Computer and Information Science; Vol. 6, No. 4; 2013
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Architectural Limitations in Multi-User Computer-Aided Engineering Applications

Edward Red[1], Greg Jensen[1], Prasad Weerakoon[1], David French[1], Steven Benzley[2] & Karl Merkley[3]

[1] Mechanical Engineering Department, Brigham Young University, Provo, UT, USA

[2] Civil Engineering Department, Brigham Young University, Provo, UT, USA

[3] Computational Simulation Software, LLC, Provo, UT, USA

Correspondence: Edward Red, Mechanical Engineering Department, Brigham Young University, Provo, Utah 84602, USA. Tel: 1-801-422-5539. E-mail: ered@byu.edu

Received: June 19, 2013 Accepted: July 17, 2013 Online Published: August 8, 2013

doi:10.5539/cis.v6n4p1 URL: http://dx.doi.org/10.5539/cis.v6n4p1

Abstract

The engineering design process evolves products by a collaborative synthesis of specifications, personnel and organizations. Unfortunately, collaborative effectiveness is thwarted by existing single-user computer-aided applications like computer-aided design, computer-aided analysis, and others. These applications and associated file management systems assign editing rights to one technical person, e.g., a designer, analyst, or a process planner. In the absence of collaborative computer-aided engineering applications, we conducted a survey to establish that product collaboration is limited to interactive, either formal or ad-hoc design sessions, social communication tools, serial model sharing, terminal/screen sharing, and to conference call interactions. Current computer-aided (CAx) tools do not permit simultaneous model changes by a collaborative team editing the same model. Although over a decade of prior research has demonstrated multi-user feasibility for computer-aided applications, the architectural breadth of this research has apparently not yet compelled developers and end-users to develop and adopt new multi-user computer-aided applications devoted to product development.

Why have collaborative engineering CAx tools not been commercialized for mainstream use? This paper uses several multi-user prototypes, including the first Computer-Aided Engineering multi-user prototype called CUBIT Connect, to expose additional architectural hurdles to implementing new multi-user collaborative paradigms. These challenges relate to variable algorithmic performance times, multi-threading and event driven client notification processes, distributed access level security, and model change management in design sessions.

Keywords: Multi-user, collaboration, collaborative architectures, change management, NX Connect, CATIA Connect, CUBIT Connect

1. Introduction

Modern computer-aided engineering applications (CAx) like CAD/CAE/CAM and related file management applications like PLM authorize one user to a model where the assigned user independently edits the design, analysis, or manufacturing parameters, or applies design and analysis algorithms. When a modeling, performance, or resource error is discovered, necessary changes divert the model version upstream in the design process cycle. In many cases improved collaboration would have discovered and corrected these deficiencies earlier in the process cycle.

Why do our design applications remain single user within a collaborative world of product engineering? And why do they not integrate the social communication tools so pervasive to society? These are puzzling questions. The obvious answer is because our computer tools have been designed to empower the individual, not product teams. Yet, considering widespread gaming collaboration, where hundreds and thousands of participants scattered around the globe simultaneously participate in a single game, these questions are even more perplexing. An engineer's charter is to bring about positive technological change by embracing best practices. It seems that we are not immune to inertial tendencies when it comes to our most important engineering applications.

The function of this paper is twofold: 1) to verify that Application Programming Interface (API) libraries, with additional Client-Server (CS) software plugins, can support multi-user versions of our most complex CAx applications; and, by doing so, 2) expose collaborative and architectural multi-user limitations in modern CAx

tools. The three prototypes consist of multi-user versions of Siemens NX CAD (NX Connect, Ryskamp et al., 2010), CATIA CAD (CATIA Connect), and the first multi-user finite-element analysis (FEA) pre-processor based on the Sandia Labs CUBIT application (CUBIT Connect, Red et al., 2011, Weerakoon et al., 2012).

We have expanded multi-user prototypes into other applications beyond CAD to identify additional architectural challenges not disclosed by prior research. By testing these prototypes across several time zones, and allowing 30 or more multi-users into a single model (Jensen, 2012), we discovered a change management problem (Undo Ctrl Z) that complicates session management model consistency among distributed clients. Other challenges surfaced: variable algorithmic performance times, multi-threading sophistication, event driven client notification processes, and distributed access level security.

Prior research has mostly focused on document applications, CAD applications, multi-user architectures (Client-Server/CS, Peer-to-Peer/P2P, thin/thick client/server, etc.), granularity of multi-user interactions (operational transforms, constraint relationships, region locking, etc.), and limited investigations into model visibility based on security philosophies and access limitations. The incompatibility of CAx application formats, file sizes, CAx diversity among suppliers and subcontractors, and the need to collaborate on design decisions have given rise to efficient CAx viewing applications; see http://www.oracle.com/us/products/applications/autoVue/index.html and http://www.techsoft3d.com/our-products. These technologies are useful for simultaneous viewing and evaluating design models by collaborative design teams, and for assigning and recording design decisions, but they don't allow users to simultaneously edit design models.

Little research has examined organizational methods or personnel knowledge databases for practical task/personnel decomposition into virtual multi-user teams. We postulate that industry will not adopt new processes, regardless of productivity potential, unless the administrative issues are well understood and are fairly practical to implement. Hannan and Freeman (1984) articulated the principles of industrial inertia in their highly referenced 1984 paper.

2. Collaborative Motivation

Having more than one person work on any given task has proven to decrease its completion time, particularly for distributed electronic collaboration (Jafari et al., 2010). In a competitive marketplace time-to-market is critical, where consumer acceptance often relates to market timeliness.

A counterargument suggests that collaboration increases costs by requiring additional personnel. But organizational inefficiencies in matching personnel to product schedules would suggest that carefully directed teaming could improve organizational efficiencies and eliminate product development mistakes that often cause the engineering design process to cycle unnecessarily.

Moreover, multi-user tools do not require that multiple personnel be applied to a particular task or model; rather, these tools provide the opportunity to do so when appropriate. Companies will be able to evaluate the trade-off between time-to-market and immediate resource allocation for themselves, but with expanded possibilities. At the session level, with predetermined workspaces and boundaries, we have shown that collaboration decreases the product development time in proportion to the number of multi-users (Jensen et al., 2012).

As reported by Owen et al. (2007), a Sandia National Labs 2005 survey determined that in Finite Element Modeling (FEM) and simulation 73% of the time is spent in developing the analysis solid model, meshing, etc. Much of the 73% is spent in cleaning up solid model discontinuities. The multi-user preprocessing prototype, CUBIT Connect, provides an ideal environment where multiple engineers can simultaneously eliminate discontinuities in node connectivity or mesh assigned model regions autonomously, thereby reducing the pre-processing time significantly.

Within industry, the greatest advantage of multi-user CAx may be the knowledge transfer and training that occurs among cross functional virtual teams. It seems that previous multi-user research has yet to trumpet this most important advantage of multi-user CAx environments. Presently, experts train novice engineers in a time consuming one-on-one process. Similar advantages will be realized at all educational levels when students are able to simultaneously engage themselves in a common design model. Since multi-users, with collaborative permission, can both view and edit each other's work on their screen it would be easier to see potential problems within the model, particularly if engineers or operators from different disciplines can engage in the multi-user sessions.

3. Previous Research

We acknowledge the previous excellent research conducted in multi-user CAx, and note the emergence of new multi-user tools like Google Docs (http://en.wikipedia.org/wiki/Google_Docs). An earlier but extensive literature review of this work is given by Red et al. (2010) and summarized and expanded in the following sections.

3.1 Architectures for Multi-User Collaboration

Researchers like Sun et al. (2006) and Zheng et al. (2009) apply CAx API's and Transparent Adaptation (TA, i.e., external software plugins and API integration) so that multi-users can edit Word and Power-Point documents simultaneously without document decomposition constraints. However, the extension to CAx objects is far more difficult. Zheng extends Sun's TA methods to AutoCAD application, referring to constrained collaboration using feature and spatial blocking as *pessimistic* approaches to collaboration, although the authors of this paper would use the term *practical* rather than *pessimistic*.

Li et al. (2008) proposed and tested architectures for replicated collaboration between client users, tackling the difficult problems of simultaneous editing of model features within common regions. The interesting, mostly serial, interactions use feature exclusion/locking methods that require numerous communication events among the clients. Feature tree dependencies could prove challenging, and the serial interactions lead to questions of session productivity.

Ramani et al. (2003), like most researchers in collaborative design, use a CS architecture where the server acts as the master to distribute model changes between collaborating client users. In Ramani's implementation the server stores the master model and the client local model is updated by command objects representing local creation, modification and deletion.

3.2 Infrastructures for Multi-User Collaboration

One objective of this paper is to show that modern CAD systems have inherent architectural limitations that make collaboration difficult. Conventional CAD systems use version control and change management to control their distributed models (Barbosa et al., 2003). Barbosa suggests that CAD models need an object model foundation, made up of objects that can be collaboratively interpreted, edited, linked and distributed, although it could be argued that the hierarchical feature trees in modern CAx systems mimic object properties.

Bonneau and Gabrielaitis (2009) use a hierarchical file structure to decompose building designs into sub-structure files, such as different floors. Different collaborators are then assigned to the sub-structure files. A manager provides oversight and control of the design evolution. Although more efficient, this approach does not allow multiple users to simultaneously design the same structures.

Li et al. (2007) used a neutral architecture to collaborate among clients using differing CAx applications, like SolidWorks and ProE Wildfire. CS architectures propagate model changes between clients using a thin server. Most of the model translation software is placed on the client computers. This is somewhat counter to the current trend of cloud serving applications where the client becomes thin (little data management) and the thick cloud server manages the data and changes. In Li's prototype clients are not able to simultaneously edit a model but must pass it to other client users in a serial sequence, similar to the serial collaboration capabilities offered in the newest CATIA Version 6 CAD application.

3.3 Security for Multi-User Model Sharing

Compared to the architectural research in multi-user CAx, model sharing security has not been a primary research area, yet a major concern to industries engaged in global product development. Our larger product development industries (automotive, aerospace, consumer products, etc.) often globally design and deliver sub-systems and related components using diverse third party suppliers. CS and P2P architectures used in the reviewed research prototypes expose IP (intellectual property) to multiple collaborators. Among the limited researchers in model security Wang et al. (2006) propose to restrict model sharing by providing lean and selective information based on collaborator need-to-know. Encryption can be used to secure the information among the networked collaborators. Role-based viewing methods, where data is partially shared among designers, can deter reverse engineering.

Product development companies use PLM systems and assigned editing rights to protect and limit access to their product models within the company. Typically, the design/model data is located behind company secured firewalls to protect the data from external access. Companies engaged in global design practices will transfer large design files among offshore facilities in a 3 shift 24 hour design cycle. These file transfers use large file and image encryption methods based on 40 bit and 128 bit algorithms.

As new collaborative methods evolve, such as the approaches discussed in this paper, security architectures will also evolve (Kushwaha & Roy, 2010). Cloud serving architectures offer numerous efficiencies and it seems inevitable that CAx applications will be deployed on company secured cloud servers, rather than on local workstations. Security considerations become more challenging (Reddy & Reddy, 2011) when files or file deltas must be moved between mirrored servers at off-site facilities.

Considering the widespread reverse engineering of products, we anticipate more stringent control of access rights to CAx models in the future. For example, Cera et al. (2001) partition 3D models according to certain geometric or design features based on user assigned access control roles. "The partitioning is used to create variable level-of-detail (LOD) meshes, across both individual parts and assemblies, to provide a model suitable for access rights for individual actors within a collaborative design environment".

3.4 Constraints and Conflict Resolution within Multi-User Collaboration

Jing et al. (2009) use a local locking mechanism for model features to avoid user-to-user conflicts over a network. User assignments, along with hierarchical naming conventions, prevent topological correspondence conflicts. Bu et al. (2006) use semantic locking, where the "region lock and object lock" resolve violations. Users can attach comments to the lock explicitly or implicitly, although it is not clear how these comments associate directly to the model, or how other multi-users can interpret them. Some serial multi-user interactions are required to resolve conflicts, such as user negotiation and version sequencing.

Lai (2009) demonstrates hierarchical constraint-based design concepts that relate to the parametric approaches used in modern CAD systems. One way to maintain model consistency and avoid conflicts among multi-users is to have them assigned different model features, particularly when these features use design actions that are reasonably independent, such as blending surfaces using filleting or chamfering within certain size constraints.

Panchal et al. (2007) present an Interval-Based Constraint Satisfaction (IBCS) Method for decentralized, collaborative multifunctional design. The authors use decision making processes represented by the range of design parameters as design constraints. The design converges and complexity reduces as the users continually compare and evolve the design towards the assigned constraints.

Model decomposition has decided advantages when models are complex and characterized by numerous feature types, like surfaces, pockets, fillets, holes, etc. Marshall (2011) and Moncur (2012) demonstrated methods for spatial and feature decomposition among multi-users. With help from Siemens personnel, Marshall showed that NX CAD could confine mouse selection to features in an assigned spatial region bounded by infinite planes. Moncur used locking and reserve feature constraints to control multi-user editing across a multi-user session, where users were not confined spatially and data consistency was to be maintained.

3.5 Previous Research Observations

Prior research has demonstrated multi-user feasibility, but in spite of its promise, the research breadth and conclusions have not yet compelled developers or end-users to press towards commercialized releases of this new paradigm. It is interesting that Zheng (2012), in an editorial forecast for the Journal of Computer-Aided Design, fails to include multi-user collaboration technologies in his discussion of next generation CAD/CAE.

We suggest that there are remaining unresolved multi-user issues, some relating to organizational efficacy, many simply inertial, and some to research breadth and practicality. Industries shy away from basic process changes which may destabilize the company. Red et al. (2012) suggest that multi-user CAx applications will change personnel/organizational assignment processes in product development, considering the globalization of product development across virtual design teams. Research on distributed multi-user specification and assignment is lacking, as are related multi-user organizational tools. We address some of these issues in this paper.

Multi-user CAx requires organizational change from project specification to multi-user team organization. These changes affect the basic engineering design process which remains mostly a serial set of activities. Yoshimura (2012) suggests that new methods are required to achieve collaborative excellence, including maximizing team expertise, and providing a more flexible framework for collaborative interaction.

In trying to understand why modern CAx tools are still single user based, we undertook a 2011 study among nine large product companies to see how their technical personnel collaborate and what tools they use. The survey results, limited to members of BYU's (Brigham Young University) ν-CAx NSF research site, are posted at http://www.ehandbook-byu.net/ (2012) (Red et al., 2013). Clearly, the results show that industry implements modern social communication tools (email, instant messaging, cell phone, wiki's, etc.) in lieu of embedded CAx collaborative tools.

Some surveyed personnel wondered why collaboration tools were not integral to the CAx applications used in their core design processes. Red et al. (2013) suggest that *social* tools are somewhat being used as a substitute for the missing collaborative elements in modern CAx applications.

4. Multi-User Prototypes: Client-Server (CS) Architectures

The prototypes described in the following sections test three CAx API libraries for multi-user capabilities and limitations. These prototypes were described in earlier papers (Red et al., 2013; Weerakoon et al., 2012) to establish that the primary CAx applications used in our largest industries can support functional multi-user architectures. This paper reviews the prototype functionality further to identify inherent architectural limitations, and hopefully encourage the development and research community to overcome these limitations. The current serial processes forced by our single user CAx applications discourage collaboration and cause tremendous resource waste.

Our prototypes use CS architectures for a number of management/administrative reasons: 1) greater industry acceptance because industry IP (intellectual property) can be more easily controlled and monitored; 2) can be deployed locally or globally; 3) can be deployed behind company firewalls; 4) provide central locations for client operator verification and data file transfer encryption; 5) similar to accepted PLM file transfer architectures used to move CAx design files among globally distributed design sites; 6) provides accessible data channels where filters can be applied to assess design iterations and extract/record process rationale; and 7) servers can be made redundant or provide for server mirroring at distributed sites. Further research and maturing is needed in these areas before industry will champion multi-user collaboration.

Technically, CS architectures can adopt a number of variations, such as thin/thick server or thick/thin client, depending on the client or server sophistication, and how the add-on software (also referred to as plug-in) is configured using the CAx API. The simplest future format will be *cloud serving* where the multi-user architectures, applications and plug-in software elements are all deployed on server reconfigurable "blades". The client stations, devoid of local CAx applications, act like a terminal design portal by using an application like HP's RGS to interact with the design model in collaborative multi-user sessions. If the server is designed to be application agnostic, then it can move commands among differing client CAx applications like Siemens NX, Dassault CATIA, Inventor, Solid-Works, etc. Agnostic simply means that the CAx application plug-in translates the design change data into a server neutral format which is then propagated to other client workstations, possibly running different CAx applications. This approach is described by Li in his 2007 paper, and provides a powerful additional advantage for international businesses.

Conflict resolution is a critical subject of multi-user CAx sessions. The prototypes that follow use models that easily decompose by region. Both the NX Connect and CATIA Connect prototypes use similar CS architectures and data propagation modes, with small differences in database formats due to API, threading, and geometric differences between NX and CATIA. In some design iterations, where users experience design conflicts, multi-users negotiated solutions and operations by headphone and textual messaging. In such cases the editing sequences were somewhat serial, but only momentary. Researchers like Bu (2009), Li (2007, 2008), and Jing (2009) consider the gamut of collaborative assignments, spatial, feature, locking, etc., but do not always consider them in the context of simultaneous model editing.

In the case of CUBIT Connect, and with other CAE applications, models enter the application already organized into modular forms. Thus, they are easily decomposed among different multi-users for model clean-up and simplification, and ultimately grid meshing It is often the case that the geometric models used in analysis must be simplified to encourage more uniform mesh formats to improve analytical convergence; see Mounir et al. (2013).

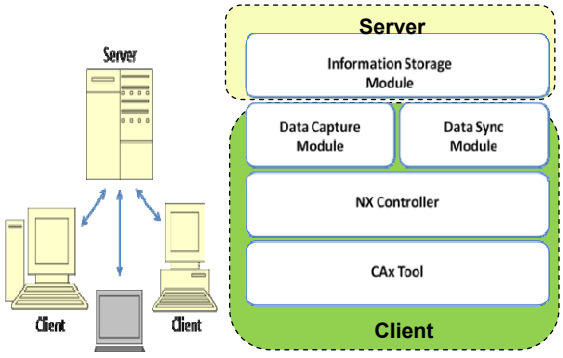
The CUBIT Connect prototype verified that multi-user decomposition for model clean-up (remove dangling edges, node discontinuities, gaps, etc.) was more straightforward because models were easily modularized into distinct components.

4.1 NX Connect

NX Connect is multi-user software integration with the CAD package Siemens NX. NX Connect is a client-server architecture where multi-users (multi-user = client) access and edit a single model simultaneously; see Figure 1. The integration software was designed using the NX C# API. Collaboration is enhanced when several users can view (zoom and rotate) the model independently without affecting other user views.

The early version of NX Connect utilizes Client-Server (CS) with a thin server and a thick client. The server stores the data for the part file, essentially maintaining the master model, and broadcasts changes to each user workstation. Each user station maintains a local copy of the part file which is constantly updated.

NX Connect uses the following custom modules shown in Figure 1b:



a) CS architecture b) NX Connect modules

Figure 1. NX Connect architecture

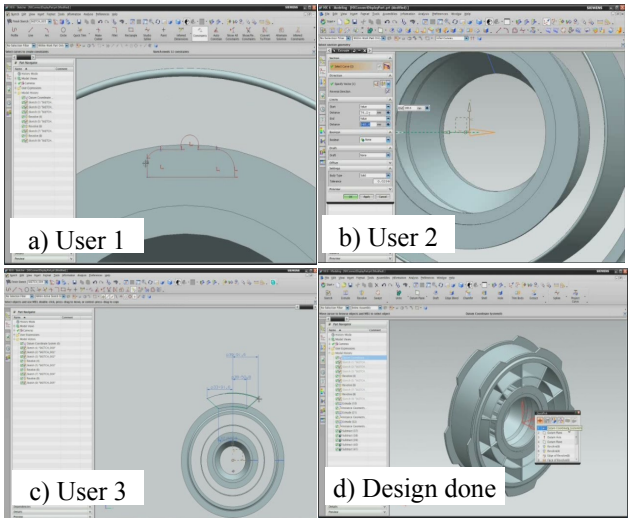


Figure 2. NX Connect front frame multi-user session

1) *Information Storage Module (ISM)* – uses Microsoft's SQL Server for data storage and a hierarchical structure to sync the part features and data changes.
2) *Data Capture Module* - monitors the NX session for changes to the part file and then passes the change information to the multi-users.
3) *Data Sync Module* - monitors the ISM for changes uploaded by another user, using these changes to alert the NX Controller.
4) *NX Controller* - converts all model edit information into primitive values for database storage, translating the primitive data and parameters back into the API constructs required on each user's computer.

After preliminary client coordinated planning, Figure 2 shows three multi-users simultaneously building a jet engine front frame. The design time experiments show that the overall time is reduced by the number of users, about 1/N, where N is the number of multi-users. The achieved design time reduction will depend on how well the design decomposition can be balanced among the users. Research is needed to develop methods to balance complex design decomposition among a potential set of multi-users given model design specifications and user experiences with similar models and associated features. Many, if not most, new products are incremental improvements to existing products models.

4.2 CATIA Connect

Both the NX Connect and CATIA Connect prototypes use a similar CS architecture. CATIA Connect is programmed in C# and uses COM to access CATIA's API. In contrast to the NX API the C# version of CATIA Connect is able to run on a separate thread. Most CAx API's do not run their API's as a separate thread. A thread allows a timed sync to run in the background every few seconds rather than requiring the user to initiate each synchronization step either manually or by incremental polling.

The multi-user test session creates pads, shafts, pockets, grooves, and circular patterns of the same jet engine front frame shown in Figure 2. The instructional steps that follow show how design steps can be decomposed and coordinated among several users (see Red et al., 2013). Users can synchronously enter or asynchronously depart a multi-user session at any time. This approach resolves the synchronicity problem that Shaojin (2010) notes when collaborating users pass incremental design files among themselves.

User 1: ***Step 1*** - To create the front frame's inner most section, make the sketch of Figure 3 in CATIA V5 on the XY plane and use it to create a 360 degree shaft around the horizontal axis.

Step 2 - Update CATIA Connect after shaft is complete. Figure 4 shows the shaft along with the middle rim and a fin created by users 2 and 3.

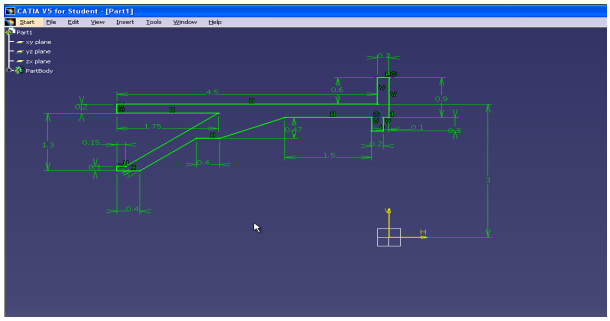


Figure 3. Sketch plane cross-section of shaft

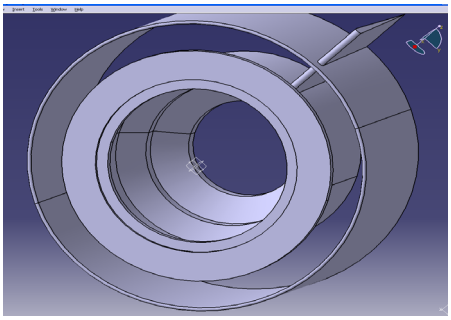


Figure 4. Revolved shaft and client additions

User 2: ***Step 1*** - Create the middle rim by making the sketch of Figure 5 on the XY plane and revolving it as a 360 degree shaft. The thickness is 0.1 inches (0.25 mm) with the front constrained along the V axis.

Step 2 - Create the sketch for the fin bosses on the surface of the middle rim, Figure 6. Create a circular pattern with 10 instances of the feature using the complete crown method and any of the rims as the reference axis, Figure 7.

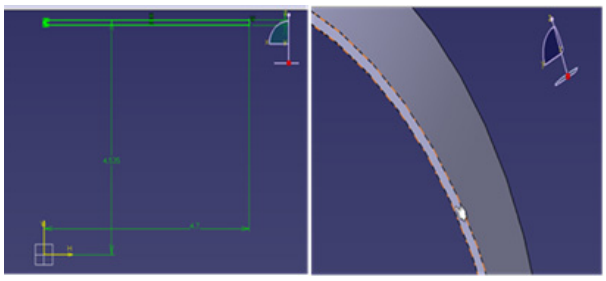


Figure 5. Sketching and revolving the middle rim

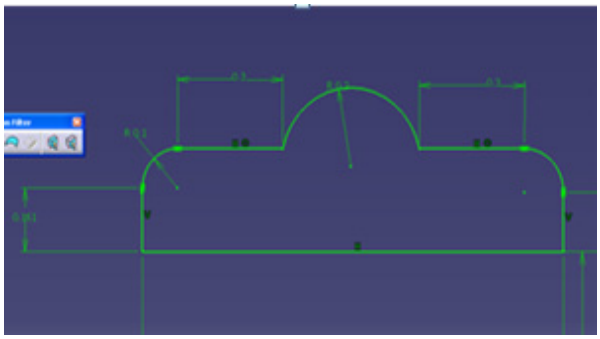


Figure 6. Sketching fin boss

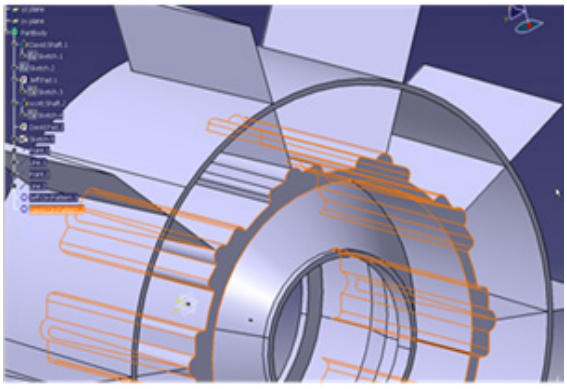


Figure 7. Apply 10 instances of extruded boss

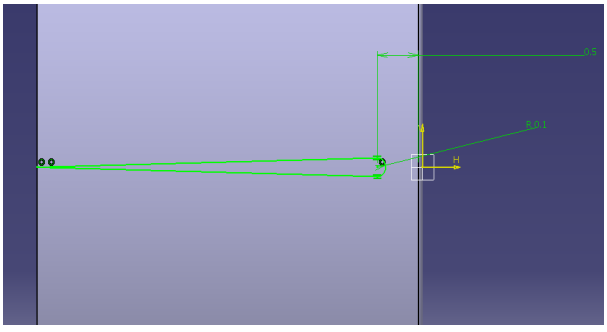


Figure 8. Sketching and revolving the middle rim

User 3: ***Step 1*** - Build the fin by making the sketch shown in Figure 8 on the new plane and pad it 1 inch (25.4 mm) to -2 inches (50.8 mm). Edge of fin is constrained to edge of rim and arc is tangent to both lines. Now update the part.

Step 2 - Pattern the fin in a complete crown circular pattern with 10 instances using any rim as a reference; see Figure 9.

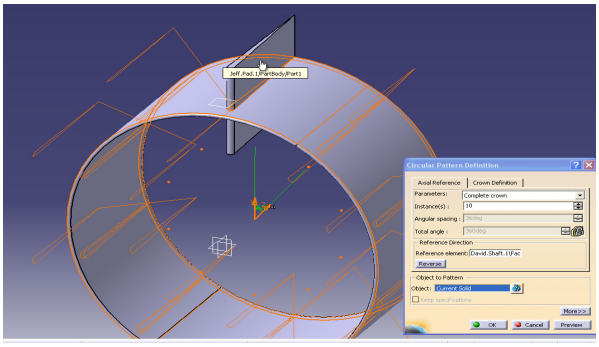


Figure 9. Sketching and revolving the middle rim

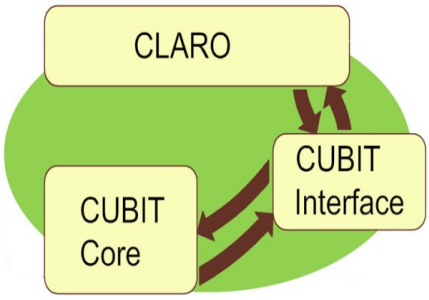


Figure 10. CUBIT Connect interfaces

If one expert is designated a session manager, and a multi-user completes an assigned task earlier than scheduled, we have seen cases where the session manager assigns that user new tasks, thus providing in-session load balancing. This is particularly effective when all engaged designers are experts, and require little in-session training.

4.3 CUBIT Connect

CUBIT is a mesh generation tool developed primarily by Sandia National Laboratories. CUBIT comprises: 1) CUBIT Core that executes the meshing algorithms; and 2) Claro, the Graphical User Interface (GUI). The Core and the GUI communicate using a C++ API called CubitInterface as depicted in Figure 10. Access to the CUBIT source code enabled us to develop our CUBIT multi-user prototype within CubitInterface using C++. CS architectures pass the change information among the clients.

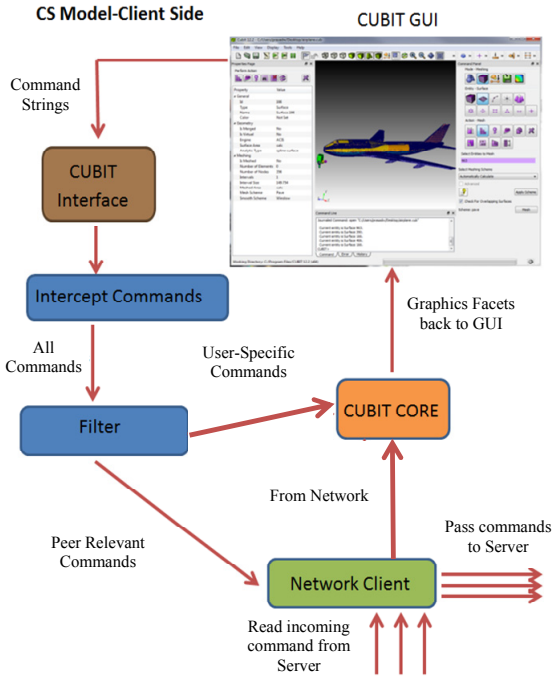


Figure 11. CUBIT client side architecture

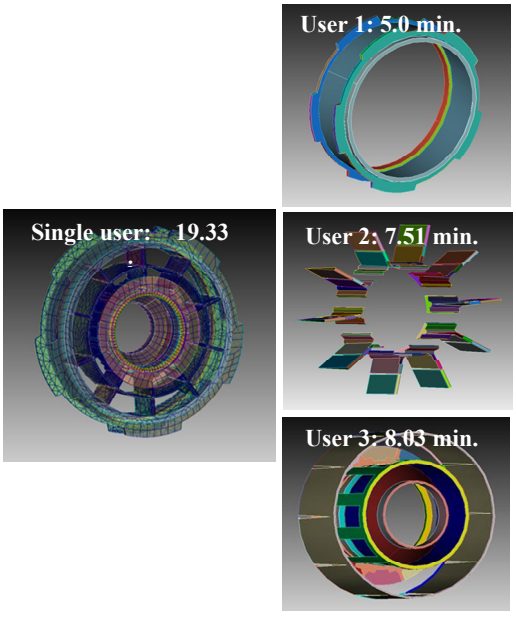


Figure 12. Three users meshing the front frame

Whenever the user makes a model change (feature change, mesh command, etc.), a command string is passed to the CUBIT Core through the CubitInterface. The command is then executed and the results passed back to the GUI for display through the same interface; see Figure 11. Weerakoon et al. (2012) provide more detail on the CUBIT Connect architecture.

The front frame model in Figure 12 was decomposed into three regions so that three clients could test CUBIT multi-user editing functionality. Three separate users manually edit elements, each in an assigned region. Multi-user regional assignment is easier in CUBIT Connect because CUBIT identifies and numbers uniquely the geometric modular components of the design model, much like an assembly. In this example the time savings was 59%, rather than 66.7%, because the editing load could not be balanced perfectly across the three multi-users.

Algorithmic Performance Delays – Whereas previous multi-user prototypes have focused on multi-user CAD and document editing, the CUBIT Connect prototype revealed an interesting problem for simultaneous analytical collaboration. The problem is that meshing algorithms can take minutes or hours to perform. Reflecting feature changes, such as meshing commands, among the clients could possibly halt all clients from editing for minutes or hours. Algorithmic delays are not a substantial problem in CAD or CAM applications, but can be significant in CAE applications like CUBIT. This problem is compounded when CPU performance varies among the clients.

Undo Command – CUBIT Connect and the other multi-user prototypes revealed that undo sequences can get complicated in a multi-user environment and really have not been addressed practically by the research community. For example, Li (2008) notes that "a feature of a product model might depend on other features and modifying an early feature may cause later features to become invalid". Gao et al. (2009) consider undo intent by a set of multi-users and uses local history buffers to record the changes made by local clients. Any user in a multi-user session could apply a feature undo function. Undo's can cascade such dependencies and cause chaotic collaboration. This is why some researchers are proponents of locked/blocked feature editing or spatial assignment (users constrained to different regions of the model), including the authors.

Given source code access to CUBIT, the following undo command method is being developed; refer to Table 1. Each command generated is sent to the server with the user ID and a time stamp. This can be done by using a JSON style string (Aziz & Mitchell, 2007). The client computer's sending module attaches a User ID to the command string before it is sent to the server for broadcast. Then the listening module on the server attaches a timestamp to the string containing both the command and the user ID; see Figure 13.

Table 1. Command string for undo commands

Command String	User ID	Timestamp
brick 10x 10y 10z	user_A	02:15:37 2011-11-30
mesh surface 1	user_B	02:19:27 2011-11-30
mesh surface 5 2 6	user_C	02:20:12 2011-11-30
node 308 move X -0.4;Y -0.08; Z 0	user_A	02:25:56 2011-11-30

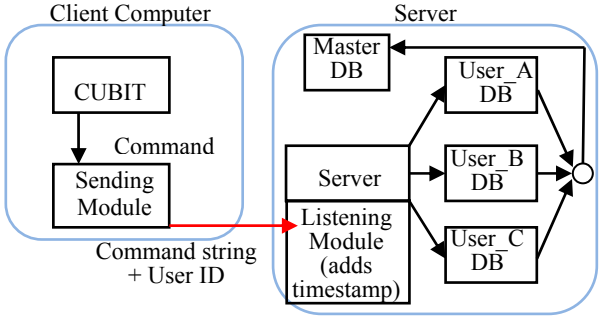


Figure 13. Undo model for CUBIT Connect

The server then manages two databases:

- Master Database (MDB): contains all the commands from all users.
- User Database (UDB): each user has their own database which consists of only the commands they generated.

The MDB maintained by server is used to sync all client computers and effectively makes their models current. When a user sends an undo command, the server looks up what command needs to be undone in the UDB and then deletes the command from there. This can be done without much difficulty if the user wants to undo the last action. The MDB is then synced with the UDB to reflect those changes.

Research is currently underway to implement and test this undo handling model and early results are promising. Figures 14 a), b), and c) show a multi-user engine block design session with three multi-users. User 1 performs a Boolean operation to create the piston chambers, see upper left of Figure 14a. Figure 14b shows the Boolean operations reflected to users 2 and 3. Figure 14c shows an undo operation performed on the piston Boolean in the local history tree. The undo operations have not yet been reflected from the server to the other two users, but this happens in a succeeding step. When a user performs an undo/redo on an operation owned by another user (server history tree), a message pops up to void the action. Negotiation between users is then used to determine the final undo/redo action.

Manual Update in Multi-User FEA Applications - NX Connect and CATIA Connect have automatic update functions built into them. When a user makes a change to the model, it gets updated on other client computers almost instantly. However, in an FEA application like CUBIT, some complex meshing algorithms can take an extended period of time to complete on the computer. This can be further complicated if the model/assembly that is being worked on is large and complex. Since CUBIT's Core and the GUI run on the same thread, the user is not able to work on the model until all those commands are completed. As mentioned before, some commands can take several minutes, if not hours, to be run.

Even if the Core and the GUI run on separate threads, meshing algorithms can utilize a considerable amount of system resources to execute. Therefore, the client computer can freeze when those commands run in the background. A manual update by client option has been implemented to overcome this latency problem at the present. However, to make the update function more streamlined, a different update model is being considered where each client computer keeps track of how long a command takes to execute completely and then that information is passed to the server. The execution time can be found by starting a counter when the command string is sent to the core and stopping the counter when the task completion message is displayed on CUBIT's command line.

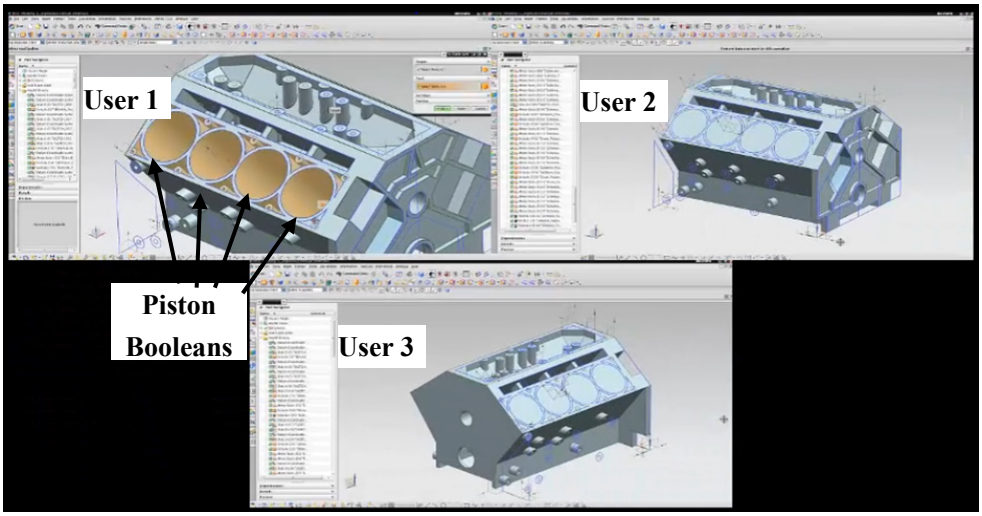
If a user wants to sync with other users, they can query the server for the time it took for the other users' commands to complete. The user can then decide whether to update now or at a later time. Another option is to update each user's changes. Since each user has a separate database containing their commands, other users can ask the server to send the updates of a specific user.

5. Architectural Considerations

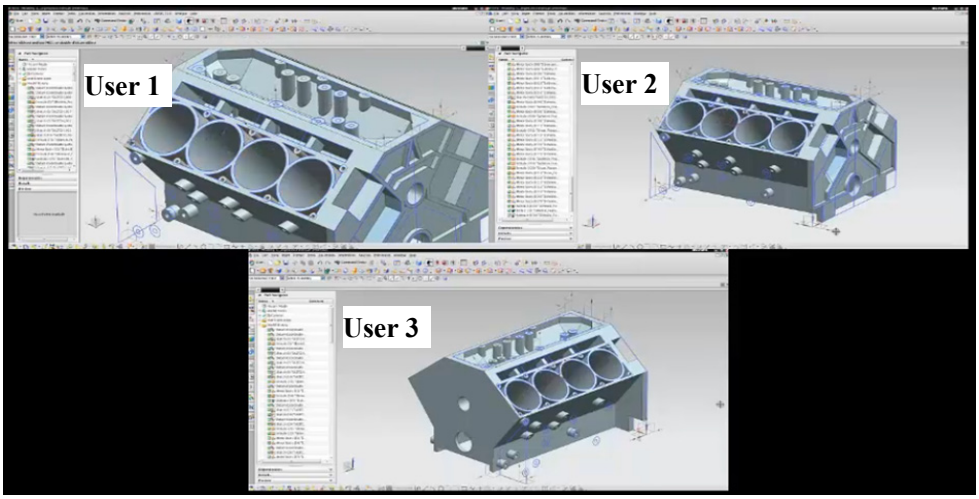
The research and prototype implementations have enabled us to draw conclusions about the limitations of single user architectures in commercial CAx applications. For example, the mostly single threaded API interfaces of current CAx applications limit the architecture and force multi-user capabilities to be implemented as an extension to the main CAx application. Thus, multi-user capabilities must be programmed bit-by-bit to extract the feature and action parameters for each multi-user action. Some actions may be difficult or impossible to program due to current API access limitations.

In CUBIT Connect we had access to the source code and therefore the event handler. This allowed us to make virtually all of CUBIT's commands multi-user with less effort. In NX and CATIA we had no direct access to the event handler; thus we used other methods such as parsing undo marks to access model parameter changes in a multi-user session.

a) Create pistons



b) Reflect edit



c) Undo pistons

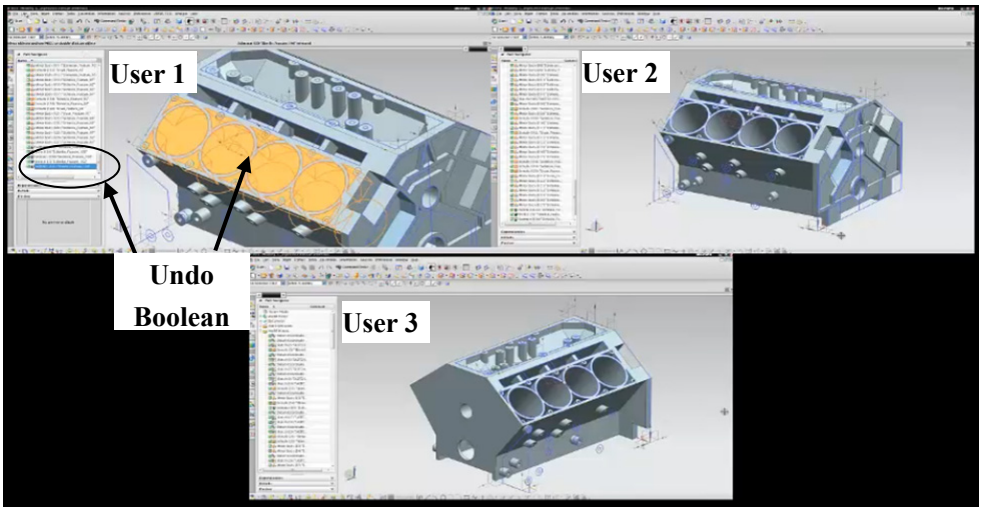


Figure 14. Create pistons undo example

Application of the CUBIT Connect prototype revealed serious algorithmic performance delays that could make multi-user session control difficult. This issue is being examined because it will require more flexible broadcasting methods between client and server as design modifications are made to a session model.

Red et al. (2013) noted that "API's provide handles to geometric objects (memory addresses) that cannot be passed to other multi-users over a network since memory locations on each computer vary." Using undo markers and extensive data structure investigation, extensive programming was needed to extract the data structure parameters. Future multi-user CAx applications must provide the parameters directly, perhaps through an object copy, rather than through object memory addresses.

CAx applications, like most PC applications, are designed for single users: display screen, single view-point, and single cursor. The user interfaces have changed little over the past two decades. CAx applications generally run on a single serial interface thread with GUI events invoking methods that are performed sequentially. GUI's are not designed for multi-user sharing, which means that users can only view the model from one viewpoint, even when using screen sharing applications over a network.

CAx GUI's use events triggered from a user GUI mouse or keyboard action to call software algorithms for model construction and change. Unfortunately, CAx API's are not designed to be interrupted by the event trigger. Multi-user prototypes typically have to use update buttons or polling cycles to observe and record user changes for reflection from the server to other multi-users engaged in collaborative modeling. A threaded multi-user architecture would use the event handler to trigger interrupts when the model is being changed, and would be more efficient. Other functionalities, such as sound alerts for client model changes, or client session entry/departure, could be easily implemented using event interrupts.

Change management is a research area needing more extensive research. Undo (Redo implied) sequences do not recognize multi-user input and user undo histories are vulnerable to undo actions by multi-users. The complexity increases if the several users are working in the same region and any undo action is complicated by CAD feature tree dependencies.

This is further complicated by algorithms that take substantial time to complete. These major architectural deficiencies fall under the category of change management. Along with multi-user distributed security issues, change management must be addressed by CAx developers before multi-user collaboration can be practical.

5.1 Multi-User Practicality

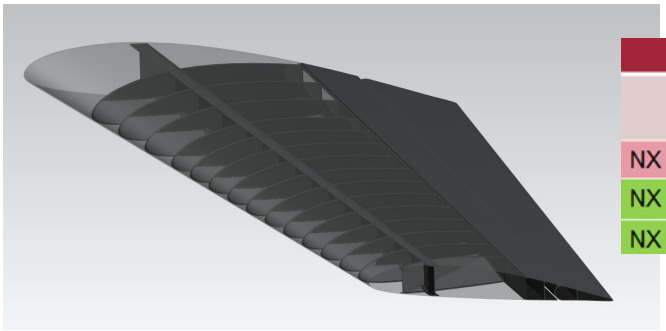
What part/model features and characteristics or contexts make multi-user practical? It seems evident that complex and larger parts might qualify, particularly when the parts spatially distribute repeating features. Our research has shown that multi-user contexts are not limited to just size and spatial distribution. The key is to find reasonable independencies among the features or spatial locations. Multi-user CAD may be the most difficult application because of feature tree dependencies, whereas other application areas such as architecture, integrated circuit and circuit board design, and manufacturing CAM have naturally occurring and reasonably independent features ripe for multi-user decomposition (different building floors, multiple IC and board layers, CAM roughing versus pocketing versus holes, threads, fillets, chamfers, etc.).

First, consider model context and model complexity. Previous researchers may have underestimated the importance of multi-user training modes or design rationale sharing modes, even for simpler parts. Users who can share a model, rather than view a model with limited input, can engage in design edits and related design decisions, either as trainer, trainee or as design reviewer or modifier. Users engaged in such sessions share the design rationale, which is more difficult in single user sessions. Users may be distributed over a number of sites and may be from several organizations such as design, analysis, or manufacturing.

Second, consider the multi-user design session context. The literature offers competing rationale for the best multi-user collaborative methods. For example, Zheng (2009) suggests that collaboration should be open, not region or feature locked, with negotiation handling of collision problems, whereas Moncur (2012), Li (2008) and Jing (2009) suggest that feature and local locking resolve many of the conflict issues. Our research shows that all these methods have appropriate validity, and that experienced users will gravitate to collaborative methods appropriate to the part, given that they are available through the user interfaces.

Figure 15 provides an interesting wing design experiment conducted by BYU researchers where the modeling efficiency improved as the collaborative session engaged 2 users (100 minutes total) and then engaged 4 users (72 minutes total). The improvement occurred because of 1) natural feature and spatial decomposition over several wing ribs and spars; 2) session leader decomposed and assigned tasks while in the active session based on user knowledge and performance; and 3) the clients competed among themselves to finish sooner. In this experiment the assignments changed dynamically (session leader made re-assignments during the multi-user session), depending on progress and work completed, reinforcing the notion that multi-user design situations can be adaptive.

If multi-user tasks can be decomposed based on reasonable feature independencies, then we expect and have shown productivity proportional to the number of multi-users in the session. Generally, this will require models that offer reasonably independent design features or easy spatial decomposition. When multi-client negotiation becomes more intense, the process becomes more serial, although the negotiated design may likely improve, as design rationale is mutually agreed upon. Marshall (2011) implemented a method to spatially decompose an object among several multi-users, where a user could only access/select the object features within the assigned space, and where planes were used as boundary constraints. A user's mouse could not select user features outside the user's assigned space. This early spatial constraint prototype does not resolve all the issues, such as objects that protrude through boundaries, where negotiation must occur.



Wing Modeling Times		
	# of Modelers	Time in Minutes
NX 8.0	1	120
NX Connect	2	50
NX Connect	4	18

Figure 15. Multi-user wing modeling times

As we have developed tools for spatial and feature decomposition, we have also used negotiation to manage our multi-user sessions. The engine frame of Figures 2-9 was easily decomposed by pre-planning because of the axisymmetric shapes and thus easily decomposed regions; no constraint planes were used since the users understood what parts they were designing. A master user blends the objects across boundaries to complete the design.

We are currently developing object tree comparison methods that notify clients of potential changes to the tree branch by another user, where negotiation may be necessary before allowing changes to cascade through the model. The model tree is maintained at the server along with the client ID for each object node and related operation. When a client tries to modify a feature or change an operation in a tree branch with divided ownership, negotiation must occur before the changes can be applied.

6. Conclusions and Future Work

Multi-user collaboration is both feasible and desirable, given that architectural limitations can be addressed, such as: multi-threading of CAx API's and GUI's, access to CAx event handlers and interrupts, undo's/redo's (change management), and API's that can provide feature parameter copies rather than address handles, along with a hosts of organizational and security features that have not really been considered by the research community. Our research prototypes were designed to discover additional limitations like these.

Our research (Jensen, 2012) has established that, at the session level, design times can be reduced in proportion to the number of collaborators. Although this result seems important, particularly to industry, previous multi-user researchers have not published productivity numbers as they scale to the part complexity and the number of clients. We have also determined that CS architectures accommodate both synchronous and asynchronous modes (random client sign-in), because the master model is always maintained in the server database and can update any client upon invocation. We assume that previous researchers see the proportionality figure as an obvious expectation, but we note that industry practitioners expect test verification. These results have been verified using the multi-user prototypes for NX and CATIA. These CAx applications are the primary applications for complex product designs in aerospace and automotive companies. Productivity in organizational decomposition and management of the multi-user paradigm is yet to be established, and a critical subject for future research.

Our research surveys have discovered that product engineers are using a large number of *social* communications tools to collaborate in the absence of embedded CAx collaborative tools. Social tools like email, instant messaging, etc., have features to record communication interactions and histories. The next step is to integrate their features into CAx applications and then associate decision histories with the current design model files. Xu

(2010), a BYU ν-CAx researcher, used a multi-user GUI prototype to integrate automatic language translation of text messages between a culturally variant team of CAx collaborators.

What has yet to be addressed by the research community is the organizational structure and related tools to decompose/manage multi-user teams assigned to designs and specifications. We are examining how company-wide pervasive databases can capture and manage the experiential capabilities of design personnel, and also incorporate their cultural and previous teaming histories. Design conflicts in multi-user sessions can be reduced by prior organizational review of design specifications, model complexity, potential for feature distribution among client users, and regional decomposition based on model complexity. This assumes that new multi-user CAx applications have the associated decomposition and feature locking tools.

We have established that CS architectures (thin server, thick client) can accommodate different tools and application types (CAD versus CAE). We are now examining cloud serving architectures where the client accesses the multi-user applications on the cloud server through a terminal display. With all applications distributed on the cloud server, no CAx application resides at the client station.

It was easier to convert CUBIT to multi-user because we had access to the source code (and event handler) and only had to filter the commands sent to peers. In the case of CATIA or NX the API functionality was limited and had to be implemented bit-by-bit. Effectively, we had to write distinct code for each implemented function, no different from prior researchers in multi-user CAD who have adopted transparent adaptation (TA) methods (e.g., Sun, 2009; Zheng, 2008).

We have discovered that variable algorithmic performance is a major impediment to multi-user design session productivity. This was discovered in our CUBIT Connect prototype. Automatic reflection of model changes between the server and active clients can halt an online client from performing any design function. We are considering more advanced architectures to transmit model changes and inform clients of algorithmic delays, and also considering how to enable distributed computational resources.

The paper has reviewed several network architectures and prototypes to measure collaborative effectiveness. Our prototypes are being transitioned to a cloud serving architecture where effectiveness will depend on network latencies, and may be ineffective across widely distributed global networks. Inertial tendencies may provide momentary reluctance for this flexible new CAx architecture but it simply offers too many advantages for global product development; see Stackpole (2012). Mirroring of multi-user operations and master model parameters using distributed cloud servers could prove effective.

Acknowledgements

The National Science Foundation, ν-CAx member companies, and BYU's ν-CAx research students are acknowledged for their funding and conducting of this research as a Center for e-Design Site.

References

Aziz, A., & Mitchell, S. (2007). *An introduction to JavaScript object notation (JSON) in JavaScript and .Net*. Retrieved from http://msdn.microsoft.com/en-us/library/bb299886.aspx

Barbosa, C., Feijo, B., Dreux, M., Melo, R., & Scheere, S. (2003). Distributed object model for collaborative CAD environments based on design history. *Advances in Engineering Software, 34*(10), 621-631. http://dx.doi.org/10.1016/S0965-9978(03)00095-4

Bonneau, P., & Gabrielaitis, I. (2009). Applying multi-user technology for modeling complex CAD objects. *Statybines Konstrukcijos Ir Technologijos Engineering Structures and Technologies, 1*(1), 89-94. http://dx.doi.org/10.3846/skt.2009.11

Bu, J., Jiang, B., & Chen, C. (2006). Maintaining semantic consistency in real-time collaborative graphics editing systems. *IJCSNS International Journal of Computer Science and Network Security, 6*(4), 57-61.

Cera, C., Regli, W., Braude, I., Shapirstein, Y., & Foster, C. (2001). *A collaborative 3D environment for authoring of design semantic*s. Drexel University Technical Report DU-CS-01-06.

Chen, L., Song, Z., & Feng, L. (2004). Internet-enabled real-time collaborative assembly modeling via an e-assembly system: status and promise. *Computer-Aided Design, 36*, 835-847. http://dx.doi.org/10.1016/j.cad.2003.09.010

Chen, X., Fuh, J., Wong, Y., Lu, Y., Li, W., & Qiu, Z. (2005). An adaptable model for distributed collaborative design. *Computer-Aided Design & Applications, 2*, 47-55.

Fan, L., Kumar, A., Jagdish, B., & Bok, S. (2008). Development of a distributed collaborative design framework within peer-to-peer environment. *Computer-Aided Design, 40*, 891-904. http://dx.doi.org/10.1016/j.cad.2008.05.006

Fuh, J. Y. H., & Li, W. D. (2005). Advances in collaborative CAD: the-state-of-the-art. *Computer-Aided Design, 37*, 571-581. http://dx.doi.org/10.1016/j.cad.2004.08.005

Gao, L., Lu, T., & Gu, N. (2009). Supporting semantic maintenance of complex undo operations in replicated Co-AutoCAD environments. *Proceedings of the 2009 13th International Conference on Computer Supported Cooperative Work in Design*. April 22 – 24, Santiago, Chile.

Hannan, M. T., & Freeman, J. (1984). Structural inertia and organizational change. *American Sociological Review, 49*, 149-164. http://dx.doi.org/10.2307/2095567

Jafari, M. M., Ahmed, S., Dawal, S. Z. M., & Zayandehroodi, H. (2010). The effects of electronic collaboration in reducing production time: product design process in SMEs. *Proceedings of the International MultiConference of Engineers and Computer Scientists.* March 17 – 19, Hong Kong, 3.

Jensen, G. (2012). Collaborative multi-user synchronous and asynchronous modeling, analysis and design. Keynote presentation. *Defense Manufacturing Conference (DMC),* November 26 – 29, Orlando, Florida.

Jing, S., He, F., Han, S., Cai, X., & Liu, H. (2009). A method for topological entity correspondence in a replicated collaborative CAD system. *Computers in Industry, 60*(7), 467-475. http://dx.doi.org/10.1016/j.compind.2009.02.005

Kushwaha, J., & Roy, B. N. (2010). Secure image data by double encryption. *International Journal of Computer-Aided Applications, 5*(10), 28-32. http://dx.doi.org/10.5120/947-1325

Lai, Y. (2009). A constraint-based system for product design and manufacturing. *Robotics and Computer-Integrated Manufacturing, 25*(1), 246-258. http://dx.doi.org/10.1016/j.rcim.2007.12.003

Li, M., Gao, S., & Wang, C. L. (2007). Real-time collaborative design with heterogeneous CAD systems based on neutral modeling commands. *ASME Journal of Computing and Information Science in Engineering, 7*, 113-125. http://dx.doi.org/10.1115/1.2720880

Li, M., Gao, S., Fuh. J. Y. H., & Yang, Y. F. (2008). Replicated concurrency control for collaborative feature modeling: a fine granular approach. *Computers in Industry. 59*, 873-881. http://dx.doi.org/10.1016/j.compind.2008.07.003

Marshall, F. (2011). *Model decomposition and constraints to parametrically partition design space in a collaborative CAx environment.* Master's Thesis, Department of Mechanical Engineering, Brigham Young University.

Moncur, R. A. (2012). *Data consistency and conflict avoidance in a multi-user CAx environment*. Master's Thesis, Department of Mechanical Engineering, Brigham Young University.

Mounir, H., Nizar, A., Borhen, L., Benamara, A., & Deneux, D. (2013). FEM simulation based on CAD model simplification: a comparison study between the hybrid method and the technique using a removing details. *Design and Modeling of Mechanical Systems* (pp. 587-596) LMNE, Springer-Verlag Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-37143-1_70

Owen, S. J., Clark, B. W., Melander, D. J., Brewer, M., Shepherd, J. F, Merkley, K., … Morris, R. (2007). An immersive topology environment for meshing. *16th International Meshing Roundtable, Seattle WA.*

Panchal, J., Fernandez, M., Paredis, C., Allen, J., & Mistree, F. (2007). An interval-based constraint satisfaction (IBCS) method for decentralized, collaborative multifunctional design. *Concurrent Engineering, 15*(3), 309-323. http://dx.doi.org/10.1177/1063293X07083083

Ramani, K., Agrawal, A., & Babu, M. (2003). CADDAC: Multi-client collaborative shape design system with server-based geometry kernel. *Transactions of the ASME, 3*, 170-173.

Reddy, V. K., & Reddy, L. S. S. (2011). Security architecture of cloud computing. *International Journal of Engineering Science and Technology, 3*(9), 7149-7155.

Red, E., French, D., Jensen, G., Walker, S., & Peter Madsen, P. (2013). Emerging design methods and tools in collaborative product development, *J. Comput. Inf. Sci. Eng, 13*, (September, 2013) (14 pages).

Red, E., Holyoak, V., Jensen, G., Marshall, F., Ryskamp, J., & Xu, Y. (2010). A research agenda for collaborative computer-aided applications. *Computer-Aided Design and Applications, 7*(3), 387-404. http://dx.doi.org/10.3722/cadaps.2010.387-404

Red, E., Jensen, G., French, F., & Weerakoon, P. (2011). Multi-User architectures for computer-aided engineering collaboration. *17th International Conference on Concurrent Enterprising.* Aachen, Germany.

Red, E., Marshall, F., Weerakoon, P., & Jensen, G. (2012). Considerations for multi-user decomposition of design spaces. *CAD12*, Niagara Falls, Canada, June (to be published in *Journal of Computer-Aided Design and Applications*, 2013.)

Ryskamp, J., Jensen, G., Mix, K., & Red, E. (2010). Leveraging design rationale to improve collaboration in multi-user CAD. *Proceedings of the TMCE*. Ancona, Italy.

Shaojin, Y., Jianjun, C., & Jindou, L. (2010). An asynchronous CAD collaborative design model. *2010 International Conference on Computer Application and System Modeling, 6*, 563-568. http://dx.doi.org/10.1109/ICCASM.2010.5620677

Stackpole, B. (2012). Engineering tools take to cloud. *Design News*. July 23.

Sun, C., Xia, S., Sun, D., Chen, D., Shen, H., & Cai, W. (2006). Transparent adaptation of single-user applications for multi-user real-time collaboration. *ACM Transactions on Computer-Human Interaction, 13*(4), 531-582. http://dx.doi.org/10.1145/1188816.1188821

Wang, Y., Ajoku, P., Brustoloni, J., & Nnaji, B. (2006). Intellectual property protection in collaborative design through lean information modeling and sharing. *ASME Transactions Journal of Computing and Information Science in Engineering, 6*(2), 149-159. http://dx.doi.org/10.1115/1.2190235

Weerakoon, P., Wu, J., Bright, T., Teng, C., Red, E., Jensen, G., & Merkley, K. (2012). A networking architecture for a multi-user FEA pre-processor. *Computer-Aided Design and Applications, 10*(3), 527-540. http://dx.doi.org/10.3722/cadaps.2013.527-540

Xu, Y. (2010). *A flexible context architecture for a multi-user GUI*. Master's Thesis, Department of Mechanical Engineering, Brigham Young University.

Yoshimura, M. (2012). Framework and methodologies for maximizing achievements of product designs by collaborative works. *Journal of Engineering Design, 23*(9), 674-695. http://dx.doi.org/10.1080/09544828.2011.651715

Zheng, Y., Shen, H., & Sun, C. (2008). Adapting single-user autoCAD system to support realtime collaborative design: issues, challenges and achievements. *Proceedings of the Tenth International Workshop on Collaborative Editing Systems in conjunction with ACM Conference on Computer Supported Cooperative Work*.

Zheng, Y., Shen, H., & Sun, C. (2009). Leveraging single-user AutoCAD for collaboration by transparent adaptation. *13th International Conference on Computer Supported Cooperative Work in Design.* Santiago, Chile.

Zheng, Y. (2012). Fundamentals of next generation CAD/E systems. Editorial, *Computer-Aided Design, 44*, 875-878. http://dx.doi.org/10.1016/j.cad.2012.05.005