Chapter XIV

Collaboration and Virtual Early Prototyping Using the Distributed Building Site Metaphor

Fabien Costantini and Christian Toinard
CEDRIC, France

Rapid Prototyping within a virtual environment offers new possibilities of working. In order to reduce the design time and define better specifications, concurrent engineering shall be addressed at the early stage of a concept phase. The presentation defines different concepts namely Virtual Prototyping, Rapid Prototyping and Virtual Early Prototyping. VEP improves the Rapid Prototyping providing important leverage on the problem of collaboration.

The state of art shows that the current solutions offer a limited collaboration. Within the context of an extended team, the solutions do not address how to move easily from one style of working to another one. They do not define how to manage the rapid design of a complex product. Moreover, the different propositions suffer mainly from the client-server approach that is inefficient in many ways and limits the openness of the system. One explains also that the Internet protocols are best suited to develop collaborative services within a VEP system. That state of art enables to explain that CORBA, MPEG-4 and multimedia protocols are not adapted to solve the problem of collaboration.

A case study is used to show that our solution enables an efficient collaboration. The paper presents a global methodology enabling different styles of work. Thus, an extended team manages easily the concurrent design of a complex product. A way to start a project among a geographically dispersed team is proposed. It enables to manage different design teams in a secure way over the Internet. Afterwards, the different teams reach a kick-off meeting to set-up the initial proposal of the specification. Then, each team works on a system design in a distributed and collaborative way. Thus, private works are merged and consolidated easily. Work reviews solve the interdependencies between the different

systems. At last, a project review enables to conciliate the different proposals into a satisfying solution. Our solution provides practical advantages namely a design in context to avoid speculation by default, a method breaking down the complexity, a decrease of the design time and an increase of the quality.

The case study is derived into a set of general requirements for a VEP tool. Thus, the major functional services are identified.

Afterwards, the paper presents a new solution to satisfy the VEP requirements. It proposes new collaboration services that can be used to distribute a virtual scene between the designers. Our solution, called the Distributed Building Site Metaphor, enables project management, meeting management, parallel working, disconnected work and meeting work, real time validation, real time modification, real time conciliation, real time awareness, easy motion between these styles of work, consistency, security and persistency.

In contrast with the other solutions, our services enable parallel work while preserving consistency. These services do not require or implement a reliable multicasting. They are fully distributed and do not require any specific quality of service from the under-laying network. DBSM can add collaboration to any standalone application.

INTRODUCTION

Motivation

Virtual Early Prototyping enables concurrent engineering at the early stage of a product design. Most of the time, an important part of a product cost is charged during the design phase. But tools to reduce the time to design a product and to examine different design alternatives are missing. VEP addresses the concept phase of a product development where most of the design choices are made. So, the concept phase is critical for the quality and the cost of the product. A VEP tool must authorize different designers to make quickly the best virtual prototype through a large set of design alternatives. So, the workers must work to explore quickly and easily different design alternatives. The concept design activities of a complex system can be performed in project-oriented teams that have important and tight deadlines. Concept design is not limited to a single enterprise. Members of a global design force can include partners, suppliers, contractors and customers. That global design force is organized into several design teams responsible for the different systems. The best specifications must be defined by these different teams through successive iterations while mixing easily individual works, alternatives for the different systems and conciliation milestone to reach a global specification. That extended team must collaborate freely and easily without forcing a given enterprise to have more privilege and information than others. At the same time, the design environment must provide the ability to respect the responsibilities and expertise of the different members.

Collaborative systems focus on natural social interactions that let people easily move between different styles of work. They propose metaphors like [Greenberg, 1998] that ease people's transitions across the different styles of work. Generally, the room metaphor is considered as a container for several documents and space where workers meet. Thus, workers easily move from one room to another and can introduce documents in the different rooms. But, these collaboration metaphors do not address the way designers can collaborate at the concept phase of complex engineering products. Moreover, these collaborative systems do not manage virtual shared worlds.

Industrial solutions like [Dassault, 1998] start from a 3D representation of the product. They authorize mainly a design review that is a limited style of collaboration. Some like [Hewlett-Packard, 1999] offer a direct modification of the shared 3D world but they use a central server to achieve a consistent update. These solutions are not easy to use since they do not address clearly how the designers can cooperate freely and efficiently. For example, they do not support mobile working, iterative validation and conciliation. Moreover, a central server is not an open solution because it requires a unique enterprise to collect the shared world. Centralization has also some usability drawbacks since each participant must have an account and sufficient right before being able to work. Thus, a central management deprived the participant of their know-how, data management and rights. Moreover, these solutions do not support an easy motion between different styles of work. For example, it is not simple for mobile workers to prepare proposals for a global task before reaching meeting to conciliate the different proposals in real time. Generally, current solutions address a single way of working which is the meeting and tackle mainly the design review.

Thus, metaphors are missing to achieve concurrent engineering at the early phase of a complex industrial product. First, collaborative models do not tackle virtual prototyping. Second, Computer Aided Design (CAD) systems only support a limited style of collaboration.

On the other side, the distributed virtual environments are guided by the historical requirements coming from the simulation of battlefields. Thus, [Barrus, 1996] [Defense Modeling and Simulation Office, 1997] [Broll, 1998] [Hagsand, 1996] [Madedonia, 1995] consider mainly how reducing the network traffic due to a high number of moving objects. Generally, a way to divide statically or dynamical the scene is provided for scalability. These systems do not support parallel working [Barrus, 1996] or do not guaranty the work consistency [Defense Modeling and Simulation Office, 1997] [Broll, 1998] [Hagsand, 1996]. Moreover, client-server pitfalls [Broll, 1998] [Hagsand, 1998] or quality of service requirements [Defense Modeling and Simulation Office, 1997] limits these solutions.

Thus, adapted distributed virtual environments are missing. First, despite they claim to be, the current environments do not address the concurrent engineering requirements because they do not preserve the consistency or limit the ability of parallel working. Second, they do not address how to move easily from different styles of work and to support different kinds of collaboration and conciliation. Third, a central server introduces a bottleneck and a failure point in the system. At last, the different Quality of Service (QoS) approaches do not inter-operate and are yet poorly deployed on the Internet. So, requiring a specific QoS from the under-laying network forces the participants to share the same network architecture and limit the ease of deployment and the openness of the solution.

As the industrial solutions and propositions from the literature do not address the collaboration services well at the early stage of a product design, this paper defines the Virtual Early Prototyping requirements. It shows that the Distributed Building Site Metaphor (DBSM) provides a good solution that satisfies these requirements. Our metaphor enables a real-time collaborative design and parallel work within a virtual shared world while preserving the consistency of the design when multiple participants are working on simultaneously. It enables different styles of work like mobile working on a global design, meeting, real time design, real time review, real time conciliation and real time validation. DBSM can be used for both co-located and distributed work. It is a fully distributed approach without any central server. The solution does not require any specific quality of service from the under-laying network. It can be deployed over the standard Internet architecture while providing a robust authentication and confidentiality. As the solution consumes a reduced bandwidth, low links can provide a real time work.

Chapter Organization

Section *BACKGROUND* provides definitions and discussions of the collaborative virtual environments. It incorporates a state of art that gives an up-to-date overview of both 3D collaborative systems and distributed virtual environments. This section explains also why existing communication standards like CORBA [Object Management Group, 1999], MPEG-4 [Eleftheriadis, 1998] and real time protocols cannot help developing collaborative services for a VEP system. It shows however that GroupWare tools and multimedia environments can be used as complementary technologies during distributed sessions.

Section *USE CASE IN AN EXTENDED TEAM* presents the style of collaboration that is supported by our VEP solution. It shows how the participants set-up a design project. Different alternatives will be design concurrently starting from an elementary proposal. Participants resolve conflicts through a day-to-day consolidation.

Section *REQUIREMENTS* defines a complete set of requirements for collaborative VEP environments. These requirements address functional needs like project management, meeting management and shared world consistency but also non-functional constraints like the ease of deployment over the Internet and heterogeneity support.

Section *THE DISTRIBUTED BUILDING SITE METAPHOR* describes DBSM that is a fully distributed solution. It uses regular email during the initial phase of a project establishment. Afterwards, the collaboration services are fully distributed using only multicast communications that are available on any standard Internet Protocol (IP) suite. DBSM is thus fully supported by the Internet architecture.

Section *A DESIGN PATTERN APPROACH FOR DBSM IMPLEMENTATION* introduces how DBSM can be implemented using Design Patterns. More precisely, it focuses on two important patterns solving respectively distributed designation and automatic scene aggregation problems.

Finally, the section *CONCLUSION* gives the lessons learned and further directions to improve the solution.

BACKGROUND

Definitions

Virtual Prototyping

It is a way of designing a manufacturing product using a CAD system. These systems are mainly devoted to have a 3D representation of the product. Each piece of the product is finely drawn showing any internal detail of that piece. These systems enable a high numerical precision of the 3D prototype. Using a CAD system requires a long-term training. Moreover, designing a product with a high numerical precision is a long-term activity. So, a time consuming design is the price you should pay to have a precise model and 3D representation of your product. The major CAD system is CATIA from Dassault Systems. It is widely used in different branches of activities like aeronautic or automotive building.

Rapid Prototyping

It is a pre-design of a manufacturing product during the concept phase of an industrial product. A Rapid Prototyping system authorizes a 3D representation of the product. In

contrast with Virtual Prototyping, a designer does not make yet a precise prototype. At this stage, Rapid Prototyping makes the main pieces of the product emerging only gradually as his ideas evolve. The different pieces of the product are roughly defined. At this stage, a designer does not think to the internal details of each piece of his product. In contrast with a CAD system, a Rapid Prototyping system is easy to learn. Moreover, defining a rough model requires only a couple of hours.

Virtual Early Prototyping

It is an improvement of the Rapid Prototyping concept. Virtual Early Prototyping occurs also during the concept of a product development. It is another tool to make a 3D representation of the developed product with the same functionality than a Rapid Prototyping system. In contrast, Virtual Early Prototyping offers a better level of design and high degree of interaction. For example, CoCreate OneSpace [Hewlett-Packard, 1999] is a basic virtual early prototyping. Thus, time to design decreases while the quality of the specification increases.

First, it enables to evaluate advanced characteristics of the product at the early stage of the development. For example, a tool like CATIA [Dassault, 2000] checks interference between components. It can propose automatically modifications to respect these functional rules. Thus, the system goes one step further in the designer's assistance.

Second, the human-machine interactions are easy than ever with an ergonomic interface that is adapted to the major operations carried out by a designer. Thus, the system is not a general-purpose tool with inadequate services but proposes the best practices to the designer. For example, best practices for the air conditioning design is a widget enabling to route automatically the air conditioner with the different air vents and compute the flow (i.e. debit and pressure) at each air vent.

Third, human-to-human interactions are supported. Different designers with a specific knowledge participate and debate the baselines of the future product. Here, the different participants examine several alternatives in order to achieve the best specifications taking into account the expertise of each designer and satisfying the major requirements of the product. For example, electricians and hydraulicians collaborate to design a new product. Different teams are formed to design in parallel the different systems in accordance with the product requirements. Each designer can propose different alternatives (i.e. 3D objects) connected with others' work (i.e. other 3D objects). These different alternatives will be used to define the best specification. The different works are related with each other by day-to-day consolidation. A validation and conciliation phase enables to respect the global requirements. For example, the electrician respects minimal distances with the fuel system.

Human-Human interactions take place within a 3D scene that is managed through a virtual environment. Thus, Human-Human interactions can be transported by a scene modification. Typically, a participant creates a new 3D object that is taken into account by the other participants later on. For example, while traveling the electrician uses his laptop to create a new electric system within the 3D scene. Despite that mobile work, the electrician will share the proposed system as soon as possible by initiating a meeting with the concerned people. Thus, other designers will work in accordance with that new electric system.

Human-Human interactions are easy to use and support different styles of work. The system does not give any unnecessary rights to a given participant even if he is the leader of the project. Thus, the expertise and property rights of the extended team are respected.

State of the Art and Justification of the Approach

Different aspects are considered in this section. First, industrial prototyping systems are briefly described in order to show the proposed services and their implementation. Second, different solutions of distributed virtual reality are presented. As a VEP system must provide efficient Human-Human interactions and collaboration within a 3D world, it is important to see if propositions from the literature can be used to develop the system. Third, an overview of different communication standards show which ones are best suited to develop a VEP system.

Throughout this section, emphasis is put on services that are missing for a VEP tool. Thus, the reader has a first level of understanding of the required services. At the end of the section, a justification of our solution is given with contrast to the other proposals.

Industrial Prototyping Systems

CATIA 4D Navigator

[Dassault, 1998] provides a collaborative navigation of the scene through synchronization of viewpoints, telepointers and annotations. Thus, a simple form of design review is achieved. Typically, the designers share one viewpoint. One designer actively pilots the viewpoint throughout the 3D scene to conduct a visual inspection of the work. The passive designers fly through the scene as controlled by the pilot. Thus, the system allows simultaneous viewing of the scene by multiple participants. Each participant has a telepointer that is transmitted to the distant participant. Basically, a telepointer enables to locate the corresponding participant within the scene. Each participant can use his telepointer to show a direction or make an annotation. Thus, the participants share the same scene, the same viewpoint and observe the different pointers. 4D Navigator provides the ability to make notes, request design modifications or ask questions through an online redlining capability.

GroupWare toolkits like Microsoft NetMeeting can be used to run 4D Navigator. Thus, the designers can make a videoconference during the 4D Navigator session. So, they talk with each other using the voice transmission of the GroupWare toolkit. Typically, voice helps to comment on the annotations achieved during the review.

On a communication point-of-view, 4D Navigator uses a server that broadcasts the scene and position of the viewpoint to each participant. So, this is a client-server architecture without any advanced distribution mechanism.

This product is limited to design review as one designer cannot modify directly the 3D scene (i.e. adding a new 3D object) with distant designers having a real time awareness of the modifications. There is no possibility for a given worker to make a private improvement of the scene before adding the modifications to the shared scene. Thus, mobile design of an extended team is not supported. Real-time conciliation of the shared scene is not supported since real-time modification of the shared scene is not permitted.

CoCreate OneSpace

[Hewlett-Packard, 1999] proposes better collaboration services. Like [Dassault, 1998], designers make distributed review meetings. On the same way, shared viewpoints and pointers are proposed and the system enables a simultaneous viewing.

The major difference is the possibility to modify on-the-fly the scene. For example, one participant creates a new 3D object by activating a modification command. The characteristics of the newly created object are sent to all connected participants in the form of a

graphical update. This incremental update guarantees that all participants see the results of the modification immediately.

The communications use a client-server architecture. After the initial upload of the scene, the participating stations render the 3D scene locally. Typically, a station sends a command to the server. The server processes the command and sends an incremental update to the different stations. The network traffic is reduced but the server has to compute the update that has to be sent. So, the load of the server can be high if each participant sent a big number of commands.

The server has to be administrated. In case of an extended team, the server shall be hosted in a central entity and the different partners must ask the central administrator access rights. So, this solution introduces set-up and administration costs. It is not feasible to have a fully distributed team as the hosting and administration is devoted to a single entity.

Mobile working and easy merging of the different works are not addressed. So, the extended team cannot alternate easily between mobile working and sharing process. So, the working phase is only supported during a connection phase with the central server. So, the designers become dependant of the connection ability and a disconnected work can be easily integrated within the shared scene.

Distributed Virtual Reality

Locales

[Barrus, 1996] allows combining different rooms into a global world. The combination relies on transformation matrixes in order to connect adjacent rooms that present communication doors between them. A room is called a locale. Thus, a user located in a room can observe the events of an adjacent room through the door. The transformation matrix allows representing the neighboring rooms and moving smoothly from one room to another. In that context, a neighboring room can be connected dynamically in order to combine the works achieved separately in different rooms. The participants use different locales to work separately. This solution provides three interesting features: graceful motion between neighboring locales, performances are improved through decomposition of the world into chunks that can be processed separately and workers combine separate objects through a dynamic connection of their locales. A server maintains each locale using a multicast address to send the update messages. By opening a network extremity with the corresponding multicast address, a client can efficiently obtain the locale updates. Thus, a client obtains information about locales surrounding the focus of attention without receiving messages from irrelevant locales. As each locale is assigned to a server maintaining its state, it is a centralized solution where the server multicasts the updates.

The locale metaphor suits well to a visit different locales that are connected through doors. A concept design activity cannot easily be mapped onto the locale metaphor since the different pieces of the global scene are not disjoint locales but systems interleaving with each other. So, the locale metaphor is not adapted to the concept design activity. At last, no security is provided.

High Level Architecture

[Defense Modeling and Simulation Office, 1997] addresses the framework of a great number of moving objects. The solution aims at a real time that is as much as possible exact. They provide sophisticated solutions to reduce the number of state transmissions and to

recover the transmission errors. [Defense Modeling and Simulation Office, 1997] defines a dynamic cutting (which is not perfect) according to the motion of the objects. Each moving object defines an area of interest and events falling into that area are observed by the moving object. The difficulty is to change dynamically the area of interest according to the motion of the object. A given object does not have the guaranty to receive all the interesting events.

Moreover, this standard defines different ordering of events that can be used in a simulation context. [Defense Modeling and Simulation Office, 1996] identifies different qualities of ordering. They are mainly interested with a logical ordering of events where the application associates logical dates to the events. That ordering supports a distributed simulation by guaranteeing that the events are processed at the corresponding logical time. These properties do not address a real time simulation or a collaborative activity.

Ownership services are defined. The ownership management allows federates to transfer ownership of object attributes. The ownership transfer is associated with publication and subscription services. Ownership acquisition requires current publication and subscription for the attribute. The standard does not define different levels of ownership like read and write permissions. It is not integrated within a schema where a shared scene is distributed among different designers in order to be processed on a private basis. So, distribution of the shared scene among mobile designers is not supported. In fact, the ownership management concerns less the collaborative design than the distributed simulation. Security is not addressed at all.

HLA does not standardize the protocols 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.

Distributed Interactive Virtual Environment

[Hagsand, 1996] considers moving objects and aims at a precise real time.

Solutions reduce the number of state transmissions. In that context, uniquely fresh events interest the recipient and the system improves the performances relying on that feature. The system uses multicasting heavily and partial replication.

With DIVE, the application has the responsibility of the placement of the copies for the shared world. It must define which object must be present in the different copies. It means that all of the states are not replicated at each copy. A modification is multicast to update the copies that have joined the corresponding address. Consistency is achieved by associated one object to a given owner for a long time. Thus, concurrent requests must wait during that period. In fact, the solution makes the assumption that concurrent modifications seldom occur.

Since partial replication means that none has a complete state of the world, a server must be used to preserve the persistency of the scene. The server receives all the update and writes the changes to a persistent storage.

The first member of a session loads the initial world from the server. A subsequent member requests its initial world state over a multicast address and receives an up-to-date copy from the closest member.

This solution provides a protocol allowing to locate the nearest copy for a given object. When an entity detects a missing event or just requests an object, it multicasts a request. The closest copy replies with its latest version of the object. The latest version of the object is not the freshest one. Another copy can have a better knowledge for the state of that object. When nobody has a better proposition, only the closest copy replies to the request.

The system implements a classical dead-reckoning module, like [IEEE, 1995], in order to keep the number of updates low: between two updates, a receiving entity evaluates the position using linear and angular velocity.

Since a server is required for downloading the scene and for persistency, an extended team cannot use the same tool during a mobile working and the sharing process. Moreover, the solution does not define consistency properties that can support distributed validation and conciliation. Security is not addressed since the solution does not provide authentication and confidentiality.

Distributed World Transfer Protocol

[Broll, 1998] considers a server for: transmission of the virtual world, reliability of the updates, persistency, supervising the connections of participants, supporting consistency. In essence, that solution is close to [Hagsand, 1996]. Proxies manage copies of the shared world to reduce the load of the server. The server and the copies are updated when a peer multicasts events. The system provides a recovery mechanism. Each peer has the ability to receive all the events. When the recovery cannot be achieved, a participant has to reconnect to the server. [Broll, 1998] provides a simple concurrency control achieved through a lock mechanism. The approach is a locking on a per-interaction basis instead of a per-object basis. The server releases the locks within a certain time. That concurrency control suits well for special requirements (like controlling concurrent motions of the objects). [Broll, 1998] integrates solutions to limit the bandwidth that is necessary when working at a VRML level of abstraction. The system improves the performances using cells. The cells define statically a mapping of the world. Each peer has to connect to a central server. The server transmits mapping and world state at the connection time. The solution does not describe how to add dynamically new cells to the world.

As being close to DIVE, this solution suffers from the same drawbacks. An extended team cannot use it to alternate mobile and sharing phases. Consistency is not addressed to support distributed validation and conciliation. The solution does not provide any security.

World2World

[Sense8, 1997] proposes a client-server architecture to distribute and synchronize simulation data. A central server receives all the modification events and forwards them to the other participants. The basic idea is that each update is sent using a point-to-point message over the User Datagram Protocol (UDP). UDP offers a better throughput than the Transmission Control Protocol (TCP). Moreover, the server can make some optimization when receiving two modifications for the same object (i.e. two translations of the same object). In that case, it will only forward the latest modification and discard the first one. This optimization is only possible for the same type of modification (i.e. two translations). The data are structured hierarchically within a server using a tree. Each node in the tree can be associated with a lock. When a client wants to modify a sub-tree associated to a node, it first has to acquire the lock before being able to update that node. The server prohibits all other users from adding, updating or removing properties of that protected sub-tree until the lock is released. Thus, the system guaranties the consistency of the concurrent operations. At last, a client has an update rate. If a client produces more updates than enabled, then the client will sent only the most recent update for each shared property. Thus, the bandwidth is limited and the server is not too much loaded.

In contrast with others, this solution does not use the multicasting of events. So, the server has to send explicitly each update to the different client through a point-to-point message. That solution uses more bandwidth than multicasting. The major advantage of that solution is the optimization that enables a central server. But, this server also introduces a bottleneck in the system and shall be administrated.

The solution does not address the distribution of a shared scene among mobile designers. There is no solution that is provided to merge the mobile works into a global and consistent scene.

Cavern

[Leigh, 1997] addresses design review and collaboration services with immersion in the virtual world. A client uses a server with supercomputing resources and high-performance networking. This centralized solution requires resource reservation and reliable transmission. Through a persistent object server, the system supports persistency. Either intermittent snapshot can be created or entire collaboration experiences can be recorded for later review. 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 collaboration where immersion generates a great volume of data with real-time constraints. The solution aims at loading large scientific data (like video or CAD format). Caches using timestamps improve the performances.

That solution cannot be used easily within an extended team. First, the solution does not address the distribution of the shared scene among mobile workers. Second, security is not considered. At last, supercomputing is not a scalable solution for an extended team that is dispersed among several companies.

Communication Standards

Internet

The Internet Protocol (IP) is the network protocol that is present at each router or station. It supports point-to-point, broadcast and multicast transmissions but is unreliable. At each Internet station, UDP and TCP are the two transport protocols that are available to transmit a message from one process to another. These two transport protocols use the IP services. Using UDP, a process can send a message to a single process (point-to-point), all the process of a unique network (broadcast) or a group of process (multicast) that is disseminated over different networks. These three kinds of communication are fully supported by any Internet station. UDP is unreliable. TCP is reliable but offers a lower throughput.

As depicted during the previous section, Distributed Virtual Environments make an intensive use of multicasting. The reason is that DVE requires that each station has its own copy of the world. That way the station can render each copy of the scene locally. A multicasting packet is the best way to send each event (creation, update, and deletion) to the group of copies. Sending each copy a point-to-point message requires N packet, when one multicast packet suffices. Thus, using a multicast transmission reduces the bandwidth. Moreover, a multicast message can be sent directly by a peer to the multicast group. It does not require a server for relaying the message to the group. Thus, multicasting can be used to avoid a server bottleneck.

CORBA

At that time, the industrial implementations support the CORBA 2 standard [Object Management Group, 1999]. CORBA is mainly a client-server architecture where the client invokes a method on the server. A client cannot invoke a method on a group of servers. CORBA solves the heterogeneity problem between the client and server.

The only way to send a message to a group is using the Event Service. But, this service is inefficient as it needs one method invocation for each recipient and each invocation is carried out using a TCP connection.

CORBA Audio/Video AV Streaming Service specification [Object Management Group, 1997] defines architecture for implementing open distributed multimedia applications. But, in essence it is a new interface to encapsulate multimedia transport protocols. As explained further, multimedia protocols do not correspond to the requirement of collaborative virtual environments. So, CORBA does not suite to Distributed Virtual Reality, as it cannot support multicasting.

Multimedia Transport

[Schulzrinne, 1994] is gaining acceptance as a transfer protocol for streaming audio and video flows over the Internet. It is now widely adopted by GroupWare toolkit like NetMeeting.

The basic idea of multimedia protocols like RTP is to send data using UDP that provides a good throughput. A multimedia protocol is then responsible to associate a date to the samples that are transmitted with UDP. When a reception buffer is full, the receiver can sort the samples according to their emission date to preserve the regularity of audio or video. Thus, the receiver is able to drop the samples without producing long silences (i.e. it does not drop consecutive samples).

In fact, collaborative virtual environments are interested mainly to find adapted solutions in case of a great number of participants. They require also preserving the consistency of the object. Playing samples at a regular rate does not reply to these requirements.

Anyway, multimedia protocols and GroupWare toolkits are communication tools authorizing designers to talk with each other during a session of collaborative virtual prototyping. Thus, the collaborative virtual environment supports real time modeling and advanced collaboration styles of design while the GroupWare product allows to interact more naturally despite distance barriers.

MPEG-4

[Eleftheriadis, 1998] is a client-server architecture. It is devoted to the transmission of an audiovisual scene where 3D objects, audio and video are transmitted. Synchronization information is transmitted at the same time. At the server side, audiovisual scene information is compressed, supplemented with synchronization information. The server passes the scene to a delivery layer that multiplexes it in one or more coded binary streams. At the receiver side these streams are de-multiplexed and decompressed. The media objects are composed according to the scene description and synchronization information and presented to the end user. The end user may interact with the presentation. Interactions are processed locally or transmitted to the server. The main objective is to allow a server to control how the receiver will behave in terms of buffer management and synchronization when rendering the audiovisual sequences. The model allows the sender to specify when information is removed

from these buffers and schedule data transmission so that overflow does not occur.

Despite some interactions can be sent to the server, the system is not specifically designed to enable the user to modify the data on the server. Moreover, if one user wants to send information to other users the interaction must be sent to server and processed before being forwarded to the distant users. At last, the system does not permit to a user to manage access rights and put locks on nodes to solve concurrent updates.

Justification of our solution

DBSM is an open platform that enables the participants to move easily from a private work (i.e. a disconnected phase) to a virtual building site aiming to collaborate on a shared global scene. That global scene is distributed dynamically among the different workers to improve the virtual building site and to propose different design alternatives. The resulting work is disseminated within the different private workspaces when leaving the building site. The global scene remains accessible when entering latter on in the building site. That way, the different workers in the building site can form a conciliation meeting. Moreover, mobile designers do not need any network access to improve a subset of the shared scene. Before entering again in the building site, a mobile worker selects the new proposal he wants to share with the other participants. Thus, an update of the virtual building site is carried out through an automatic merging of the different proposals.

The virtual building site can be compared to the building of a house. When a worker is in the building site, he observes the evolutions carried out by the other workers. At the same time, he enters new assemblies into the building site to adjust them in accordance with the other elements. The building site is the place for real time cooperation, validation and conciliation to respect the specification of the house. The main difference with a real building site is that a designer brought out at home the virtual building site to improve his elements or define new assemblies.

The solution authorizes different styles of working.

First, the designers prepare their work before reaching the building site, as any reasonable worker would do. This disconnected work improves a subset of the global scene that was collected during a previous meeting.

Second, they collaborate in real time by modifying, discussing alternative proposals during a meeting. The global scene reforms automatically through implicit relationships. Despite the distribution of the global scene, our solution guaranties a consistent progression of the work. Thus, the designers do not spend time to rebuild previous results.

Third, this metaphor can be used to perform a co-located work where participants are in the same room and discuss directly while observing a copy of the global scene on their machine. Moreover, a distributed work, where participants are physically at different sites, is enabled. Thus, participants discuss through chatting facilities and GroupWare toolkits while interacting in real time on the global shared scene.

Fourth, parallel working is fully supported in the different styles of work. During a preparation phase, parallel work is achieved without any communication. During a meeting, parallel work is achieved while preserving the work consistency.

At last, different styles of validation, conciliation and responsibility sharing are supported. Designers share the responsibility of the different systems. Thus, each participant controls easily how others can modify his work. The participants can build and juxtapose different alternatives to achieve real time conciliation. Real time modification can be also help to make the conciliation. Each designer can run a validation tool that processes the local copy of the shared scene. Since, each designer has a fresh and consistent copy the validation

tool has the guaranty to process an accurate and correct scene. A precise consistency property is provided to support validation and conciliation. Thus, the designers define in real time the best specifications while preserving others' responsibilities.

The next section describes a case study where DBSM is used for virtual early prototyping of a new aircraft. The design of a new aircraft involves several teams to design eleven systems. That case study talks you through our solution for the concurrent engineering of aircraft systems. The time to design the aircraft is reduced because of parallel work, validation facilities and usability of the tool. The quality of the aircraft is increased since a wide range of alternatives can be defined, examined and conciliated.

A major point of our solution is to be fully distributed through standard best effort services. Thus, it is deployed easily on any IP (Internet Protocol) network infrastructure. It does not require any specific quality from the under-laying network like resource or bandwidth reservation. In contrast with [Broll, 1998] [Hagsand, 1996], our solution implement neither a reliable multicast nor an ordered multicast [Toinard, 1999] (i.e. a causal and total order) that are well known to scale poorly. Moreover, a reliable or ordered multicast does not guaranty the work consistency. Logical ordering used in parallel simulations is useless as it addresses reproducibility. Concurrent design is not confronted with reproducibility but with parallel working and consistency.

Here, parallel working is not limited and consistency is achieved using a lightweight protocol that is processed uniquely at required time. 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. As the ownership transfer runs over an unreliable transport (UDP Unreliable Datagram Protocol), one multicast and two point-to-point acknowledgments are required. In contrast with a reliable multicast, ownership transfer is a low cost protocol that does not recover the errors in a continuous way and preserves the work consistency. If required (typically on user demand) a peer resynchronizes its copy with the current states of the distant nodes to get a fresh copy of the global scene.

At last, our solution can be deployed securely over the Internet. It uses standardized authentication tools that are embedded in any email client. A project can be set up and meetings are scheduled using email. Email serves also to distribute a session key. Thus, confidentiality is achieved using that session key.

So, our solution enables a straightforward and lightweight implementation on any standard workstation.

USE CASE IN AN EXTENDED TEAM

To design the different systems of the virtual prototype, several teams are involved. These teams may be geographically distributed among several countries and companies. Thus, several partners including contractors and suppliers cooperate at the design phase. The case study shows the style of working and organization that is typically supported by our metaphor. It describes that the designers start working from an initial proposal and converge towards the optimal specification through several iterations.

Suppose for example that three teams are disseminated at different sites. The project coordinator is at site 1. Electrical and hydraulic teams are located also at site 2 and 3.

Project Start-Up

In order to initiate the design, the project coordinator sends an email to each team manager including the project definition, the role of each team, the milestone dates and other organizational data. Each team manager sends an email to reply and accepts the definition, the role and the organizational proposal. When the project coordinator receives all the replies, he sends a last email to confirm the start-up of the project.

Since emails are transmitted between the different sites and teams using the standard Internet, security must be guarantied. Here, S/MIME enables each team manager to authenticate the project coordinator. In turn, the project coordinator authenticates each team manager when he receives the reply. Confidentiality of the exchanged data is also provided in a standard way through S/MIME.

Kick-Off Meeting

The second phase is to schedule a kick-off meeting where the project manager will distribute the initial virtual prototype which can be an empty shape that the designers fills throughout the design process. For example, the initial virtual prototype can define the shape of a new car. It would have no practical sense to change the shape since it does not really evolve during the life of the car. Thus, automotive designers can fill a given shape/volume with a motor, sites, gears, air-conditioned and so on.

The project coordinator sends an email to propose a kick-off meeting. Each team manager acknowledges the proposed schedule. Scheduling the meeting can require to exchange several emails. At last, the coordinator sends a final email to confirm the negotiated schedule. That final email includes the communication address and encryption data that permit a straightforward and secure meeting. Thus, each team member can invite securely another team member to the scheduled meeting through email transmission of the received data.

It is practical management architecture since generally the team members can change and each team manager wants to decide which member must participate to the meeting. It authorizes a decentralized management: the project coordinator controls which team can participate and each team manager controls independently the participation rights for his team members.

Unauthorized person cannot reach the meeting since they do not receive the appropriate communication address and protection key. Even if an unauthorized person succeeds to listen the meeting address, she cannot use the encrypted data.

That security mechanism is simple to use, since for the end user point-of-view it relies only on a widely standardized secure email. Thus, the end user does not have any specific system to learn. Moreover, it increases the usability of the solution since a mobile designer can use the system from any unprotected communication channel. Thus, a participant can reach securely the meeting all over the world from any available network extremity.

After that scheduling phase, any invited designer can reach the meeting independently at the scheduled time. For that purpose, he uses a menu of the VEP tool to select the corresponding entry of his agenda. The VEP tool establishes automatically the communication channel with the distant participants. Since the system relies mainly on multicasting, the end user does not have to connect to any server in order to reach the meeting. That communication establishment simplifies the interface since a designer enters into the meeting just by knocking at the scheduled door without having to know a server address.

For the kick-off meeting, all the participants of a given site are collocated in the same room. Thus, they can talk directly with each other. To talk with a distant team, each team manager typically starts a NetMeeting session with the distant manager. The distant network extremity is transmitted using the VEP tool by creating a chat object within the shared world. Thus, the basic problem to discover the network extremity of a distant NetMeeting session is easily resolved through the VEP tool. It is a very practical way to establish a NetMeeting session. Thus, the teams benefit at the same time from video-conferencing and scene sharing facilities. When the project coordinator enters into the meeting, he selects the initial prototype he wants to introduce within the shared scene. Thus, any participant gets a copy of the initial structure he will have to work on. During the kick-off meeting, the project coordinator transmits the requirements of the aircraft as direct interactions on the shared scene but also as NetMeeting data. These requirements can be discussed in real-time through the scene sharing and NetMeeting session. The different teams work in real-time on the shared scene to allocate the space of the major systems whose they are in responsible. At the end of this first meeting the designers have the requirements and a first level of specification. Each participant leaves the meeting by storing locally his copy of the shared scene. Thus, he brings at home the environment (the initial prototype plus the distant systems) and his own systems. Typically, he will use his local store latter on a disconnected basis to improve his work in accordance with the requirements and initial specification.

It is a very fast and efficient way to cooperate.

First, the different participants do not physically move to a common location to reach the meeting. They simply reach the virtual building site from their regular working location. A team member can even reach the building site from any temporary location where he currently moved. Thus, a meeting has not to be canceled or delayed because of participants that are traveling.

Second, they work in real-time during that first meeting to directly define and share a first iteration of specification. Since the system is easy to use and the scene directly shared, the delay to transmit and share a first level of iteration is reduced. As each participant observes in real-time the distant interactions, that first iteration is a design context. Thus, the first iteration does not contain speculation about the distant intentions and choices.

Third, that first iteration respects responsibilities of each other since the interaction respects the role devoted to each team. For example, the electrical team cannot create a regular hydraulic system since that operation is permitted only to the hydraulic team. In fact, for proposing alternatives to a given problem (i.e. a fuel system located close to an electrical system), the VEP tool can manage exceptions like an electrician creating an hydraulic system. In that case, the resulting system is displayed with a special rendering (i.e. use of a zebra texture to distinguish the exceptional system).

Fourth, the operations (creation, update and deletion of object) are carried out in a consistent way according to the protections of the associated object. That property has a practical impact on the progression and the correctness of the design. Thus, a work has not to be redone because of a synchronization problem between two parallel workers. Moreover, that property avoids an inconsistent modification of the considered object.

Distributed Design

Within the distributed organization, each team designs independently the system whose it is responsible.

For example, the air-conditioned team uses the result of the first iteration to improve the air-conditioned by adding new ventilation routes. This operation is carried out on the global scene resulting from the first iteration. Thus, each participant (or team) can work on a disconnected basis without communication with the other participants (or teams). That distributed design relies on a human center approach where private work provides the baseline of further real-time interactions and discussions. Thus, a teamwork enables to complete an individual work.

This approach presents several practical advantages. First, it is an efficient collaboration principle since the designers shall not work at the same time on the same project to achieve their work. Second, it increases the productivity since a designer can efficiently prepare same work outside the virtual building site without any external perturbation. Third, it is a very natural and efficient approach of collaboration where the private work serves as a baseline for the team cooperation.

Assume that the air-conditioned team built a new air-conditioned system during a previous meeting based upon the first iteration. Afterwards, each member prepared several improvements on a private basis.

Then, the air-conditioned team participates to a collocated meeting to join the private works. Thus, each member introduces his work. These different private works are used to reform automatically the improved system. Each participant gets a copy of that new proposal on his laptop. During that meeting, the participants consolidate the proposal to introduce it latter as partial specification for the second global iteration.

Since the authorized designer has moved the air-conditioner, the systems designed by the other members are no more connected to the air-conditioner. Different design choices are possible to resolve the problem (i.e. move back the air conditioner, update the other systems). As the team is aware that the proposal could also conflict with other systems (i.e. that have not been introduced into the meeting), they prepare two design alternatives. The first one includes the new air-conditioner position and the second one corresponds to the previous position of the air-conditioner. Thus, the team has consolidated the proposal by producing two distinct alternatives.

Each participant can check interference between components using a validation tool like proposed in [Dassault, 2000]. The validation tool can process the local copy of that participant since his copy contains all the network data and information required by the interference checker. The VEP tool displays the results of the flow simulation as shared objects for mutual awareness. It is a very powerful property since each team guaranties valid proposals (i.e. where, for example interferences are detected and solved in real-time).

So, using DBSM present several advantages.

First, a team proposes a correct system where inconsistencies between the private contributions of the team members have been resolved and requirements are satisfied. Thus, a future conciliation phase is shortened since the global conciliation will only check that new entries are correct.

Second, each consolidation and validation becomes independent of possible inconsistencies in the distant systems. Thus, despite the interdependencies between the different systems it becomes feasible to achieve a partial validation reducing the range of control to only one system. Without partitioning between the concurrent evolutions of the different systems, it could be completely impossible to do a partial validation of each system. Moreover, a global validation would become even more difficult then.

Third, partitioning is an efficient specification approach to design a complex system since it is impossible for a designer to take into account all the interdependencies in real-time. Thus, a team focuses on his problem without being flooded by other constraints.

Fourth, day-to-day consolidation and validation enable to update the system with a private work as soon as possible. Thus, the team in charge of the system can work quickly on an up-to-date system.

The consolidation and validation phases have the guaranty to process a consistent and fresh state of the shared scene. Without that technical characteristic, any consolidation and validation would be useless since the result could be irrelevant due to inconsistent proposals. As can see, a technical property in the field of distributed systems can have major impact on the usability of the solution.

Work Review

Different work reviews can be necessary to resolve conflicts between several systems and discard a maximum of alternatives.

We will conduct a simple example because the main point is the understanding of the basic principle of the work review.

Let us consider a first example. The hydraulic and electrical managers organize a meeting to review the possible problems between different alternatives. When entered into the meeting, each designer observes a shared scene including the proposed hydraulic and electrical networks. Thus, immediately the participants observe unsatisfied constraints. Typically, a collision is detected between the two systems. In order to resolve the collision, the two designers can define in real time several alternatives. [Costantini et al., March 2000] describes practical ways to resolve such a collision.

Several advantages are provided. First, conflicts are resolved as soon as detected and in cooperation with the designer responsible of the other system. Second, the VEP tool enables to sketch different alternatives since the update of a subset of the scene is a rapid way to define a new alternative. Thus, a maximum number of solutions are browsed within a short time. Third, design in context enables to define real alternatives.

Once again, the ability to process a consistent and fresh copy of the shared scene enables design in context to solve real problems.

Through frequent reviews, the different teams avoid speculation by default, improve the quality of the solution and decrease the design time.

A second example involves the air-conditioned and hydraulic systems. The two corresponding teams start a meeting to review several alternatives and resolve collisions between their two systems. As described previously, the air-conditioned team prepared several alternatives to solve collisions. Since the global scene merged, the air-conditioned manager observes several collisions with the hydraulic pipes. Instead of updating immediately the shared scene to cope with the conflicts, he introduces another alternative within the shared scene and computes again the collision. Thus, the manager can select the alternative that minimizes the collisions. Afterwards, a direct update of the selected alternative enables to resolve the remaining collisions.

In contrast with the previous way of working, the prepared alternatives speed-up the review and augment the number of solutions the participants can examine.

This is only a demonstration example. But, all along the design several opportunities are given to the designers to prepare different alternatives. First, a designer can have several proposals that he prepares during the private work. Thus, he can quickly propose his ideas during a meeting to the other designers and discuss their respective advantages. Second, a

team offers several possibilities that satisfy the requirements. Thus, the global interdependencies will be exercised with a greater degree of freedom. Third, the work review naturally leads to different alternatives to resolve the conflicts and satisfy the overall requirements. Obviously the number of alternatives increases with the number of teams and systems. That is why several team-to-team reviews can be used to eliminate in parallel the irrelevant alternatives. Thus, the number of explored possibilities increases.

REQUIREMENTS

As a VEP system shall augment the Rapid Prototyping concept with efficient Human-Human interactions, collaboration and distribution facilities, this section focuses on the corresponding requirements.

This section formalizes the different styles and phases of working presented during the previous case study section. This section corresponds to the proposal made in [Costantini et al., Tech. Report 2000], further details can be found in that paper.

One does not address here the needs associated with the respect of functional constraints and human-machine interactions. In fact, these two latter needs are not specific to a VEP system. Moreover, sharing a 3D world enables to compute locally functional constraints and human-machine interactions. So, much emphasis is being placed on Human-Human interactions, collaboration and distribution needs.

Requirements of virtual early prototyping are now described. The objective is to enable the designers to quickly set-up and discuss proposals producing a 3D prototype. Thus, the best specification is defined in a collaborative way.

Designer Definition

Designer's name: each designer has a unique name (i.e. "Franck Cossgatoichega from Virtual Engineering Design Inc. at 205 Smart Avenue London in England"). A designer shall provide his name to participate in a specification activity.

Project Management

A project is a formal organization of multiple designers that carry out the global specifications of an industrial product.

Project name: a designer creates a project through its name definition (i.e. " Specifications of Paradise the third millennium space shuttle").

Dynamic project membership: a project membership sets up the names of the different designers that are project members. Within the project, a designer is simply named member with reference to his project membership. The group is built dynamically by adding or removing members during the lifetime of the project.

Responsibility sharing: different responsibilities are associated to the designers. For example, a designer shall define the electrical routes while another is responsible for the mechanical structure. Responsibilities evolve during the project lifetime. Moreover, a responsibility cannot be used to perform unauthorized actions during the project lifetime.

Project negotiation: a project definition is negotiated among the relevant participants to set up accordingly the responsibility and role. That negotiation phase occurs before exchanging significant data. At any time, a project can be renegotiated while respecting current responsibilities. The project negotiation can use various channels (like email, fax,

phone call, etc…). Received information authorizes to negotiate the project participation. This information cannot serve to take unauthorized roles or actions.

Distributed Specification

Each designer can prepare an individual 3D scene. This 3D proposal is elaborated using local tools. Afterwards, proposals are merged into a global specification. Thus, an extended team produces a distributed specification.

Individual proposal: it contains 3D objects but also organizational data namely geographical positions, responsibility data, annotations and computing resources. A unique designer makes an individual proposal. It is a subset of a global specification. It can contain results from a previous global specification. An individual proposal is produced mainly on a private basis during a disconnected phase. But, it can be introduced latter on into the global specification during a meeting phase.

Global specification: though the designers elaborate individual proposals, these local proposals can be used to define a global specification of the product. A global specification is the union of different individual proposals. Different iterations of a global specification are produced. Each iteration is produced mainly during a meeting phase.

Parallel work: two individual proposals are carry out simultaneously by two different designers. Thus, concept phase can be shortened as much as possible. Parallel work must be supported during both a disconnected and meeting phase.

Disconnected Work

It is a way to work on an isolation basis at the specification without any network connection. Typically, it enables a mobile designer to prepare some work on a laptop during a travel time. It is also a way to prepare at office a work before integrating the extended team. Thus, disconnected work can be seen as a preparation phase or a local improvement of the global specification. That disconnected work produces a sub-set of the global specification.

Responsibility control: a disconnected designer can only make actions in accordance with role and rights that have been granted by the project management or during a meeting phase. For example, if a designer works on a version retrieved during a previous meeting then he can only modify objects whose rights have been granted to him during this meeting.

Local validation: each designer can validate his individual proposal during a disconnected work. Thus, he checks that his local proposal respects the functional rules of the system. Typically, he verifies that his local proposal is correct before entering a meeting. This local validation does not guaranty correctness among the different local proposals. So, a global validation will be necessary during a meeting.

Local modification: a designer can improve locally a global specification by modifying it. Typically, he recovers a global specification during a meeting work and continues to improve it on a disconnected basis. Thus, he produces locally a new proposal that will be used during a further meeting.

Parallel work: two disconnected tasks are carried out simultaneously. Thus, two disconnected designers produce in parallel two different local proposals. These two local proposals will be merged in a further meeting.

Meeting Management

Designers, located in different sites, reach independently a meeting for joining their different proposals. Thus, a global specification is automatically obtained and shared. That

global specification is submitted to a discussion process. The global correctness is validated through a real time conciliation and real time modification.

Dynamic discovery: the different designers discover dynamically a meeting when it is created. Thus, a designer can decide to reach a new meeting after discovering it.

Real time membership: different designers join independently the meeting. Meeting membership is updated when a new designer is in. During a meeting, a designer is named participant with reference to his meeting membership. Membership is accessible to each participant. Knowledge of distant members provides information describing each role and responsibility.

Privacy: a meeting can be restricted to certain participants. Authorized participants are project members that are allowed to reach the meeting. Unauthorized members cannot reach a meeting or cannot use the information that is transmitted during that meeting. Typically, only authorized member can process the information exchanged during a meeting. If a member receives unauthorized information, he cannot interpret or process it. Thus, confidentiality is achieved during a meeting.

Global scene merging: each designer joins the meeting with an individual proposal. These individual proposals are merged into a global shared scene. Thus, a global specification is build by an automatic union of individual proposals satisfying the responsibilities of the different designers.

Real time validation: each designer validates the global scene by checking that others do not make conflicting or incorrect proposals. The system must guaranty that a validation is processed on a consistent scene. Otherwise, processing a validation would detect non-existing conflicts. Thus, the global specification is validated during the meeting.

Real time modification: a designer improves the global scene by modifying his proposal or others' proposals. The modifications are carried out while satisfying the different responsibilities. It reduces the design time because the designers collaborate in real time to improve the specifications through an immediate modification of the 3D scene.

Real time awareness: during a meeting, any participant observes the distant interactions as quick as allowed by his processing speed. For that purpose, the system minimizes the flow of data that must be processed by a station.

Real time conciliation: the system provides the required services to negotiate the improvement that must be carried out. First, a designer can juxtapose a contra-proposal with the current proposal. Thus, different proposals for a subset of the global scene are defined and compared in real time. Second, the conciliation must be done on a consistent scene. This is a constraint similar to the real time validation. Thus, the participants have the guaranty that their consensus is negotiated using consistent data.

Parallel work: two independent tasks are carried out simultaneously. Thus, the meeting does not degrade the ability of parallel working. For example, two participants are able to create, modify or delete two different objects simultaneously. So, in case of concurrent modifications the performances do not degrade due to inadequate locking mechanisms.

Guided visit: the system provides the required tools to facilitate the overview of the global scene. Typically, one participant guides the visit authorizing the others to follow him in real time. Thus, the participants review the specification in a synchronized way. Typically, shared viewpoints and annotations are supported. Moreover, GroupWare tools enable to talk with each other despite the distance barrier.

Iteration

The designers can iterate several times moving easily from a disconnected work to a meeting work. A new phase of work enriches the work carried out during the previous phases. Thus, moving between different styles of work does not impoverish the global specification.

Individual iteration: an individual proposal contains a subset of the global specification. A designer builds an individual proposal starting from a given subset. Thus, the designer works on the interesting view. During iteration he modifies the subset. During a further meeting, the resulting proposal can be used to update the global specification. During an individual iteration, he creates, modifies and deletes objects of his proposal. So, the subset changes, increases or decreases according to the different operation carried out during the individual iteration.

Distributed global iteration: the starting state of a global iteration is a collection of individual proposals. The different participants process a global iteration in a distributed way. During a meeting phase, the distributed modifications produce results of a global iteration. Those results define a new global specification. When multiple global iterations are processed within different meetings, the corresponding global specifications incorporate results from individual iterations that are interlaced among those meetings.

Consistent progression: a new iteration shall incorporate the consistent results from previous iterations. Thus, the system guaranties a consistent progression of the work and the designers do not spend time to rebuild previous results. Let us give some counter-examples. First, a participant has the ability to destroy distant works because the system does not manage correctly the rights that are not persistent from one meeting to the next one. Second, during the same meeting, an old state wrongly supersedes a newer state of a given object because the system does not preserve object consistency. So, clearly the system must avoid those inconsistent behaviors in order to guaranty a consistent progression within a single meeting but also between different meetings.

Constraints

Non-functional requirements shall be respected to guaranty usability and efficiency of the solution.

Simplicity: solution must be simple and lightweight. Simplicity enables good chances for the solution to be implemented. A lightweight solution can run on any standard workstation.

Portability: solution must use only standardized and widely supported services and protocols. Thus, the solution can be ported easily through recompilation and minor changes on different operating systems (Unix systems, Windows, MacOS, etc…).

Internet deployment: the solution can be deployed easily over the Internet. It shall not require any specific quality of service from the under-laying network. Thus, the solution runs widely over any kind of under-laying network.

Heterogeneity: the collaborative system authorizes heterogeneous machines to inter-operate. Thus, different platforms can participate in a meeting.

Zero administration: the system requires no administration effort to be deployed. Thus, any standard user manages projects and meetings without any system or network administration skills. It means also avoiding a central server where a system administrator is required to set-up user with customized profiles providing access to the collaboration data. Even if administration procedures are required, they shall be completely distributed among the designers and easy to use.

Low bandwidth: the solution can run with low bandwidth networks, including 33.6 modem links. Even with low links, good real time performances must be achieved. So, the system must support mechanisms reducing the bandwidth consumption without degrading real time awareness.

Modularity: collaboration services must be independent of application and virtual reality environment. Thus, any application can use the collaboration services within any kind of virtual environment.

Persistency: a designer has the ability to make his work persistent. Thus, he stores different steps of his progression to be able to retrieve them latter on.

Security: the collaboration services guaranty authenticity and confidentiality. Through authentication, it is determined who is participating before revealing sensitive information or entering into a collaboration process. Thus, a malicious people cannot usurp the identity of another designer. Confidentiality guaranties that only authorized person can access the information.

THE DISTRIBUTED BUILDING SITE METAPHOR

One proposes a novel solution answering to the VEP requirements that have been described in the previous section. This solution is called the distributed building site metaphor. This metaphor defines the specifications of a distributed environment allowing to share a 3D world. It is a fully distributed approach without any central server. That metaphor gives general specifications of the distributed system. It defines how the properties are provided. In fact, the metaphor gives the principles of the solution. The major assumption is that the metaphor uses only an unreliable multicast transport. It also requires regular email for the preparation phases. Thus, project management and meeting scheduling use email. Afterwards, the solution uses multicasting to share the 3D world and support several modes of real time collaboration.

Project management

Project negotiation: the designer creating a project becomes the manager of his project. Afterwards, the project management can be transmitted among the different participants.

As a project manager, the creator establishes the project attributes, namely project name, project membership, responsibility of participants and various collaboration, graphic or organizational data. The manager sends an email including the project definition to the different members.

Each recipient replies using email. He accepts the project as is or asks some modifications. For example, a member can request other responsibilities. That negotiation phase does not require any specific tools. But, a robot can assist a member by processing an email entry, negotiating the reply with the local member and replying to the sender using an email. Thus, different email exchanges can be involved. At the end of that negotiation phase, the manager sends a project confirmation to the participants.

Security is achieved using standard authentication and encryption procedures like S/MIME that are integrated within any email client (i.e. Netscape Navigator or Microsoft Outlook). For that purpose, any participant uses a digital signature (i.e. a X509 certificate including the user public key, the private key is kept secret at the user side) to be authenticated.

Management transfer: the manager can transfer the project management at any time to another person. For that purpose the project is renegotiated using the same protocol. The difference is that the confirmation message includes a signed text from the granting manager. The signed text was computed using the private key of the granting manager (i.e. the creator). The new manager of the project includes the signed text within any further email. Thus, the owner gives the proof that he really received the project management from the previous manager.

Dynamic project membership: project membership is set-up dynamically during the project lifetime. Entering or departure of any member is under the control of the current manager. Updating the project membership is achieved by running again the project negotiation protocol. Project membership can be different from meeting membership.

Initial scene transmission: the project confirmation can include the initial 3D scene to be used for the project. It gives the initial definition of the product. Considering a project of aircraft engineering, the aircraft structure would be the initial scene to be transmitted to each participant. Thus, each designer could design the systems corresponding to his responsibility (air-conditioned, seats, electricity, hydraulic…).

Moreover, an initial scene can been downloaded latter on using FTP or HTTP services. So, there is no obligation to achieve any transmission of the initial scene at the confirmation time.

Meeting Preparation

After the project negotiation phase, a scheduling phase enables to prepare a meeting.

Scheduling: any project member can send an email to a group of participants to schedule a meeting. Typically, this group is a sub-set of the project members. But, the group of participants can be extended to any required designer. Each member replies with an email. Several emails can be exchanged. Finally, the project initiator sends a latest email to confirm the meeting.

Like project negotiation, authentication and confidentiality is achieved through secure email and X509 certificates. Thus, a mutual authentication is guarantied between the meeting initiator and participants.

Address allocation: during the scheduling phase, the initiator proposes a multicast address. Each recipient replies if the requested address is not already allocated for another purpose. The reply can contain a range of available addresses. In the latter case, the manager selects a new address and processes the scheduling negotiation again.

In practice, each recipient can use a local directory service (i.e. Lightweight Directory Access Protocol [Wahl, 1997]) to reserve the requested address. By consulting already reserved addresses, he replies with the available addresses. In case no directory service is configured, a recipient simply accepts the proposed address.

This is a temporary allocation that will be used at the beginning of the meeting to contact each other. Further, we describe how conflicting address are detected and resolved automatically during the meeting.

Key distribution: the scheduling phase serves also to distribute a private key PK_S. PK_S enables a symmetric encryption during the scheduled meeting.

The first email proposes a schedule and enables the receiver to authenticate the meeting initiator using the included certificate. Each recipient X replies with his own certificate. Thus, the initiator authenticates each participant X. When sending the confirmation message to X, the initiator sends a private key PK_S. That private key is encrypted through secure email (i.e. S/MIME). Thus, X gets the private key PK_S that will be used later on during the meeting.

Thus, a session key PK_S is securely distributed. First, mutual authentication is achieved.

Second, only an authenticated participant X can decrypt the confirmation and gets the session key PK_S.

The system uses standardized cryptographic systems: PK_S is a DES key. Thus, any standard tool can be used to generate that key.

The manager can use a robot to send the confirmation email. A participant can also use a robot to retrieve and store the session key PK_S. Those robots provide automation but the user can also make the operations without any assistant robot. In fact, required operations are simple as various tools for those cryptographic systems are now widely available and easy to use.

Distributed Scene Tree

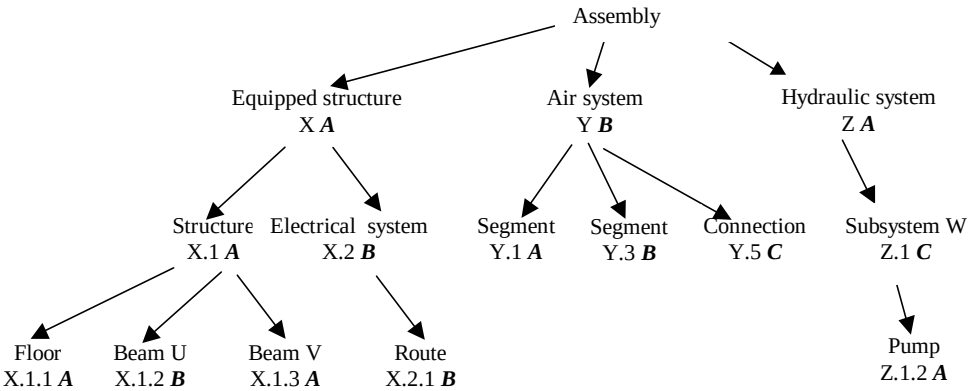
Scene tree: the application uses a scene tree that is composed of different objects. The scene tree is an abstraction of the application scene. It describes graphical information but also data that are relevant for the collaboration activity. Each object defines a sub-tree where the leaves are elementary nodes (i.e. a beam or an electrical route). Each node contains attributes namely geometrical attributes, organizational attributes, collaboration attributes and processing resources. A node is typically a graphical object. Organizational attributes consist of father-sons' relationships. Collaboration attributes contains mainly protocol states and protection attributes. A node can also contain computing resources like code to animate the corresponding object.

Figure 1 shows a simplified scene dealing with an aircraft subassembly design. User *A* owns the equipped structure and hydraulic system. *B* owns the air system. As *C* does not manage any system, his is not owner of any system. When a designer owns a node *N*, it means that he can do operations like updating attributes (i.e. size, position, protection attributes, etc…) associated with *N* or creating a child node *N.M*. Here, designer *A* owns the nodes *X, X.1, X.1.1, X.1.3, Y.1, Z, Z.1.2*. Designer *B* owns the nodes *X.2, X.1.2, X.2.1, Y, Y.3*. Designer *C* owns the nodes *Y.5, Z.1*.

Distributed scene: any central server does not maintain the global scene tree. It is automatically built during a meeting. The global scene is accessible as a whole if all the designers are present in the meeting.

Figure 2 presents the partial scene that is processed if only designers *B* and *C* are present. The important feature is that designer *B* and C can work despite the absence of *A*.

Figure 1: Simplified scene dealing with an aircraft subassembly design



In that case only B and C's nodes are processed. All the nodes, owned by *B* and *C*, are present in the partial scene. Each node of the partial scene can be processed without any restriction. For example, *C* adds a pipe (*Z.1.3*) to the subsystem (*Z.1*) while *B* changes some attributes of the route (*X.2.1*).

During a meeting each designer has a copy of the scene. Any modification is processed on the local copy and a modification event is multicast to the distant copies. When receiving the event, a distant peer processes the modification on his local copy. Thus, any distant participant observes the modification.

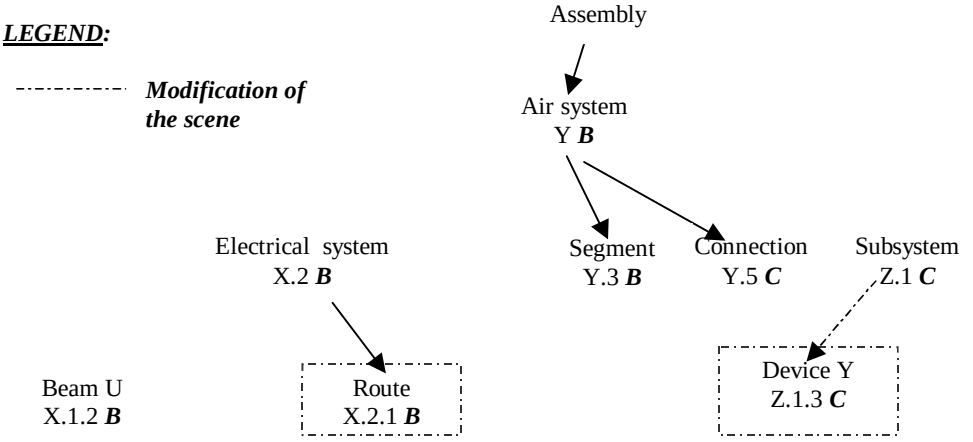
As will be described further, each participant has the ability to bring his nodes in an isolated workspace in order to continue working on a disconnected basis. Thus, the scene is distributed among the different designers. The scene is processed on a distributed basis during both a meeting and a disconnected work.

Protection: each node has protection attributes (read right and write right). The owner manages the protection attributes. Thus, the owner prevents or authorizes the modifications from the other designers. For example, disabling write permission the owner prevents distant participants from deleting or modifying the corresponding node. Disabling write permission prevents also other designers from attaching a new branch at the corresponding node.

Distributed designation: at first level of the scene tree, a unique name is attributed locally when a designer creates a new node. A unique name contains the IP address (*@IP*) of the creation machine plus a local date (e.g. name *X=@IPA,dateA* defines the *Equipped Structure* of figure 1). The subsequent nodes are designated according to their creation time (e.g. *@IPA,dateA,1.3* corresponds to the first branch and fourth leave that have been created by *A* for the node *@IPA,dateA*). Only the owner can create a child node (e.g. *A* created the node *@IPA,dateA,1.3* as owner of *@IPA,dateA,1*). That way, each computation peer defines unique names in a distributed way. Those distributed names can be created on a disconnected basis. It is an interesting approach as a designer does not need any name server to create a unique name. The father-son relationships are implicitly maintained by the distributed names and thus the system does not have to explicitly store those relationships.

If a designer does not have the right to create some type of nodes (i.e. a designer that is an air system manager does not have the privilege to create an hydraulic system), the application controls the permission. The distributed system provides the basic services enabling the application to respect the different responsibilities and permissions. The

Figure 2: Partial scene dealing with an aircraft subassembly design



application uses the project definition (that was transmitted to each designer) and protection attributes of the nodes to control the validity of the requested operation. In figure 2, the application uses the protection attributes of *Z.1* and decides that participant *C*, as owner, has the permission to create the child node *Z.1.3*.

Disconnected Work

Isolated workspace: during a disconnected work, a user builds an individual proposal within an isolated workspace. He creates, deletes or modifies nodes within that isolated workspace without any communication at all. Thus, a designer can work on a disconnected basis using a subset of the global scene. The subset comes from an initial transfer of the scene or from a previous meeting. That subset contains all the nodes that belong to the designer but it can also contain nodes from other designers.

For example, designer *A* is the manager of the assembly project. He sends during the project negotiation the initial scene as represented by figure 3. According to the project definition, designer B starts a disconnected work and creates an individual proposal within his isolated workspace as depicted by figure 4.

Figure 3: Sending during the project negotiation: The initial scene

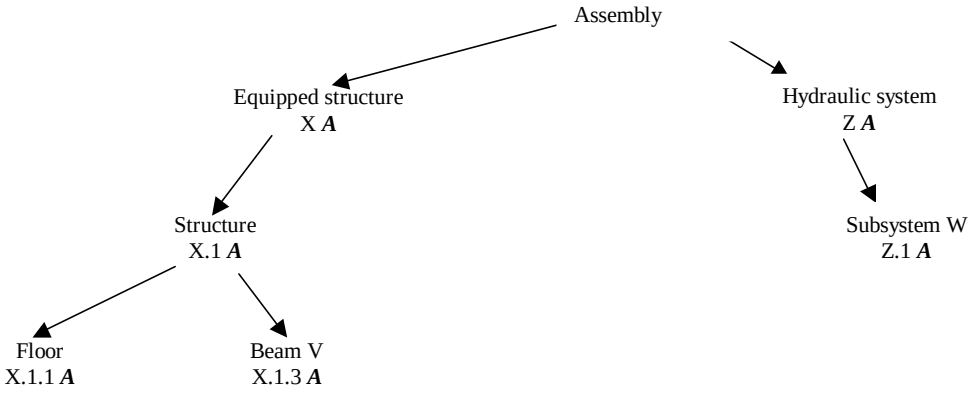
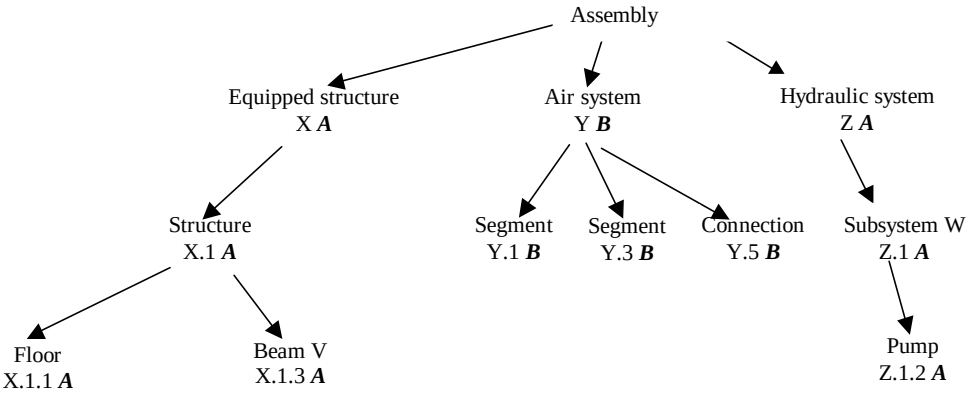


Figure 4: Starting a disconnected work and creating an individual proposal within isolated workspace



As described latter on, the ownership of a node can be transferred during a meeting. For example, after the first meeting the isolated workspace of designer *C* looks like figure 1. It means that the ownership of the nodes *Y*.5 and Z.*1* was transferred to designer C during the first meeting. After the first meeting, *C* can then modify these two nodes (i.e. he can update their attributes, delete those nodes, create child nodes) during a disconnected work.

Thus, on a disconnected basis an isolated workspace permits some improvement starting from a previous meeting.

Responsibility control: the application can control that an operation is in accordance with the designer's responsibilities.

For example, the application enables designer *B* to create an air system because he has the corresponding responsibility included in the project definition. After the first meeting, the application can also permit designer *C* to update the attributes of node *Y.5* during a disconnected work.

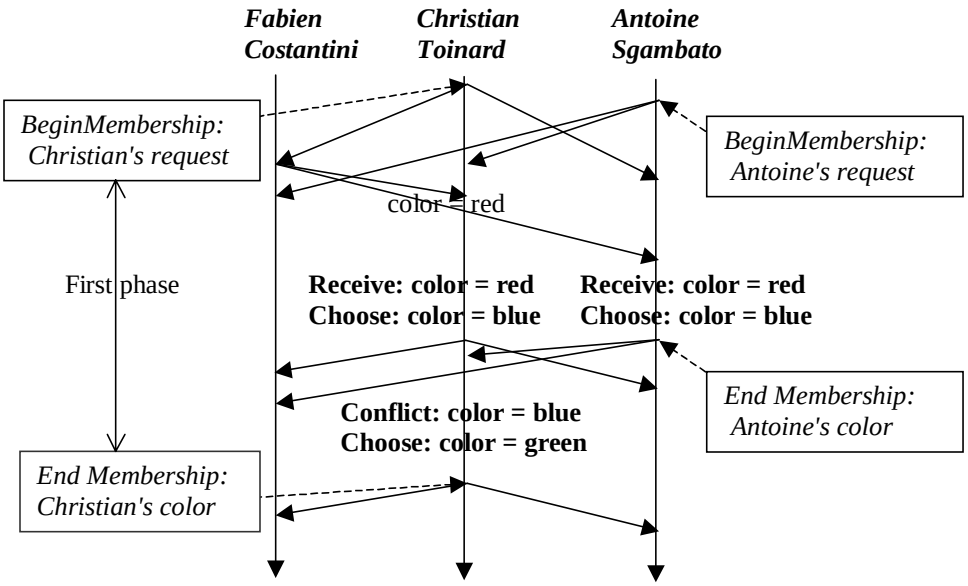
Parallel work: two disconnected tasks are carried out simultaneously.

For example, designers *A* and *B* work in parallel starting from the initial scene that is described in figure 3. *A* modifies the scene by adding the pump *Z.1.2*. In the same time, *B* builds the air system as presented in figure 4. Thus, two disconnected designers produce in parallel two proposals. These two proposals will be merged during a further meeting. For example, the first meeting builds automatically the scene presented in figure 4.

Meeting

Basically, a meeting runs in four phases. The first phase is a real time membership. The second phase achieves a global scene merging. At the end of that second phase, each participant has a copy of that global scene. That second phase uses a synchronization protocol that guaranties each participant to have exactly the same copy. During a third phase, different operations can be processed namely real time modifications, real time review, real

Figure 5: Christian entering concurrently with Antoine



time validation and real time conciliation. Fourth, the participants leave the meeting bringing a subset of the shared space in their isolated workspace.

Real time membership: during that first phase, an entering peer multicasts the participant' name. Distant participants reply as multicasting their names. Thus, a new participant maintains locally his knowledge of the meeting membership. The project definition enables a participant accessing to the role and responsibility of any distant member. Moreover, the protocol allocates a session color to each participant. This is a user-friendly mechanism allowing to color the objects according to their owner.

Let us describe the protocol more precisely. The entering peer multicasts a request including its network extremity (address IP + port number). In response, the distant peers multicast their name, network extremity and color. To terminate the membership phase, the entering participant multicasts a second message with the chosen color. When two peers are conflicting for the same color, the peer with the smallest extremity will change its color and multicast a new message. Thus, each participant builds its local membership knowledge of other participants.

Figure 5 shows *Christian* entering concurrently with *Antoine*. The conflicting colors are resolved at the end of *Christian*'s membership phase and Christian's peer multicasts a new message. The mechanism can have minor effects like seeing user *Christian* changing from the blue color to the green color during the same meeting.

Global scene merging: after the first phase, a second protocol enables each participant to share exactly the same scene. For that second phase, a participant chooses the individual proposal that he wants to add to the global scene. Receiving the proposal, a distant participant multicasts his own proposal. So, the entering peer receives the distant individual proposals that are assembled to compose a copy of the global shared scene. At the same time, the distant peers update their local copy with the entering proposal. A node that does not have a read permission is not multicast. That way, distant participants cannot observe unauthorized nodes.

As the meeting was scheduled previously, the application knows the list of all the participants that planed to come. Thus, the application can prohibit to multicast the nodes of the proposal that are owned by other participant. It has two advantages. First, it reduces the set-up time of the copies. Second, it avoids sending old-fashioned nodes, as the owners will enter their up-to-date states.

Thus, each peer has a local copy of the global scene at the end of that second phase. Moreover, the global scene respects the responsibilities because the modifications carried out during a disconnected work have been done according to the protection rules.

In fact, the entering peer uses a synchronization protocol. This synchronization avoids overloading the network by simultaneous multicasts from the different peers. Moreover, it recovers the transmission error guarantying each peer to have an exact copy of the shared scene. This protocol is described precisely below.

Copies synchronization: this synchronization protocol is used during the global scene merging. But, a peer can use it at any time to resynchronize the copies with the freshest state of each node. As each owner has an up-to-date state for his nodes, fresh states are collected from the different owners.

Peer *A* (i.e. an entering peer) multicasts a synchronization request including a list of nodes owned by *A*. Each element of the list contains the name of a node plus the current version number of that node. Afterwards, *A* multicasts a set of messages to provide a state for each announced node. When a distant peer *B* received all the states from *A*, it multicasts a synchronization reply including a list of nodes owned by *B*. This list is followed by the corresponding states.

To avoid network saturation if all the peers reply at the same time different mechanisms are proposed.

First, the network extremities can be used to schedule concurrent replies. Any peer acquired the network extremity of the distant peers during the membership phase. When *A* finishes to multicast its states, the peer with the smallest network extremity will start to reply. Thus, the network extremities order the replies.

Second, each peer *P* includes a list in its synchronization request or synchronization response. If a distant peer *R* misses a node of that list, then *R* asks *P* to retransmit a state providing the node' name. Moreover, *R* must receive a state with a version that is superior or equal to the announced version. Otherwise, *R* asks *P* to retransmit a state in order to receive a fresher version.

This solution is efficient in many ways. First, it guaranties each peer to have a fresh copy. Second, during that protocol multiple senders do not overload a peer as a unique peer can multicast at a time. Third, it enables to recover the transmission errors. So, at the end each peer has recovered a consistent copy including all the freshest states.

This service can be used before starting a validation or conciliation procedure. A participant can also use this service when he thinks that he misses lots of states. In that case, the peer sends a synchronization request asking for point-to-point replies. Distant peers replies directly to the requester using point-to-point exchanges. Thus, a requested peer does not receive any response from the other peers and its computation load is reduced.

Real time modifications: a designer can create, delete or modify a node according to the permission rules defined by the protection attributes and project definition. At that time, his peer multicasts an event to the distant peers. Each event contains the name of the corresponding node and all its attributes namely graphical data, node version and ownership. A receiving peer computes a new state for its local copy. To prevent old events to overwrite a newer state, the receiving peer uses the version number to distinguish those old versions.

Ownership transfer: to modify nodes owned by others, a peer must first request the ownership to the current owner. The owner refuses the ownership transfer when a write attribute is disabled. Otherwise, the requesting peer receives the current state and ownership at a time.

The transfer runs in three phases. First, the requester multicasts a message to locate the owner. Second, the owner replies to the requester with a point-to-point message including the ownership and the current state of the node. Third, when receiving the reply, the granted peer sends a point-to-point acknowledgement that terminates the transfer. When receiving that acknowledgement, the granting peer sets the ownership to false being sure that the granted peer received the ownership.

The ownership transmission must be reliable. For that purpose, a local number is associated with the request. The owner replies with the same number. When receiving the reply, the requester sends an acknowledgment including the same number. The requester resends his request in absence of response. The granting peer resends the reply in absence of acknowledgment. Faulty situations, where an object is without any owner due to a transmission error, are thus avoided.

When receiving a request from *A*, the granting peer sets the node to pending. Thus, it does not reply to any request that arrives from another requester *B*. That concurrent peer *B* multicasts periodically the request waiting that the new owner *A* replies.

Freshness of a copy: a copy can contain old states for the nodes owned by distant participants. But, owned nodes obviously are up-to-date.

There is impossibility to guaranty that a copy has only fresh nodes. The reason is that the transmission time of a modification cannot be bounded. Moreover, it cannot be the same for the different peers. Even if the system uses an under-laying network guarantying the same transmission time (which is hard and costly) for the group of participants, one cannot guaranty such a property if a transmission error occurs. In practice, transmission time is thus unbounded.

Moreover, the important point is that an old state cannot overwrite a fresh one. Here, the version number maintained by the owner guaranties that the old versions are discarded.

Object consistency: the ownership transfer guaranties that any modification is carried out starting from the latest state of that node. The reason is that a peer must recover the ownership before being able to modify that node. Thus, inconsistent modifications, where a peer would modify a node without being owner, are avoided. This is a very important property that guaranties a consistent progression of the work. It assures a designer will not modify a node without observing its latest state. Thus, a designer will not have to redo a work that was not been observed by a concurrent worker.

This basic mechanism can be used to achieve object consistency. For that purpose, a peer requests the ownership of the sub-tree composing the object. Thus, the designer is guaranteed to have a latest state of the sub-tree. Then having the latest version, he can decide if it is worth being modified.

One can also imagine a participant requesting the ownership of the whole scene. Thus, he makes sure to have a consistent scene. In contrast, the synchronization protocol provides a fresh copy of the whole scene without requesting the ownership transfer.

Real time awareness: at any modification, an event passing authorizes synchronization of the distant peers. In contrast with a point-to-point transmission, multicasting reduces the transmission time by emitting the event once and by propagating it simultaneously to the different participants. Moreover, a receiving peer speeds up the delivery of recent events throwing away old pending event. Thus, the workload is reduced because the receiving peer does not process old-fashioned updates. This mechanism enables to speed up the relevant events in the pending queue. Missing states are not recovered. It is not necessary for a distant participant to observe all the state of a given node because the owner maintains the correct state.

Parallel work: two tasks dealing with two distinct nodes are processed simultaneously. These two tasks do not communicate with each other.

Two tasks for the same node are serialized. The current owner processes the first task. Afterwards, the ownership transfer is carried out with the distant task. Thus, the second task starts processing the node after the end of the first task. Serialization is requested to guaranty a consistent progression of the work as explained previously. Thus, parallel working is limited only when required.

Real time review: a shared viewpoint can be created by one of the many participants as a standard graphical node. Afterwards, the ownership of the shared viewpoint can be transmitted to any participant that wishes to guide the visit. A guiding request is multicast to the distant peers. Observing the guiding request, a receiving participant can start the visit. Thus, his local viewpoint is synchronized with the shared viewpoint. At any time, a guided participant can stop the visit to recover his local viewpoint. Thus, the participants review the specification in a synchronized way through a simple shared node.

During the review, any participant can place an annotation by creating a new node whose attributes define a plain text. Participants can work together by jointly selecting and referring to specific scene nodes. For example, a participant moves his telepointer to select

and highlight a node (setting the attributes with a typical color) in order to referencing this element. Thus, the distant participants observe the telepointer motion and the highlighting refers the participants to the node involved.

Address allocation: a multicast address was reserved during the meeting preparation. If another application uses the same multicast address, DBSM will detect automatically the situation because received messages cannot be decrypted using the session key *KS*. DBSM multicasts automatically a new address. Receiving the proposed address, each peer will join that new multicast address. That way, the session is not interrupted.

Before proposing a new multicast address, DBSM joins and listens that address. If DBSM does not receive any message during a period, the new address is free and can be multicast to the other peers.

It should be noted that currently only a small set of application uses the multicast addresses. Thus, the probability that the chosen address be in use at the same time is really low. Moreover, using an occupied address does not prohibit the application of running. Reading irrelevant messages simply slows down the application.

Real time validation: this functionality is inherent in the application using our metaphor. DBSM provides the basic services enabling the application to propose adapted services. In fact, the application can run any kind of tool for validating the distributed work. For example, that conflicting proposals can be detected through a local computation namely a collision detection module, segregation module or flow simulation tool. These tools come as part of the application itself. They are embedded modules processing the local copy.

The basic support, provided by DBSM, rests on the multiple ways to have an exact copy.

First, after the scene merging each participant gets a complete copy of the global scene. So, the end of the merging phase is the right time where each participant can run a validation tool.

Second, at any time after real time modifications a participant can use the synchronization procedure. Thus, synchronization enables the different participant to run a validation tool after real time modifications.

Third, a point-to-point synchronization enables a requesting peer to recover a consistent copy without forcing the distant peers to do the same. Thus, a single peer runs a validation tool without disturbing distant participants.

At last, any peer has good chances to be up-to-date without using the synchronization service. The first reason is that the transmission time of each event is shortened. Each modification message vehicles a small amount of data (i.e. node name, object type, size, position, color) and emitting a multicast message takes less time than sending multiple point-to-point messages. The second reason is that the probability of loosing a message is low because the size of each event is small and each router is not congested by a big queue of point-to-point messages. The third reason is that a receiving peer discards the old states. Thus, a peer is not congested with old states.

A validation computation should have good performances as it processes only the local copy. Thus, a real time calculation is inherent in the associated tool.

Real time conciliation: on the same way, the application can use the basic services of DBSM to build conciliation procedures. Various possibilities can be imagined. DBSM provides basic properties.

Let us consider how to negotiate the required improvements. A designer can duplicate a sub-set of the scene. Original node can be write-protected but the duplicated node belongs to the requester. Thus, he can update the duplicated node to propose another solution. The

different participants can visualize the current proposal and new proposal at the same time. Thus, the participants discuss and compare the two proposals accordingly. A real time modification of the global scene is another way to make a new proposal.

A basic property that can be used by a conciliation procedure is the synchronization procedure that guaranties the different participant to have a consistent copy. Thus, having the same view they can discuss in a consistent manner. They did not loose time thinking on old-fashioned data.

Chatting facilities support the discussion process. A chatting object is associated with a node that can be created by any participant and written by everyone. Moreover, that tool can be completed with a GroupWare toolkit providing voice transmission.

Leaving: a leaving participant selects the nodes that he wants in his isolated workspace. All the owned nodes are automatically included in his isolated workspace. Nodes, owned by distant participants, can be selected. Thus, the leaving participant defined the sub-set of the shared scene he includes in his isolated workspace. The operation is processed locally by marking the excluded nodes within the local copy. Unmarked nodes build up the isolated workspace. It is thus a lightweight procedure. At the end of the selection, the peer multicasts a leaving message to inform the distant participants of its departure. Then, the peer leaves the multicast address.

It shall be noted that the leaving message does not need any acknowledgment. Unreliable leaving suffices as the peer has already the required nodes within his local copy. Before sending the leaving message, the peer will normally save the isolated workspace within a local store. Thus, the designer is able to recover his isolated workspace at any time. Further, the saved workspace enables a disconnected work.

DBSM Qualities

Simplicity: the solution uses unreliable multicasting and a lightweight consistency protocol. Thus, complex solutions are avoided by using a novel approach that focuses on consistency.

Portability: the solution uses standard multicasting and email communication. Thus, the solution can be easily ported on any machine supporting Internet.

Internet deployment: the solution does not require any specific quality of service from the under-laying network like resource reservation or reliable multicasting. It is deployed using the standard ability of routers to find a path for each member of the multicast address. In some case, tunnels are set up to go through deficient routers. Thus, the solution is deployed easily over the Internet.

Heterogeneity: some standard encoding rules, as used by ASN.1 compilers, can solve the heterogeneity problem. At the time of writing, our implementation transmits each message as a character string. It is possible because the state of a node has a small size. Thus, string conversion does not increase greatly the required bandwidth.

Zero administration: a user requires an email and an application using the collaboration library. Thus, any designer uses the collaboration services in a straightforward way. It does not require administration privilege or installation of specific servers. The system relies on a standard configuration of electronic mail and routers.

Performances: they should be good because of eliminating completely client-server bottlenecks and reliable multicasting pitfalls. Various techniques improve the performances (i.e. speed up of events, synchronization with load control). Moreover, dead reckoning can be used when the shared viewpoint moves quickly within the scene.

Low bandwidth: the different communication protocols use uniquely the multicasting of events. Thus, less bandwidth is required than sending a message to each peer. Moreover, each peer receives only a small amount of data, as the modifications are simply incremental updates.

Modularity: a copy of the scene is stored as application data. A local interaction event is processed directly by the application to update the local copy accordingly. Update uses the services of the virtual environment for a 3D representation of the local copy. It uses the collaboration services for transmitting an event to the distant peers. Thus, application, collaboration services and virtual environment are implemented as independent modules.

Persistency: at any time a designer can decide to store locally a sub-set of the nodes. This sub-set includes owned nodes. It can include also distant nodes. Thus, persistency is achieved in a fully distributed way through local savings. A global scene is thus distributed in the different permanent stores available among the designers.

Security: certificates enable the authentication of the designers. Thus, any participant checks the identity of another user. The application can then define what that authenticated user is permitted to do according to his responsibility and the required protections. Moreover, encryption guaranties the confidentiality. Each message is encrypted with the session key *KS*. Thus, uniquely the authorized members that securely received the session key *KS*, can decrypt the messages that are transmitted during a meeting.

A DESIGN PATTERN APPROACH FOR DBSM IMPLEMENTATION

In this final section, we focus on how two important aspects of the DBSM can be implemented with a Design Patterns [Gamma, 1994] approach.

The first pattern illustrates the distributed designation problem. The second pattern shows how the scene aggregation could be implemented. For a more complete discussion about this patterns see [Costantini et al., December 2000].

A Pattern for Distributed Designation

Name

Hierarchical Designation Manager

Example

Consider the example (Figure 1) of the previous section. Distant designers can set-up in a collaborative way a virtual prototype and attach simulation modules to the different sub-systems. Typically, designers cooperate to allocate space for different sub-systems and connect these sub-systems to show their relationships. During the process, designers can interact with the virtual objects hierarchy and provide real-time inputs to the simulators onto a virtual shared scene.

Figure 1 shows a simplified scene tree dealing with an subassembly design. We want these sub-system objects to be managed in a fully distributed and efficient way while keeping consistency of work.

Context

In a collaborative framework, workers need to share objects on the network. These objects have owners and may depend on other objects. Then, we need a component that manages a distributed applicative scene tree that allows these objects to be handled hierarchically in a fully distributed way.

Problem

The nodes of the global scene must keep unique names in time and space among the different private spaces and replicas.

The solution must guaranty several properties:

- Unique names should be easy to manage and use.
- Two private spaces or replicas cannot contain the same object with two distinct references. For example, during a disconnected phase, the pump can be present in two different private spaces and still be designated with the same name. During a meeting, two replicas use a same name to access the pump object (local or distant).
- If a name designates two distinct nodes, then uniquely one node is active and the other is a pending node that cannot be processed. The pending node must disappear and be replaced by the active one. The system must guaranty that a pending node exists only within a private space. In contrast, two versions of the same node have the same name.
- The distributed scene tree may not reflect the applicative scene tree: objects can be private and remain in the local private space or be public in the global shared space.
- A user creating a node must be able to use a dynamic Internet Protocol address. Despite a dynamic address allocation, the name must remains unique in time and space.

Solution

A peer (i.e. an entity processing a replica or a private space) computes locally a unique name when a user creates a distributed new node (i.e. a Network Decorator Leaf1 or Network Decorator Composite as defined in the sequel) within the scene tree. For example, a unique name contains the Internet Protocol (*IP*) address (*@IP*) of the creation machine, a local timestamp and the position within the tree (e.g. *@IPA,stampA,1.3* corresponds to the first child and third grandson of the node *@IPA,stampA*). A timestamp includes a date, corresponding to the local creation date, plus a random number. Moreover, a node maintains locally the child names. For example, node *@IPB,stampB,1* stores locally the list of its children (*@IPB,stampB,1.1*, *@IPB,stampB,1.2*, ..). To create or to delete a name, a user must be the owner of the father. For example, to create name *@IPB stampB,1.3* the user must own node *@IPB,stampB,1*.

Thus, a node (e.g. *@IPA,stampA*) can be created at the first level on a disconnected basis without being attached to any particular shared scene. It enables a user to prepare some work without knowing in advance in which scene he will enter his work. For example, one designer prepares an electric sub-system and another designer prepares an hydraulic sub-system. Afterwards, they reach a meeting to conciliate their works. Figure 6 gives the automatic merging of the corresponding scene tree. The root of tree corresponds to the name of meeting. It is seen that the unique names do not correspond directly to the graphical position of the objects (e.g. the first vent is named *@IPB,stampB,1,3* while the second vent is named *@IPB,stampB,1,1*).

Figure 6: Automatic merging of the corresponding scene tree

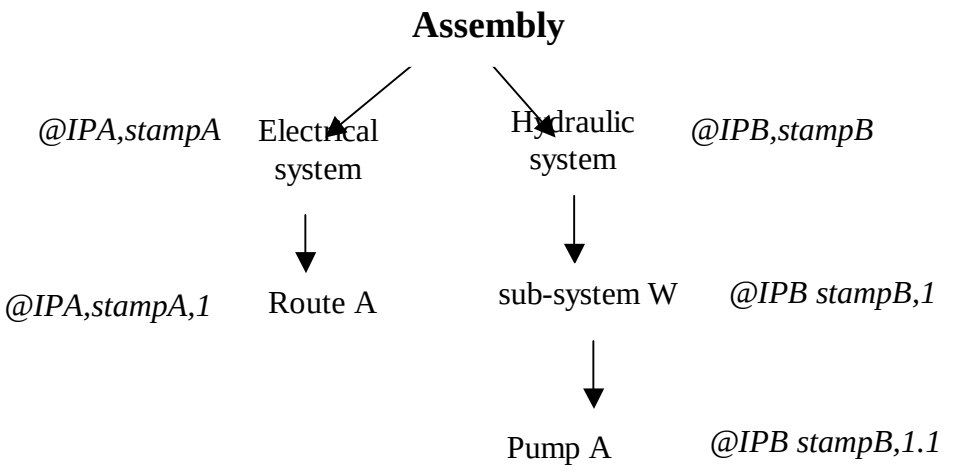
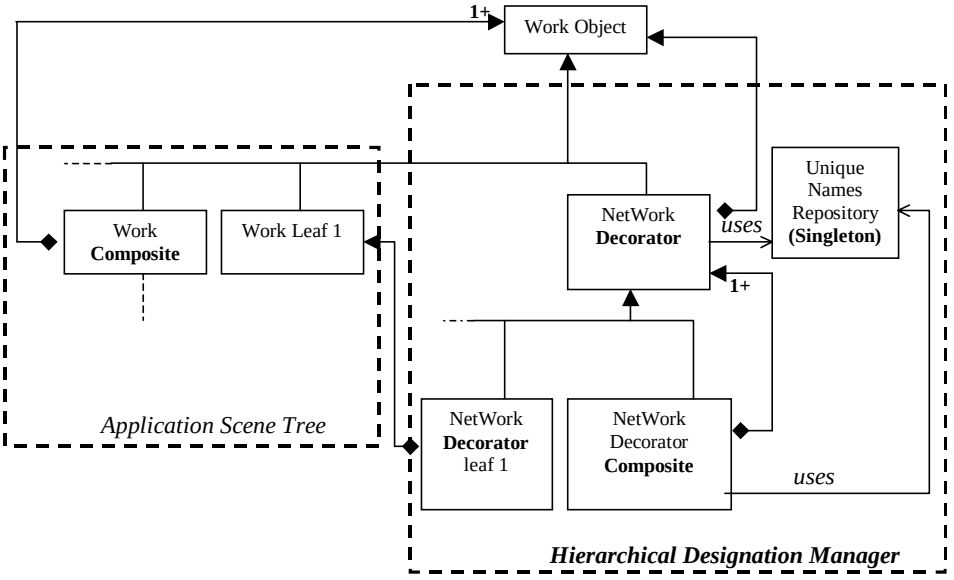


Figure 7: Structure



Participants

Applicative Part

- Work object:
  - Abstract work object class that defines all the generic attributes and methods that a VR work object may have (position, orientation, color, …).
- Work Leaf1 (Pump):
  - An concrete applicative object that may be wrapped by a Network Object Decorator on demand
- Work Composite inherited objects (Route):
  - Concrete groups of applicative objects.

Distributed Part

- Network Decorator:
  - Abstract class reflecting the generic network mechanisms of the Work Object abstract class. For example, it can implement at this level the distribution of abstract Work Object methods like moveObject(), resizeObject(),…
- Network Decorator Leaf1 (PumpDecorator):
  - This object inherits from an abstract Work Decorator and wraps a concrete applicative object from the scene tree.
  - It furthers additional network control to abstract Work Object methods in order to update the replicates of the under-laying object on the network.
- Network Decorator Composite:
  - This group forms, with the Network Decorator, the composite part of the distrib uted object tree implementation.
- Unique Names Repository:
  - This Singleton implements the local Unique Naming access methods (CreateUniqueName(), RemoveUniqueName(), IsNameDistant(), …) and is used by the Network Decorator objects. It also handles the dynamic construction of the global name list by storing the distant unique names received from the remote participants.

Collaboration

When entering a meeting, Network Decorator objects substitute local Application Scene Tree objects. On the others participants sides, Network Decorator are constructed and then create replicates of the distant objects. Note that the same network classes manage the local objects and their replicates. So there are as many concrete Network Decorators implementations as there are concrete applicative implementations.

Consequences

Network properties

- The first property is satisfied. The unique names are easy to create since the Unique Names Repository creates each name locally. Thus, a unique name can be created both during disconnected work and during meeting. A unique name is independent of the geographical position within the 3D scene. So, objects can move without changing

their names. As an example, A Network Decorator Object can update its underlying applicative concrete Work Object position but its name at the network level remains the same. A unique name defines the position of the node within the global scene tree. Thus, a name is unique, in time and space, but defines also a position within the distributed scene tree. It is useful since the application retrieves directly from a name which node is the father. Moreover, an application can decide to accept unordered messages notifications and stack them until it has resolved all the dependencies between them. For instance, a Network Decorator Composite Object node notification may results in several concrete Network Decorator object notifications. As hierarchical information is attached to the notification, we can implement a best effort algorithm that wait for all dependent nodes notifications before it updates locally the replicates.

- The second property is satisfied since the object name is defined at creation time and remains unchanged all along the life of that object.
- Third property is guarantied because to create a name (*@IPB,stampB,1.1*) a user must be owner of the parent node (*@IPB,stampB,1*). So, the parent (a Network Decorator Composite object) must be present to create a child node. Using the local list of children, the parent guaranties that an active name cannot be reused. To be reused, the owner of the active name must be present and must delete the name. If the same name exists for another object, it can be only for a user that is neither present nor owner. So, it can only be a private space that contains the pending name. Moreover, the pending name will disappear when the user will reach a new meeting.
- Latest property enables a communication peer to change its IP address without any difficulty. Already created names remains unique because the probability that two machines use the same address to produce the same name (same local time and same random number) is close to zero.

Moreover, existing works can be reused through first level nodes that do not contain any reference to a specific shared scene.

Structure considerations

- Network implementation is well decoupled from objects implementation: the Network Decorators does the network job, so almost no specific and complex network code obscures the existing code and therefore simplify the developer task.
- The existing applicative object structure must be stable: adding new applicative objects types means adding new Network Decorators. However, note that many generic features may be already implemented in the abstract Network Decorator class that reflects the Work Object generic methods augmented with network constraints. So even then, the effort of creating a new concrete Network Decorator class should be minimized.
- In some situations, the Network Decorators may have to know about the private internal attributes of their underlying applicative objects in order to be able to dispatch object updates or to construct the replicates. This problem can be solved in most cases by using the Memento Pattern [Gamma, 1994] which will enables to interface the applicative object with his decorator by furthering only a subset of the real object state to the decorator without violating encapsulation of this underlying object.
- Maintaining two different trees synchronized (a distribution one and an applicative one) is not easy. However, this structure allows to choose clearly what should be distributed and what should remains in the private local space.

- Direct object distribution may not be acceptable for performance reasons. In some situations, it is better to implement a Distributed Facade [Brown, 1999] instead of sending several dissociated object notifications. In our example, the distributed objects are lightweight objects that need to send few messages to their replicates. Moreover, for more abstract command notifications that doesn't apply specifically to an object structure, it can be necessary to implement a Command oriented pattern which can be expressed as a combination of the Command pattern and the Decorator Pattern (see Compounding Command [Vlissides, 1999]). The decorator part will correspond to the network transmission of the wrapped command.

See Also

Decorator, Singleton, Composite, Memento, Command [Gamma, 1994], Compounding Command [Vlissides, 1999], Distributed Facade [Brown, 1999].

A Pattern for Automatic Scene Aggregation

Name

Scene Aggregation Helper

Example

Collaboration requires moving easily between disconnected works and meeting works. When virtual prototyping, each participant improves during a disconnected phase different nodes of the global virtual scene. Thus, a disconnected workspace includes nodes from the previous meeting but also new nodes created during the disconnected phase. During a further meeting, each participant brings the disconnected nodes into the shared scene tree. Thus, different entering nodes reform automatically a shared scene tree.

Context

This service is present typically when mobile workers collaborate at a virtual prototyping. Merging of disconnected works must be supported while satisfying a minimal "consensus". It is obvious that disconnected works cannot provide a global specification that satisfies all of the prototype requirements. For example, two disconnected designers can produce two conflicting improvements. During a meeting, the system can compute the shared scene to check for requirement respect. Detected conflicts must be resolved by interactions between the designers and direct modifications of the shared scene. End-users do not want the system to resolve automatically the conflict since they want to negotiate and conciliate the different proposals. Despite it is not feasible to guaranty a global consistency for the different entering proposals, the system must at least guaranty that the different proposals are associated with correct functional locations to guaranty a minimal consensus on the shared work.

Problem

A participant must be able to bring pieces of the scene puzzle and gets a copy of the global scene. The different pieces must go to the right place of the puzzle. A right place is not only a 3D position but also a correct functional location within the scene tree. For each arrival, the distant participants must observe the added pieces. Moreover, each peer must recover from transmission errors and gets a fresh state for each node of the global scene.

Solution

An application peer creates a Network List Object (i.e. an object including all of the entering nodes, see further details in the structure section) to define the nodes entered by the local user. At the creation time, the Network List Object multicasts several list messages to the distant peers for creating distant replicas. Each list message can be viewed as a part of the whole Network List Object. When a peer received the total number of parts, it has a complete copy of the Network List Object. Several messages are required since UDP (User Datagram Protocol) limits the size of a multicast message.

A list message is defined as follows [*LIST*: V_L, *TotParts*, *NPart*, (G_1,V_1),…, (G_N,V_N)] where V_L is the version number of the list, *TotParts* is the total number of parts forming that list, *NPart* is the part number and each couple (G_X,V_X) defines the unique name and the version number for an object *X*. Using *TotParts* parts, a peer announces the objects it enters into the meeting. Afterwards, the announcing peer multicasts an object state [*State*: V_L, G_X, V_X, T_X, S_X] for each object *X* of the list where T_X is the type of *X* and S_X is the state associated with V_X. Thus, a distant peer recovers all of the announced objects. At the end of the scene aggregation, each peer has a copy of the shared scene.

Since a distant peer has the list of the objects, it can ask the retransmission of a missing object *X* by requesting a version number equal or higher to V_X. Thus, a producing peer has to retransmit uniquely the current version V'_X of *X* ($V'_X \geq V_X$).

Since each state [*State*: V_L, G_X, V_X, T_X, S_X] transports a list version V_L, a receiving peer that misses the corresponding list message can ask a retransmission by requesting the list part for the object G_X. Generally, the producing peer will retransmit the corresponding list part [*LIST*: V_L, *TotParts*, *NPart*, (G_1, V_1), …, (G_X, V_X), …, (G_N, V_N)]. When the producer added or deleted an object, the requested version V_L is no more available since the list changed. In that latter case, the producer retransmits all the new parts [*LIST*: V'_L, …] with $V'_L \geq V_L$. After receiving the new list, the requester has the responsibility of asking the retransmission of missing states based upon the received list.

It should be noted that a selective list [*SLIST*: V_L, T_L, *TotParts*, *NPart*, (G_1, V_1),…, (G_N, V_N)] transmits only object references of type T_L. A selective list is a subset of the list V_L.

Participants

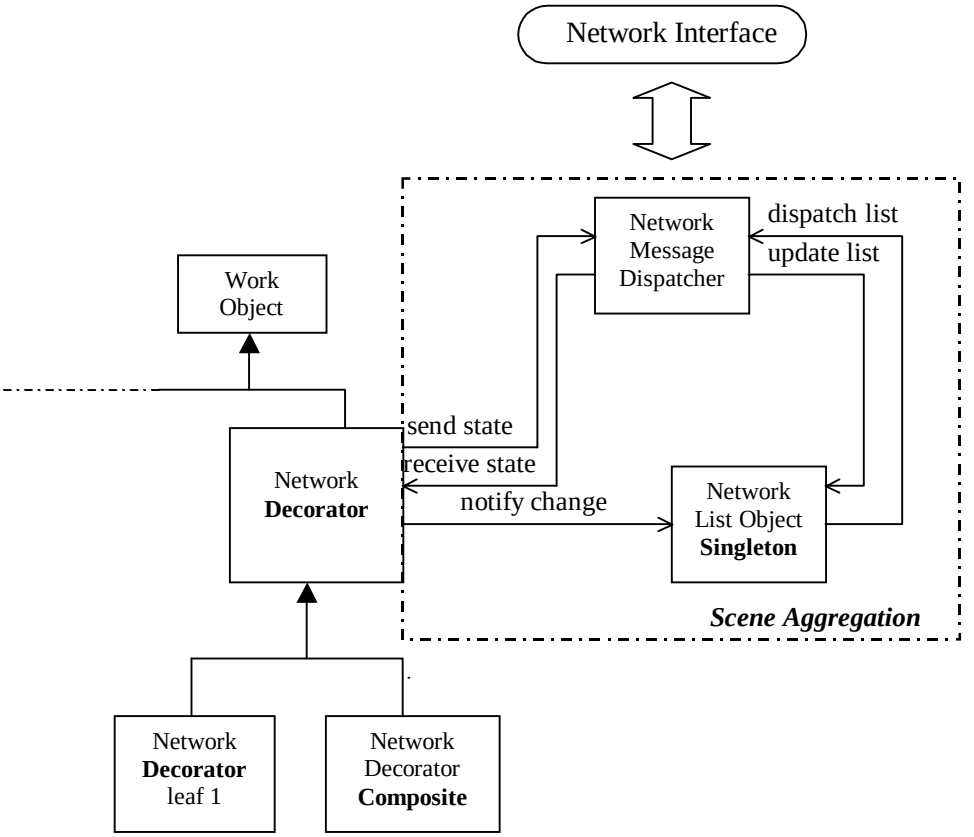
- Network List Object:
  - There is as many list objects as meeting participants
  - It manages several attributes like (G_1, V_1),…, (G_N, V_N) and list version V_L
  - On demands, it furthers additional network control to update the replicates of the Network List Object on the network

The other participants have been described for the previous pattern.

Collaboration

Typically, a GUI uses the Scene Aggregation when the local participant decides to enter prepared works. Thus, a new Network List Object is created and transmitted to update the distant copies. Each Network Decorator (Leafs and Composite) uses the Network List Object each time a modification is processed on the corresponding objects (Leafs and Composite). Thus, each Network Decorator object modifies Network List Object of the local participant. On demands (from a local user or distant request), an update of the Network List Object is transmitted to the distant peers. For example, if a replica wishes to re-synchronize with fresh values then it sends a request for the Network List Object to check freshness of

Figure 8: Structure



distant objects. When a Network List Object is newly created for a distant participant, the creation method uses the local Network List Object to update the entering participant with the local Network List Object. Since a communication peer receives the version number of the Network List Object but also of the different Work Objects, it can ask retransmission of missing list messages or object updates.

Consequences

Network properties

- Each entering peer brings pieces of the scene puzzle using the Network List Object. A couple (G_X, V_X) transmits a piece X. The entering participant gets a reply from distant participants. Thus, he completes the global scene as assembly of the different responses. At the end of the scene aggregation, each participant gets a copy of the shared scene.
- Each piece X goes to the right place since the name G_X contains the location within the scene tree.

- Each time a participant enters, the distant copies of the global scene are updated using the list messages. Moreover, the responses give an opportunity for the copies to resynchronize and to recover fresher states.
- When a receiving peer detects a transmission error (missing announced objects or missing list parts), he asks for newer versions. Thus, retransmission only gives the fresh version of the missing object or missing list. Retransmissions of out-of-date states are thus avoided.
- Since the List Object is transmitted only on demands to the distant peer, the system does not retransmit redundant information i.e. the state of the Decorator Object that caused the List Object update.
- List Object gives information about the global progression of the corresponding participant. It enables a receiving peer to quickly check if the current state of a participant is fresh or if states are missing. It provides a basic support to further send negative acknowledgement about missing values.
- It shall be noted that a List Object can evolve while the local peer processes its Decorated Objects. The List Object evolves consistently with the current state of the peer. There is no need to locking the List Object while a peer processes its Decorated Object or vice versa.

Structure Considerations

- Network implementation is well decoupled from any application processing: the cooperation between the Network List Object and the other Network Decorators does the network job. So, application code only have to create a List Object on user request to get a copy of the shared scene.
- As the List Object is processed either through local or distant request a Compounding Command [Vlissides, 1999] can be used. The decorator part enables to wrap and transmit the command to a distant peer.
- The communication between the Network Decorator and the Network List Object can be implemented efficiently with Observer [Gamma, 1994]. In effect, the Decorator will be the subject and Network List Object will observes the notify updates.

See Also

Compounding Command [Vlissides, 1999], Observer, Singleton [Gamma, 1994].

CONCLUSION

Our solution is efficient since the design time can be reduced. The key points of that reduction are the ability to work in parallel through a widely distributed specification, the easy motion between private and shared work, the good support for conciliation and the incremental validation of the global specification.

Second, the number of examined solutions increases. The key points are the ability of working anywhere in the world, easy consolidation and conciliation of private works into a global solution, collaboration in real time with distant participants and good usability of the collaboration services.

Third, the distribution can be integrated within any standalone application. Thus, a rapid prototyping tool can be developed, experimented and improved on a standalone basis.

Afterwards, collaboration can be integrated to support concurrent engineering without modifying the interface of the prototyping tool.

Fourth, deployment has been proven to be very simple. The major reasons are namely a fully distributed solution, no hosting and management of any server, efficient security using secure email to distribute a session key, use of the standard Internet architecture and automated components to manage the network channel and security.

DBSM works on both Unix and NT. It has been used in a virtual prototyping application like described in [Costantini et al., March 2000] and [Costantini et al., May 2000].

Future works will provide several improvements. New human-human interactions will be added using the feedback of the users. The resynchronization service will be improved through a new management of object versions [Costantini et al., November 2000]. A solution will be developed to cope with participants that cannot be reached using multicast and a better confidentiality will be provided using Deffie-Hellman shared secrets [Costantini, 2001]. eXtensible Markup Language will be used to access various CAD and Product Data Management systems.

REFERENCES

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

Broll W., *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, 1998.

Brown K., Eskelin P., Pryce N. *A mini-pattern language for Distributed Component Design. 6th. Pattern Languages of Programs Conference*. Urbana, Illinois. August 1999.

Costantini F., Sgambato A., Toinard C., Chevassus N., Gaillard F., *An Internet Based Architecture Satisfying the Distributed Building Site Metaphor.* IRMA2000 Multimedia Computing Track, Anchorage, Alaska, 21-24 May, conference proceedings, pp.151-155, published by IDEA Group Publishing ISBN 1-878289-84-5, 2000.

Costantini F., Sgambato A., Toinard C., Chevassus N., Gaillard F., *Distributed Building Site Metaphor For Concurrent Engineering.* MICAD2000, 19th International Conferences dedicated to CAD/CAM/CAE and New Technologies for Design and Manufacturing, Paris, France, 28-30 March, conference proceedings, pp.187-194, 2000.

Costantini F., Sgambato A., Toinard C., Chevassus N., Gaillard F., *VIRTUAL EARLY PROTOTYPING USING THE DISTRIBUTED BUILDING SITE METAPHOR.* Technical Report CNAM-CEDRIC submitted to WSCG'2000, 2000.

Costantini F., Toinard C., Chevassus N., *Secure mobile replication for collaborative virtual reality.* Intelligent Multimedia Computing 2000, Rostock-Warnemünde, Germany, 9-10 November, conference proceedings, 2000.

Costantini F., Toinard C., Chevassus N., *Patterns for Collaborative Distributed Virtual Reality*. ICSSEA 2000 conference proceedings, Paris, France, 5-8 December, 2000

Costantini F., Toinard C., Chevassus N., *Collaborative design using distributed virtual reality over the Internet.* Internet Imaging 2001, San-José, California, conference proceedings, 2001.

Dassault Systems CATIA Marketing Team, *4D Navigator*, http://www.catia.ibm.com/prodinfo/, 1998.

Dassault Systems, http://www-3.ibm.com/solutions/engineering/escatia.nsf/public/v5_public, 2000.

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

Defense Modeling and Simulation Office, *HLA Time Management Design Document V1.0, August*, U.S. Department of Defense, Washington D.C., 1996 .

Eleftheriadis A., Herpel C., Rajan G., Ward L., *Text for ISO/IEC FCD 14496-1 Systems,*

ISO/IEC JTC1/SC29/WG11CODING OF MOVING PICTURES AND AUDIO, May 1998.

Gamma et.al., *Design Patterns: Elements of reusable Object-Oriented Design*, Addison Wesley, 1994.

Greenberg S., Roseman M., *Using a Room Metaphor to Ease Transitions in Groupware*. Research report 98/611/02, University of Calgary, 1998.

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

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

Hewlett-Packard, *CoCreate OneSpace: the Award-winning, Web enabled, CAD-Independent, real-time Collaboration Solution*, 1999.

IEEE Std 1278.1, *Standard for Distributed Interactive Simulation Applications Protocols*, IEEE Computer Society, 1995.

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

Object Management Group, *Control and Management of A/V streams specification*, OMG Document telecom/97-05-07 edition, 1997.

Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Edition 2.3, 1999.

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, 1995.

Schulzrinne H., Castner S., Frederick R., Jacobson V., *RTP: A Transport Protocol for Real Time Applications*, Internet Draft, 1994.

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

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

Vlissides J., Helm R. *Compounding command*. C++ Report, April 1999.

Wahl M., Howes T., Kille S., *Lightweight Directory Access Protocol (v3)* Request for Comments 2251, 1997.

Sense 8, World2World Release 1 Technical Overview, 1997.