



## A Flexible Context Architecture for a Multi-User GUI

Yue Xu<sup>1</sup>, Edward Red<sup>2</sup> and C. Greg Jensen<sup>3</sup>

<sup>1</sup>Brigham Young University, [yue.xu@pace08.com](mailto:yue.xu@pace08.com)

<sup>2</sup>Brigham Young University, [ered@byu.edu](mailto:ered@byu.edu)

<sup>3</sup>Brigham Young University, [cjensen@byu.edu](mailto:cjensen@byu.edu)

### ABSTRACT

This paper focuses on flexible context architectures for a multi-user GUI (MUG) that serves several users engaged in collaborative computer-aided applications (CAx). Our work extends previous research into multi-user GUI's by proposing a flexible context interface between users working on the same part at distributed locations. The investigation will consider how distributed users, through user interfaces, interact to simultaneously build models and how interaction context might be presented to regulate the way they wish to interact.

A prototype implementation integrates a MUG with NX Connect, a multi-user CAD prototype for Siemens NX. NX MUG uses agent software to render the user prototype outside of NX and NX Connect. This generalizes the interface so that it could be used with other engineering applications where several users wish to collaborate. NX MUG enables users to view a collaborating user's workspace, send/receive messages between multi-users, and is capable of translating text interactions in different languages, while Skyping with other users. This research will enhance the existing NX Connect multi-user prototype by providing collaborative interaction support among the multi-users.

**Keywords:** collaborative design, multi-user CAx, multi-user GUI.

**DOI:** 10.3722/cadaps.2011.479-497

## 1 INTRODUCTION

Product development practices assign the steps of design, analysis, and process planning to specific departments and ultimately to responsible individuals, all within a highly controlled serial process [18]. The collaborative goal is to co-develop parts by designers distributed over different locations where "the associated interfaces run across a number of workstations" [2].

Single user architectures, including the user interfaces that characterize both commercial computer-aided application software and computer operating systems, greatly inhibit collaborative engineering, in spite of the numerous research efforts into product team cooperation. A flexible context allows "the activities of one user to be reflected on other users screens and is supported by sharing

application information” [2]. Bentley et al. [2] state that sharing is “the principle means of promoting cooperation, and the real-time presentation and manipulation of shared information is the main function of cooperative, multi-user interfaces”.

The objective of this paper is to build flexible context architectures into a multi-user GUI (MUG) that serves several users engaged in collaborative CAX. We have already developed a multi-user client-server collaborative software prototype at BYU called NX Connect. We used the Siemens NX API (Application Programming Interface) to extend the single user commercial CAX software to a multi-user environment where several users can simultaneously construct and edit the same model. NX Connect does not have a formal multi-user GUI to provide a desired flexible context for the collaborative activities. This paper will extend previous research into multi-user GUI’s by proposing a flexible context interface for multi-user interaction, and also provide a simple prototype using currently available tools. The investigation will consider how distributed users, through user interfaces, interact to simultaneously build models, and how interaction context might be presented to regulate the way they wish to interact.

## 2 REVIEW OF MULTI-USER TECHNOLOGIES

This section uses six categories to examine previous work: 1) multi-user GUI’s; 2) ideal-user interface; 3) client – server application; 4) existing MUG architectures and infrastructures 5) multi-user awareness and agent software; and 6) motivation and challenges

### 2.1 Multi-User Interface

In a global collaborative design environment there is the need to co-develop parts by designers at different geographical locations. Geographical separation makes it difficult to frequently gather involved parties and departments for a meeting [5]. Sosa notes that even engineers who sit in the same room and work on different parts of the same model create conflicts when they try to assemble the parts together [23]. Macfarlane suggests that if engineers start a complicated model on the same screen, they may avoid “no match” situations before assembling the parts [17]. A multi-user environment challenges system capacity and reveals problems like process collisions [20] among collaborating users in the same space.

One example of a multi-user interface is CoMaya [28]. CoMaya converts single-user Maya into a real-time collaboration using Autodesk and Verse tools. CoMaya allows two or more users to collaborate simultaneously to make changes to a single file. The session maintains the original user interface and functionalities of Maya, but adds the CoMaya user interface as well. User A’s actions immediately appear on User B’s workspace in real-time. CoMaya uses target object high-lighting and semantic sound effects [28] to indicate concurrent work. Either of these effects can be applied to the same object and attribute, or to the same object with different attributes. With fast local response, users can modify a part of the model, or navigate different regions of the model, from different perspectives.

### 2.2 Ideal User-Interface

An ideal user-interface (UI) will help a target group easily access the application and reduce the support cost. One UI design principle is consistency, which enables the users to “build an accurate mental mode of the way it works” and require less training and support costs [31]. Similarly, most collaborative design companies try to retain the single user interface and add multi-user view windows or view sessions.

An ideal UI will deliver the messages effectively. Equivalently, an ideal multi-user UI will promote cooperation and multi-user interaction. It will provide users with encouragement and help collaboration take place with one user’s awareness of others’ activities. Most existing multi-user applications produce the illusion that the users are the only ones in the system. This illusion makes the users unaware of the presence of others and prohibits collaboration. A good way to overcome the illusion and enhance multi-user awareness is to establish and maintain a common context. This context is communications between users of different cultures and personalities, and “allows the activities of

one user to be reflected on other users' screens that are supported by sharing application information" [2].

### 2.3 Client – Server Application

The Client-Server (CS) architecture, also known as a centralized architecture, uses a central server program to process user input events and display output events which are routed by way of local client programs [2]. This architecture simplifies access management and data consistency. Accordingly, the great advantage of the CS approach is simplicity. But this architecture does not support end-user tailoring, and it is vulnerable to "failure of the central node (server), and delayed feedback as all input and output events must travel over a network" [2].

The literature considers two CS architectures that enable co-design: thin server/strong clients; and strong server/thin clients. In the former the server exchange and broadcasts CAD information from one client to client. The server, as a registration system for multi-user application session, acts as a messenger between clients. Clients are "equipped with whole CAD systems and some communication facilitators" [29]. In the latter architecture the server side carries out the major modeling activities with visualized-based manipulation clients [24].

Our prototype, called NX Connect, is built on a thin server with strong clients. The server acts as a messenger to distribute model changes between collaborating users while maintaining all necessary information. Each client is equipped with the full CAx application and generates a local copy of the part file that can be updated during the design session. Since each client application performs the computations, a real time multi-user experience is delivered.

### 2.4 Existing MUG Architectures and Infrastructures

Multi-user applications essentially rely on "an underlying distributed communication layer to perform message passing between computers" [8]. Most of the architectural styles found in multi-user systems follow a philosophy where application data (the model) is kept separate from interface code (the view) [15]. Collaboration transparency follows this style. It will not modify the actual application after collaboration support is added to the application [8]. Such systems handle collaboration implicitly and "replicate display output and adopt an approach based on sharing the presentation of an application" [2]. Therefore these systems give each user "a common frame of reference" or a common context, which support the What You See Is What I See (WYSIWIS) context [2][15]. However, pure WYSIWIS sharing cannot be used on MUG development, especially when multi-users build the same model where different tasks need to access shared data. Different users may require "different presentation-level or WYSIWIS information sharing" or some of the users may consider all levels of presentations. Richard, et. al [2] state that collaboration transparency cannot provide this level of flexibility because the system is required to exploit knowledge of the shared task undertaken by each user. If that is the case, the alternative approach is a collaboration aware application.

In contrast to the transparent approach, the collaboration aware application develops special purpose applications, which recognize levels of cooperation and handle collaboration explicitly. The collaboration aware application provides facilities to "explicitly manage the sharing of information," allowing sharing to be presented in a range of different ways to different users [2]. This approach often demonstrates how the information is performed and modified within the application, and how the decision of MUG's management is embedded in the protocol. Furthermore, the collaboration aware applications lack a supporting infrastructure and inhibit interface tailoring of sharing policy, which often require the developer to conduct the application from scratch [2]. So the transparent approach is more popular for developers to apply to MUG applications.

Besides the transparency approach and collaboration aware applications, there are other methods available - blackboard architectures and Groupware toolkits widgets. Gelernter, et al. [7] and Martin, et al. [16] point out that Blackboard Architectures allow "multiple processes to communicate by reading and writing information from a global data store" and provide "a flexible framework for problem solving by a dynamic community distributed process". Blackboard architectures are "an associative memory with multiple independent cooperating knowledge sources, competing hypotheses, levels of abstraction and feedback to the sources" [1]. Another option is Groupware toolkits widgets. Hill, et al. [7] suggest that Groupware toolkits provide reusable GUI widgets and communication infrastructure

components [10]. It also “integrates with a set of popular Integrated Development Environment (IDE)” [10]. The user interface of multi-modal interaction supports multiple forms of interaction, such as a combination of speech and gesture [4][3].

## 2.5 Multi-User Awareness and Agent Software

Multi-user awareness is “the up-to-the-moment understanding of another person’s activities in a group environment” [10]. This includes “information about who is using the system, where they are working, and what they are doing” [10]. Gutwin [8] suggests that multi-user awareness is “an important part of collaborative activity. If workspace awareness is difficult to maintain, collaboration becomes more difficult”.

Agent software is a popular and effective approach to increase multi-user awareness between users. Researchers have had experience in multi-user interfaces for many years. Liu, et al. [26][18], propose that independent agent-based methods be deployed to enhance the multi-media communication between collaborating personnel. The intelligent agents can apply rules to communicate among themselves and distribute change information among collaborators.

P2P (peer to peer) is “any distributed network architecture composed of participants that make a portion of their resources (such as processing power, disk storage or network bandwidth) directly available to other network participants, without the need for central coordination instances (such as servers or stable hosts)” [21]. Each site has intelligent agent software to support interaction between multiple users. Agent software can transmit the collaborative network and capture audio and video streams locally for each user; meanwhile another agent might record design changes, and update them between users based on users’ privileges. Douglas, et al. [6][18], apply an agent-based approach to facilitate real-time interactions between player entities in a P2P game architecture.

Shen[22] proposes Augmented REALITY concepts that display models between the collaborating team members, but only one member has the right to modify the model. Wallace, et al [26][18], “discuss a multi-user X window application that associates multi-colored cursor software to time-share the system cursors among all the users.” User conflicts occur without carefully orchestrated interactions [9]. “Multi-cursor ICEWM has allowed multiple users to work on different applications, but the window manager is still time-slicing single system cursors [25][18]”.

## 2.6 Motivation and Challenges

Many of the multi-user GUI’s for the research prototypes that we have identified orchestrate the user interactions, because the interfaces have no flexible means of interpreting the actions among the users, or even regulating how they choose to interact. It is one thing to continually update the screens with all changes among users, but it is another thing to investigate what type of change information might be useful among a set of users and to permit the regulation of that data among the users. It is also necessary to provide an informative collaborative environment for multi-users with awareness information [10]. “Generic, customized, and groupware-specific components” [9] should be contained in the MUG.

Compared to the widgets in a single user interface, distributed communications protocol and the rendering and visualization of multi-user data are fairly complex in a multi-user environment. First, a multi-user interface needs to express which user is managing a component. Second, a multi-user interface needs to determine whether a local user or a remote user manipulates the widget [10]. The visualization for remote users is often different from that of local users within a widget [10].

## 3 CONTEXTUALLY FLEXIBLE MUG ARCHITECTURE

This research has the following objectives:

1. Investigate and develop architectures for multi-user interfacing of localized and distributed collaborators applying CAX applications, including agent software that can be configured for user context preference.
2. Determine how to integrate filters to provide personalized and selective contextual presentation of collaborating user interactions. A filter provides a function/algorithm to limit collaborating data

input, to provide a different view of that data, or to promote multi-user awareness. Filters could be mapped to Multi-User GUI (MUG) menu items or selection icons, etc.















3. Develop MUG prototypes that are consistent in look and feel with current commercial CAX GUI's, but can be invoked or hidden. The intent is to develop software libraries that can be linked to CAX API libraries, and then eventually become a member of the API library after commercialization by the CAX company. In addition, a prioritization method will be considered that allows manager oversight of multi-user sessions.
4. Demonstrate that MUG's can enhance multi-user CAX; and that product personnel of different cultures can navigate CAX collaboration effectively, in spite of cultural differences.

The following sections outline the proposed MUG architecture that provides user flexibility to configure the contexts to meet the interface styles and cultural constraints for each user in a collaborative session. The following sections describe a limited MUG prototype built on current API's and other software to emulate many of the desired features for the objective MUG. We then compare the desired MUG with our prototype to show current limitations in CAX API's, CAX architectures built for single users, and current computer OS's and associated networking technologies.

### 3.1 MUG Architectural Concepts

As an interface layer on top of the CAX application the proposed MUG will separate the model from the view as discussed in section 2.4. Contextual changes in a user's MUG will not affect the application data inherent to any Multi-User Software (MUS) used to communicate CAX application data among multi-users. NX Connect is one example of a MUS. The MUG architecture should also be general enough to integrate into multi-user CAX applications regardless of distribution network type, such as CS, P2P, hybrid forms of CS and P2P, or cloud serving.

Tab. 1 shows a proposed context list. Multiple icon selection sequences could be used to vary context preferences for each user in a collaborative session.

													
MUG On/Off	Multiple Displays	Divided Display	Timer	Security Tokens	Emotive	Text	Translation	Video	Audio	Skype	Record	Supervisor	MUG ISM

Tab. 1: MUG contexts.

#### 3.1.1 Context preferences

Users will have a variety of options for communicating with each other under preferred context settings.

*Display control contexts* - Each user will set a display mode context, which identifies whether user remote sessions and their screen displays are to be observed locally, and, if so, how and how often? Context preferences will be set by each multi-user to 1) share an entire screen between selected multi-users; 2) share only the CAX workspace screen, not including the GUI; 3) set the timing for refreshing multi-user windows on a user's local display(s); and 4) if the user is the supervisor, set security tokens that limit the model information transmitted between multi-users. For proprietary reasons each multi-user will have rights, from none to full access, and can block certain aspects of model viewing by other multi-users.

Effective multi-user display contexts will depend on the related display equipment, being most effective in rooms where large multiple displays can be used to mirror the session windows of networked multi-users. Multi-user display collaboration will be limited on a single display screen, unless the display is very large or high resolution projectors are used to partition multi-user sessions on a large wall screen.

Data compression techniques will be very important to propagating multi-user session display data among multiple users, although techniques such as windows terminal sharing and HP's Remote Graphics Software (RGS) clearly demonstrate that network bandwidths are capable of mirroring

distributed terminal displays, also capturing mouse cursor and keyboard events relative to a screen layout.

*Communication contexts* - The proposed MUG will allow multi-users to communicate by 1) video, 2) audio/voice, 3) text, and 4) emotive icon. Each user will set communication context preferences, unless a supervisor overrides the preferences. *Video contexts* depend on each user having a webcam or other video source, and enabling it for multi-user viewing. Each user will have the right to specify those multi-users to view, and filter out those not to view, unless supervisor override.

Using *audio contexts* and related equipment (headphones, external microphones, etc.) a multi-user can enable other users to enter an audio session, or allow the user to send voice mail to a multi-user. When combined with video interfacing, this can be an effective way to exchange production information and design details. Modern VOIP and Skype applications abound with this capability already. Researchers at BYU have embedded Skype capabilities into a Siemens NX CAX application, calling the prototype NX Skype, but the prototype does not include video collaboration.

Audio contexts may not be effective when different cultures are collaborating and located in different time zones. In this case each user will set a *text context* that would include no text communication allowed, or text communications allowed in a language of preference and with a notification context, such as text inbound alert followed by user response, streaming without requiring user response, or a fixed time interval before text viewing or response. In the fixed time interval mode text is buffered for later viewing and response; this mode is important to users located in different time zones. When different cultures are texting, the user can set a translation context which consists of 1) no translation, show in native language received, or 2) translate into current language preference.

Icons will provide a way for users to express themselves emotionally, and provide an effective rhythmic order for users to visually manage applications and improve awareness in a collaborative environment. *Emotive context* preferences will enable users to use emoticons in their communication, and to relate them to the audio and texting communications between multi-users. Emoticons within a message can enhance dialogue and lighten the mood. "Humor and sarcasm don't come across well in text" [30].

*Supervisory control contexts* - A future MUG will provide organizational contexts, by providing both a teaching context and a management context. In the teaching context a supervisor/instructor can remote control another user's cursor while tutoring the user. A supervisor can also assign tutor authority to any multi-user team member under restrictions, such as limiting the power distance [11]. A supervisor will be able to limit multi-user model access, including team participation scope, and participating timelines. Some participants will only access part of the team resources while others may have full access. The supervisor will be able to delete a user's account or change the user's role on a team.

*Security contexts* - Although mentioned earlier, security contexts will allow supervisors to set the collaborative access rights to CAX application model data. Extending participation rights to supply chain personnel who can identify current problems in obtaining material resources, or supply chain limitations in globally distributed design and manufacturing facilities, will counter potential resource problems early on.

*Recording context* - The proposed MUG architecture recognizes the importance of multi-user CAX for generating product development histories, including critical design and manufacturing intent and decisions. This information is almost impossible to attain/maintain in the current single user application interfaces. Thus, each MUG user will set a record context to record or not record multi-user session data, along with several contextual preferences to designate which of the communication means are to be recorded: audio, text, emoticons, etc. It is assumed that the MUS application will record the CAX model changes and maintain a master product model.

### 3.1.2 MUG modules

A MUG is broken into two modules: MUG Client Application (MUG CAPP) and a MUG Information Storage Module (MUG ISM). Each multi-user applies a CAPP to program and regulate the interaction with other multi-users logged into an application session. A CAPP will be deployed differently depending on the network type: CS, P2P, or cloud, but the basic functionality will not change. The ISM functions to store critical user authorization information, e.g., each user's technical background and cultural information, at a critical location server.



The CAPP provides tools and a GUI to serve two primary functions: 1) program and store contextual preferences for multi-user collaboration; and 2) transmit, receive, and store session data propagated between multi-users. Common terms used to describe the tools used are MUG filters and MUG agent software, respectively. A MUG filter is any method, such as a widget application, used to set the flexible context state, whereas agent software can be configured to package and transmit multi-user session data, according to the current filters set by the user. This data would include user contexts, user awareness information (there/not there, active/not active, session type, security restrictions, etc.), filtered application data packets (what application data is transmitted and viewable; e.g., some users may be restricted to viewing only a subset of a design assembly), and network communication speed/stability data.

Other widgets will expand the preference settings. For example, A Radar View Widget can reflect a single user's activities on other user screens. A Text Widget can provide text display and allow users to display and edit texts. A Translator Widget can bring real-time, in-place translating services. A security key widget can ensure all files are transferred in a secure environment.

Each user will use their GUI CAPP to connect to other users. The user information data is stored by

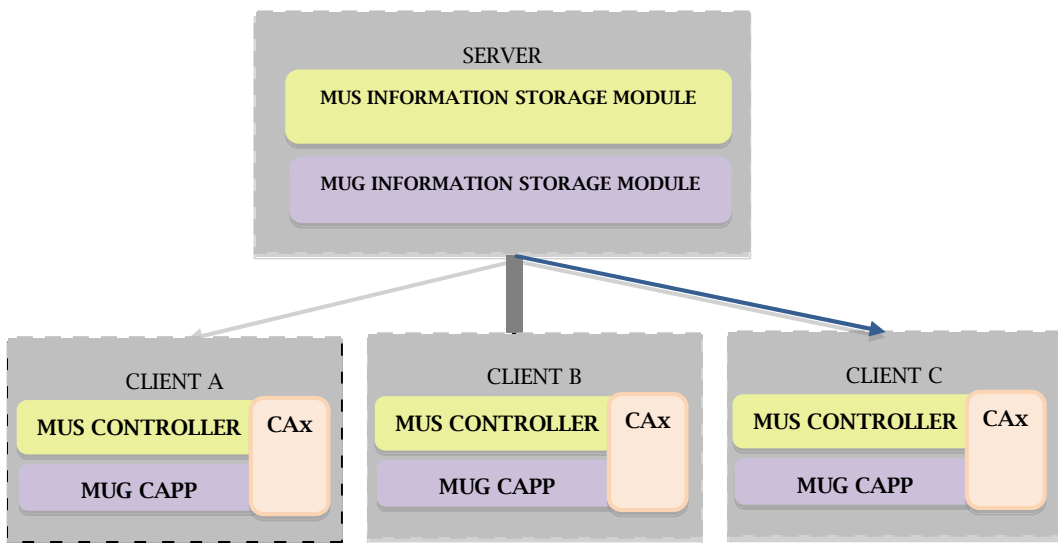


Fig. 1: MUG architecture in CS mode.

the MUG Information Storage Module and can be quickly accessed by other users' MUG. The number of new features for managing interaction contexts will increase with advances in collaborative networks and collaborative CAx applications.

The MUG Information Storage Module will store authorization information and user identification information. This module interfaces to the MUG CAPP of each client and stores or retrieves data as needed by the client's MUG CAPP. The module stores MUG context-related information and organizes data types and user information. The MUG architecture is general enough to be distributed in several network architectures: 1) CS; 2) P2P; or 3) cloud server.

### 3.1.3 MUG in CS mode

Fig. 1 shows how the proposed MUG components are deployed over the Server and Client when multi-users engage in a collaborative Multi-User Software (MUS) application. Presently, MUS prototype software such as NX Connect acts as an application layer on top of single user CAx applications, where API calls are used to observe, record, and transmit data modifications among multi-users. Future versions of CAx applications will integrate this functionality directly into a CAx application interface.

A server will include both the MUG ISM and the MUS ISM. But at each client computer, a MUS Controller will convert design modification information into primitive values and transmit them to the

MUS ISM. The data will be stored and transmitted to multi-users according to permission and context settings. At each user station the CAX application will construct corresponding features in their CAX application, depending on the permission settings for each user and the desired contexts.

There are some important questions still to be answered. Where should user context filters and permission filters be applied - at the user computer, at the server, or at both? Should all multi-user session data be sent to the server, where each user's context and security filters can be stored and thus applied? This method permits all multi-user and application data to be recorded at a single server, enhancing session security, and minimizing data hacking at a user's computer, but will certainly increase data network density and server data storage requirements. Conversely, applying filters at each user's computer to strip down the data packets will mandate more sophisticated and localized security software to protect proprietary data from unauthorized viewing/storing, but minimize the data to be sent.

The MUS controller function is critical. If data capture and data synchronization in the controller are mainly processed at the client side, as shown in Fig. 1, a CS with a thin server and strong client will be utilized. If the data related to multi-user activities are computed on the server side, a strong server with thin clients will be utilized.

### 3.1.4 MUG in P2P mode

Fig. 2 shows how the MUG components could be deployed in the P2P mode. Peers will function as both clients and servers to other peers as shown in Fig. 2. MUS and MUG ISM's will reside in each peer computer and be linked to each other in the overlay network. Each peer will employ an interfacing protocol or flood through the network to route a search to other peers for multi-user session data. In P2P networks, once a network opens up to the public domain, users begin to access a portion of resources. The more peers that join the network, the more resources become available to each peer. This may cause team-shared files to be unsecured.

For example, an ideal P2P architecture may not be secure for the data storage requirements of a complex CAX collaboration. It seems there has to be a central infrastructure node, which serves to enforce security and store the "master changes", even though the MUS and MUG are deployed over all the peers. This central node might be a supervisor's peer computer or any other peer computer so designated.

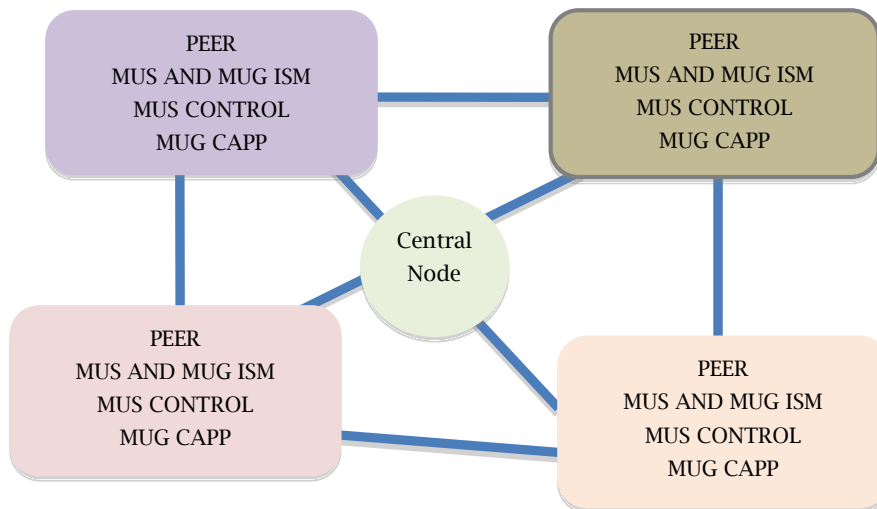


Fig. 2: MUG architecture in P2P mode.

In an ideal P2P case all data is passed among the users and filtered at each peer computer, including sensitive data that should not be viewed by all multi-users. The central node infrastructure, which deviates from the ideal P2P, would store sensitive data in the central node and not allow it to be



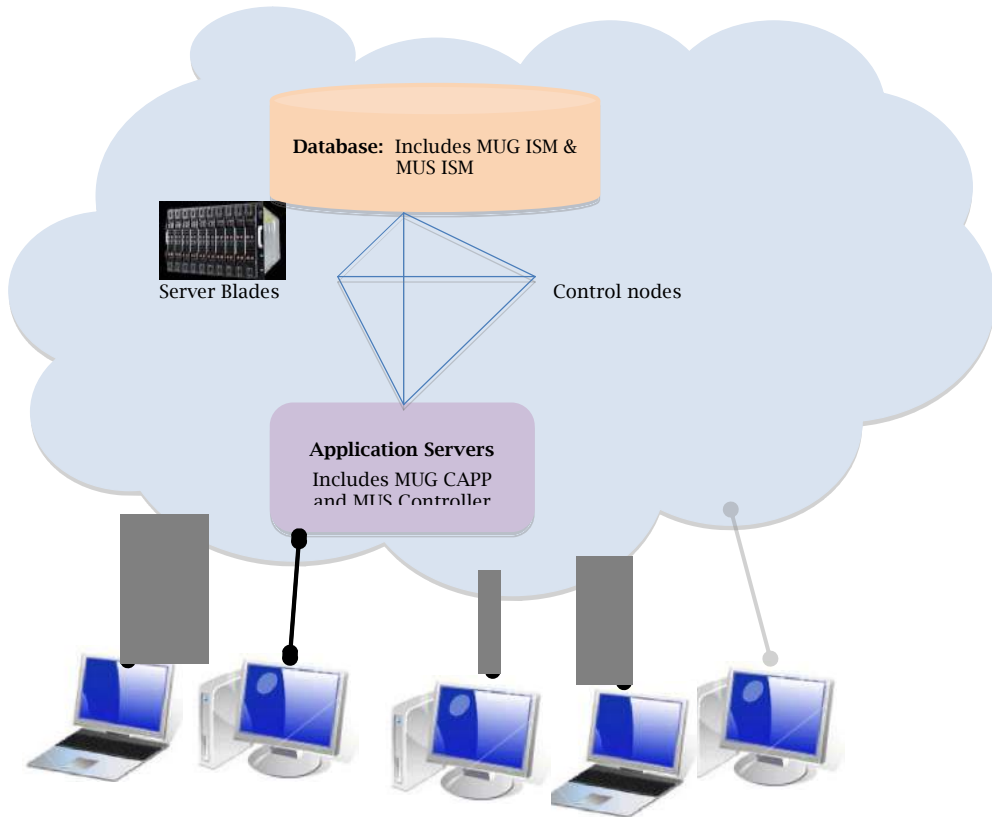


Fig. 3: MUG architecture in cloud serving mode.

generally distributed to the peer computers. Each peer could then remotely access this data but not make a copy in the local machine. The central node will pass and filter the regular data as other peers but have a higher data storage requirement to process and store sensitive data.

### 3.1.5 MUG in cloud serving mode

The MUG ISM and MUS ISM will be stored on several virtual servers hosted on a large data center labeled as "Database" in Fig. 3. MUG CAPP and MUS applications will be stored in cloud application servers that are labeled as "Application Servers".

Typically "Application Servers" have many blades, each of which can host applications. "Central Nodes" in the figure will be programmed to monitor the network traffic and administer the system to make sure everything runs properly by reporting event information through the collector component to the server. Protocol and Middleware used in "Central Nodes" will set rules and maintain communications among the computers in a network. "Server Blades" will handle all the heavy workload generated by the running applications. Client computers functioning as display terminals will access MUS and MUG applications remotely through interface software like Remote Graphics Software (RGS). Where a MUS application is replicated for each client, a MUG would also be replicated for each client. Multi-user clients see the same application interfaces in cloud serving as they see in CS architectures.

An interface like RGS is used to connect client computers and blade servers shown in Fig. 4. RGS sends user mouse and keyboard events from client computers to blade workstations. These events will be interpreted by RGS software and processed by the applications on the blade servers. A modified

multi-user RGS environment would transmit the interactive desktop images running on blade workstations to client computers via a TCP/IP network.

The Session Allocation Manager (SAM) optimizes the deployment of client/application sessions among the blade servers. Presently SAM only allows the user to log into SAM once or establish a re-connection to the user's previous remote computing resources. SAM also manages connections between client computers and remote computing resources (like Siemens NX, MUG CAPP, MUS controller, etc.). The database in SAM will store the properties of the computing resources, client computers and RGS and will determine how to allocate user requests based on the computing resources.

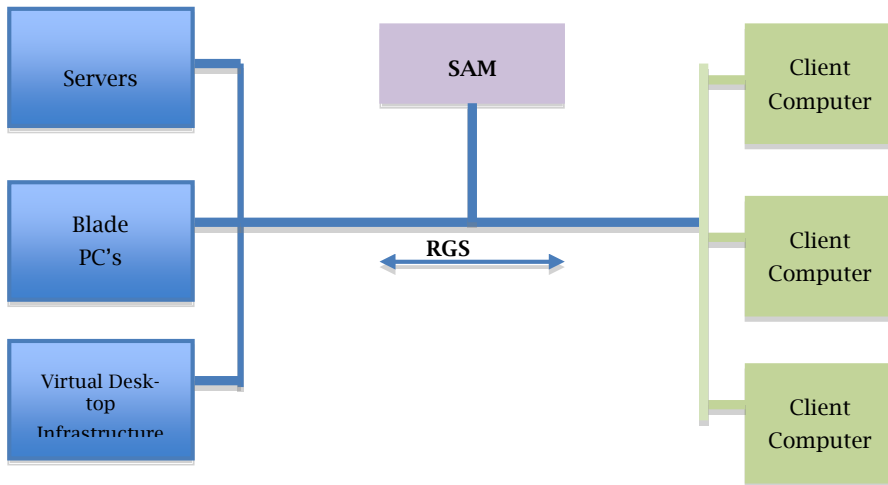


Fig. 4: RGS architecture (Source: HP Website).

Security will be enhanced due to the centralization of data, and the performance will be monitored and scalable. But users will lose control of local data and the performance of CAx applications will be greatly influenced by high network load or low bandwidth.

### 3.1.6 Agent software

Agent software is programmed to specify how one user interacts with other users and exhibits a high degree of autonomy. It is configured to observe, record, and transmit user actions among a set of collaborators. Intelligent agents apply rules to communicate among themselves and distribute change information among collaborators [18]. In the proposed MUG architecture the agent software uses the context settings to determine what information is to be transmitted. Again, an issue to be resolved in the future, and which depends somewhat on resources, is whether the agent software strips the data packets locally, depending on context settings, or whether the agent software transmits the context settings to a server to strip the packet.

The advantages of agent software are: 1) simplifying the complexities of distributed computing and 2) overcoming the limitations of current user interface approaches, such as speed of performance, error rates, and retention over time [12]. Agent software will periodically check if any information has been changed and then broadcast updates to all users.

The agent interface can receive a query regarding the collaboration environment, such as context settings for an on-line user, and report the query results to the users shown in Fig. 5. In the CS mode a regional server, at some remote location, receives requests from the interface agent software and collaboratively responds to the request.

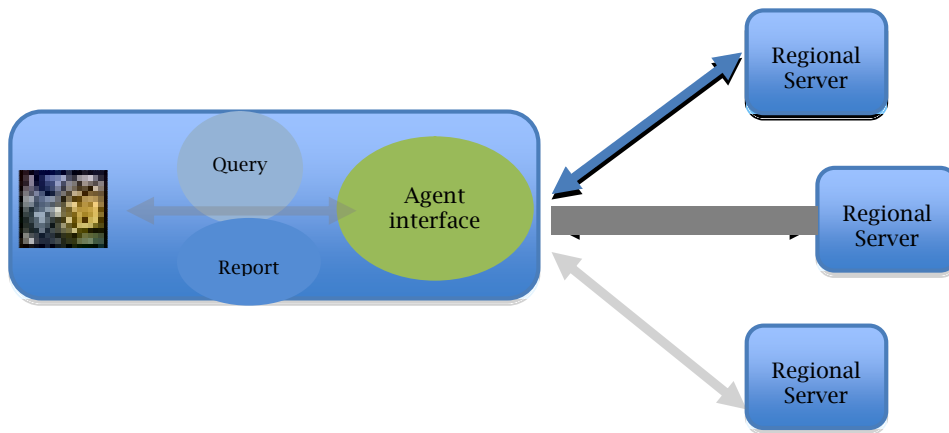


Fig. 5: Agent software workflow.

One useful feature of agent software is to integrate video and audio streaming systems that can be broadcast to the set of collaborating users in a MUG. The agent software may use any of the architectures previously discussed, such as the P2P and CS architectures. CS can handle account management and present status information. P2P is used for in-band real-time streaming data that includes audio, video, and text messaging. It is scalable and robust because the removal of nodes has no great impact on the network.

A future MUG will display the context filters described previously. As an example, consider text translation. A user can use a filter widget to pre-set the native language of choice, type in that language, and based on the communication and security contexts, allow agent software at each user's display to translate the texts into the native language set for each user.

#### 4 MUG PROTOTYPE CONSIDERATIONS

In this section we configure our prototype, considering the limitations in current CAX API's and other tools that are available.

##### 4.1 CAD API

A CAX application provides API's to extend the design environment, enabling engineers to design and produce better products. Zhou et al. [27][14] used "the APIs of multiple CAD systems in order to develop a CAD neutral module which will be able to interact with the different commercial CAD packages enabling more collaborative design efforts to take place." Kao and Lin [13][14] used "the CAD API to pass model information over a LAN as well as the Internet as part of a collaborative CAD/CAM system in development."

A CAD API also helps the system extend capability and compatibility. Tucker's research involved the development of an API translator, which was able to convert API functions and code from one CAD system into functions and code for another system. Tucker's research demonstrates a partial solution to "a pervasive problem in CAD which is the collaboration and file sharing between groups or companies which use different CAD packages" [14]. He developed an API translator, which was able to convert API code and function for one CAD system into code and functions for another CAD system. It is important to design an adaptable and user-friendly platform. We suggest that a future MUG would develop a collaborative library that could be integrated into the API of any commercial CAX API library.

## 4.2 NX Connect

NX Connect is an add-on MUS to Siemens NX and allows multiple users to access and make changes to a single part file simultaneously. It is a collaborative CS prototype without a multi-user MUG; thus, all collaborative activities in NX Connect have to be orchestrated. Before building an NX MUG as a demonstration prototype, we need to understand how the NX Connect application is limited by the Siemens NX architecture and related API.

## 4.3 Limitations of Siemens NX

There are a number of limitations in Siemens NX that can be related to single user architectures. These limitations would be present in most CAx application API's.

1. There is no direct access to the event handler; undo marker files are utilized to access model parameter changes in a multi-user environment.
2. API's provide handles to geometric objects (memory addresses). These cannot be passed to other multi-users since memory locations on each computer vary. Thus, NX Connect required extensive programming to extract the parameters from the data structures. CAx applications inherently know how to pass geometry parameters because they generate files that conform to IGES/STEP standards for neutral file exchange. Multi-user CAx will need API's that provide the parameters directly, like copying the object, rather than just memory addresses to the object.
3. CAx applications have a concept of a single display screen, a single viewpoint, and a single cursor. The NX GUI's are built for single users and are not intended to be shared by multi-users. Only one user can modify the model at one time, regardless of how many are viewing the display screen. Users cannot simultaneously view the model from different viewpoints or simultaneously edit the model parameters in the same GUI.
4. There are no event interrupts available that signal parameter changes inside NX. NX Connect polls for parameter changes from undo marker files. Although polling is simple, it would be more efficient to replace polling with interrupts. The interrupt will be raised only when the new update occurs. Furthermore, busy-wait polling is perfectly appropriate in a simple single-purpose system, but in a multi-tasking system, especially in real-time computing, interrupts are preferred [32]. Interrupts can keep the computer from busy-wait polling, then switch a context to an interrupt handler. Interrupts make it easy to add new functions such as a sound alert when the server receives a new update and sends it to each client.

## 4.4 Network considerations

In CS architectures where the CAx application resides locally on each PC (or workstation), screen updates will only be as effective as the network speeds, and may be affected by network latencies and reliability issues. LAN networks within companies will generally be fast. Image rendering within company network infrastructures will likely be very effective and allow multiple personnel from disparate organizations to collaborate.

Multi-user groups that are globally distributed may have to use buffering methods to update changes periodically, rather than attempt real-time screen updates. The proposed MUG allows users to set timing contexts and thus CS architectures should support multi-user collaboration effectively in the near future.

A cloud serving architecture will also depend on network latencies, and may be ineffective across widely distributed global networks, unless the cloud servers are also distributed and updated accordingly using update timers and buffers. Cloud serving collaboration will be very effective in local infrastructure networks, and provide for enhanced security.

Software like RGS minimizes the information to be transmitted from the user station to the cloud server, since only the screen pixels, mouse cursor, and keyboard events need to be transmitted. Compressing/decompressing algorithms at the client and in the cloud minimize the data packets to get realistic performance.

If NX Connect and the prototype MUG are distributed in a cloud, using RGS display updating methods, the current thin-server-thick-client CS architecture will be dramatically changed. This

architecture will make the data more secure and a centralized architecture will reduce management costs by consolidating resources to one-point management.

## 5 NX MUG PROTOTYPE

The objective of our MUG prototype is to assist users to communicate with each other through NX

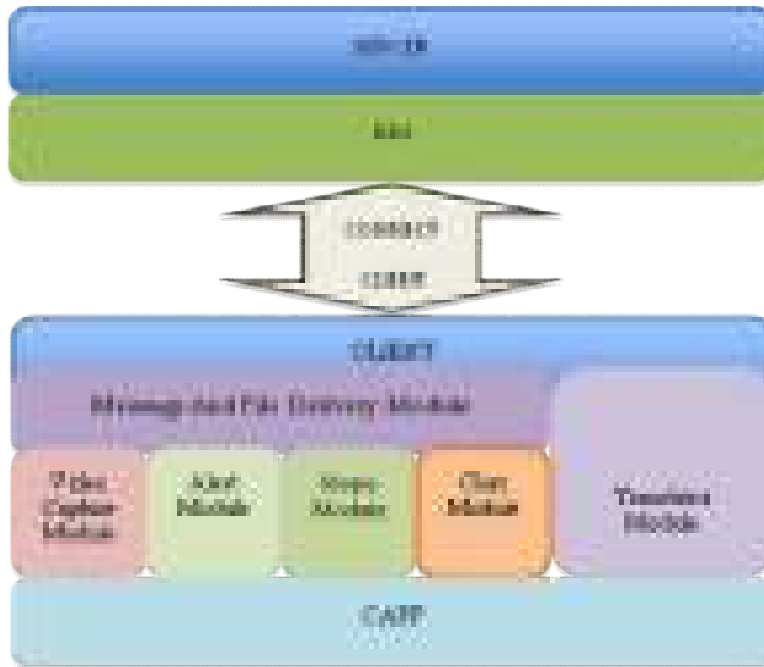


Fig. 6: NX MUG prototype architecture

Connect when multiple users are making changes to a single model; see Fig. 6. NX MUG uses agent software to connect to NX Connect and the multi-user clients. To link NX Connect to NX MUG the programming codes were placed so that NX MUG is invoked right before entering multi-user mode. This action assures that NX MUG and NX Connect use the same server and serve the same group of clients. User awareness data and filters are set in each client's MUG.

A client's display window is captured by continually calling Windows system API at a fixed frequency, capturing screen images (or workspace images), and grabbing the mouse movements. The mouse is drawn on the images and saved in bmp format; the images are compressed through the compression algorithm and are converted into .jpg format. Then the captured images are transmitted over the network as a byte (input/output) stream. Clients receive the data packages in a byte stream and assemble them to a JPEG file. Finally, these images are displayed in the client's MUG video window.

Chat messages are transmitted via UDP sockets, first stored as a data I/O packet that includes other authentication information. The client software will extract the data and display the text messages in the client's MUG. The alert function has two modes, a broadcasting mode and an alert mode. In the broadcasting mode a message is sent to the server and the server forwards the message to inform all clients to submit the update information. In the alert mode an alert is sent to the target person in the group, which is similar to the principle of sending chat messages in the form of packets.

The translation tool is implemented by polling Google Translator through custom developed C# web-based tools. The SKYPE function linked to the program was written by another graduate student [19]. He integrated Skype in NX.

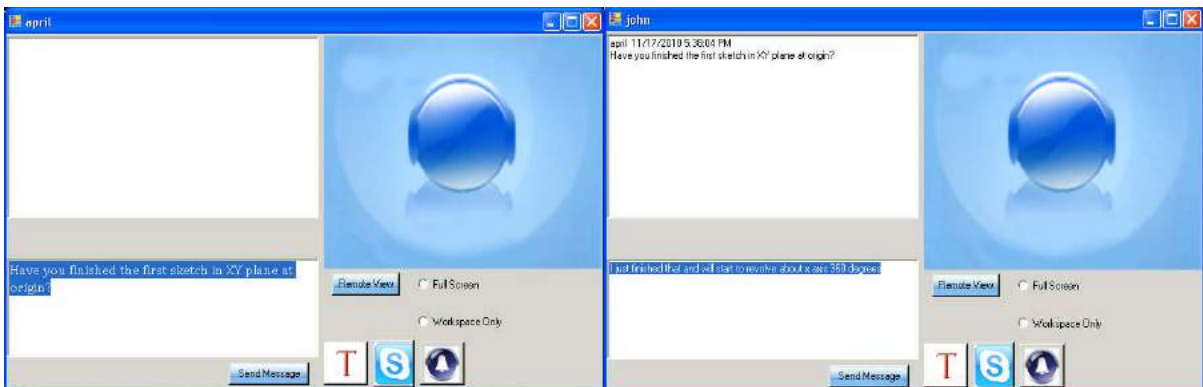
The multi-user functions are controlled by the MUG CAPP. The server only plays an interactive role, which is responsible for reading the corresponding information from the ISM database for *CurrentUser* and *CurrentUserProfile*. The server interacts with the clients to control user login and registration, including distributing user status, establishing UDP connection for different clients, and broadcasting any user's updated information.

### 5.1 NX MUG Implementation

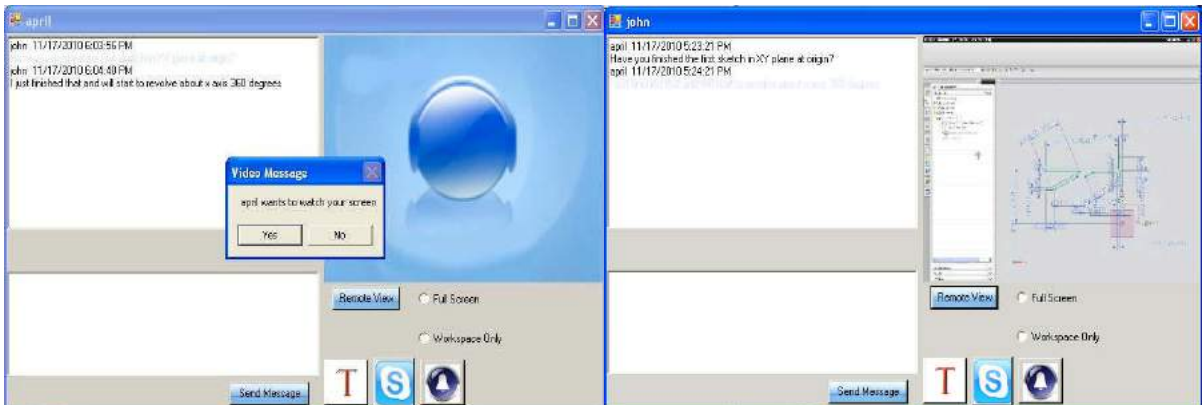
Three participants collaborated in the demonstration, and spent some time coordinating and decomposing their tasks beforehand. They logged in as John, Lu and April and the group collaborated to design a turbine engine front frame that required multiple features.

In the figures below, John started to sketch the inner ring and asked April if she finished the first sketch in XY plane at origin.

April responded to John that she just finished the sketch and would start a revolve 360 degrees about the x-axis.

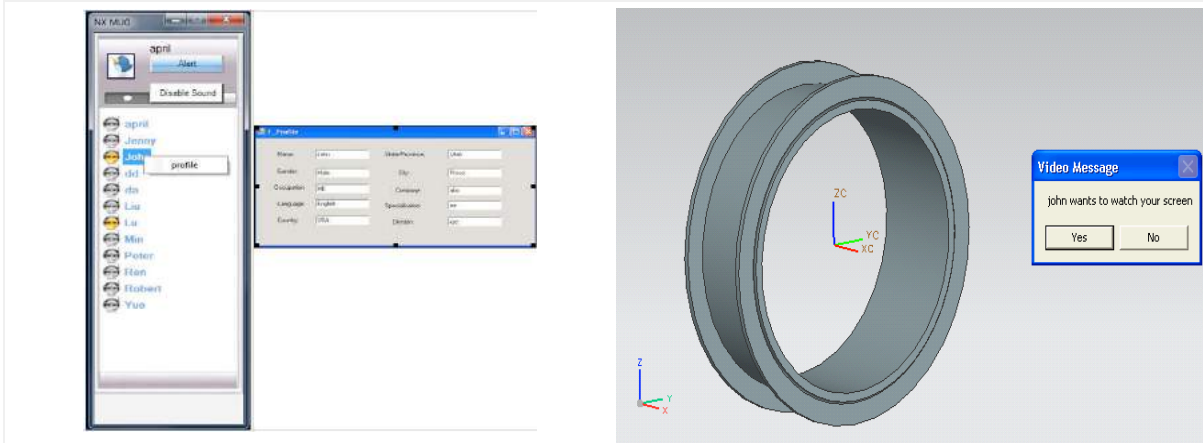


April wanted to watch John's screen to make sure there was no intersection of their respective parts. John accepted her request and gave April full screen access.

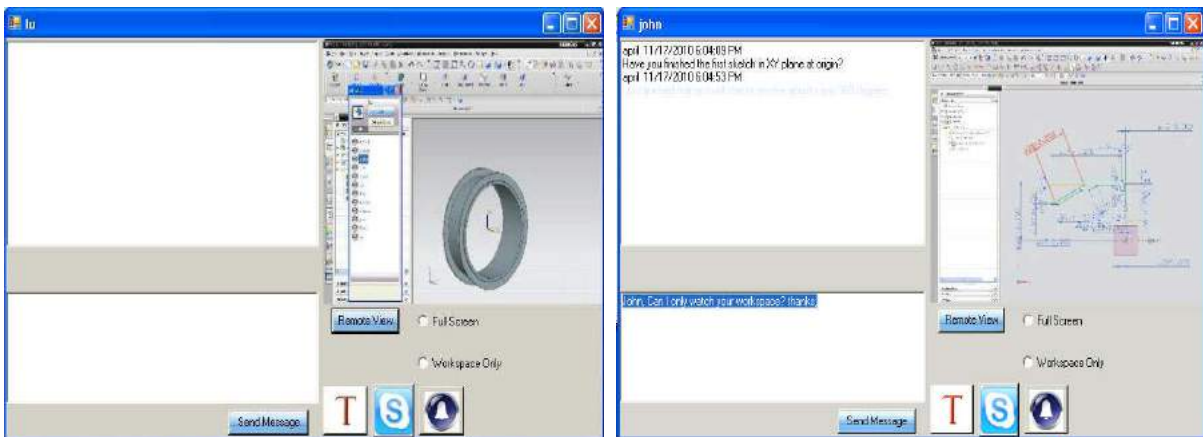


Meanwhile, April was checking other user profiles. John wanted to watch Lu's workspace and Lu accepted John's request as shown later.

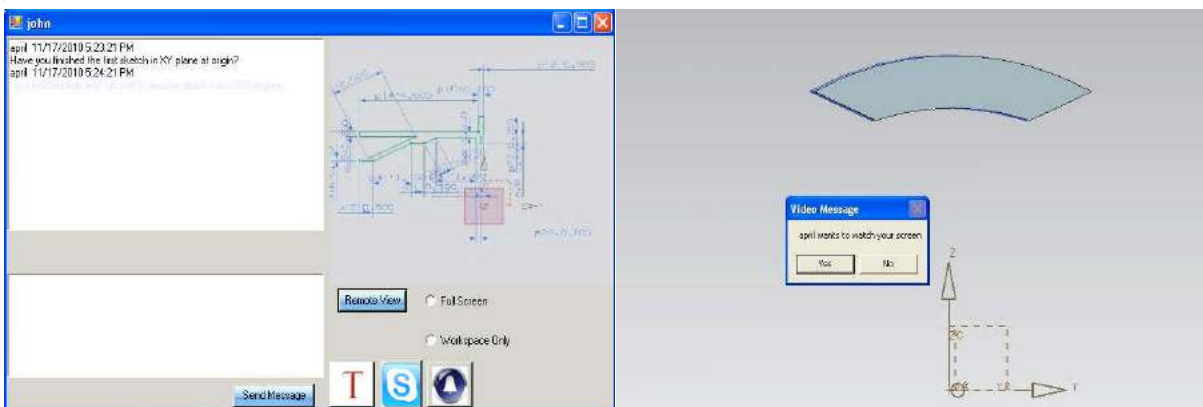




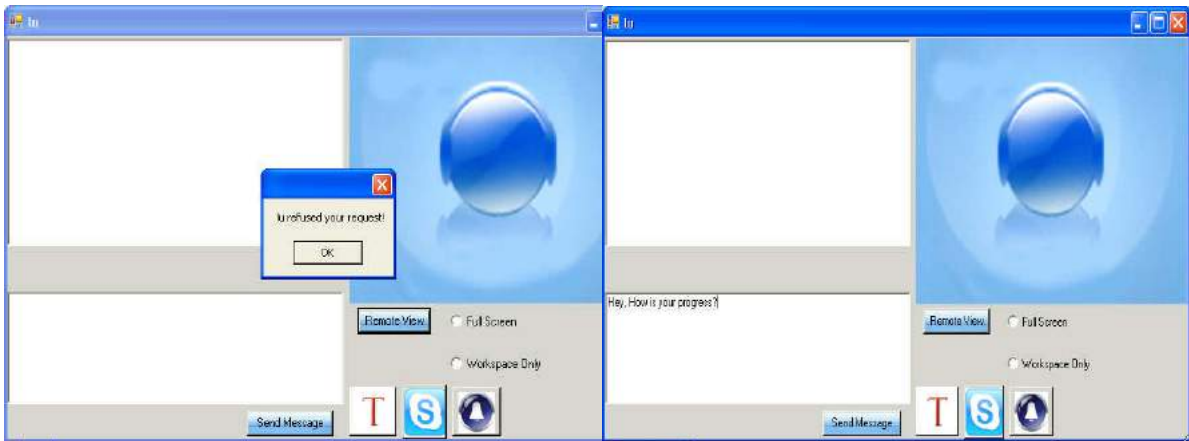
Later April sent John a text requesting that she be able to view his workspace.



John changed his view setting based on April's request. Then, April sent a request to watch Lu's screen.



But Lu was in the middle of editing the model, and he refused April's request. Later, April asked Lu about his progress and she typed and sent the message -"hey how is your progress" in her native language (English) to Lu.



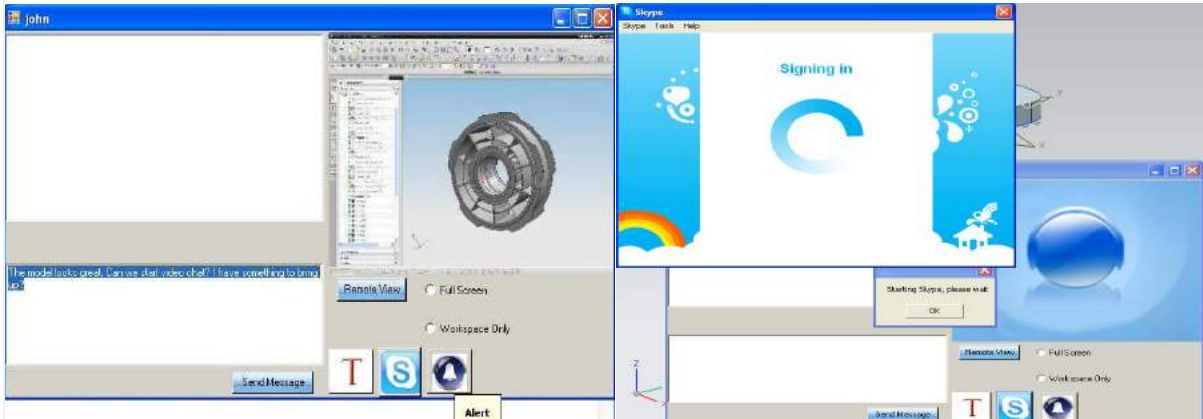
Lu received the message and translated it into Chinese, Lu's native language, using Google Translator accessed through the MUG.



Shortly thereafter, April sent several text messages to Lu and Lu used the Google Translator. He figured that it might be easier to understand April's intent using the remote view.



Later, April sent an alert to Lu and John and notified them that the model had appeared to be finished. April decided to conclude the session using a video chat rather than texting, and each of them evoked Skype through NX MUG as shown in Fig. 8.



Tab. 2: NX MUG demo.



Fig. 8: Skype session.

## 6 CONCLUSIONS

A future NX MUG will be capable of all the current prototype features, plus those that follow, assuming that future CAX applications will be modified to incorporate these capabilities: 1) integrate some MUS capability or provide API interfaces to an integrated MUS; and 2) provide direct access through CAX API to event buffer, using interrupts to notify when parameter changes occur and allowing API's to copy the data object if necessary. It is also presumed that display communication software like RGS can be modified for multi-user interaction/awareness. In conclusion:

- NX MUG can run as application concurrently with CAX application or will have API set for direct integration into CAX applications, where CAX has some MUS capability.
- NX MUG architecture will allow it to be used in CS, hybrid P2P, or cloud serving modes.
- NX MUG will apply security measures to restrict/manage collaboration between users (what models they can view, change, copy, etc.). This will require some interfacing between the NX MUG application and the CAX MUS.
- Future NX MUG can set all the contexts discussed in the paper, including display control contexts, communication contexts, supervisory control contexts, security contexts, and recording contexts.
- NX MUG integrates auto text translation so that each user communicates in a language preference/context.
- Users set up multi-user display sessions, partitioned on one screen or over multiple displays.

## 7 REFERENCES

- [1] Bahill, T.: Applications of Blackboard Architecture, systems and industrial engineering, University of Arizona, 1997.
- [2] Bentley, R.; Roden, T.; Sawyer, P; Sommerville, I.: Architectural Support for Cooperative Multi-user Interfaces, Computing Department, Lancaster University, 1994.
- [3] Coutaz, J.; Nigay, L.; Salber, D.; Blandford, A.; May, J.; Young, R.: Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE properties, Proceedings of the INTERACT'95 conference, S. A. Arnesen & D. Gilmore Eds., Chapman & Hall Publ., Lillehammer, Norway, 1995, 115-120.
- [4] Coutaz, J.: Software Architecture Modeling for User Interfaces, Laboratoire CLIPS(IMAG).
- [5] Eiji, A: Knowledge and Skill Chains in Engineering and Manufacturing Information Infrastructure in the Era of Global Communications, Proceedings of the IFIP TC5 / WG5.3, WG5.7, WG5.12 Fifth International Working Conference of Information Infrastructure Systems for Manufacturing 2002 (DIIDM2002), November 18-20, 2002 in Osaka.
- [6] Douglas, S.; Tanin, E.; Harwood, A.; Karunasekera, S.: Enabling Massively Multi-player Online Gaming Applications on a P2P architecture, Proceedings of the International Conference on Information and Automation, December 15-18, 2005, Colombo, Sri Lanka, 7-12.
- [7] Gelernter, D.: *Mirror Worlds*. Oxford University Press, New York, 1993.
- [8] Gutwin, C.: "Workspace Awareness in Real-time Distributed Groupware", Calgary, Alberta, Dec., 1997.
- [9] Hill, J.; Gutwin, C.: *Traces: Visualization of Interaction*, Department of Computer Science, University of Saskatchewan, Saskatoon, SK, Canada, 2001.
- [10] Hill, J.: *A Direct Manipulation Toolkit for Awareness Support in Groupware*, University of Saskatchewan, August, 2003.
- [11] Hofstede G.: *Cultures and Organizations: Software of the Mind*, New York: McGraw-Hill U.S.A. 2003
- [12] Jo, C.; Chen, G.; Choi, J.: *A framework for BDI Agent-based Software Engineering*, Studia Informatica Universalis.
- [13] Kao, Y.-C.; Lin, G.: Development of a Collaborative CAD/CAM System, *Robotic and Computer-Integrated Manufacturing*, 14, 1998, 55-68. [DOI:10.1016/S0736-5845\(97\)00014-8](https://doi.org/10.1016/S0736-5845(97)00014-8)
- [14] Larson, B.-M.: *Exploring the Common Design Space of Dissimilar Assembly Parameterizations for Interdisciplinary Design*, Dept. of Mechanical Engineering, Brigham Young University, August 2008.
- [15] Lauwers, J.-C., Lantz, K.-A.: *Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems*, Proceedings of CHI '90, April 1-5, Seattle, Washington, ACM Press, 1990, 303-311. [DOI:10.1145/97243.97301](https://doi.org/10.1145/97243.97301)
- [16] Martin, D.; Cheyer, A.; Moran D.: *The Open Agent Architecture: A framework for building distributed software system*, Artificial Intelligence center.
- [17] McFarlane, R.-D.-P.: *Network Software Architectures for Real-Time Massively-Multiplayer Online Games*, M.S. Thesis, McGill University, School of Computer Science, Feb., 2005.
- [18] Red, E.; Jensen, C.; Holyoak, V.; Marshall, F.; Xu, Y.: *v-Cax: A Research Agenda for Collaborative Computer-Aided Applications*, Vol. 7, No. 3, , 2010, 387-404.
- [19] Ryskamp, J.; Jensen, C.; Mix, K.; Red, E: *Leveraging Design Rationale to Improve Collaborative Co-Design CAD Environment, Tools and Methods for Competitive Engineering*, Volume 1, 2010, 475-486.
- [20] Ammar-Khodja, S.; Perry, N.; Bernard, A.: *Processing Knowledge to Support Knowledge-based Engineering Systems Specification*, *Concurrent Engineering*, 16, 2008, 89-101.
- [21] [DOI:10.1177/1063293X07084642](https://doi.org/10.1177/1063293X07084642)
- [22] Schollmeier, R.: *A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications*, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE, 2002.
- [23] Shen, Y.; Ong, S.; Nee, A.: *Collaborative Design in 3D Space*, VRCAI 2008, Singapore, Dec. 8 -9.
- [24] Sosa, M.; Eppinger, S.; Rowles, C.: *A Network Approach to Define Modularity of Components in Complex Products*, *ASME Journal of Mechanical Design*, 129 (11), 2007, 1118-1129.
- [25] [DOI:10.1115/1.2771182](https://doi.org/10.1115/1.2771182)

- [26] Steves, M.; Knutilla, A.: Collaboration Technologies for Global Manufacturing, Proceedings of the ASME International Mechanical Engineering Congress and Exposition (IMECE): Symposium on Manufacturing Logistics in a Global Economy, 1999 .
- [27] Stødle, D.: User Interface Components to Support Simple and Efficient Use and Control of Large, High Resolution Displays, University of Tromso, 2005.
- [28] Wallace, G.; Bi, P.; Li, K.; Anshus, O.: A Multi-cursor X Window Manager Supporting Control Room, <http://www.cs.princeton.edu/omnimedia/papers/multicursor.pdf>.
- [29] Zhou, H.; Han, M: CAD/CAM Optomechatronics, in Education and Training in Optics and Photonics, OSA Technical Digest Series (Optical Society of America, 2003), paper EWF7.
- [30] Comaya Official Website, <http://cooffice.ntu.edu.sg/comaya/video.htm>.
- [31] ELSEVIER, [http://www.sciencedirect.com/science?\\_ob=ArticleURL&\\_udi=B6TYR-4D9D8W6-3&\\_user=10&\\_coverDate=04%2F15%2F2005&\\_rdoc=1&\\_fmt=high&\\_orig=search&\\_origin=search&\\_sort=d&\\_docanchor=&\\_view=c&\\_searchStrId=1550457045&\\_rerunOrigin=google&\\_acct=C000050221&\\_version=1&\\_urlVersion=0&\\_userid=10&md5=ffa3385aa56f68566088a7e0dc44c8cf&searchtype=a](http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6TYR-4D9D8W6-3&_user=10&_coverDate=04%2F15%2F2005&_rdoc=1&_fmt=high&_orig=search&_origin=search&_sort=d&_docanchor=&_view=c&_searchStrId=1550457045&_rerunOrigin=google&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=ffa3385aa56f68566088a7e0dc44c8cf&searchtype=a) , 2004
- [32] Technology & The Internet by Andy Finn:
- [33] <http://www.tafinn.com/andyfinn-us/Writing/Technology/emoticons.htm>
- [34] User Interface Design Tips, Techniques, and Principles, Amysoft,
- [35] <http://www.ambysoft.com/essays/userInterfaceDesign.html>
- [36] Wikipedia, [http://en.wikipedia.org/wiki/Polling\\_%28computer\\_science%29](http://en.wikipedia.org/wiki/Polling_%28computer_science%29)