Collaborative Multimedia in Scientific Design *

Vinod Anupam Chandrajit L. Bajaj

Department of Computer Sciences,
Purdue University,
West Lafayette, IN 47907
{anupam,bajaj}@cs.purdue.edu, 317-494-6531, FAX: 317-494-0739

Abstract

We discuss the application of multimedia in scientific design, and describe a multi-user distributed and collaborative scientific manipulation environment, Shastra, implemented on the multimedia desktop. We highlight salient features of the underlying collaboration infrastructure – an application conferencing substrate that enables user level cooperation. We demonstrate that, in conjunction with shared contexts, multimedia interfaces – incorporating text, graphics, audio and video, greatly empowers users in the process of collaborative scientific design.

Keywords:

Concurrent Engineering; Multimedia Communication; Distributed Conferencing; CSCW; Groupware; Virtual Team Support; Application Interoperability; Information Sharing.

1 Introduction

Multimedia workstations have become commonplace due to recent advances in electronics, computer and communication technology. Current audio, video, and graphics processor architectures, coupled with high speed networking and compression techniques, have presented us with a very powerful tool – today's desktop system. Computer mediated mechanisms built on top of these systems provide us with the means to exchange multimedia information, and will revolutionize how we collaborate in the scientific setting.

Our goal is to depart from traditional single user scientific manipulation systems and build multi-user (collaborative) scientific design and analysis environments, by harnessing current computing technology and utilizing multimedia functionality and performance. The objective is to develop the next generation of scientific software environments where multiple users, *e.g.* a geographically distributed collaborative engineering design team, create, share, manipulate, analyze, simulate, and visualize complex three dimensional geometric designs over a heterogeneous network of workstations and supercomputers.

We have adopted the approach of integrating a collection of function-specific tools into a distributed and extensible environment, where tools can easily use facilities provided by other tools. Isolation of functionality makes the environment modular, and makes tools easy to develop and maintain. Distribution lets us benefit from the cumulative computation power of workstation clusters. Tool-level cooperation allows us to exploit the commonality that is inherent to many scientific manipulation systems. An enabling infrastructure of communication and interaction tools, display and visualization facilities, symbolic processing substrates, and

*Supported in part by NSF grants CCR 92-22467, DMS 91-01424, AFOSR grant F49620-93-10138, NASA grant NAG-1-1473 and a gift from AT & T

simulation and animation tools saves avoidable re-implementation of existing functionality, and speeds up the application development process.

The collaborative scientific environment provides mechanisms to support a variety of multi-user interactions spanning the range from demonstrations and walk-throughs, to synchronous multi-user collaboration. In addition, it facilitates synchronous and asynchronous exchange of multimedia information which is useful to successfully communicate at the time of design, and to share the results of scientific tasks, and often necessary to actually solve problems. The infrastructure provides facilities to distribute the input of low computation tasks – to obtain the parallelism benefit of distribution, and the output of compute intensive tasks – to emphasize sharing of resources among applications. It provides a convenient abstraction to the application developer, shielding him from lower level details, while providing him with a rich substrate of high level mechanisms to tackle progressively larger problems.

2 Overview

The rest of this paper is structured as follows. We introduce Shastra in the context of related work in this section. In Section 3, we describe the architecture and highlight salient features of the system. We also briefly describe building blocks of the runtime system. We present specific details of an example application for distributed and collaborative problem solving in Section 4. There, we show how users exploit Shastra facilities in the collaborative design process.

2.1 A Scientific Manipulation Environment

Shastra [1] is an extensible, distributed and collaborative geometric design and scientific manipulation environment. At its core is a powerful collaboration substrate – to support synchronous multi-user applications, and a distribution substrate – which emphasizes distributed problem solving for concurrent engineering. Shastra provides a framework for distribution, collaboration, session management, data sharing and multimedia communication along with a powerful numeric, symbolic and graphics substrate. It enables rapid prototyping and development of software tools for the creation, manipulation and visualization of multi-dimensional geometric data.

The Shastra environment consists of multiple interacting tools. Some tools implement scientific design and manipulation functionality (the Shastra Toolkits). Other tools are responsible for managing the collaborative environment (Kernels and Session Managers). Yet others offer specific services for communication and animation (Service Applications). Tools register with the environment at startup, providing information about the kind of services that they offer (Directory), and how and where they can be contacted for those services (Location). The environment supports mechanisms to create remote instances of applications and to connect to them in client-server or peer-peer mode (Distribution). In addition, it provides facilities for different types of multi-user interaction ranging from master-slave blackboarding (Turn Taking) to synchronous multiple-user interaction (Collaboration). It implements functionality for starting and terminating collaborative sessions, and for joining or leaving them. It also supports dynamic messaging between different tools. Tools are thus built on top of the abstract Shastra layer, which is depicted in Figure 1.

2.2 Related Work

A teleconferencing approach to modeling and analysis of empirical data is presented in [3], where the authors hypothesize about a collaborative scientific visualization environment. A discussion of an interactive visualization environment for 3D imaging is presented in [7], where the the authors adopt the electron microscope to perform as a computer peripheral. An environment like Shastra makes it convenient to build collaborative visualization and manipulation facilities, that support resource sharing in a distributed setting.

[1] Shastra is a Sanskrit word which means a branch of knowledge

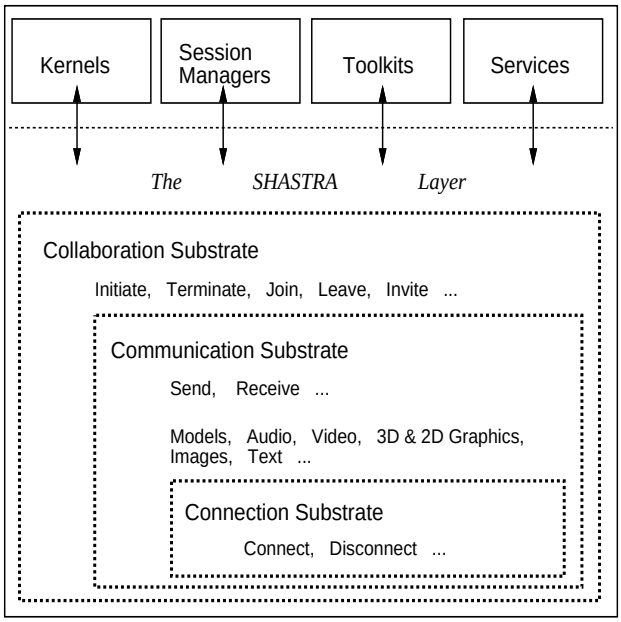

Figure 1: The Shastra Layer – A connection, communication and collaboration management substrate. Shastra tools inter-operate using facilities provided by this layer.

Our intention in Shastra is to provide a distributed heterogeneous environment for multimedia collaboration and to facilitate the development of collaborative applications by providing a rich substrate. Specifically, we focus on the scientific domain, though the collaboration facilities are flexibly adapted to other areas.

Groupware emphasizes on using the computer to facilitate human interaction for problem solving. Ellis *et al* present an overview of the state-of-the-art in [6]. Research in collocation via multimedia facilities has resulted in systems like MONET [11], CECED [4], and MERMAID [12]. Colab is a collaborative meeting facility which permits multiple users to simultaneously manipulate drawn objects [10]. Rendezvous proposes a powerful architecture for multi-user applications [9].

These systems provide simple collocation (audio-video communication), or content independent sharing of drawing and viewing surfaces. Shastra lets us build applications with shared drawing and viewing surfaces by supporting content dependent sharing – the applications are collaboration aware, and support synchronous multi-user manipulations of application-specific objects. This adds a new dimension to the kind of cooperation that can occur in collaborative problem solving, because it permits cooperative manipulation and browsing of objects in the context of applications that manipulate them. It supports cooperation in the design (problem-solving) phase, as well as in the analysis (review) phase.

3 Shastra – System Features

The Shastra architecture is described in detail in [2]. Here, we present salient features. The design of Shastra is the embodiment of a simple idea – scientific manipulation toolkits can abstractly be thought of as objects that provide specific functionality. These objects exchange messages, automatically or under user command, to request other objects to perform operations. At the system level, Shastra specifies architectural guidelines and provides communication facilities that let toolkits cooperate and exchange information to utilize the functionality they offer. At the application level, it provides collaboration and multimedia facilities allowing the development of applications in which users cooperate to solve problems. A synergistic union of these two ideas lets us design sophisticated problem solving virtual machines.

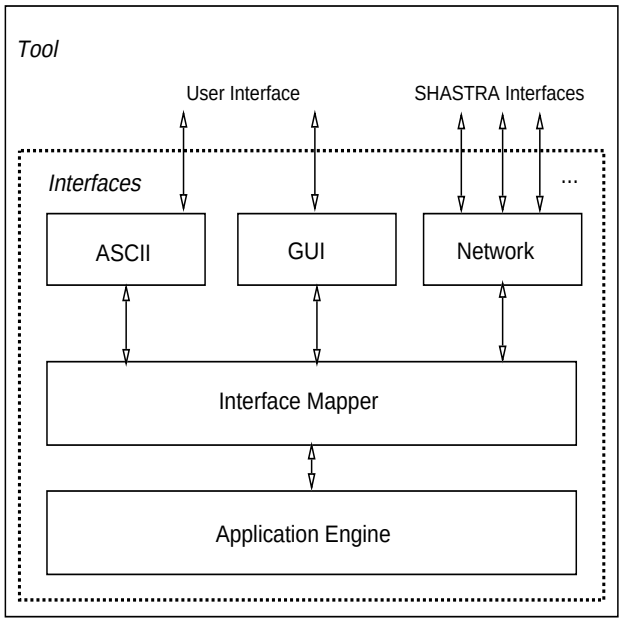

Figure 2: High Level Block Architecture of a Shastra Tool – Users interact with tools via the user interface. Tools interact simultaneously with multiple tools via the network interface.

The high level block architecture of all tools in Shastra is depicted in Figure 2. This architecture makes it easy for tools to connect to other tools and request operations, synchronously as well as asynchronously. A tool has an application specific core – the Application Engine, which implements the core functionality offered by the tool. Above the core is a Functional Interface Mapper which invokes functionality embedded in the Engine in response to requests from the the Graphical User Interface, ASCII Interface or the Network Interface. The GUI is application specific. The ASCII interface is a shell-like front end for the application. Tools communicate with other tools in the environment, via the Shastra substrate, through an abstract Network Interface, which multiplexes multiple simultaneous network connections, and implements the Shastra communication protocol [2].

The entire set of connected Network Interfaces of Shastra tools implements the abstract Shastra layer at runtime (see Figure 1). It maintains the collaborative environment, provides access to functionality of different systems, and provides facilities for initiating, terminating, joining, leaving and conducting collaborations.

3.1 Tools

Shastra Tools are the building blocks of the runtime system. Kernels and Session Managers are management tools, responsible for maintaining the distributed and collaborative environment. Shastra Toolkits provide scientific design and manipulation functionality, and Service Tools provide mechanisms for communication and animation. Toolkits and Service Tools are collectively referred to as Front Ends, or simply Fronts, since they are the actual sites of user interaction. Any Front can access the Shastra environment to instantiate tools locally or on remote sites, and to terminate previously instantiated tools. Fronts can connect directly to each other to exchange data in client-server or peer-peer settings using the Shastra substrate.

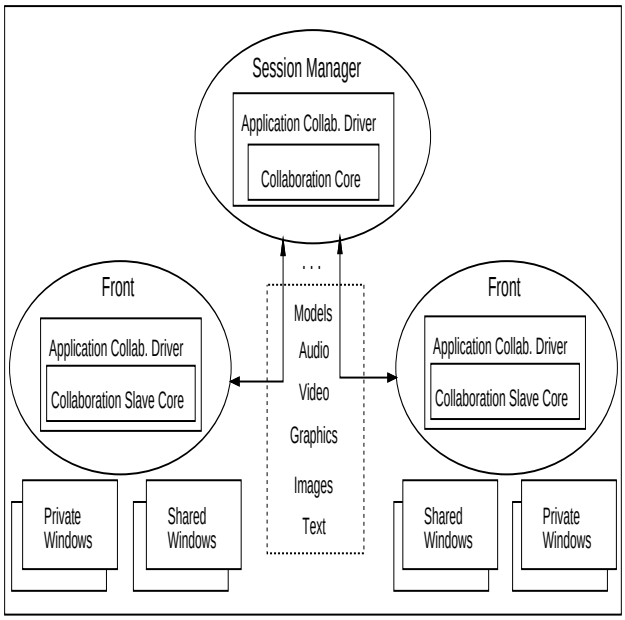

Figure 3: Architecture of a Collaborative Session – Fronts have separate private and shared workspaces. They maintain shared workspaces on behalf of the Session Manager. The substrate supports media-rich interaction between multiple Fronts.

3.1.1 The Kernel

The Shastra Kernel is responsible for maintenance of the runtime environment. It keeps track of all instances of tools in the distributed system. It consists of a group of cooperating Kernel processes. A Directory facility lets users dynamically discover what tools are active in the environment at any time. A Location facility provides contact information about where the tools are running, letting applications dynamically connect to each other to access functionality.

3.1.2 Session Managers

A Session Manager is a management tool in the Shastra environment. It maintains a collaborative session and handles details of connection and session management, interaction control and access regulation. It is a repository of the shared objects in a collaboration, and keeps track of membership of the collaborative group. A collaborative session in Shastra is started by a user through a Front. One instance of a Session Manager runs per collaborative session. The Session Manager provides a multicast facility needed for information exchange in synchronous multi-user conferencing. It has a constraint management subsystem which resolves conflicts that arise as a result of multi-user interaction, enabling maintenance of mutual consistency of operations. It also provides a floor control facility based on baton–passing. The architecture of a typical collaborative session in Shastra is depicted in Figure 3. Figure 4 shows a view of the Shastra world, where different tools interact to support a collaborative environment. The Shastra collaboration architecture uses a replicated computation model for the multiple user system – a copy of the application (the Front) runs at each site involved in the collaboration. The main benefits derived from this replication are heterogeneity and performance.

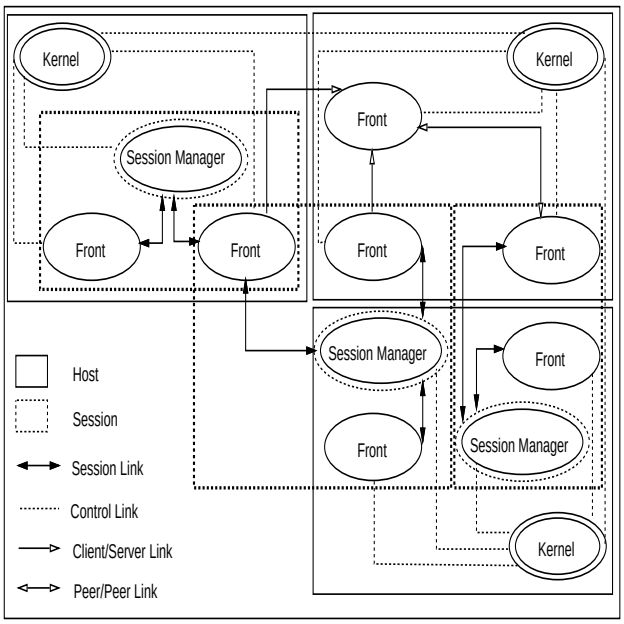

Figure 4: Information flow in the Shastra Environment – Sessions can span many hosts, and Fronts can participate in multiple independent collaborative sessions.

3.1.3 Toolkits

Currently Ganith, Shilp, Vaidak, Bhautik, Splinex and Rasayan are scientific tools under the Shastra umbrella. These toolkits are powerful standalone systems which operate on application-specific models. They have been integrated into the Shastra environment, and permit concurrent engineering and distributed problem solving by providing access to their functionality to other toolkits. This interoperability enhances the functionality of each toolkit.

The Ganith algebraic geometry toolkit manipulates arbitrary degree polynomials and power series [2]. It is used to solve a system of algebraic equations and visualize its multiple solutions. It incorporates techniques for multivariate interpolation and least-squares approximation to an arbitrary collection of points and curves, and C^1-smoothing using low-degree implicit patches. Other Shastra toolkits use the algebraic manipulation capability it provides at its network interface – curve and surface intersection, interpolation, and approximation functionality.

Splinex is a curve and surface modeling toolkit that provides interactive creation and manipulation of implicit and parametric splines in Bernstein-Bezier and A-spline bases [2]. It provides Bezier and A-spline surface manipulation capability in the environment.

Shilp is a boundary representation based geometric modeling system [2]. Current functionality of the toolkit includes extrude, revolve and offset operations, edit operations on solids, pattern matching and replacement, boolean set operations, fleshing of wireframes with smooth algebraic surface patches, and blending and rounding of solid corners and edges. These operations can be invoked by local users and by remote toolkits. Figure 5 shows a site during collaborative polyhedron smoothing in Shastra using Shilp and Ganith. Conferenced Shilp instances use multiple remote instances of Ganith to interpolate faces of a polyhedral car model in parallel, to produce a curved surface model with C^1-continuous surface patches. The toolkits communicate via their network interfaces, and Ganith services Shilp requests.

The Vaidak Medical Image Reconstruction Toolkit is used to construct accurate cross-sectional, surface and solid models of skeletal and soft tissue structures from CT (Computed Tomography), MRI (Magnetic Resonance Imaging) or LSI (Laser Surface Imaging) data. These models can be used by Shilp for design

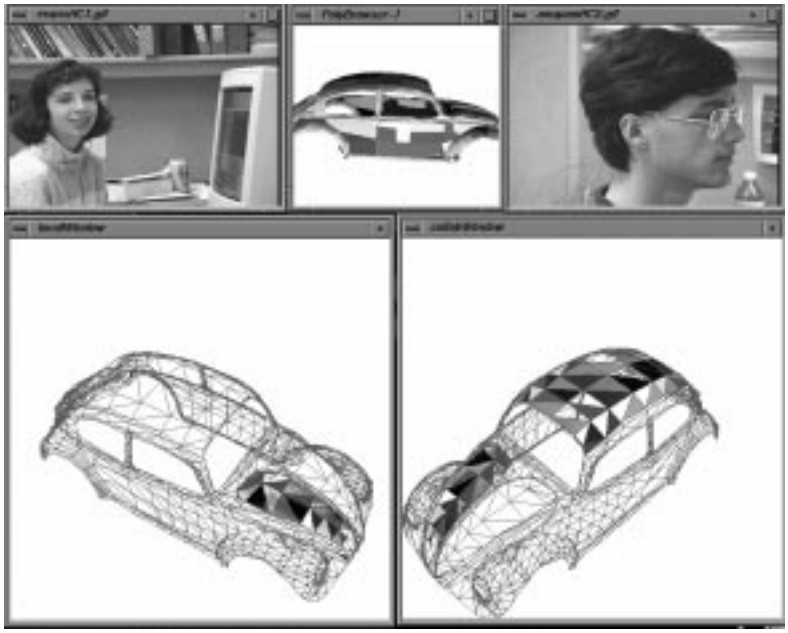

Figure 5: Collaborative Polyhedron Smoothing in Shastra – The original polyhedral car model (top-center), one designer's part of the shared task (bottom-left) and the shared, partially complete curved surface car model (bottom-right) are shown. Images of a supporting video-conference are also shown.

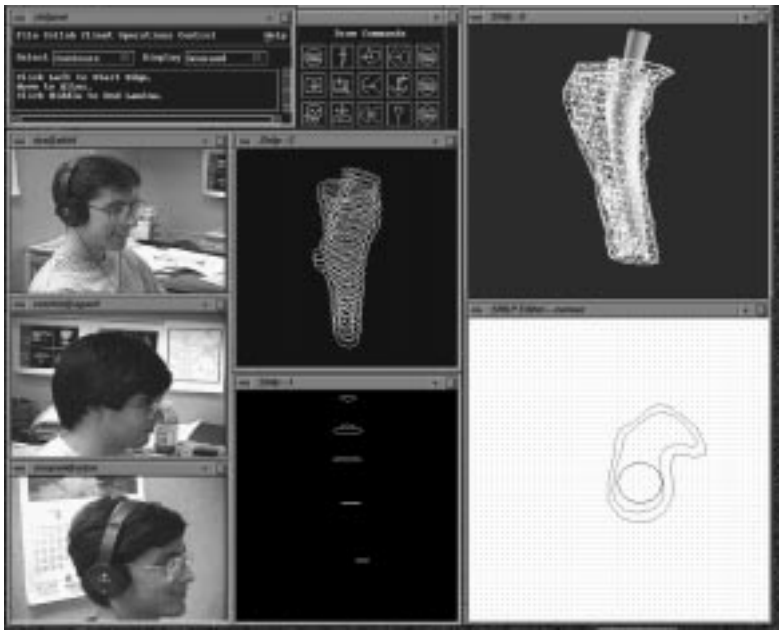

Figure 6: Collaborative Customized Hip Implant Design – A designer uses Shilp to interactively create a geometric model of a hip implant (right-top), by generating cross-sectional contours of the implant (bottom-center and right) from a sectional model of the femur (center) created in Vaidak. A video conference is used for communication.

Figure 7: Stress Analysis Visualization – A designer uses Bhautik to analyse stress under loading patterns to optimize custom implant shape for an artificial implant for a human femur (top-left). The designers use video-conferencing for communication.

activity, and by Bhautik for physical simulations. In a distributed problem solving scenario, a geometric model of a human femur is reconstructed in Vaidak (See Figure 6) and manipulated in Shilp.

The Bhautik physical analysis toolkit provides mesh generation facilities and a graphics interface to set up, perform and visualize physical simulations on geometric models created interactively using Shilp, or models reconstructed from imaging data by Vaidak. Figure 7 shows a load transfer finite element analysis used in custom design of hip implants [2], using Vaidak, Shilp and Bhautik toolkits.

Rasayan can compute and visualize the "docking" of drug and protein molecules under molecular Brownian motion. It provides mechanisms for analysis and visualization of the potential energy surfaces of the molecules and the stationary points on these surfaces.

3.1.4 Services

The current set of Shastra services contains communication and animation tools. The objective is to provide a media-rich communication substrate for the design of multimedia applications, by relieving application developers of the burden of low-level manipulation of devices and media formats. In the scientific setting, especially in design and analysis, most of the information shared by a collaborating team is oriented towards structured 3D graphics, and is typically application specific. However, inclusion of facilities for text, image, audio and video communication has greatly enhanced the quality of interaction, enabling the design of more sophisticated applications.

SHA-TALK is a text communication tool which supports synchronous n–way textual conversations. It is useful for designers who do not have multimedia communication facilities on the desktop (See Figure 8).

SHA-DRAW is a Shastra environment sketching tool, which facilitates the generation and display of simple 2–D pictures, using a rich set of primitive operations. A collaborative session consisting of SHA-DRAW applications lets a group of collaborators synchronously create and edit simple 2–D sketches on shared whiteboards. Figure 11 depicts one site in a collaborative sketching session.

SHA-POLY is a collaborative visualization and graphical-object browsing and manipulation tool. It

Figure 8: One Site in a 3–Way Text Conference – SHA-TALK provides a simple textual conferencing facility which is especially useful when other communication methods are unavaliable. Image bitmaps identify the owner of a text panel.

Figure 9: Shared Visualization – A group of researchers uses SHA-POLY to share volume visualization images of a head with cutaways (top-center and right), and a cadaver (center). The images are generated by Vaidak from large volume data sets.

Figure 10: Video Support for Design – A researcher (top-right) uses a live SHA-VIDEO window (top-left) to verify topological accuracy of a reconstructed femur model (bottom-left, bottom-right) in Vaidak.

supports shared viewing of 3-D models using different display and visualization techniques in a synchronously conferenced setting. Figure 9 shows one of three sites with independent private windows, and a shared conference window, for volume visualization of large medical data sets.

The SHA-PHONE service is used to record and playback audio information stored in multimedia objects. An n-way audio conference is conducted by setting up a collaborative session consisting of SHA-PHONE instances.

SHA-VIDEO handles image data (without sound) – both still images and motion video. It is used, both directly and by other tools, to playback and record video information stored in multimedia objects. A collaborative session consisting of SHA-VIDEO applications provides the mechanism to conduct a silent video conference. In Figure 10 a researcher uses a live video window to confirm the topological accuracy of a reconstructed femur in Vaidak.

GATI is an animation server that provides distributed and collaborative real-time interactive animation in two and three dimensions. The system supports a high level animation language based on a commands/event paradigm.

4 Collaborative Problem Solving

We now describe an example of multi-user cooperative design in the context of Shastra. We have built an application for collaborative set operation based design using Shilp and SCULPT, two Shastra toolkits. It permits a group of designers to cooperatively create a 3-D model by performing set operations on simpler models in Shilp using SCULPT as a back end to perform the actual operations. Shilp is a geometric modeling system. SCULPT is optimized to perform set operations – Union, Intersection, Difference and Complementation on polyhedral geometric models [8]. We use Shastra's application level cooperation substrate and user level collaboration substrate to link the two applications. The result is a powerful multi-user interactive design facility.

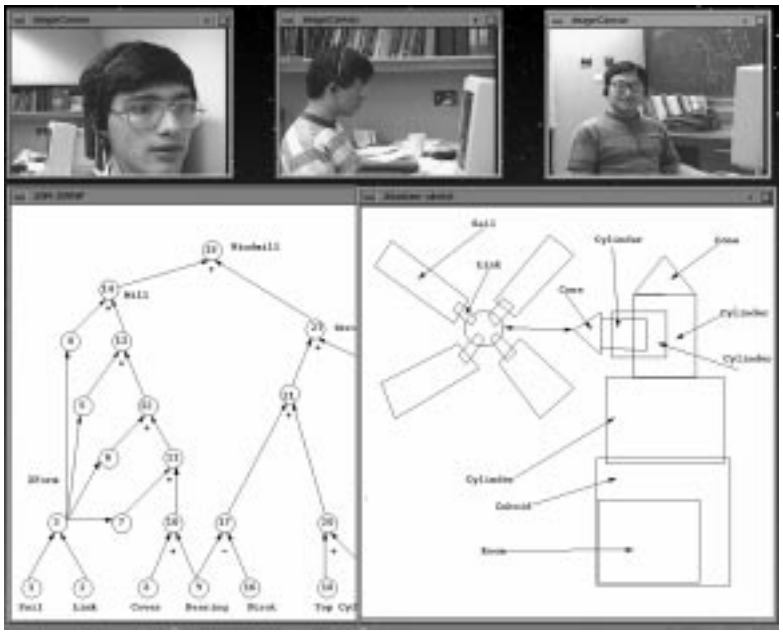

Figure 11: Using SHA-DRAW for Shared $2D$ Sketching – Collaborating designers (top row) use shared windows to create a sketch (botom-right) and a design graph (bottom left).

4.1 Motivation

The design paradigm in Shilp emphasizes on creation of complex models by performing operations on simpler models in a hierarchical fashion. Set operation based design (Constructive Solid Geometry) is a very flexible way of creating intricate $3-D$ designs. Conventional systems support this method in the single user setting. A designer creates a design by going through the multiple steps involved. This application presents a departure from the traditional method – a collaborative design environment where a group of designers cooperatively create large designs. It provides facilities to enable cooperation between multiple users.

In boundary representation based solid modeling systems, generation of the results of set operations is compute intensive. Also, the design process can be represented as a tree in which the lower levels are often parallelizable – a group of designers can work independently on those parts. The application improves throughput of the design operation by providing a collaborative environment from the conception phase through the final design realization phase.

4.2 The Startup Problem

One of the challenges of this design scenario is establishing a starting point. This issue is traditionally resolved by conducting a physical meeting where a design team is identified. Communication and information exchange between team members enables them to synchronize at a starting point, and gives them all an idea of the final design. Shastra eliminates the need for such a physical meeting by providing mechanisms to create a design group and media-rich support for a design brainstorming session (see Figure 11). In conjunction with a SHA-VIDEO and SHA-PHONE conference, SHA-DRAW provides a powerful interaction environment.

A user running an instance of Shilp, queries Shastra to find out other active users in the environment. He uses the messaging facilities of Shastra to invite some of them to a text conference, using SHA-TALK, and explores their interest in a particular design. Subsequently, he decides on his working group. Before the design process is started, some members of the team may have a mental picture of the object that they

are attempting to design, while others may not. One of the designers initiates a collaborative brainstorming session using SHA-DRAW, the multi-user sketching tool. If audio and video processing hardware is available, he invokes instances of SHA-PHONE and SHA-VIDEO, and initiates the relevant sessions. Sites without video hardware can use the software-only playback facility to display incoming video streams and generate outgoing streams. Audio-visual communication, provided by concurrent SHA-PHONE and SHA-VIDEO sessions (see Figure 11), leads to rapidly establishing the design goal. The group interactively creates a rough sketch of the intended design. Alternately, a stored image, or a live image of an actual physical object, or of its picture, can be broadcast to the group using SHA-VIDEO. At the end of this phase, the entire team has a good idea of the task at hand.

The designers use SHA-DRAW to set up the dependencies of various parts of the intended design in graphical form. A directed design graph, where nodes are solid models and edges are dependencies of the destination nodes, is subsequently created. The leaf (0 in-degree) nodes represent existing or primitive solid models, and internal nodes are intermediate models in the design process. A designated root node represents the final design goal. Directed edges indicate that the destination node is the result of an operation on all of the source nodes. Annotations in the graph indicate the operation needed to obtain the destination node from the source nodes (see Figure 11).

4.3 Design Outline

The operation is performed in two phases – design graph generation and model computation. The design graph, which is created in a SHA-DRAW collaboration in the context of a sketch or video image of the final model, is a succinct summary of the entire design task. The image and/or the sketch is stored with the graph for future reference. The graph is converted into a form amenable to this operation, with maximum in-degree of the nodes being 2, since only unary and binary operations are supported. An automatic DNF (Disjunctive Normal Form) decomposition is the simplest transformation, but doesn't produce the most efficient design graph. The design team cooperatively restructures the design graph, in a SHA-DRAW collaboration, to meet the requirements.

In the model computation phase, a designer graphically positions models using the Shilp user interface. This is done to set up models in appropriate configurations for set operations. The actual operations to generate intermediate and final models of the design graph are performed in Shilp by automatically requesting geometric services from SCULPT.

4.4 The Shastra setting

In a single user setting, a designer would compute the various nodes of the graph sequentially. The final model would be checked for goodness, and the computation process repeated till a satisfactory model was obtained.

In the multi-user setting, a collaborative Shilp session is initiated by one of the Shilp users in the environment. He specifies, to the local Kernel, the Shilp users who will be invited to participate in the session, and (by default) becomes the group leader. The Kernel instantiates a Session Manager, which starts a session with the group leader as its sole participant, and then invites the specified users of concurrently executing remote Shilp instances to participate. Users who accept are incorporated into the session. The Session Manager is responsible for providing access to shared objects and context at all participating sites.

Any participant can leave the session at any time, by simply de-linking from it. Users of other instances of Shilp in the environment can query the system to discover ongoing sessions, and request participation. The group leader regulates whether or not they are allowed to join the session. He can also invite other Shilp instances to participate in the session.

Every participating Shilp session creates two shared windows in which all the cooperative interaction occurs. More local windows can be created if desired. One shared window contains the design graph that is used to regulate the entire operation. The second window contains models as they are created or introduced to the session. Users introduce leaf node objects into the session by selecting them into the shared window.

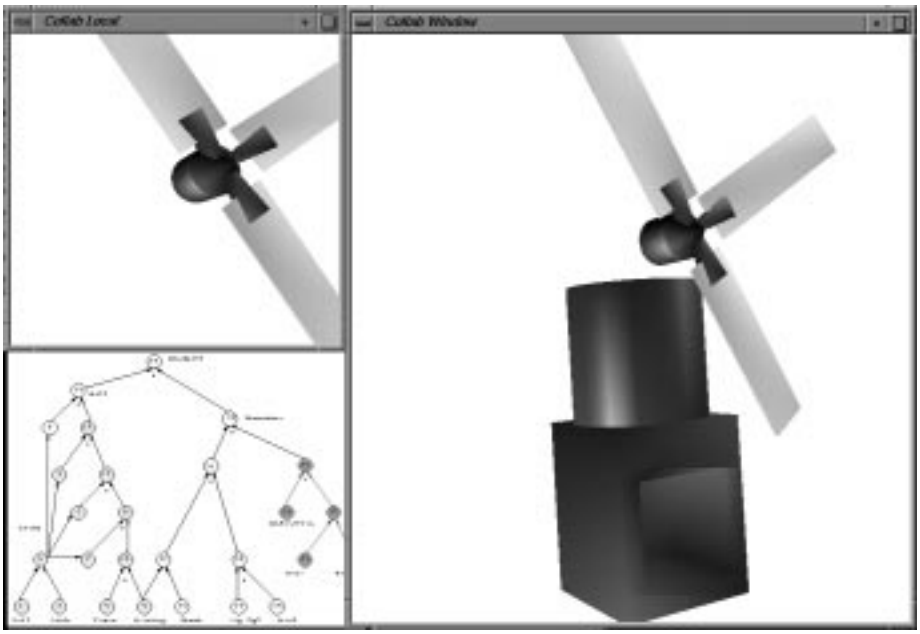

Figure 12: One Site in a Design Collaboration – The design graph (bottom-left) is used as a context to monitor progress and regulate the task. The designer sees the incomplete shared model (right) and the locally designed part (top-left).

The Session Manager partitions the design graph into slightly overlapping zones. The partitioning is based on the number of people in the collaborating group, and on the number of subtasks left in the operation (the number of uncomputed nodes in this case). It aims to minimize the number of shared nodes of partitions. Shared nodes in the graph are regions of contention in this collaboration scenario since they constitute dependencies in an otherwise parallelizable situation. The partitioning also aims to distribute the leaf nodes equally among the designers, since they usually represent nodes that have to be interactively created. It defines a scenario for fair, minimal-conflict cooperative interaction. The partitioning is dynamically altered as users join and leave the session. The group leader can explicitly specify and alter the partitioning. The partitioned design graph is displayed in a shared window, and serves as a context to regulate the collaborative operation, since it captures the state of the operation.

The partitioned zones are assigned to the collaborating designers, and are colored differently for identification. Every user is responsible for filling the intermediate nodes in his zone by first positioning the models on the incoming edges, and subsequently performing the actual set operation. This process is repeated till a satisfactory design is created. Figure 12 shows one site in the design of a simple windmill model. Figure 13 shows another site at the end of the operation.

4.5 Collaborative Interaction

The Session Manager regulates all interaction relevant to the operation at participating sites. Interaction can occur in two modes.

In the Regulated mode, the user responsible for a zone creates all the models internal to the zone. All other users are denied access to the interior of a zone by the session manager. If the source nodes for an intermediate node are filled, a user locks the intermediate node by selecting it in the Graph Window. The session manager allows him access to models in the source nodes. The user interactively positions these models and performs the appropriate set operation, and the resulting model is assigned to the intermediate node, which is subsequently unlocked. At the boundary of a partition, users of adjacent zones must agree

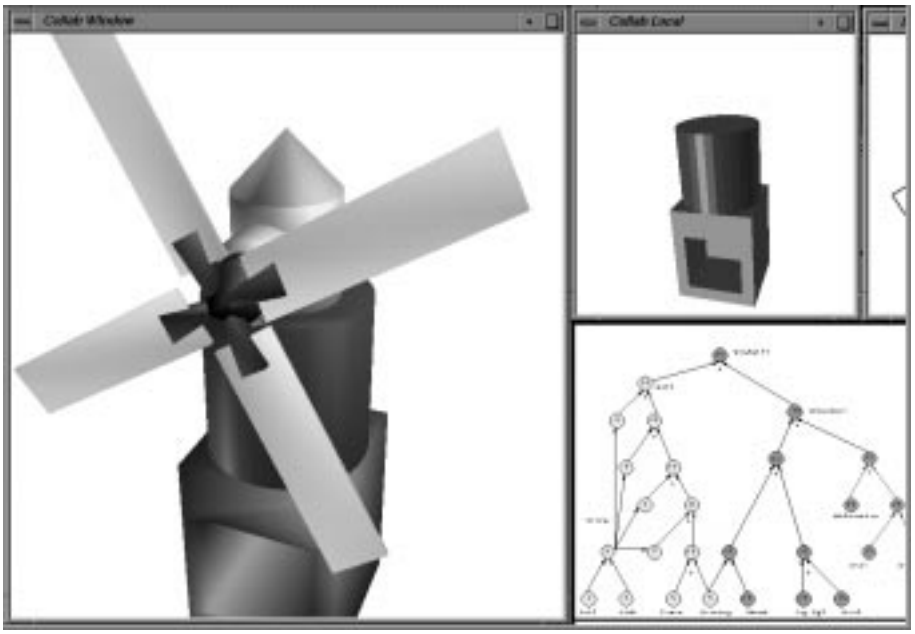

Figure 13: Another Site, at the End of a Collaborative Design Session – The designer sees a completed shared model (left), the locally designed part (top-right) and the shared design graph (bottom-right).

about the models at the boundary nodes so that inconsistencies are not created in the design.

A good group design protocol for this setting is to resolve boundary condition issues at the start of the operation, to prevent unnecessary cycles due to inconsistencies later on. This involves computing the subgraphs rooted at boundary nodes first, till satisfactory models at those nodes are obtained.

All operations are performed via the (central) session manager which is responsible for keeping all sites up-to-date, so that the users have a dynamically changing and continuously updated view of the operation in the shared windows – the design graph and intermediate models. Changing a node requires all of its dependencies to be re-evaluated. The operation is completed when all the nodes of the design graph have been evaluated. Any site with Copy permission can then extract the model from the session and save it.

4.6 Access Regulation and Collaboration Modes

The collaboration infrastructure of Shastra supports a two-tiered permissions-based access regulation mechanism. It is used to structure a variety of multi-user interaction modes at run-time. It allows a high degree of tailorability and flexibility in Shastra's CSCW applications in the domain of interaction as well as data sharing and access control. The regulatory subsystem supports **Access**, **Browse**, **Modify**, **Copy** and **Grant** permissions for collaboration sites as well as for shared objects [2]. These permissions control what actions different users in the conference can take, and what objects they can operate on. In addition, tools can define and use new permissions for tool-specific actions. Permissions are controlled by the group leader or his designees. The regulatory subsystem also provides a mechanism to enforce and regulate floor control based on turn taking. Users can dynamically configure the interaction mode and permissions to suit the task.

In the brainstorming phase, for example, it is useful to allow everyone equal access to all operations and objects, to support free flow of ideas.

In the Unregulated mode for this operation, the partitioning merely suggests a minimal-conflict setting, and the session manager doesn't regulate interaction beyond what is specified by collaboration permission settings for the site and the object. In this mode a user can access any node if he has **Access** and **Modify**

permissions for the collaboration.

The session manager allows only one user to manipulate a "hot spot" in the graph– where there is a possibility of contention – at any particular instant. It uses the first-come-first-served paradigm to decide which user gets temporary exclusive control. The last completed operation specifies the model associated with the node.

The baton passing facility of the system can be used for floor control – to take turns to set boundary nodes. Alternately, designers can use the auxiliary communication channels to regulate access, and to decide which users will set those nodes.

At one extreme, the Shastra implementation for collaborative design can be used by a single designer to design a solid model much like in a non-collaborative setting. Allowing other users to join the session with only **Access** and **Browse** permissions sets up the environment like an electronic blackboard to teach novice users the basics of the design mechanism. An appropriate setting of collaboration permissions and turn-taking can be used to allow hands on experience with the task. In conjunction with the audio and video communication services of Shastra, this becomes a powerful instructional environment.

In a different situation, a group of n designers can set up an Regulated collaborative design session and collaborate to design an object. Each designer performs only the designated part of the shared design, and a speedup of as much as *problem size / maximum partition size*, can be achieved. Novice designers can join the ongoing session with only **Access** and **Browse** permissions, and get familiar with group dynamics of a collaborative session. In yet another situation, a group of n designers can start an Unregulated collaborative design session. Judicious use of the auxiliary communication facilities (Audio, Video and Text) to regulate design operations in a cooperative manner can let the team acquire a speedup factor of up to n.

4.7 Heterogeneity Issues

A Shastra conference consists of multiple tool instances at different sites. This localizes platform specific dependencies in the tool. It permits the Session Manager to view tools as high level application objects, without having to concern itself with low level details of how things are actually done. This approach supports the Shastra system on a wide variety of hardware platforms. Specifically, tools can take advantage of available hardware graphics facilities, video compression and decompression, and audio processing hardware. This greatly simplifies multimedia interaction management.

4.8 Heterogeneous Collaboration

The above described application is an example of a homogeneous collaboration – the collaborative design task is supported on a collection of instances of the same tool (Shilp in this case). We are currently building an environment for collaborative design of custom hip and knee implants, using different toolkits of Shastra coupled with a computer aided manufacturing facility. This puts us in the realm of heterogeneous collaborations – here collaborations are supported between instances of different tools, which operate on different types of models or data.

The architecture paradigm of Shastra has greatly facilitated the kind of inter-application cooperation required to build such a system. The Medical Image Reconstruction toolkit (Vaidak) is used to build a model of the patients femur from cross-section information (CT/MRI images). A designer uses Shilp to custom design an implant for the femur (see Figure 6). A physical analyst using Bhautik conducts a stress-strain analysis to evaluate the load transfer occurring between implant and bone (see Figure 7). The design team iterates over this custom design process till an optimally shaped customized implant is obtained. Multimedia communication facilities in the form of SHA-VIDEO, SHA-PHONE and SHA-POLY conferences permit a rapid exchange of rationales for design choices, interpretations of analyses and iterative shape modification and analysis.

5 Conclusions

We have described applications that demonstrate that collaboration in the scientific design setting is facilitated by multimedia support as well as information exchange. Shastra is a powerful distributed and collaborative toolkit prototyping environment. It provides a rich substrate for design of such systems. The multimedia aspect brings powerful communication primitives to the desktop. The integration of $3-D$ graphics into the environment adds a new dimension to the potency of this environment, as visual processing on powerful graphics engines becomes more common. Collaboration support in the environment, in the form of communication facilities and application development substrate, makes it easy to develop synchronous multi-user applications, and problem solving tools.

Though a scientific manipulation environment has been the focus of our implementation, the facilities easily abstract out to a variety of situations requiring similar substrates. The collaborative layer is generic and can be used to implement the heart of systems for collaborative editing, code viewing and quality assurance tools, software development environments, multi-user electronics CAD, architecture CAD and mechanical CAD tools, and interactive multi-player games etc.

Acknowledgements

A preliminary version of this paper, titled "Collaborative Multimedia Scientific Design in Shastra," appeared in the Proceedings of the First ACM International Conference on Multimedia, Anaheim, California, ACM Press, August 1993, 447-456.

References

[1] Ahuja, S., Ensor, J., Horn, D., (1988), "The Rapport Multimedia Conferencing System", *Proc. ACM SIGOIS*, Mar. 1988.

[2] Anupam, V., Bajaj, C., (1993), "SHASTRA - An Architecture for Development of Collaborative Applications", *Proc. of the IEEE Workshop on Enabling Technologies and Integrated Concurrent Engineering*, Morgantown, West Virginia, April 1993, IEEE Computer Society Press, pp. 155-166.

[3] Carlbom, I., Hsu, W., Klinkner, G., *et al*, (1992), "Modeling and Analysis of Empirical Data in Collaborative Environments", *Comm. of the ACM*, June 1992, pp. 73-84.

[4] Craighill, E., Lang, R., Schlager, M., Garcia-Luna, J., (1992), "Environments to Enable Informal Collaborative Design Processes", *Proc. First Workshop on Enabling Technologies for Concurrent Engineering.*

[5] Crowley, T., Milazzo, P., Baker, E., *et al*, (1990), "MMConf: An Infrastructure for Building Shared Multimedia Applications", *Proc. ACM CSCW 90*, Oct. 1990, pp. 329-342.

[6] Ellis, C., Gibbs, S., Rein, G., (1991), "Groupware: Some Issues and Experiences", *Comm. of the ACM*, Vol. 34 No. 1, Jan 1991, pp. 38-58.

[7] Mercurio, P., Elvine, T., Young, S., *et al*, (1992), "The Distributed Laboratory", *Comm. of the ACM*, June 1992, pp. 54-63.

[8] Naylor, B., and Thibault, W., (1987), " Set Operations on Polyhedra using Binary Space Partitioning Trees", Computer Graphics Vol. 21 No. 4, 1987.

[9] Patterson, J., Hill, R., Rohall, S., Meeks, M., (1990), "Rendezvous: An Architecture for Synchronous Multi-User Applications", *Proc. ACM CSCW 90*, 317-328.

[10] Stefik, M., Foster, G., Bobrow, D., *et al*, (1987), "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings", *Comm. of the ACM*, 30, (1), 32-47.

[11] Srinivas, K., Reddy, R., Babadi, A., *et al*, (1992), "MONET: A Multi-media System for Conferencing and Application Sharing in Distributed Systems", *Proc. First Workshop on Enabling Technologies for Concurrent Engineering*, Vol. 1, pp. 21-37.

[12] Watabe, K., Sakata, S., Maeno, K., *et al*, (1990), "A Distributed Multiparty Desktop Conferencing System", *Proc. ACM CSCW 90*, 27-38.