AN INTERNET BASED ARCHITECTURE SATISFYING THE DISTRIBUTED BUILDING SITE METAPHOR

Fabien Costantini[#], Antoine Sgambato[#], Christian Toinard[#], Nicolas Chevassus[+], François Gaillard [+]

[#]CEDRIC – Centre D'Etudes et de Recherche en Informatique CNAM
292 rue Saint-Martin 75141 Paris Cedex 03 France
costanti@cnam.fr, sgamba_a@cnam.fr, toinard@cnam.fr
[+]AEROSPATIALE-MATRA Centre Commun de Recherches
12 rue Pasteur BP 76 92152 Suresnes Cedex France
nicolas.chevassus@aeromatra.com, francois.gaillard@aeromatra.com

ABSTRACT

Virtual reality offers new possibilities of cooperation for the concept phase of a product development. The deployment of a cooperative system suffers mainly from the client-server approach that is inefficient in many ways and attributes a leading role to a server site. Moreover, requiring a specific quality from the under-laying communication restricts openness and usability of the solution. At last, current solutions focus on reliable multicasting and manage moving objects but they address poorly consistency and parallel working.

The paper describes the Distributed Building Site Metaphor. That solution enables a real-time 3D cooperative design and parallel work within shared virtual worlds while preserving the scene consistency. Designers move easily between different styles of work. They prepare the shared work on a standalone basis. During a meeting, the shared work reforms automatically. The participants move from a collaborative design to a review activity or real-time conciliation.

The main focus is the core services and protocols supporting efficiently the different properties of our metaphor. Communication services over standard IPv4 or IPv6 protocols are proposed. They use intensively the multicast ability of IP protocols and support a fully distributed approach without any centralized control.

Keywords: communications services, collaborative design, advanced man-machines interactions, virtual reality.

1. INTRODUCTION

Interaction of workers is a challenging area of research in cooperative design. Virtual reality is used to support the cooperation in design of manufacturing products. Most of the time, solutions focus on design review. Less often the system supports a collaborative design activity. Currently, the CAD/CAM (Computer Aided Design and Computer Aided Manufacturing) systems require a long term working in order to design a new product with a high numerical precision. They support poorly the collaborative design and early stage of the concept phase.

Historically, distributed virtual reality is associated with simulations of battlefields like [Maced95]. These systems mainly tackle the problem of how reducing the network traffic due to a high number of moving objects. Generally, a way to divide statically the scene is provided either for scalability or persistency.

[Barru96] proposes to divide a large virtual world into different locales according to a geographic cutting. A given user can create and modify a locale. The users can combine their separate locales together. The combination corresponds to a connection of the different locales (each locale can be seen like a room). A user is aware of the modifications carried out on neighboring locales. A user can move to any neighboring locale. The work is consistent because a locale belongs to a single user (i.e. the creator maintains the locale). Thus, the locales are distributed statically beyond the different users. Any user can place a new object into a locale. He is responsible for the maintenance of his object. So, an object statically belongs to his creator.

Like [IBMDa98][e-Vis99], industrial solutions provide a collaborative review of the scene through the view point synchronization, telepointers and annotations. Thus, a simple form of design review is achieved. GroupWare toolkits [Green96] [Netsc98] do not manage a virtual scene. Collaboration consists in a shared white board, session management, telepointers and chatting facilities. Moreover, [Netsc98] provides audio transmission and integrates the application [IBMDa98]. [e-Vis99] provides a solution where conferencing and review of a 3D scene are closely integrated but in essence it tackles the real-time design review. Generally, the users can run the scene navigation directly from a group session and benefit from the GroupWare collaboration services.

[Leigh97] addresses design review and cooperation services with immersion in the virtual world. This is a client-server solution. A client must specify a Quality of Service (QoS) in order to have the desired bandwidth and jitter. These requirements are due to the nature of cooperation where immersion generates a great volume of data with real-time constraints.

[HLA97] is a standard addressing mainly simulation of battlefields. It defines services but do not address how these services shall be implemented. Protocols are not standardized in order to get a freedom of solutions. In practice, a run-time is under development. It assumes resource reservation (i.e. ReserVation Protocol) and reliable multicasting. These requirements seem necessary in the event of a great number of state transmissions associated with moving objects. This standard defines different ordering of events where the application associates logical dates to the events. It guaranties a reproducible simulation. But, collaborative design is interested with the parallel working and consistency and not with the reproducibility.

2. DISTRIBUTED BUILDING SITE METAPHOR

To address virtual early prototyping, we have defined a new solution formerly called the Distributed Building Site Metaphor (DBSM) [TCS99]. DBSM is not limited to design review. Different designers meet into a building site to cooperate at the concept design of a product. Each designer can prepare some work into an isolated place before entering in the building site. Thus, he brings prepared work in the building site. Within the building site, the designers work in parallel to improve the work of other participants and to complete their own work. When leaving the building site, each participant decides which pieces of the global shared site he wants to carry at home in order to improve these pieces on an isolation basis. During a further meeting, the building site will reform automatically by reunifying the global shared work. Parallel working is supported during the isolated and the meeting phase but the work remains consistent DBSM also defines a distribution solution where client-server and quality of service pitfalls are avoided. Thus, different properties and functions are provided in a fully distributed way and do not require a specific quality of service from the under-laying network.

The following features, each of which provides important leverage on the problem of collaboration, define our metaphor:

- application tree: the application maintains a scene tree that is composed of different objects. The scene tree is an abstraction of the application scene. That scene describes graphical information but also data that are relevant for the design activity.

- hierarchical object: each object defines a tree where the leaves are elementary nodes (i.e. a beam or an electrical route).
- object protection: An object and elementary nodes is associated with protection attributes (read right and write right). Thus, a user prevents or authorizes the modifications from the other users.
- shared workspace: the scene tree is maintained by the system. It is accessible during a meeting between several users. The different users enter independently into a meeting. The meeting allows the users to reform the global scene tree by unifying their isolated workspaces. The scene is processed as a whole when all the users are present. Only a subset is accessible when users are missing. Each user creates, deletes or modifies nodes in line with the protection rules.
- isolated workspace: a user prepares a subset of the shared space on a standalone basis. Thus, a user creates, deletes or modifies nodes without any communication at all.
- isolated nodes: when leaving a meeting, a user defines which nodes he wants in his isolated workspace. At the time of leaving, two kinds of nodes can be distinguished. The nodes owned by the user are brought into his isolated space. The other nodes can be selected by the user in order to get a clone or a duplicate in his isolated space.
- parallel working: during a meeting two nodes owned by two different peers are processed in parallel without synchronization between the two tasks. Moreover, two isolated spaces provide a parallel working that does not use any communication at all.
- concurrency control: the system supports concurrent requests for the same node. These concurrent requests are serialized in order to maintain the consistency. Consistency is provided in a distributed way through the ownership transfer.
- shared work consistency: the work remains consistent because any modification is achieved from the latest state of the considered node. The owner grants state and ownership at a time to a requesting peer.
- real time awareness: during a meeting, any participant observes the distant interactions as quick as allowed by his processing speed. Event passing authorizes a synchronization of the distant peers. The transmission time is as short as possible. A peer speeds up the delivery of recent updates.
- distribution of the scene tree: the global scene tree is not maintained on a central server. It is completely distributed over the different users through node relationships. During a meeting, the global tree reforms automatically using the local relationships of each owner.
- dynamic distribution of the scene tree: during a meeting, the ownership of any node is distributed dynamically according to the user requests. So, the management of a node is distributed dynamically.
- conciliation: during a meeting, the participants use a conciliation service to combine gracefully their separate works. They propose different alternatives and define the modifications for a subset of nodes.
- work persistency: different levels of persistence are provided from participatory persistence to continuous persistence.
- full distribution: the solution does not rely on any central server. A server does not maintain the shared workspace. Consistency is not provided through a state server or a lock server for the concerned object. Persistency is completely distributed. The system does not use any server to manage the arrivals and departures of the participants.

These features confer different key benefits that are described in [TCS99]: aggregate and scattering of objects, advanced collaboration for the design activity, efficient collaboration, communication efficiency, parallel working and consistency, fast animation of the scene, freedom of conciliation, fault resistance and efficient persistency.

3. CONTRIBUTIONS OF OUR IMPLEMENTATION

The contribution of our metaphor can be found in [TCS99]. In that paper, the core services supporting the metaphor are described. Our Internet architecture respects the full distribution constraints of our metaphor. New protocols, running over standard IPv4 or IPv6 networks, are defined.

As many solutions [Maced95][Hagsa96][Broll98], our system uses best effort communications available on any standard IP (Internet Protocol) machine with UDP (Unreliable Delivery Protocol) and standard multicast ability.

Our services do not implement a reliable multicasting for the event transmissions like [Hagsa96][Broll98]. Systems building a reliable multicasting are confronted with the acknowledgment implosion problem that is still an on going work. Moreover, a reliable multicasting does not guaranty the work consistency. It only guaranties that a participant will receive all the events [Broll98] or fresher ones [Hagsa96]. Solutions like [Hagasa96][Broll98][HLA97] do not guaranty that the modification of an object is achieved from the latest state of that object. Our solution guaranties that semantic in a fully distributed way. With our solution, faulty copies of an object do not affect the work consistency because the owner always has a consistent state for the considered object. The inconsistencies are recovered when required through the ownership protocol. This is a low cost solution where the inconsistencies are recovered uniquely when necessary with a point-to-point protocol.

Parallel working is due to the distribution of the application tree. Thus, two distinct tasks (processing two distinct nodes at the same time) are processed simultaneously without communication between them.

The consistency protocol, corresponding to the ownership transfer, is processed when a peer requests a node that is owned by another peer. Only the requesting peer and the granting peer are involved in the transfer. Thus, the consistency protocol does not limit the parallelism between two distinguish nodes. Moreover, that protocol enables concurrent operations for the same node while preserving the consistency.

Most of the time, the solutions limit the ability of parallel working [Barru96] or do not guaranty the work consistency [Hagsa96][Broll98]. Our solution provides a high level of parallel working while preserving the consistency.

Ordering properties like causal and total ordering [TFC99] introduce higher complexity than a reliable multicast. Ordered multicasts are well known to scale poorly. Moreover, they only capture the causal relationships from the application that are associated with communication causal relationships [Lampo78]. Ordered multicasts introduce high overhead. The basic idea is to circumvent these ordering problems with a reliable and low cost consistency protocol used at the right time.

The system does not use protocols, like the Real Time Protocol [RTP96], that are dedicated to audio or video transmission over best effort network (IPv4 or IPv6). Using such protocols, the application manages the received samples to play them at a regular rate. Thus, the receiver does not produce (audio or video) distortions because of irregular arrivals. This kind of protocol is not applicable in our context because the problem is not to play at a regular rate but to achieve the consistency of work. Real time awareness is not confronted with a sender emitting a flow at a regular rate. So, real time awareness is achieved in straightforward way by using a best effort multicasting.

In contrast with simulations of battlefields or network games, our metaphor of working does not present the problem of a high number of moving objects. So, dead reckoning or solutions to filter the events [Hagsa96] [Broll98] [HLA96] shall not be considered as core protocols and necessary services.

The current solutions rely more or less on a client-server approach. [Hagas96][Broll98] are closed. A primary server and its copies are updated when a participant multicasts events. Copies (also called "Proxies" [Broll98]) are used to reduce the load of the server. But, the primary server maintains the exact state of the system.

Our solution provides scene graph sharing, meeting management, consistency, concurrency, persistence and conciliation in a fully distributed way. The scene graph is distributed without any special server, proxy or relay having a better knowledge than a common peer. A peer does not contact any specific server in order to download the world. The meeting is not managed by any server to maintain the list of participants. Each peer builds and maintains his local knowledge of other participants. Consistency does not involve any lock manager or server. Persistency is achieved in a distributed way with pieces of the global work that are saved into different local stores. Conciliation is a distributed protocol.

That full distribution is a key point of our solution. Thus, the system starts without any administration procedure. The lack of bottleneck allows reaching good performances. The following paragraph describes the core services and protocols.

4. SERVICES AND PROTOCOLS

Let us describe the core services and the associated protocols implementing our metaphor. Though these protocols define an efficient way to implement our solution, other implementations can be proposed to satisfy our metaphor.

4.1. Copy Activation

Using the *Activate()* function, a peer gets a copy of the objects currently present in the scene. First, that function establishes a session membership. Second, it sends the owned objects to the distant participants and recovers the other objects from the distant peers.

Let us describe the first phase of that function.

Each participant is associated with a distinguished name (i.e. *Nicolas Chevassus at AerospatialeMatra*). Moreover, the protocol allocates a session color to each participant.

The entering peer multicasts a request to a class D address [Deer89]. In response, the distant peers multicast their name, network extremity and color. When two peers are conflicting for the same color, the peer with the smallest extremity will change its color. To terminate the membership phase, the entering participant multicasts the chosen color. Thus, each participant builds its local membership knowledge of other participants. Figure 1 shows *Christian* entering concurrently with *Antoine*. The conflicting colors are resolved at the end of *Christian*'s membership phase.

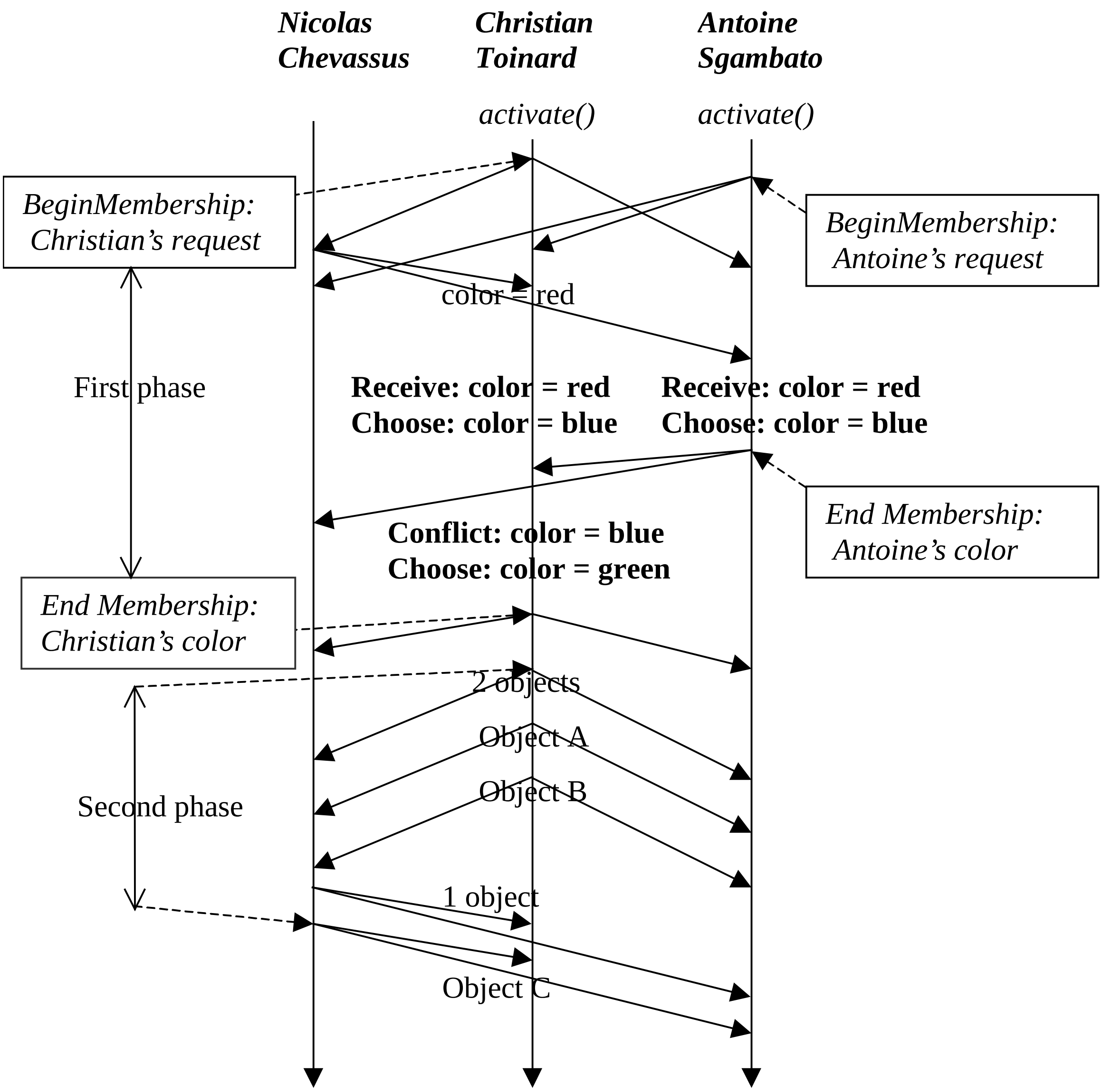

Figure 1

The color assignation is a user-friendly mechanism allowing coloring the objects with the color of their owner. The mechanism can have minor effects like seeing user *Christian* changing from the blue color to the green color during the same meeting.

During the second phase, the entering participant multicasts its objects (number of owned objects followed by each object' state). Thus, he brings his prepared objects into the building site (i.e. in figure 1 *Christian*'s brings two objects). The state of each object contains a set of small attributes like its type, size, and position. The state is transmitted at start-up and shall not be transmitted periodically. In response, a distant participant multicasts his objects (i.e. in figure 1 *Nicolas*'s replies with object *C*). Thus, the new participant gets a copy of the objects owned by the distant participants. Moreover, multicasting enables the other participants to re-synchronize their state with each other's during the entering of the new participant.

4.2. Object Modification

Multicast efficiency is widely adopted within the distributed virtual environments. But, our solution includes several improvements.

The owner peer modifies one object using a request *modifObject(reference& obj)* that multicasts the new state to the distant peers. Each modification increments a local version number for that object. The version number is included into the object state. Thus, the version is transmitted to the distant peers.

A receiving peer updates its copy with the new state. A recipient ignores the received version if the current version is newer. Thus, an old state can not overwrite a newer one.

That update message synchronizes all the receiving peers with the new version. The transmission is as short as possible because multicast requires less transmission time and bandwidth than sending a point-to-point message to each participant.

In order to reduce the processing time, a peer ignores a pending version when a newer one is arrived. Thus, a peer speeds up the recent versions by throwing away the old ones.

Missing state are not recovered. It is not necessary for a distant participant to observe all the state of a given object because the owner maintains the correct state. An inconsistency is recovered when required with the ownership transfer.

4.3. Ownership Transmission

To modify an object, a peer must first be the owner of that object. A peer requests the ownership of a given object using *requestTransfer(reference& obj)*. That function multicasts a request to locate the owner. In figure 2, *Christian*'s peer processes a *requestTransfer* before enabling the update of A. The owner sends a point-to-point reply to the requester. The reply contains the correct state of the requested object. The ownership transmission must be reliable. For that purpose, a local number (i.e. number *X*) is associated with the request. The owner replies with the same number. When receiving the reply, the requester sends an acknowledgment to the granting peer. That third message contains the same number. The requester resents his request in absence of response. The granting peer resents the reply in absence of acknowledgment. That acknowledgment terminates the transfer. Faulty situations, where an object is without any owner due to a transmission error, are avoided.

Thus, a user recovers the latest state of the requested object before modifying that object. In figure 2, the user interaction occurs on a consistent state at the end of the transfer. Ownership transfer is a low cost protocol which requires three messages and which is processed at the right time. In contrast with a reliable multicast, it does not require recovering the errors in a continuous way and preserves the work consistency.

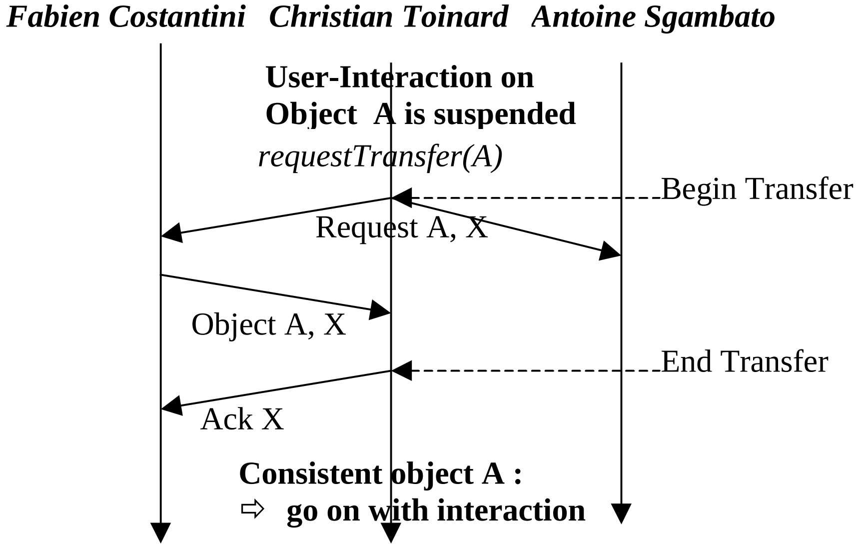

Figure 2

4.4. Re-synchronization

A peer loosing the modifications during a faulty period is de-synchronized. It uses the function *askStates()* to ask the distant peers to multicast their current state. Thus, the requesting peer recovers a state from the different owners. At the same time, the other peers can use the multicast messages to re-synchronize their copy with fresher values.

A peer can decide on its own to use the re-synchronization function (for example to re-synchronize periodically its copy). Most of the time, it is the human user that decides to be refreshed and fires the call of the re-synchronization function.

4.5. Security

The solution can be integrated within a Virtual Private Network, through IP Security [IPSec98], in order to guaranty authentication, confidentiality and integrity.

These basic security functions can be completed or even replaced by other protocols.

Let us describe a protocol. Each peer caches the current owner. Thus, a requester checks that the reply comes from the cached owner. Otherwise, the requester multicasts a security control to verify the ownership. With multiple replies, a malicious peer is detected and voting is used to reach a Byzantine agreement.

Different standardized solutions can be integrated to achieve security properties.

4.6. Membership Control

Our services support "loosely control session" in the sense that each peer maintains a session membership but does not explicitly check membership permissions. This functionality can be provided by a separate control protocol like SDP (Session Description Protocol) [SDP98] or any GroupWare toolkit that manages private sessions. In replacement or complement, a specific control mechanism can also be integrated in our core services for checking permissions before sending object states.

5. CONCLUSION

The paper presents core services and protocols supporting the Distributed Building Site Metaphor. These services authorize different participants to start a session by bringing in their own works. These different entering works compose the global scene. The global scene is modified in a shared way and is distributed beyond the participants when leaving the meeting.

In contrast with the other solutions, our services support parallel working while preserving the work consistency. These services do not require a reliable multicasting. Real time awareness is supported with best effort services. Thus, the core services define a lightweight and efficient solution over standard Internet Protocols. The defined services are fully distributed. Persistency and conciliation services can be built in a distributed way using the core services.

Currently, these core services have been integrated within an existing application. That collaborative application satisfies DBSM. It is deployed at Aerospatiale-Matra for collaborative design at the early stage of the concept phase.

REFERENCES

[Barru96] Barrus,J.W., Waters,C., Anderson,D.B. 1996. *Locales :Supporting Large Multiuser Virtual Environments*, IEEE Computer Graphics and Application. November, Vol.16, No.6, pp.50-57.

[Broll98] Broll,W. 1998. *DWTP-An Internet Protocol For Shared Virtual World*, International Symposium on the Virtual Reality Modeling Language VRML'98, ACM SIGGRAPH, conference proceedings, pp.49-56.

[Deer89] Deering, S.E. 1989. *Host Extensions for IP Multicasting*, Request For Comments 1112.

[e-VIS99] EAI. 1999. *E-VIS Internet Enabled Collaboration for Manufacturing – Executive White Paper*, http://www.e-ViS.com.

[Green96] Roseman, M. and Greenberg, S. 1996. Building real time groupware with GroupKit, a groupware toolkit. ACM Transactions on Computer Human Interaction, , ACM Press, 3(1), p66-106.

[HLA97] Defense Modeling and Simulation Office. 1997. *HLA Data Distribution Management Design Document Version 0.5*, February, U.S. Department of Defense, Washington D.C., http://www.dmso.mil/project/hla.

[Hagsa96] Hagsand,O. 1996. *Interactive Multiusers VEs in the DIVE system*, IEEE Multimedia, Vol.3, No.1, pp.30-39.

[IBMDa98] IBM/Dassault Systèmes CATIA Marketing Team. 1998. *4D Navigator,* http://www.catia.ibm.com/prodinfo/

[IPSec98] Thayer,R., Doraswamy, N., Glenn, R. 1998. *IP Security Document Roadmap*, Request For Comments 2411.

[Lampo78] Lamport, L. 1978. *Time, Clocks and the ordering of events in a distributed system*. Comm. ACM 21, July, 558-565.

[Leight97] Leigh,J., Johnson,A.E., Defanti,T.A. 1997. *Issues in the Design of a Flexible Distributed Architecture for Supporting Persistence and Interoperability in Collaborative Virtual Environments* Supercomputing, conference proceedings, pp.15-21.

[Maced95] Macedonia,M.R., Zyda,M.J., Pratt,D.R., Brutzman,D.P., Barham,P.T. 1995. *Exploiting Reality with Multicast Groups*, IEEE Computer Graphics and Applications, Vol.15, No.5, pp.38-45.

[Netsc98] Netscape. 1998. *Netscape Conference*, http://home.netscape.com/communicator/conference/v4.0/index.html.

[RTP96] Casner,S., Frederick,R., Jacobson,V. 1996. *RTP: A Transport Protocol for Real-Time Applications*, Request For Comments 1889.

[SDP98] Handley,M., Jacobson,V. 1998. *SDP: Session Description Protocol*, Request For Comments 2327.

[TC98] Toinard,C., Chevassus,N. 1998. *Virtual world objects for real-time cooperative design of manufacturing systems*, Workshop European Conference on Object Oriented Programming, Lecture Notes in Computer Sciences, Object Oriented Technology ECOOP'98 Workshop Reader, Springer Verlag, conference proceedings, pp. 525-528.

[TCS99] Toinard,C., Chevassus,N., Sgambato,A. 1999. *Collaborative Virtual Reality and the Distributed Building Site Metaphor*, WSCG'99, The 7-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media'99, conference proceedings, pp. SP/111-117.

[TFC99] Toinard,C., Florin,G., Carrez,C. 1999. *A Formal Method to Prove Ordering Properties of Multicast Systems,* ACM Operating Systems Review, Vol. 33, No.4, pp.75-89.