this assistance, including a veiled means of end user program within an ICN environment.

2.3 Supporting Coordination

The coordination model of collaborative work defines the precedence among executions of decomposed activities for any particular unit of work being processed by the group. It is this aspect of supporting group work where researchers have the most diverse opinions of the proper approach. The "manual office" provides coordination support through standing procedures, group knowledge, and explicit routing sheets. In the situated work perspective, it is important that these vague mechanisms be preserved while the computation steps become automated. In the workflow camp, the precedence is implemented as a rigid part of the computer system.

We believe it is necessary to support the situated work perspective when a new system is introduced into a group, since the actual coordination model used among the group of humans is probably not really known to any single individual. By enabling an analyst to study the work and to choose a model will result in an inadequate model of how coordination actually occurred in the manual office. This is most observable by noticing that the coordination model is typically embedded within an informal social model held by the group; if the system attempts to enforce any other precedence (or *scheduling*) mechanism using the computer, the effect on the social model will be significant — enough so that it generally causes the system to be disliked and rejected by the group members.

On the other hand, following Malone's view that computers ought to be used to handle coordination [28], the system ought to be sufficiently robust to allow the group to create their own evolutionary coordination model. That is, the system must handle a spectrum of values along the z-axis of the characterization model in Figure 1. where the control of the selected point on the axis belongs to the group rather than to a model analyst.

We believe an MBVE provides the means by which incremental, evolutionary coordination models can be created and used by the group. The coordination model is a part of the environment, perhaps used to document the way work is done in the generic case, to indoctrinate new group members, and to troubleshoot non-generic cases. If the environment is sufficiently flexible to allow the group to treat coordination as a "hint" or as a directive, then the group can decide how any aspect of coordination should be handled within the group's work. Thus the group can work at the situated work end of the spectrum for some parts of the work, and at the workflow end of the spectrum for other parts of the work. The virtual environment provides the means for information communication regarding the coordination, and the model provides the mechanism for documenting the agreed on means of collaboration.

3 Using the MBVE to Support Design

Models provide a rich language for describing and documenting knowledge about any particular process, system, organization, or other topic of study. Models are always associated with some real system (though it may not have actually been built), yet differ by focusing on a subset of the properties of the real system deemed to represent aspects of the real system important to the analyst. For example, an economic modeler of rockets would ignore the physics of the rocket in favor of detail regarding the cost of designing, building, launching, and maintaining the rocket. A fuel engineer might focus on various physical aspects of the rocket such as its weight, temperature, velocity, etc., completely ignoring the details of its computer guidance system. Hence models are most useful to people that accept the assumptions associated with the model and who believe the critical aspects of the real system have been captured by the model; one would not expect the economic modeler and the fuel engineer to agree on the same model of a rocket.

When a group agrees to collaboratively work on a problem, each has his/her own set of criteria for what is important about the work. Much of this opinion is informal, though as the size of the group grows, it is important for the group to agree on the critical aspects of the work and to reach agreement/consensus on how those aspects of the work will be handled. Formal workflow models perform exactly this function when used by a group to *describe* a process or set of procedures they use to conduct work (although they all usually recognize the model is insufficient for actually *doing* the work). This is the rationale driving Malone et al's Process Handbook to study different ways of doing work in an office [29].

The MBVE couples this formal modeling — for processes or for other work conducted by the group — with virtual environments to enable a set of geographically-separated team members to meet in an electronic meeting room with full modeling support tools. This is an exciting avenue of development for at least four reasons:

1. It provides an effective way for people to create and edit a common model.
2. It is a testbed for fine-grained group interaction.
3. It is a testbed for exploring notions of alternative *views* of the same thing in a virtual environment.
4. It provides a mechanism for preserving the informal discussion where rationale for a group's decision is saved.

In the remainder of this discussion we focus on the last aspect of the system, exploiting the MBVE as a means for capturing and managing a group's rationale for model design decisions.

3.1 Design Rationale Capture and Administration

The MBVE provides an alternative to extensive travel for a group of collaborators located in physically distributed places. Physical distribution enables flexible team-building, flexible work locations, etc. Perhaps more importantly, a meeting in the MBVE can be recorded, capturing the context, thoughts, atmosphere, implicit agreements, and innuendos of the participants. Further, *snippets* of the meeting can be indexed and referenced from individual parts of the formal model. For example, suppose a conflict arose among a group of designers. They might meet in the MBVE, informally identify important criteria in the conflict, propose different solutions, discuss the pros and cons of the proposals, and eventually reach consensus (according to the group's organization and psychology). Since the meeting is conducted in the MBVE, it is possible to index critical snippets of the meeting and to reference those snippets from the formal model.

There are several open questions in this brief scenario: How should indices be chosen? Who can index snippets? Is it necessary to save all parts of all meetings? How can a snippet be identified within a full meeting? As a starting point, we propose to ignore logical compression issues in favor of testing the feasibility of identifying, indexing, and referencing the meeting's parts. We will also initially allow any meeting participant to mark a meeting snippet and to enter a reference to the snippet into the formal model.[3]

The entire meeting might be physically compressed, e.g., using MPEG-2, then saved in a large (hierarchical) storage. In our prototype, we will save raw images on disk/tape servers, then allow references to directly address snippets. Follow-on work would need to investigate space issues.

As a final remark about broad set of possibilities in such a system, we observe that it may be possible to derive an "inverse garbage collector" to detect references to snippets, to save the snippets in relatively accessible levels of a storage hierarchy, and to migrate the bulk storage of the MBVE session to lower levels of the hierarchy. This, too, is follow-on work enabled by the research outlined here.

Notice this approach establishes a base platform for considering different time, different place work. The design rationale archive allows an individual to browse the formal model, observing prior meetings in the MBVE as required. The extension of this work is to incorporate agents for the human designers in the MBVE. A simple agent is empowered to participate in a meeting and to make simple decisions on behalf of its human user, even when the human user is not actually using the MBVE at the same time as another human user. The simple form of such an agent participant would just apply a few simple design rules encoded by its human user. However, this experiments with an architecture in which agents could be arbitrarily complex.

3.2 Meeting Capture and Management

Information describing a meeting in the MBVE is necessarily formatted in data structures — objects — exchanged among the clients. This information can be captured to archive any session in a MBVE. In our study, we do not address the nature of the storage hierarchy required to support a production MBVE, but rather, assume a flat file space to store meetings and meeting snippets.

Part of the capture technology must also include an annotation facility to allow users to mark snippets and to attach a set of reference keywords. It should also be possible to mark and annotate after a session has been completed. Our initial approach to this function will be patterned after conventional methods for marking scanned images and videotapes.

The modeling system will then be modified to allow users to attach snippet references to various model components. A model browser can then be used to inspect the model, including following pointers to meeting snippets. A user browsing the model can also view the meeting snippets.

[3]Note the natural extension of techniques used in commercial image scanning products. Note also the application of video queueing and marking techniques used in videotape editing.

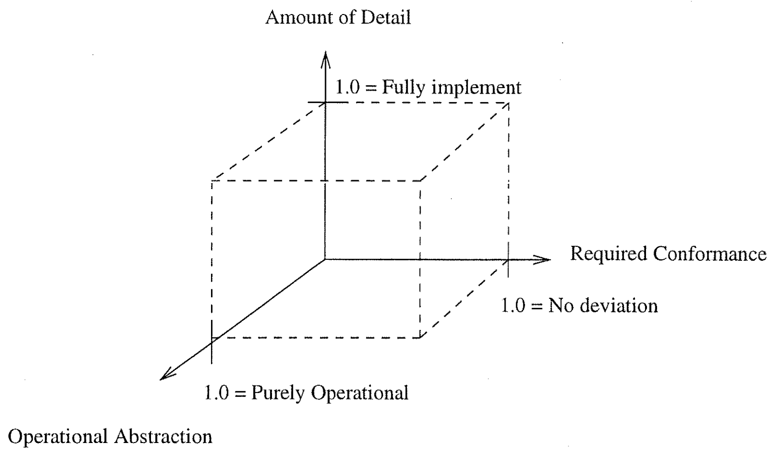

Figure 2: The Model Domain Space

4 Using the MBVE to Support Enactment

The enactment system is the target of the coordination function in the process model, i.e., we want to be able to drive simple enactment implementations from the goal-based ICN model. We have derived another informal characterization of process models used for enactment. Process models cover a large space of approaches, ranging from operational to declarative, informal to formal, and descriptive to prescriptive. Historically, different researchers have adopted some domain in this space, and directed their work at that domain. The process models that we are developing are intended to be sufficiently flexible to be applied to all points in the space, depending on the nature of the model that is required for that part of the process.

A model can be used to represent parts of the process that *must* be performed for the process to be acceptable (e.g., the part of the work that is controlled by federal regulations), and parts that *may* be performed by a computer or human (e.g., a mechanism to describe a recommended way of accomplishing the work). The language might also allow one to represent parts of the process at very abstract levels as well as very detailed levels; we believe that very detailed representations correspond to enactment implementations. Along a third dimension, the model supports representations that are highly operational or highly declarative; goal and intentional specifications are considered to be highly declarational and a C encoding of a sorting algorithm is considered to be highly operational.

Thus, the model is intended to represent processes according to the way the model is to be used, as defined by three different criteria: the amount of *conformance* that is required by the organization for which the process is a model, the *level of detail* of the description, the *operational* nature of the model (i.e., to what degree does the model describe *what* is required rather than *how* the process works (see Figure 2). In this domain space, systems that represent only *structured* work are in a subspace where $x \rightarrow 1$, $y \rightarrow 1$, and $z \rightarrow 1$; systems intended to address *unstructured* work are in a space where $x \rightarrow 0$, $y \rightarrow 0$, and $z \rightarrow 0$. The domain for which our traditional ICN model was intended is the plane defined by $0 \leq x \leq 1$, $0 \leq y \leq 1$, and $z=1$. Workflow enactment systems based on models such as the ICN could be characterized as a point in the space with $x=1$, $y \rightarrow 1$, and $z = 1$. Systems that focus on exception handling are in a space where $x \leq 1-k$, $(0<k<1)$ and $0 \leq y \leq 1$

and $0 \leq z \leq 1$. Goal-based systems are in a domain in which z=0, but x and y vary according to the specifics of the model; e.g., an inferencing system assumes that y<1.

The revised ICN model is intended to address the full space, with different parts of the model addressing different subspaces according to the need for that part of the model. For example, if part of the work is highly structured, operationally specified, and required to be accomplished according to the specification, then it should be modeled differently than work for which only the goal is known. The model should allow one to represent a process for which parts are operational and required, while the way that other parts are executed is arbitrary, provided that the executions satisfy the intent.

An ICN is a precedence graph representation of steps in the process; each step belongs to a *region* whose type is (informally) defined by a point in the space in Figure 2. For example, a type "R" (for required work) region might be represented by a point in the space at (1, 1, 1); the model can be represented as a conventional ICN subgraph composed entirely of operationally specified steps.

A type "A" region (for assisted work) also uses an operational-style specification, but the submodel in the region can be interpreted as one approach to accomplishing the work (in the absence of a declarational specification). This type of specification is a point in the space such as (0, 1, 1); it is used when a process designer has one notion of how to accomplish the work, but is not willing/able to abstract the steps into a declarative form; the A-region work can be used directly, or it can be used to (manually) infer the intent of this part of the work. This is how traditional (R-region) workflow models are often used — as an example of how work might be done.

A type "D" region (for declaration region) represents a part of the model that defines *what* the region is intended to accomplish, rather than a description of *how* the work must/might be conducted. This type of region occurs at (0, 1, 0).

A type "G" region (for goal-based region) is at (0, 0.25, 0), meaning that it lacks detail compared to a D-region, but specifies the goal and intent of the work in the region.

Using the Model Space. Once models have been designed and analyzed, some tasks within the overall procedure will be well-understood with certain regions constraining the way some work is to be performed. For example, parts of a purchasing process are well-understood (e.g., how a purchase requisition should be completed) and can be represented by an A-region; others parts are constrained by federal regulations (e.g., interactions with potential vendors must meet specified criteria) and can be represented by an R-region in the model; and other parts may be ill-defined but required (e.g., selecting a vendor based on competitive, but not equivalent, offers) and will be represented by a D-region or G-region.

The purpose of the MBVE is to provide the electronic mechanism to study the handling of process exceptions and to otherwise troubleshoot process operation. It is a virtual environment containing knowledge and tools relevant to the process, and which enables humans to interact with one another to solve problems that require group attention.

5 An MBVE Prototype

There is a companion working paper describing more of our approach in the design and implementation of MBVE systems [33]. In this section, we describe our general assumptions and approach to demonstrate the feasibility of the MBVE.

The MBVE requires a significant minimum threshold of hardware support. Specifically, our initial work has relied on the existence of sophisticated graphics controllers in each workstation. Because we have an older SGI Indigo workstation, and because we have used the MR Toolkit [36], our initial prototyping work depended on the SGI GL interface. Subsequently, we have removed our dependency on GL and on MR Toolkit by evolving the prototype software to use VRML to describe scenes, and Mesa OpenGL to implement these scenes on X displays. This has allowed us to experiment with other hardware platforms including HP RISC workstations and Pentium-based workstations with sophisticated graphics controllers.

The SGI/HP/Pentium environment we use is a UNIX environment. Because of our interest in openness, and in acquiring as many tools from the public domain as possible, UNIX is a natural environment in which to conduct our early prototype development. However, we are also convinced the MBVE domain will make strenuous performance demands on the system software. Therefore, we propose to use the Mach microkernel as the basis of a specialized server, designed explicitly to support the MBVE.

A second critical area of the design and implementation is in the means for representing data in the MBVE. Like many of our colleagues, we are focusing on the use of distributed objects to implement the MBVE and its components. While we believe VRML will provide a means for us to promote the inclusion of various components from the public domain, it is necessary for us to cast VRML components as objects shared across distributed hardware platforms. Our interest in distributed objects is driven by the MBVE application rather than by the technology. Therefore, we have attempted to characterize our requirements for distributed objects so that the implementation can provide explicit support for this type of objects rather than providing generic support for all types of objects. We will briefly describe these requirements below.

Third, the MBVE is an electronic meeting room, thus it needs to use contemporary technology for implementing multimedia artifacts, specifically audio and video streams. There is a rapidly growing community of researchers investigating the issues in such applications, resulting in advances in compression (e.g., see [9]), high-performance delivery (e.g., see [3]), inter stream synchronization (e.g., see [27]), etc. Our approach here is to apply others' research in a high-performance local environment rather than to derive new protocols. Our work in this area is described in [33].

Fourth, capturing and managing meeting snippets is conceptually straight-forward, though the engineering details are intimidating. A snippet is an arbitrarily large block of multimedia information to be saved on mass storage for later reference. At this point, we intend to simply prototype a snippet storage server without making use of standard or heuristic compression techniques. We rely on the continued development of the relevant technologies to solve much of this part of the problem, though we believe it is important to begin to experiment with the functionality now.

5.1 The VE Application Software

The MBVE is constructed by first providing a virtual enviroment (VE) to be used for collaborative meetings, then to augment the VE with model-specific applications. There are two levels at which VE applications are constructed. First, artifacts appearing in the environment must be designed. Second, a specific VE must be defined in which users navigate and interact with other users and with artifacts. The specific model support within the VE is one such application (set). The VE specification defines the virtual space and all artifacts in it. The behavior of the VE — its applications — is defined by the collective behaviors of the artifacts in the VE.

The system employs a client-server architecture with artifacts logically implemented on the server, and human-computer interactions occurring at client workstations. The client supports and controls its user's perception of the VE, but the server controls the session and shared artifact semantics. Different users may see different visualizations of the logical MBVE depending on the client implementation they are using.

A virtual room/space is defined by first defining a collection of classes representing different kinds of artifacts used by the corresponding session. A session is instantiated by executing an extension of VRML scripts to create the desired set of objects representing the artifacts in the virtual space.

5.2 MBVE Objects

Generic Artifacts. Each artifact in a VE is part of a VRML description of the initial state of the VE. When the VE is instantiated, the VRML is used with class descriptions to instantiate each object in the room with the appropriate VRML appearance. As objects change in the VE, incremental changes to the VRML description are used among the constituent physical computers involved in the session. It is also possible to regenerate a new VRML description of the VE after it has changed; this allows VE states to be loaded, changed, and saved for subsequent meetings.

Occupant Artifacts. An *occupant* is an object representing a user in a virtual space. The first version of the MBVE represents an occupant as an object having volume but no mass. Therefore, other occupants can see it, though they cannot "physically" interact with it. The state and behavior of the occupant artifact are defined by a class, `Occupant`. Conceptually, an occupant is the aggregate of at least an object of type `Eye` and another object of type `Hand`. An `Occupant` can be either a `Browser` or an `Editor`:

The `Browser` Base Class. A browser object navigates among objects in a session according to user directives, and renders artifacts on a local display according to the viewing perspective of the occupant's `Eye` instance. The occupant cannot cause the state of any object other than its own `Hand-Eye` sub objects to change.

The `Editor` Base Class. An occupant of type `Editor` can perform browsing operations, and can also create new artifacts and modify the location of existing ones. Any change in location of a passive object, P, by an active object A, requires A.Hand be directly associated with P in the virtual space. This means the hand can only move objects by making contact with P, grasping P, moving A.Hand, then releasing P. P cannot be "thrown" or "hit" by A.Hand to cause independent motion by P. A Type 8 object could have independent motion, e.g., caused through the application of some virtual force.

Since there may be many objects and occupants in a session, the interaction with shared artifacts is managed by a logical *interaction manager*. This interaction manager arbitrates the colocation of objects with mass in the virtual space. For example, A.Hand might implement the interaction manager function by having A.Hand check the virtual space where it is about to move to see if another object with mass already occupies the space; if not, A.Hand moves into the space.

6 Conclusion

The hardware technology to support distributed, collaborative work among a group of people is in a relatively advanced stage compared to the state of software for CSCW. The software problems are difficult along two fronts: software technology to make cost-effective use of the hardware has lagged the hardware technology substantially. Second, and the main emphasis of this paper, there is no wide-spread agreement as to the nature of support the distributed hardware should provide to its users.

We see a trend in computer technology to derive cost-effective software extensions to inexpensive hardware so that it supports desktop multimedia conferencing, virtual environments, and other applications of virtual reality technology. Simple electronic meeting room systems represent the leading edge in practical applications of this logically immersive functionality. Support of the computation aspect of work has begun to emerge, though the emphasis in this area is in honing personal computing tools. A third aspect of CSCW we consider is the amount of automatic coordination the computer system provides to the group. One school of thought argues for minimal coordination support, on the basis that this aspect of the work is handled best by humans recognizing the context for the work and understanding the social structure of the organization performing the work. Another school believes that computers should be delegated the task of coordinating the work.

In this paper we argue the case for model-based virtual environments. Our argument is that the distributed computer system should provide a flexible enviroment that is highly logically immersive to provide broad bandwidth communication among the group workers, and so that the communication medium explicitly supports informal/social communication among the group members. The MBVE includes an embedded formal model to help frame meetings among participants, to provide an efficient mechanism for documenting agreements, to provide a framework on which informal rationale can be organized and saved as VE meeting snippets, and to provide a "war room" for trouble shooting cases that can only be solved with situational analysis. The goal of this paper is to argue the merits for MBVE as a possible mechanism to accommodate CSCW applications where the group members control the amount of coordination support in the work as suits their taste.

References

[1] Special Issue of AT&T Technical Journal on Multimedia, September/October 1995. Nikil Jayant, Technical Reviewing Editor.

[2] Ronald M. Baecker. *Readings in Groupware and Computer-Supported Cooperative Work*. Morgan Kaufmann Publishers, Inc., 1993.

[3] Anindo Banerjea, Edward W. Knightly, Fred L Templin, and Hui Zhang. Experiments with the tenet real-time protocol suite on the sequoia 2000 wide area network. In *Proceedings of the Second ACM International Conference on Multimedia*, pages 183–191, 1994.

[4] David A. Berkley and J. Robert Ensor. Multimedia research platforms. *AT&T Technical Journal*, 74(5):34–45, September/October 1995.

[5] Richard Blumenthal and Gary J. Nutt. Supporting unstructured workflow activities in the bramble icn system. In *Proceedings of ACM 1995 Conference on Organizational Computing Systems*, pages 130–137, August 1995.

[6] Sara A. Bly, Steve R. Harrison, and Susan Irwin. Media spaces: Bringing people together in a video, audio and computing environment. *Communications of the ACM*, 36(1):28–47, January 1993.

[7] Duane K. Boman. International survey: Virtual environment research. *IEEE Computer*, 28(6):57–65, June 1995.

[8] Bull S. A. *Introduction to FlowPATH*, May 1992. Manual No. 44 A2 60XM.

[9] Didier Le Gall. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):47–58, April 1991.

[10] C. Ellis, S. J. Gibbs, and G. L. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):38–58, January 1991.

[11] Clarence A. Ellis and Gary J. Nutt. The modeling and analysis of coordination systems. In *ACM 1992 Conference on Computer-Supported Cooperative Work*, 1992. workshop position paper.

[12] Clarence A. Ellis and Gary J. Nutt. Modeling and enactment of workflow systems. In *Advances in Petri Nets 93*, pages 1–16, June 1993. Invited paper.

[13] Clarence A. Ellis and Gary J. Nutt. Workflow: The process spectrum. Submitted as a workshop paper, January 1996.

[14] Clarence A. Ellis and Jacques Wainer. Goal based models of collaboration. *Collaborative Computing*, 1:61–86, 1994.

[15] Scott Elrod, Richard Bruce, Rich Gold, David Goldberg, Frank Halasz, William Janssen, David Lee, Kim McCall, Elin Pedersen, Ken Pier, John Tang, and Brent Welch. Liveboard: A large interactive display supporting group meetings, presentations and remote collaboration. In *Proceedings of Chi '92*, pages 599–607, 1992.

[16] Lennart E. Fahlen, Charles Grant Brown, Olov Stahl, and Christer Carlsson. A space based model for user interaction in shared synthetic environments. In *Proceedings of Interchi '93*, pages 43–48, April 1993.

[17] Domenica Ferrari, editor. *Proceedings of the Second ACM International Conference on Multimedia.* ACM, 1994.

[18] Robert S. Fish, Robert E. Kraut, Robert W. Root, and Ronald E. Rice. Video as a technology for informal communication. *Communications of the ACM*, 36(1):48–61, January 1993.

[19] Thomas A. Funkhouser. RING: A client-server system for multi-user virtual environments. In *1995 Symposium on Interactive 3D Graphics*, pages 85–92. ACM, 1995.

[20] Jania Gajewska, Jay Kistler, Mark S. Manasse, and David D. Redell. Argo: A system for distributed collaboration. In *Proceedings of the Second ACM International Conference on Multimedia*, pages 433–440, 1994.

[21] David Gelernter and Nicholas Carriero. Coordination languages and their significance. *Communications of the ACM*, 35(2):97–107, February 1992.

[22] Jonathan Grudin. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM*, 37(1):93–105, January 1994.

[23] Special Issue of IEEE Computer on Computer-Supported Cooperative Work, May 1994. James D. Palmer and N. Ann Fields, Guest Editor.

[24] Special Issue of IEEE Computer on Virtual Environments, July 1995. David R. Pratt, Michael Zyda, and Kristen Kelleher.

[25] Special Issue of IEEE Computer on Multimedia Systems and Applications, May 1995. Arturo A. Rodriguez and Lawrence A. Rowe, Guest Editors.

[26] Hiroshi Ishii, Minoru Kobayashi, and Kazuho Arita. Iterative design of seamless collaboration media. *Communications of the ACM*, 37(8):83–97, August 1994.

[27] Lian Li and Nicolas D. Georganas. MPEG-2 coded- and uncoded- stream synchronization control for real-time multimedia transmission and presentation over b-isdn. In *Proceedings of the Second ACM International Conference on Multimedia*, pages 239–246. ACM, 1994.

[28] Thomas W. Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, March 1994.

[29] Thomas W. Malone, Kevin Crowston, Jintae Lee, and Brian Pentland. Tools for inventing organizations: Toward a handbook of organizational processes. In *Workflow 95 Conference Proceedings*, pages 57–78, 1995. Also appears in Proceedings of 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises, April, 1993, and as MIT Sloan School technical report CCS WP No. 141, Sloan School WP No. 3562-93.

[30] J. F. Nunamaker, Alan R. Dennis, Joseph S. Valacich, Douglas R. Vogel, and Joey F. George. Electronic meeting systems to support group work. *Communications of the ACM*, 34(7):40–61, July 1991.

[31] Gary J. Nutt. A Simulation System Architecture for Graph Models. In G. Rozenburg, editor, *Advances in Petri Nets '90*, pages 417–435. Springer Verlag, 1990.

[32] Gary J. Nutt. A study of virtual planning rooms for evolutionary software design environments. A proposal to ARPA under BAA 95-40, November 1995.

[33] Gary J. Nutt, Joe Antell, Scott Brandt, Chris Gantz, Adam Griff, and Jim Mankovich. Systems support for a virtual planning room. Technical Report CU-CS-800-95, Department of Computer Science, University of Colorado, Boulder, December 1995.

[34] Gary J. Nutt, A. Beguelin, I. Demeure, S. Elliott, J. McWhirter, and B. Sanders. Olympus: An Interactive Simulation System. In *1989 Winter Simulation Conference*, pages 601–611, Washington, D.C., December 1989.

[35] Heikki Saastamoinen, Markku Markkanen, and Vesa Savolainen. Survey on exceptions in office information systems. Technical Report CU-CS-712-95, Department of Computer Science - University of Colorado, Boulder, 1994.

[36] Chris Shaw, Jiandong Liang, Mark Green, and Yunqi Sun. The decoupled simulation model for virtual reality systems. In *Proceedings of SIGCHI 92*, pages 1–8, 1992. (Available as ftp:menaik.cs.ualberta.ca.).

[37] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar. Wysiwis revised: Early experiences wtih multiuser interfaces. *ACM Transactions on Office Information Systems*, 5(2):147–167, April 1987.

[38] Lucy A. Suchman. Office procedure as practical action: Models of work and system design. *ACM Transactions on Office Information Systems*, 1(4):320–328, October 1983.

[39] Harrick M. Vin, Polle T. Zellweger, Daniel C. Swinehart, and P. Venkat Rangan. Multimedia conferencing in the etherphone environment. *IEEE Computer*, 24(10):237—268, October 1991.