Multi-User Computer-Aided Design and Engineering Software Applications

Edward Red, David French, Ammon Hepworth, Greg Jensen and Brett Stone

Abstract This chapter will introduce multi-user computer-aided engineering applications as a new paradigm for product development, considering past collaborative research and the emerging wave of cloud-based social and gaming tools. In a historical context, computer-aided design and engineering models have become much more complex since their inception in the middle of the twentieth century. However, the way design teams approach these models has, at least in one sense, not changed much; a given model can still only be accessed by one user at a time, despite the fact that the entire design team needs to evolve the model. Single user applications have become a productivity bottleneck and do not provide interfaces or architectures for simultaneous editing of models by a collaborative team. Single user applications convert any hope for process concurrency into a serial sequence of design activities. When the single user designer experiences difficulties, the process halts until the designer can reach out to other experts to resolve the problems, which usually requires some form of external collaboration. Unfortunately, single user applications are deficient when it comes to complex and globalized product development. The chapter herein will consider how multi-user architectures will change the single user paradigm from serial to simultaneously collaborative, promote new on-demand access methods like cloud serving, and bring long hoped for efficiencies to product development. We will investigate three research areas of

E. Red (✉) · D. French · A. Hepworth · G. Jensen · B. Stone
Brigham Young University, Provo, UT, USA
e-mail: ered@byu.edu

D. French
e-mail: davidfrench11@gmail.com

A. Hepworth
e-mail: ammon.hepworth@gmail.com

G. Jensen
e-mail: cjensen@byu.edu

B. Stone
e-mail: brettstone87@gmail.com

importance to this emerging paradigm: (1) multi-user CAx architectures, including cloud serving; (2) multi-user CAx requirements; and (3) multi-user CAx standards. Of these three, architectures are most investigated, with numerous proof-of-concept prototypes, while requirements and standards, the least investigated, partially explain the reason for non-adoption and non-commercialization of this powerful new paradigm.

Keywords Computer-aided design · Multi-user · Cloud serving · Organizational management · Collaborative design · Concurrent engineering

1 Introduction

Decades ago, engineers and other technical personnel gathered around large drafting tables to review and blend their design contributions, and consider technical requirements and pressing schedules, Fig. 1. Collaboration was natural and transparent in this hands-on environment, at least until the ensuing digital age of desktop workstations and computer-aided applications (CAx) encouraged new methods of decomposing product design among technical individuals, where one person is assigned a part model design. Today, file-based control systems like product lifecycle management (PLM) track and maintain model file version changes, using secured check-in and check-out procedures.

Collaboration has become increasingly difficult as multiple designers are unable to simultaneously enter an editing session, although they can screen share the model. As products become increasingly complex, profit incentives have decomposed the various system (airplane, ship, tank, etc.) components among globally distributed suppliers. These suppliers often use different vendor supplied CAx applications to produce their contracted component, resulting in model variations and inconsistent file formats, referred to as noninteroperable data. Distribution and integration of these models among the supplier chain usually results in many model file conversions because of installed CAx heterogeneity. Because of cascaded file conversion errors, suppliers may find it necessary to replicate the model in their native CAx application. The annual cost associated with interoperability is on the order of hundreds of millions of dollars (USD) (Brunnermeier et al. 1999).

Computer-aided engineering applications like CAD/CAE/CAM (CAx) have continued to grow in complexity and capability, but have remained single user while the concurrent demands of building a new class of technically evolved products like modern transportation systems have escalated. Concurrency is further complicated by business practices where large product companies now depend on distributed global supplier chains to build their system components. Each supplier's CAx tools may vary and secondary CAx model file translations introduce errors and extend the development cycle (Contero et al. 2002).

Fig. 1 Collaboration in 1979

Fig. 2 Gaming session

Client-server gaming and related cloud serving architectures have demonstrated that teams can closely and simultaneously collaborate in complex, dynamically changing landscapes, using distributed cloud servers that manage changes to model data (Kim 2002; Fig. 2).

While cloud-based applications centralize data and make it accessible over great distances, even across continents, CAx applications isolate and protect data for a single user. A single designer creates and details a component model using modern computer-aided design (CAD) applications. A single analyst applies computer-aided engineering (CAE) tools to determine whether a component can withstand the structural loading, or elevated temperatures, or whether the component is aerodynamically stable. Finally, single users apply computer-aided manufacturing (CAM) and computer-aided process planning (CAPP) to determine and program the manufacturing processes and machines to make the component.

Surveys (Red et al. 2013a, b) have shown that technical personnel engage in some form of collaboration at least 50 % of each day (Fig. 3), either in formal or ad hoc meetings or by applying a number of social media tools like conference

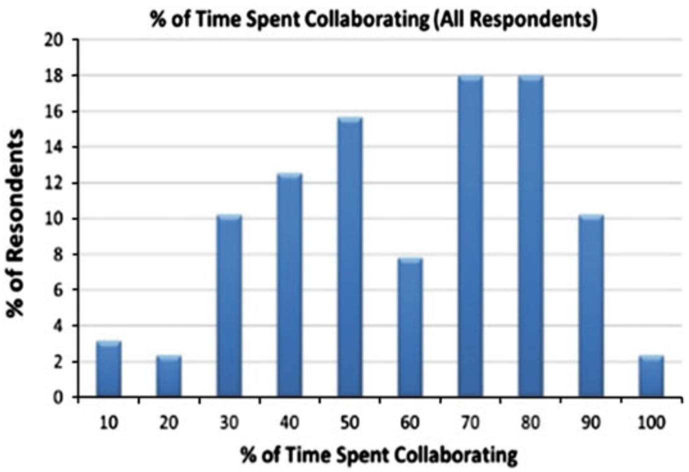


Fig. 3 Daily % collaboration

Fig. 4 Conference call design meeting

calls (Fig. 4), email, texting, instant messaging, and mobile phones. These social tools necessarily compensate for the collaborative deficiencies in modern CAx applications.

Multi-user CAx applications provide real-time collaboration and thus enable Cloud-Based Design and Manufacturing (CBDM), although expanding the concepts described by Wu et al. Wu defines CBDM as follows (Wu 2012):

> Cloud-Based Design and Manufacturing refers to a product realization model that enables collective open innovation and rapid product development with minimum costs through a social networking and negotiation platform between service providers and consumers. It is a type of parallel and distributed system consisting of a collection of inter-connected physical and virtualized service pools of design and manufacturing resources (e.g., parts, assemblies, CAD/CAM tools) as well as intelligent search capabilities for design and manufacturing solutions.

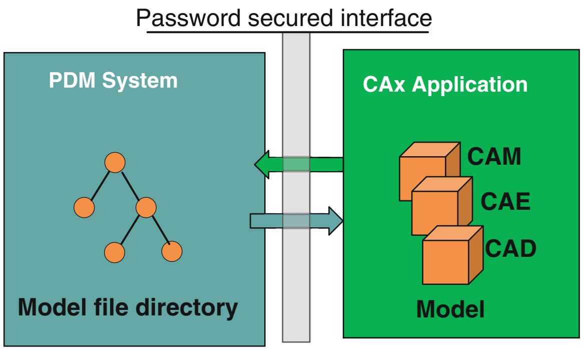


Fig. 5 PDM to CAx file exchange

Although Wu's pooling of CBDM resources refers primarily to cloud computing, we generalize CBDM to include human resources and data consistency resources, in addition to computer hardware and software. Multi-user CAx directly supports this generalized vision by distributing and parallelizing human resources to increase design process efficiency. We suggest that the CDBM vision will not be realized until the product model data becomes interoperable (CAX agnostic), regardless of consumer resource application of choice. Otherwise, the model's data format and integrity will be compromised by conformal translations to each user's application environment.

To understand the power of multi-user CAx applications, this chapter will consider three research areas: (1) multi-user CAx architectures; (2) multi-user CAx requirements; and (3) multi-user CAx standards. The last two have not been well researched, but will be important to consider nevertheless.

2 Multi-user CAx Architectures

We will consider three architectural areas: (1) collaborative networks; (2) collaborative interfaces; and (3) model sharing and conflict resolution.

2.1 Collaborative Networks

Today, network concerns in single user CAx relate to how fast a design model file can be exchanged between a local workstation and PDM server. The PDM server secures and manages the model file revisions, and other product-related data, Fig. 5. Speed is less of a problem when the network is behind company firewalls where high speed LAN's can distribute complex model files in less than an hour. But when companies engage in around the clock, 24 h global design, FTP file delivery may take hours to transfer encrypted models of large size (100's of MB) over the Internet.

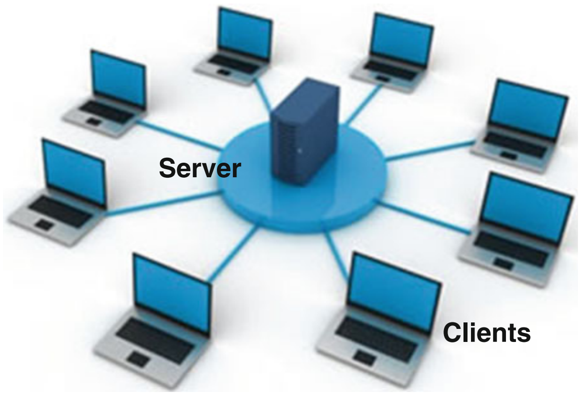


Fig. 6 Client-server architecture

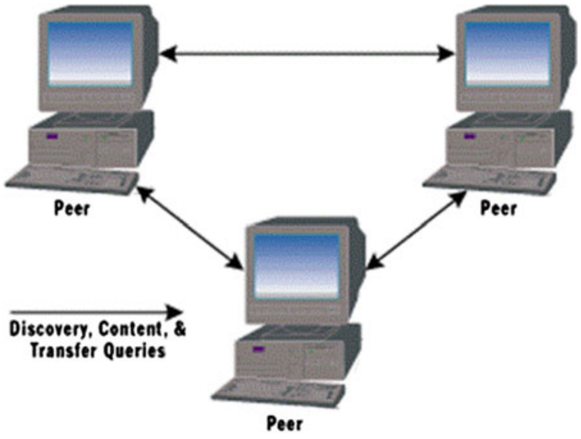


Fig. 7 Peer-to-peer architecture

Contrast single user file exchange with a multi-user network where several (or many) users, referred to as clients, simultaneously edit the model. Rather than files being passed over the network, model changes are passed between clients as small data packets, easily accommodated by even the slowest networks. With this perspective we note that two multi-user architectures dominate the collaborative networks: (1) client-server (CS, Fig. 6) and peer-to-peer (P2P, Fig. 7).

2.1.1 Client-Server

Most collaborative multi-user prototypes use client-server networks. The client-server variations configure the client workstations as thick clients or thin clients. Li's et al. review paper notes client-server dominance, lack of data security, and the difficulty of real-time interaction over networks (Li et al. 2005). Li distinguishes thick clients having a full model from thin clients where a partial model is mainly used for visualization.

Thick clients will run the CAx application on the each client's workstation and integrate software plug-ins to detect and transmit client model edits to the server where they are then reflected to other collaborating clients. These methods are referred to as "transparent adaptation" by Sun et al. (2006, only applied to MS Word and PPT) and Zheng et al. (2009, Co-AutoCAD) because the CAx application source is not modified.

BYU's client-server multi-user prototypes—Several multi-user CAx prototypes have been developed at Brigham Young University's NSF Center for e-Design site. These prototypes were built on native application API's (using C++, C#, Visual Basic) as software plugins to verify that mainstream CAx applications could function in multi-user mode, and to discover architectural limitations that might impede multi-user functionality. These prototypes allow for modeling in both single and multi-user modes. In multi-user mode, users visualize contributions from collaborators in real-time. This allows users to simultaneously contribute to a model and respond in real-time to input from others. Table 1 presents their functionality and limitations, while Figs. 6 and 7 show their architectures, although CS was most dominant. The CUBIT CAE prototype has access to the source code for CUBIT Core and thus does not entirely depend on API's for the implementation. The limitations of Table 1 will be considered later in Sect. 3 on *Multi-User CAx Requirements*.

We have implemented the Fig. 8 architecture on a laboratory LAN, on a college cloud server, and across an Internet WAN, all with similar effectiveness because small data packets easily transmit across networks. When using a cloud server, we applied Hewlett-Packard's Remote Graphics Software (RGS) to effectively turn the local workstation into a terminal screen, with the CAx application instances and multi-user application server running on remote server blades. We will discuss the security implications of these architectures later.

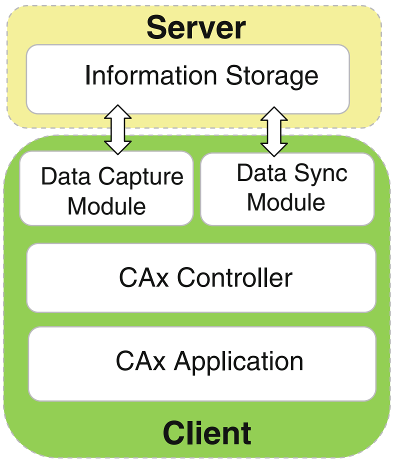
The architecture in Fig. 9 differs somewhat in implementation because we had access to the CUBIT Core source code. This allowed us to thread client interactions with the server using Windows Named Pipes (NP) (Microsoft Developer Network 2012) for inter-process communication (IPC) and TCP/IP sockets for network communications. Two networking clients (External and Internal Clients) reside on each local computer, with the Internal Client embedded into the source code.

The External Client (EC) organizes the different type of messages through a serialization process. The multi-user actions require that client ID's be appended to any GUI instruction as a message structure. Examples of message structures established in the EC are command message, master trigger, and database reset. Commands messages are generated from the CUBIT GUI, Fig. 9b, and delivered to the CUBIT core to process (e.g., "create sphere" or "mesh volume 1").

Master trigger messages are messages initiated by the user to resynchronize their model with the server database for unusual situations such as network down, or when clients engage the design process asynchronously and thus join the work environment at different times. Message structures help the server distinguish between different CUBIT Connect operations so it can respond accordingly.

Table 1 BYU's ν-CAx site prototypes

Vendor	Type	Name	Function	Limitations
Siemens NX	CAD	NX Connect	CAD	Single user architecture (kernels, single-threads, GUI's, API's configured for a single user)
				Limited access to event callbacks
				Single API thread
				Single user GUI
				Memory handles to entity data
				Few social media tools
				Some API functionality errors
				Single user undo/redo
				Models stored as files
Dassault Systèmes CATIA	CAD	CATIA Connect	CAD	Same as NX except CATIA API runs on separate thread
AutoDesk Inventor	CAD	Inventor Connect	CAD	Same as NX
Sandia CUBIT	CAE	CUBIT Connect	CAE structural analysis	Same as NX, except greater access to event handler
				Multi-user synchronicity limited by meshing times
				Meshing time user variability
				P2P model updating latencies, thus move to client-server version
				Entity ID's computer fixed



Client-server module functions:

- **Information Storage Module** – uses a relational database for data storage and a hierarchical structure to sync the part features and data changes.
- **Data Capture Module** - monitors the CAx session for changes to the part file and then passes the change information to other users through the server.
- **Data Sync Module** - monitors the information storage module for changes uploaded by another user, using these changes to alert the CAx Controller.
- **CAx 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.

Fig. 8 Client-server architecture for BYU's CAx (CAD) multi-user prototypes

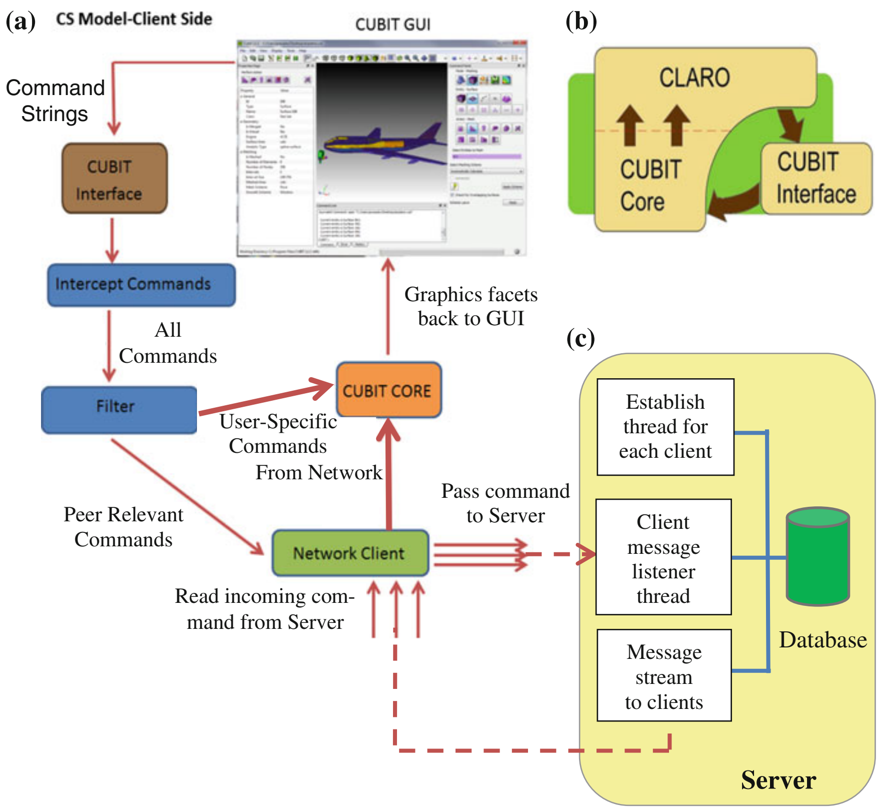
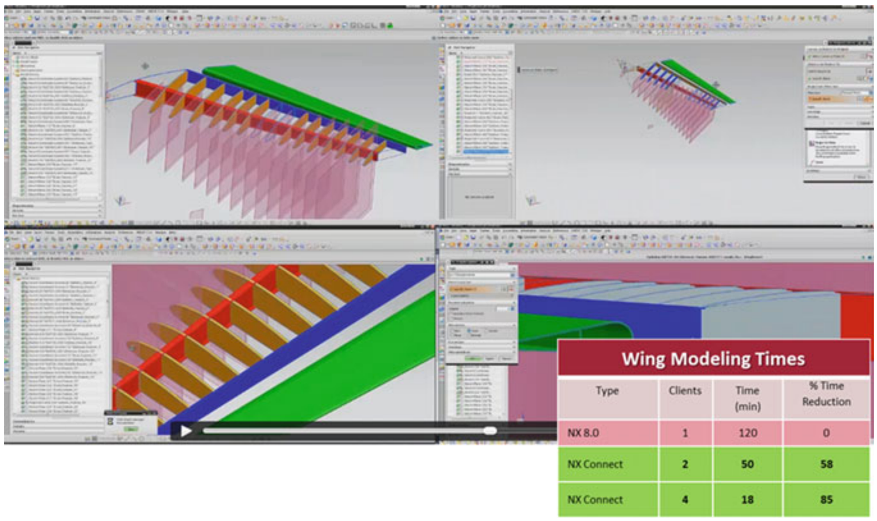


Fig. 9 CUBIT Connect: **a** client architecture; **b** CUBIT interfaces; **c** server functionality



Wing Modeling Times

Type	Clients	Time (min)	% Time Reduction
NX 8.0	1	120	0
NX Connect	2	50	58
NX Connect	4	18	85

Fig. 10 NX Connect with 4 clients developing a wing structure simultaneously

It should be mentioned that meshing algorithms, unlike CAD algorithms, can take minutes or even hours to complete. Since CUBIT Connect uses a thick client architecture, each client has to perform those same mesh operations to stay consistent with each other, and users must wait while those operations are performed. CUBIT, like most single user applications, runs on a single thread; thus, the CUBIT GUI freezes while a complex and time-consuming algorithm runs in the core. These actions are untenable in a multi-user environment and required changes to our server/client methods to permit multi-users to control when to update from clients, along with passing CPU meshing times for each meshing command among the clients for informed decision making.

In contrast, Fig. 10 shows an instance in a NX Connect modeling session where four multi-users simultaneously design a wing structure. Because the team leader was able to decompose tasks based on expertise, actual time reduction (85 %) was better than T/N (80 %), where T is the single user time and N is the number of multi-users; see the inset table.

Figure 11 shows a CUBIT Connect design session where three users simultaneously mesh three components of a race car developed in collaboration with 26 universities over 4 years as part of the PACE Global Vehicle Project. The question of how to filter and/or stack update commands from other multi-user clients is still being investigated. Algorithmic delays, not a substantial problem in CAD applications, and a minor problem in CAM applications, can be significant in CAE applications. Current research is considering operational vectors that are tagged to each multi-user and that can be applied at optimal times.

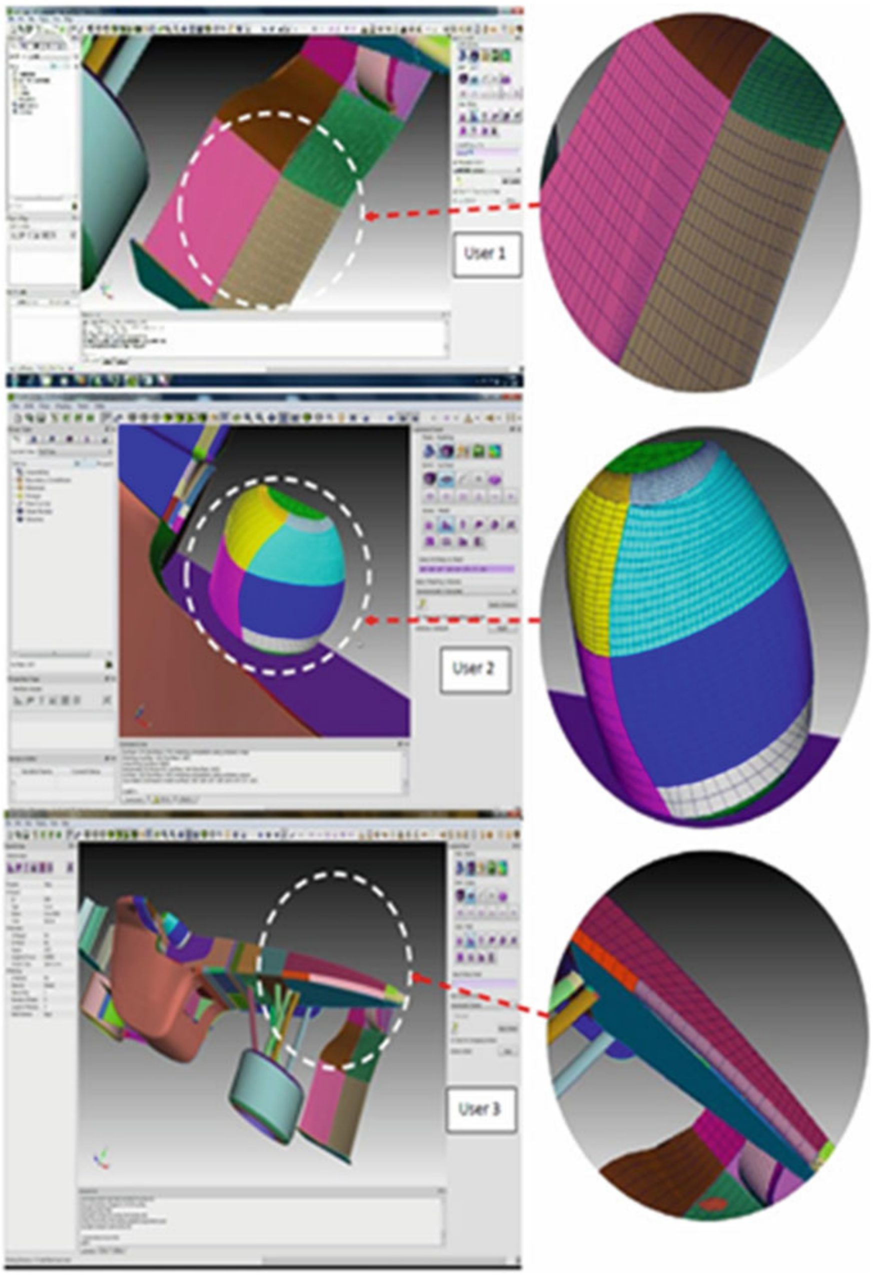


Fig. 11 CUBIT Connect session with 3 clients simultaneously meshing a race car

2.1.2 Peer-to-peer

An alternative architecture connects clients directly as peers, with each workstation replicating the software used to manage the change control for the model (Jing et al. 2009; Fan et al. 2008). In the gaming community, peer-to-peer (P2P) has many adherents, partially because the model data (game space) can accept some weak inconsistencies (Bharambe et al. 2006, http://www.cs.cmu.edu/~ashu/gamearch.html; GauthierDickey et al. 2004; Wang et al. 2004; Douglas et al. 2005; Endo et al. 2007). The common arguments are that client-server suffers from robustness and single point of failure (Bharambe et al.), lack of scalability (GauthierDickey et al.), and lack of dynamic reconfigurability (Ramakrishna et al. 2006).

P2P architectures also have weaknesses: (1) data inconsistency; (2) variable network latencies; (3) weak security; (4) additional client software; (5) network congestion; and (6) requires Internet service provider (ISP) cooperation. Thus, P2P solutions are often cast as hybrid solutions using traffic density allocated servers to maintain the gaming data. The solutions are often clustered as hybrid P2P/client-server architectures, with additional network activity software to monitor traffic, and measure client responsiveness among the players.

CAx applications cannot tolerate any ambiguity in the form of data inconsistency. Any entity data error will propagate disastrously through a model's feature tree. Data centralization and protection has long been a stable feature of PDM/PLM file management systems. Client-server architectures offer improved data security while mirrored servers can offer redundancy. In addition, design models can be stored and protected within the server database. Client-server architectures can be deployed behind company firewalls as a server-centered LAN or as a local cloud server, depending on the client to server network configuration.

2.2 Collaborative Interfaces

Real world CAx applications and their respective interfaces are single user, with few exceptions like GoogleDocs: one cursor, one active application window, and one mouse or mouse-like input device. The application expects a serial stream of user actions, and the application GUI's are generally configured as a single thread to react to mouse (or touch) events and to data entered into text fields.

The collaborative environment changes drastically when several users can simultaneously edit a common model, each user carrying a different set of experiences, capabilities, and cultural attitudes. Researchers echo the need for user awareness. Liu et al. (2008) propose that intelligent agent software be deployed at each client workstation to support awareness interactions among multi-users, such as capturing and transmitting audio and video streams for each user. Another agent might capture design changes, and update these changes among users, according to user privileges.

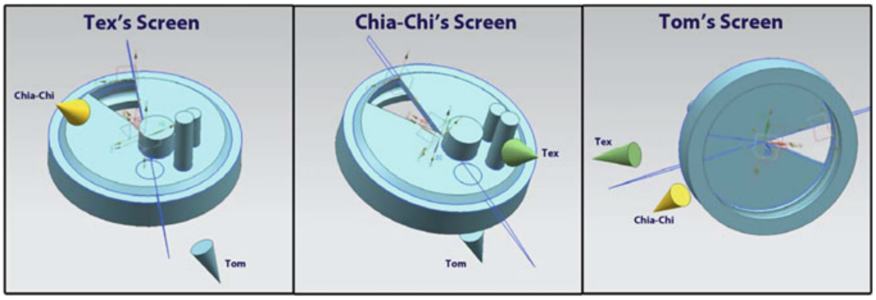


Fig. 12 Three client perspective visualization demonstration

Table 2 Flexible context filters

MUG On/Off	Multiple Displays	Divided Display	Timer	Security Tokens	Emotive	Text	Transla-tion	Video	Audio	Skype	Record	Super-visor	User Data

The BYU NXConnect prototype allows users to share their view perspective with other users who are in the same model (Nysetvold and Teng 2013). This improves the quality of their collaboration while keeping bandwidth usage low compared to existing screen share technologies. Since each NXConnect client has an up-to-date model, the only additional data that needs to be sent to facilitate view sharing is the view translation, rotation, and scale information, which can be described with as few as twelve double-precision floating point numbers. In addition, not only can users choose to follow another user's viewing perspective, they can also see cones that represent the view perspectives of other users (Fig. 12).

This capability enhances each user's awareness of the viewing perspective of other users within the same model. Knowledge of what another user is looking at may even allow a user to infer what the other user is currently working on.

Xu (2011) proposes a flexible context interface between users working on the same part at distributed locations, and defines the context as options in a multi-user GUI (MUG), Table 2. MUG uses agent software filters to render the user prototype outside of NX and NX Connect. The interface can then be used with other engineering applications where several users wish to collaborate. NX MUG, as configured for Siemens NX, enables users to view a collaborating user's workspace, send/receive messages between multi-users, and is capable of translating text interactions into different languages. Although Xu did not implement all the filters shown in Table 2, multiple icon selection sequences could be used to vary context preferences for each user in a collaborative session.

Other researchers have experimented with multiple cursors or touch control, but default to time slicing a single cursor event (see http://sourceforge.net/projects/multicursor-wm/#; http://lifehacker.com/5080196/teamplayer-enables-multiple-input-device). Focusing on the importance of GUI collaboration, Baomin Xu et al. (2009) proposed sending Windows GUI events to multi-users rather than model changes made through the CAx API, but this seems difficult in the face of so many GUI versions. Although GUI's change constantly, they are built on a fairly stable API library.

2.3 Model Sharing and Conflict Resolution

Having multiple users concurrently contribute to the same model introduces data consistency challenges that are not present in the single user scenario. Syntactic and semantic conflicts can occur, and the order in which features are created can vary between clients. Approaches for preventing data inconsistency include, spatial decomposition (Red et al. 2010, 2011, 2012, 2013b), collaborative constraints (Panchal et al. 2007; Lai 2009; Chen et al. 2004; Marshall et al. 2011; Ram et al. 1997), locking and blocking of model features (Bu et al. 2006; Moncur et al. 2012; Hepworth et al. 2013a), user negotiated feature access control (Zheng et al. 2009), and role-based and lean model editing access (Cera et al. 2003; Wang et al. 2006).

Figure 13 shows how simple planar constraints can be used to confine multi-users to a certain space, so that mouse events are only effective for the user's assigned space (Marshall 2011). Once fully implemented, these methods will work for even the most complex models. Objects that cross boundaries will require negotiation among adjoining users for editing rights. Extending these concepts to other constraint surfaces simply require that the entity boundaries be compared against the equations in Table 3, or for other possible constraint surface types.

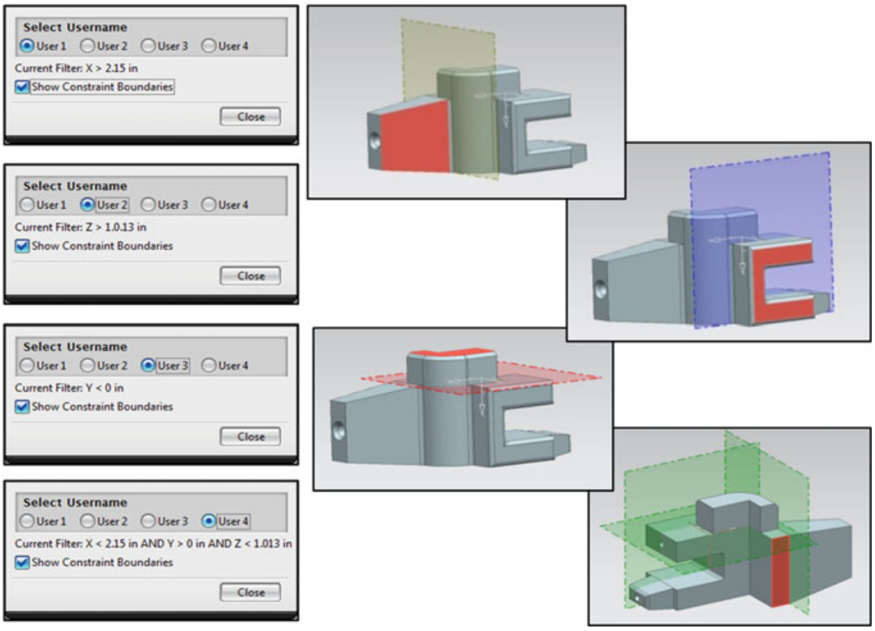
Marshall (2011) implements the constraint equations by a selection filtering tool. The selection filtering portion of the implementation is integrated within the CAD system (mouse cursor combined with feature selection ray cast normal to the viewing window) and has a single dialog window that allows for selection among different multi-users. Depending on the user, a selection filter is applied to all possible selections based on four constraint planes. This early prototype allows for selection of edges and faces, which make up a model P where any spatial point $\mathbf{p}$ is described by coordinates x, y, and z.

$$\mathbf{p} \in P \tag{1}$$

Let X, Y, and Z represent the x, y, and z ranges of points $\mathbf{p}$ in P such that

$$x \in X;\ y \in Y;\ z \in Z \tag{2}$$

For the model of Fig. 10, the constraints in (3)–(6) have been implemented using inch units. ACCEPT P means a feature on the model is selectable by the multi-user. A selectable feature can be edited by the multi-user.



Select Username
[✓ User 1] User 2 User 3 User 4
Current Filter: X > 2.15 in
[✓ Show Constraint Boundaries]
Close

Select Username
User 1 [✓ User 2] User 3 User 4
Current Filter: Z > 1.0.13 in
[✓ Show Constraint Boundaries]
Close

Select Username
User 1 User 2 [✓ User 3] User 4
Current Filter: Y < 0 in
[✓ Show Constraint Boundaries]
Close

Select Username
User 1 User 2 User 3 [✓ User 4]
Current Filter: X < 2.15 in AND Y > 0 in AND Z < 1.013 in
[✓ Show Constraint Boundaries]
Close

Fig. 13 Geometric constraint limiting of user feature selection

User 1. only select edges and faces for which

$$\text{if any } x \in X > 2.15,\ \text{ACCEPT } P \tag{3}$$

User 2. only select edges and faces for which

$$\text{if any } z \in Z > 1.013,\ \text{ACCEPT } P \tag{4}$$

User 3. can select edges and faces for which

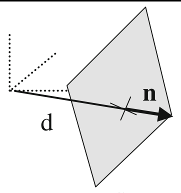
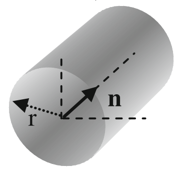
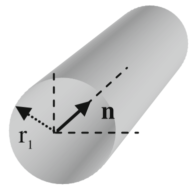
$$\text{if any } y \in Y < 0,\ \text{ACCEPT } P \tag{5}$$

User 4. can select edges and faces for which

$$\text{if any } x \in X < 2.15 \text{ and } y \in Y > 0 \text{ and } z \in Z < 1.013,\ \text{ACCEPT } P \tag{6}$$

Normally, a feature would highlight as the mouse hovers over it to show what would be selected if the user were to click the mouse. However, if a feature is not selectable, the feature will not highlight when the mouse hovers over it. The constraint boundaries can be toggled on/off for visibility and are colored differently for each user.

Table 3 Constraint boundaries for spatial decomposition

Constraint	Graphical	Constraint Surface	Constraint forms (IE = Inequality; EQ = Equality)
Plane (unbounded)		$\mathbf{n}^T\mathbf{p} = d$ $\mathbf{n}$ = outward plane normal $\mathbf{p}$ = point in plane d = plane distance	$\mathbf{u}$ = user selected point $\mathbf{n}^T\mathbf{u} < d$ (inward, IE) $\mathbf{n}^T\mathbf{u} \leq d$ (inward, EQ) $\mathbf{n}^T\mathbf{u} > d$ (outward, IE) $\mathbf{n}^T\mathbf{u} \geq d$ (outward, EQ)
Cylinder (unbounded)		$(\mathbf{p} - \mathbf{v})^T\mathbf{e} = r$ $\mathbf{n}$ = cyl axis unit vector $\mathbf{e}$ = unit vector normal to cyl axis unit directed at $\mathbf{p}$ $\mathbf{v}$ = point on cyl axis $\mathbf{p}$ = point on cyl surface r = cyl radius	$\mathbf{u}$ = user selected point $(\mathbf{u}-\mathbf{v})^T\mathbf{e} < r$ (inward, IE) $(\mathbf{u}-\mathbf{v})^T\mathbf{e} \leq r$ (inward, EQ) $(\mathbf{u}-\mathbf{v})^T\mathbf{e} > r$ (outward, IE) $(\mathbf{u}-\mathbf{v})^T\mathbf{e} \geq r$ (outward, EQ)
Conical Frustum (bounded, reduces to cone if $r_2 = 0$)		$r_c = (\mathbf{p}-\mathbf{v})^T\mathbf{e}$ $(r_2 \leq r_c \leq r_1)$ $\mathbf{n}$ = cone axis unit vector $\mathbf{e}$ = unit vector normal to cyl axis directed at $\mathbf{p}$ $\mathbf{v}$ = point on cone axis at base where $r_c = r_1$ $\mathbf{p}$ = point on conical surface r_c = cone radius at $\mathbf{p}$ h = frustum length	$\mathbf{u}$ = user selected point *Step 1:* $d = (\mathbf{u}\text{-}\mathbf{v})^T\mathbf{n}$ *Step 2: Inward* IE: if $(0 < d < h)$ and $r_c = r_1 + \mathrm{d}(r_2 - r_1)/\mathrm{h}$ $r = \lvert\mathbf{u} - \mathbf{v} - \mathrm{d}\mathbf{n}\rvert < r_c$ EQ: if $(0 \leq d \leq h)$ and $r_c = r_1 + \mathrm{d}(r_2 - r_1)/\mathrm{h}$ $r = \lvert\mathbf{u} - \mathbf{v} - \mathrm{d}\mathbf{n}\rvert \leq r_c$ *Step 2: Outward* IE: if $(d < 0)$ or $(d > h)$ or $r = \lvert\mathbf{u} - \mathbf{v} - \mathrm{d}\mathbf{n}\rvert > r_c$ given $r_c = r_1 + d(r_2 - r_1)/h$ EQ: if $(d \leq 0)$ or $(d \geq h)$ or $r = \lvert\mathbf{u} - \mathbf{v} - \mathrm{d}\mathbf{n}\rvert \geq r_c$ given $r_c = r_1 + d(r_2 - r_1)/h$

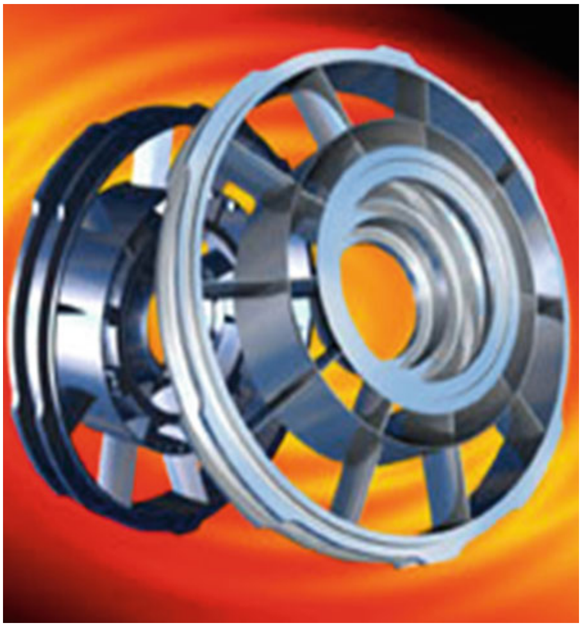


Fig. 14 Front frame easily decomposed

Spatial decomposition, a divide and conquer approach, will be useful for complex and large models, or for models that have obvious shape independencies, such as the jet engine front frame in Fig. 14. Three design regions (inner case, outer case, radial vanes) can be decomposed by cylindrical and conical constraint surfaces. But decomposition may not always involve just looking for different model shapes, since users may be assigned based on model feature expertise as discussed by Moncur (2013).

2.3.1 Abstract Decomposition

Figure 15 shows an abstract representation of a complex design space divided into multi-user regions that we will refer to as design regions, d_i ($i = 1, n$), and where design space model D is the sum of these regions. Region d_i is the set of features, attributes, operations, and/or geometry associated with the design region. If only spatial decomposition is used, then D represents the volume which bounds the model geometry. If only feature decomposition is used, D represents all features that can be edited in the CAx application. Thus, the design space can be represented by $D = \Sigma\, d_i$, assuming that the decomposition is full, i.e., the set of design regions span the entire design space. Because of regional dependencies, the design space representation can be more complex as will be shown later.

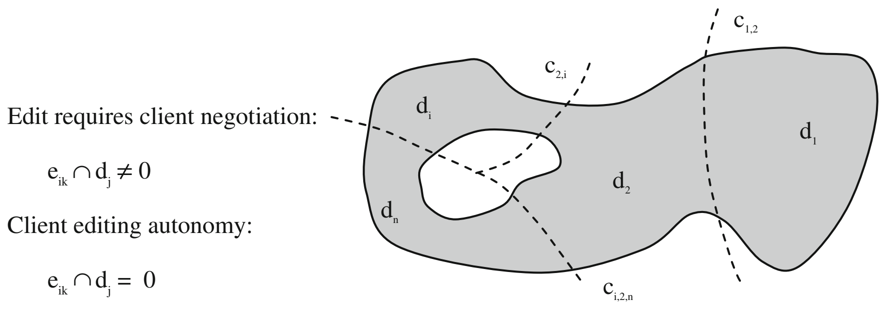
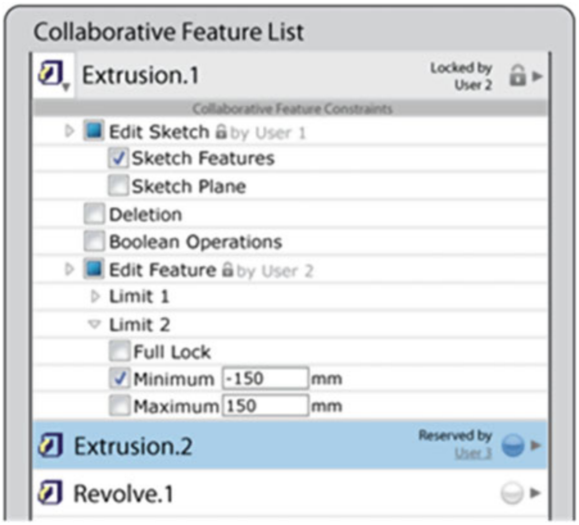


Fig. 15 Spatial decomposition

The set of elements k that define region i is represented as $e_{i,k}$. In multi-user *autonomous* decomposition, it is desired that all k elements reside within region i as depicted by $e_{i,k} \cap d_j = 0$ for all $j \neq i$. A simpler algorithm compares $e_{i,k}$ against all constraints $c_{i,j,k}$ bounding region i. If $e_{i,k}$ satisfies all constraints, then the client can edit the element without constraint. If $e_{i,k}$ does not satisfy all constraints, then element $e_{i,k}$ is locked and multi-users assigned to the intersected regions must be notified, where negotiation allows the now unlocked element to be edited.

The boundaries between two design regions are represented by user constraints $c_{j,k}$, where j, k refers to the constraint between multi-user regions j and k and where $c_{j,k} = c_{k,j}$. User constraints can be represented as geometric equations or feature sets, depending on the design space. Some design regions are defined by only one constraint equation, such as d_1 (constrained only by $c_{1,2}$) and d_n (constrained by $c_{\mathrm{i},2,n}$), while others may require comparison against several constraints, such as d_2 (constrained by $c_{1,2}$, $c_{2,\mathrm{i}}$, $c_{1,2,\mathrm{n}}$) and d_i (constrained by $c_{2,i}$, $c_{1,2,n}$).

As a simple example consider the problem of developing a manufacturing process plan to machine a part. The part model could be comprised of several feature sets that encompass operations required to manufacture a part. The feature set could be used to decompose the process plan into regions represented by operations like these: (1) roughing tool paths; (2) semi-roughing tool paths; (3) finishing tool paths for surface features; (4) pocketing and slotting features; (5) drilling, tapping, and threading features; (6) profiling features; (7) fixturing hardware and setup; and 8) tooling. Some of these operations are reasonably independent of other operations (e.g., pocketing as compared to drilling/tapping/threading) and could be used to decompose the process planning among several multi-users for simultaneous process planning. When a design region is totally independent of other regions, it will not need a constraint relationship to constrain the assigned user actions and thus $c_{i,..} = 0$. Independent regions, or mildly independent regions, are the best candidates for multi-user decomposition. Regions that are strongly dependent, i.e., connected by dependent features, will require cooperation and intense interaction between the multi-users to simultaneously edit the design space.


Collaborative Feature List

Extrusion.1 — Locked by User 2

Collaborative Feature Constraints

- Edit Sketch by User 1
  - ✓ Sketch Features
  - ☐ Sketch Plane
- ☐ Deletion
- ☐ Boolean Operations
- Edit Feature by User 2
  - Limit 1
  - Limit 2
    - ☐ Full Lock
    - ✓ Minimum -150 mm
    - ☐ Maximum 150 mm

Extrusion.2 — Reserved by User 3

Revolve.1

Fig. 16 Collaborative feature list

Some constraints may be common to more than one multi-user region, e.g., $c_{\mathrm{i},2,n}$ as shown in Fig. 15. For example, a multi-user assigned to region i will be constrained by those relationships that contain i in the constraint subscripts: $c_{2,i}$ and $c_{\mathrm{i},2,n}$. Thus, user design region i is defined by the associated feature set associated and constrained by those functions $c_{i,j,}$ with included i subscript.

Since modern CAx applications use feature representations, Moncur displays a representation of a part's feature list, Fig. 16 (Moncur et al. 2013). A user can change a 3-state toggle to reserve, lock, or release the selected feature. When changed, the new feature state is propagated to the other multi-users. When a feature is reserved, it means that the user who reserved the feature is able to edit or delete the feature. A *reservation* can be made by any user at any time, even if the feature is already reserved by another user (although this does not hold true if the feature is locked by another user). Because a reservation can only be held by a single user at a time in a collaborative design session, it prevents two users from editing the same feature at the same time. A reservation could be thought of as a per-feature edit token, or a "soft" lock for edit permissions for a given feature. This allows for open collaboration between users and allows them to work on a model while preventing update and delete conflicts.

A color scheme could be applied to the rendered features inside the CAx application to help other users see which features are available for editing, Fig. 17. Lock symbol decals could also be applied to features to convey editing state to other users.

Features can also be locked or reserved in groups. Thus, an entire branch of features can be locked by locking the parent-feature in the tree. As new features are added to a model, they can inherit any lock/reservation properties from their parent feature. This provides a way to effectively partition a CAx model by feature rather than shape.

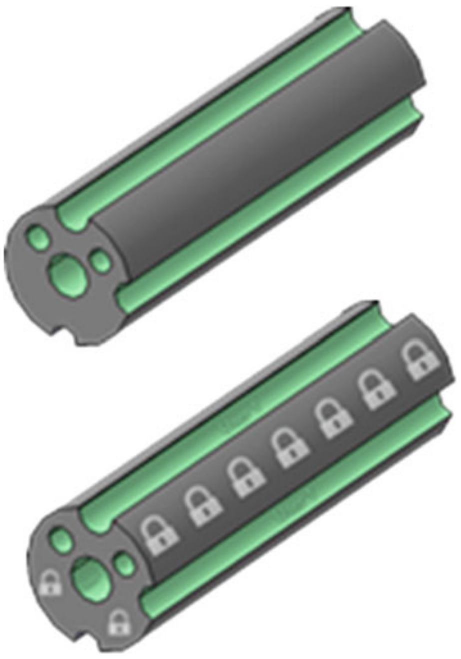


Fig. 17 Reserving/locking using colors/decals

Hepworth et al. (2013b (Automated…)) extend this approach by automatically reserving features during a feature edit operation. This prevents multiple users from simultaneously making conflicting edits to the same feature (feature-self conflicts). Feature-self conflict prevention is expressed mathematically by (7)–(10), given that F is the set of all features in part P, and in time interval Δt, and where f is a feature in part P, Δt. U is the set of all users in part P, Δt, and u is a user in part P, Δt. A part contains:

$$f \in F(P, \Delta t) \tag{7}$$

$$u \in U(P, \Delta t) \tag{8}$$

To prevent feature-self-conflicts for a given edit operation in part P, there must exist a unique set E, Δt, which contains only one feature f and one user u where

$$E(P, \Delta t) = \{f, u\} \tag{9}$$

for interval

$$t_{\text{beginEdit}} \leq t \leq t_{\text{endEdit}} \tag{10}$$

This is enforced by allowing only a single instance of set E to exist in part P, Δt, where Δt is the time interval from the beginning to the end of the edit operation.

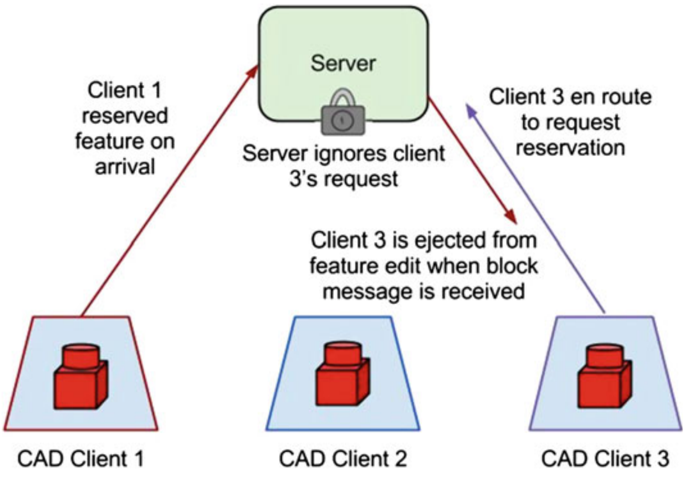

Server
Client 1 reserved feature on arrival
Server ignores client 3's request
Client 3 en route to request reservation
Client 3 is ejected from feature edit when block message is received
CAD Client 1
CAD Client 2
CAD Client 3

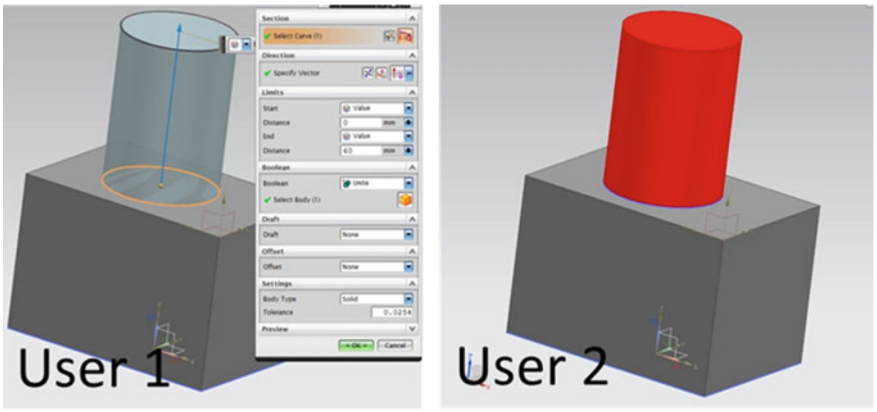
Fig. 18 Server reservation method prevents simultaneous, multi-user feature editing

Otherwise, feature-self conflicts occur if users on separate clients simultaneously edit the same feature.

Hepworth implements a method to prevent feature-self-conflicts by reserving a feature on the server on a first come first serve basis. When a user begins an edit operation, a message is automatically sent to the server requesting reservation of that feature. If the feature is not already reserved, the user is allowed to continue editing the feature. However, if the feature is already reserved, any other client request will be denied. When the user completes the edit operation, another message is automatically sent to the server removing the reservation. Figure 18 is a diagram depicting Client 1 reserving a feature while Client 3 makes a request and is ignored.

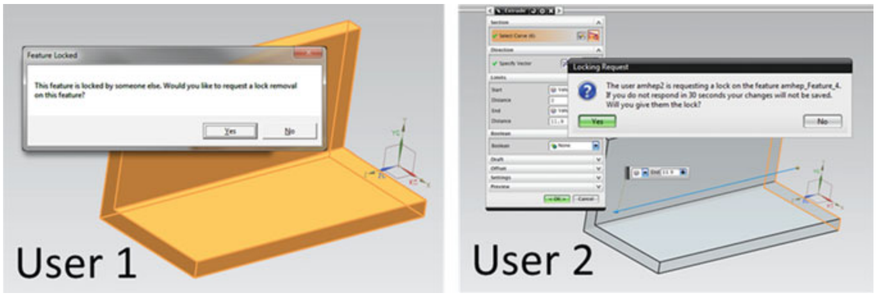
In addition to the reservation on the server, a client blocking mechanism is implemented to apprise users that a feature is reserved and prevent users from editing reserved features. When a feature is reserved by a user, all other users receive a message, blocking the feature. A block forcibly prevents users from editing a feature and changes the color of the feature to a predefined blocking color. The color helps to communicate the reservation and also show where users are currently working in the part. Figure 19 shows an example a one user editing a feature while a second user has the feature blocked.

Users may also request that reservations be removed on blocked features; see Fig. 19. This prevents users from having features blocked for an unnecessarily long period of time. Reservation request removal is done by sending a message to the reservation owner requesting reservation removal. The user has the opportunity to accept or reject the request (Fig. 20). The user has a limited time to respond before the request is automatically granted to the requester. This allows users to remove reservations on blocked features even if the owing user is unavailable to respond to the message (Hepworth et al. 2013b (Automated…)).



Start: Value
Distance: 0 mm
End: Value
Distance: 60 mm
Boolean: Unite
Select Body (1)
Draft: None
Offset: None
Body Type: Solid
Tolerance: 0.0254

Fig. 19 User 1 edits an extrusion and it is blocked on user 2 (shown in *red*)



Feature Locked

This feature is locked by someone else. Would you like to request a lock removal on this feature?

Locking Request

The user amhep2 is requesting a lock on the feature amhep_Feature_4. If you do not respond in 30 seconds your changes will not be saved. Will you give them the lock?

Boolean: None

Fig. 20 User 1 requests reservation removal and user 2 receives the request

2.3.2 Undo/Redo in Multi-user CAD

Modern CAx applications allow the user to undo/redo their last actions (often implemented as CTRL + Z = undo; CTRL + Y = redo). But in a multi-user session, undo actions cascade through the feature tree which may have been constructed by several users through dependent operations. Li et al. (2008) note 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 multi-user undo intent and use local history buffers to record the changes made by local clients. Because any user in a multi-user session can apply a feature undo function, dependencies can cascade and cause chaotic collaboration.

We have successfully applied an undo command method to our NX Connect prototype. The undo method we implemented only allows users to undo their own recent commands—not the commands of other users. Each model-changing command a user performs creates an undo action construct that contains both the data required to undo the command and the data required to redo the command.

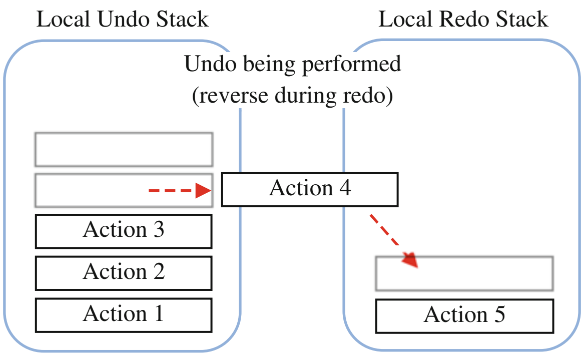


Fig. 21 Local Undo/Redo stacks

A timestamp records the time at which the command is performed. The action constructs are then placed on top of the local user's undo stack (see Fig. 21). None of the local user's action constructs or undo/redo stacks are sent to the server or to other clients. Each user has their own undo and redo stacks which only store that user's actions.

When a user performs an undo command, the action construct on the top of their undo stack is used to perform a local undo. This action construct contains the data to perform the undo operation. After undo is completed locally, the action construct is transferred from the user's undo stack to their redo stack as shown in Fig. 21. After the undo succeeds on the local client, a message is sent to the server, which forwards it to the other clients so they can perform the same modeling operation.

Figure 22a, b, and c show a multi-user engine block design session with three users. User 1 performs a Boolean operation to create the piston chambers, see upper left of Fig. 22a. Figure 22b shows the Boolean operations reflected to users 2 and 3. Figure 22c shows an undo operation performed on the piston Boolean in the local history tree. The undo operations have not yet been sent to the other two users, but this happens in a succeeding step.

3 Multi-user CAx Requirements

This section considers three main areas: (1) multi-user CAx functionality requirements, considering the deficiencies currently existing in single user CAx applications; (2) administration and management of product requirements in multi-user collaborative environments, and (3) human elements in multi-user team forming. It is important to note that the multi-user functionality already demonstrated in early prototypes like NX Connect and CATIA Connect exceeds single user functionality when it comes to reducing design times and sharing design rationale. Current CAx API's can be utilized to make CAx applications behave in a multi-user mode, but with extensive programming and API workarounds.

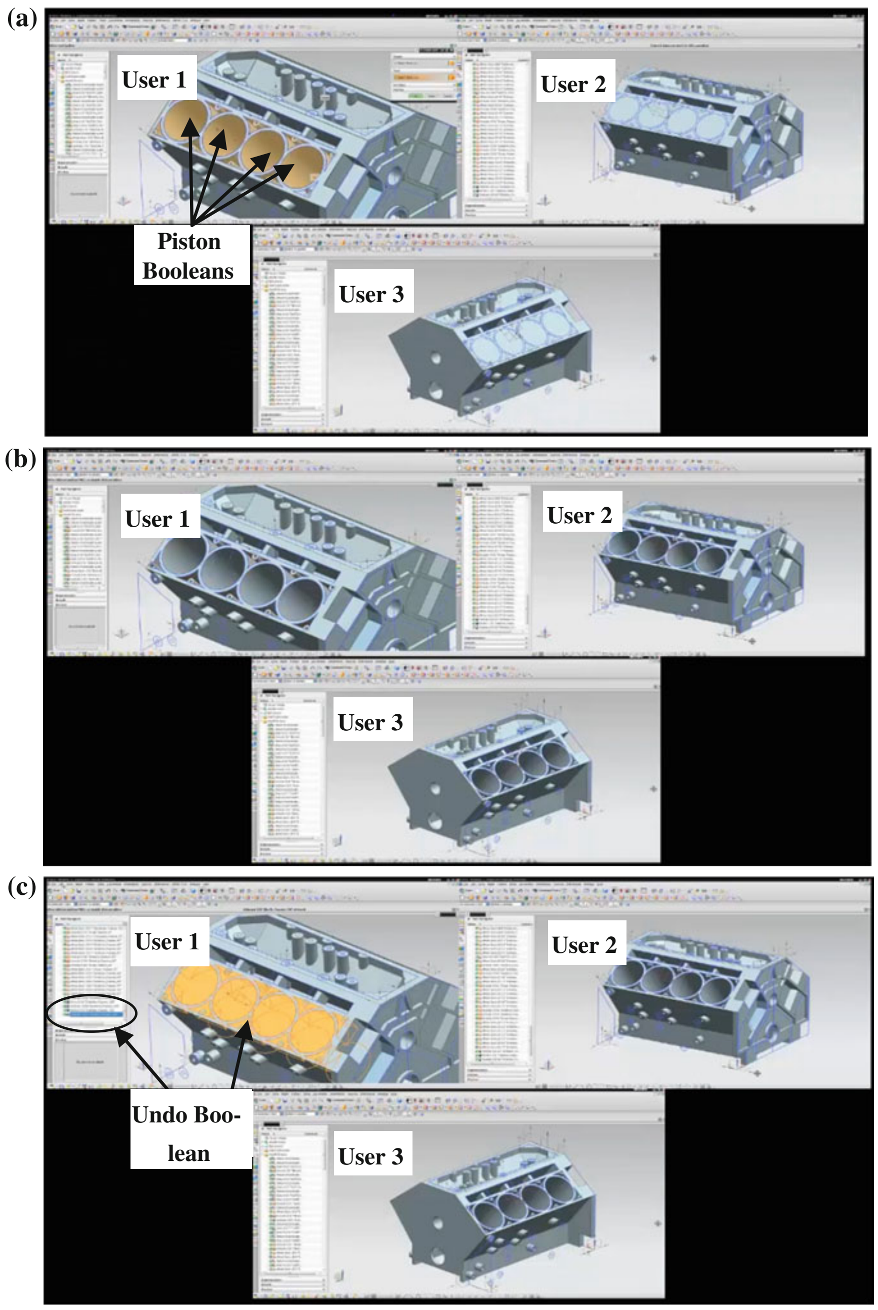


Fig. 22 Create pistons undo example. **a** Create pistons. **b** Reflect edit. **c** Undo pistons

Table 4 Desired multi-user features

Single user limitations	Multi-user implications	Importance (1 most)
Single threaded kernels*, GUI's, API's (*Parasolid, ACIS, etc.)	The advantage of threading is that multi-users can share a kernel, GUI, or API, based on computational and network resources. Threading would reduce the number of site licenses required for multi-user sessions. Presently, each client must have a separate CAx license.	2
API entity memory "handles"	Handles, i.e., memory addresses, are allocated differently in each computer. Current multi-user plug-ins require extensive programming to discover the data structure format for each entity and related operation that must be passed between clients. API's should provide object/data structure copies for data exchanged among clients in multi-user sessions.	1
API functional errors	Many of the less used API's have functional errors, and should be fixed.	2
Single user undo/redo	Single user undo/redo operations do not work in multi-user CAx, but we have demonstrated that simple methods such as tracking of local user actions can provide acceptable undo/redo functionality. Single user CAx applications will have to be converted to client-server architectures because a database must track operations performed by each user in the session.	1
Models file-based	Client-server architectures with server databases provide more accessibility to all product related data, and permit multiple users/clients to access and edit models simultaneously.	1
No access to event handler	Without access to a CAx event handler, prototypes must track temporary session files associated with undo markers, or track Windows GUI events. A simple event handler API to trigger alerts as users edit model data would replace more complex plug-in programming and replace polling timers.	1

(continued)

Table 4 (continued)

Single user limitations	Multi-user implications	Importance (1 most)
CAx algorithmic time variability	Algorithms with significant latency can hinder the multi-user process. Possible work-arounds or solutions include providing time-to-completion estimate for each operation, requesting that software providers improve algorithm speeds (e.g. by parallel processing), and multi-threading the application core/GUI/API such that the local user could continue to work while another user's command is being applied to the local user's model. Multi-user CAx applications faced with operational latencies will also require asynchronous functionality so that users can enter and leave an active session as desired. The users must also be able to control when they wish other client operations to update their model. For CAx application with severe algorithmic latencies, autonomous decomposition approaches will be most effective.	3
Entity numbering schemes	Entity numbering schemes are not uniform across CAx applications, with entity numbering often order dependent. In such cases the server must maintain each user's modifications as ordered and then reflect them to the clients in the same sequence. More uniform and predictable numbering algorithms conforming to multi-user model architectures would simplify server implementations.	2
Limited social tools	Social tools for collaborative awareness are critical to effective multi-user teaming and should be integrated into CAx applications. Surveys have revealed that product development personnel are using many such tools to collaborate because these tools are not prevalent in CAx applications.	1

3.1 Multi-user CAx Functionality Requirements

Table 4 lists several important single user CAx architectural limitations, some of which are inherent to a particular CAx application, like CAE/FEA preprocessing of meshes. Table 4 recommends some changes to improve multi-user viability. Many of these limitations are not difficult fixes and would reduce multi-user server complexity, such as entity copies rather than memory address handles, integrated social tools, fixing API errors, predictable entity numbering schemes, and exposing an API to the primary event handler. Multiple threading and algorithmic latencies will prove more challenging.

3.2 Multi-user Administrations of Product Requirements

Liu et al. (2008) recognize the reluctance to use collaborative CAD, which he says can be related to network latencies, security, and cultural issues, and because software companies have not adopted collaborative advances. We suggest that nonadoption is more closely related to potential disruption of normal business processes; stable companies are very reluctant to experiment with their internal processes.

Regardless of potential, industry will not adopt new processes unless the administrative issues are understood and easily implementable. Refer to Hannan and Freeman's (1984) popular paper to review entrenched principles of industrial inertia. We suggest that any company which has applied virtual teaming approaches can be comfortable with multi-user CAx processes. And multi-user tools simply provide an opportunity when appropriate, and do not need to be applied. Structurally, they require reliable networks, task assignments, schedules, administration, and team leadership, not really different from localized process teams. With predetermined workspaces and boundaries, collaboration decreases the product development time in proportion to the number of multi-users (Jensen 2012).

A less trumpeted advantage of multi-user CAx is the knowledge transfer and training that occurs among cross-functional virtual teams. Presently, experts train novice engineers in a time-consuming one-on-one process. Since multi-users, even technical personnel from different disciplines, can view and edit each other's work on their screen, potential problems are more easily identified early on and fixed.

Red et al. (2012) state that:

> The advent of multi-user computer-aided applications (CAx)……will change personnel/organizational assignment processes in product development…..collaborating personnel/organizations will enter design sessions and simultaneously edit/review design spaces. This paradigm shift will require new methods to be developed that decompose development tasks over personnel/organizations at both local and global locations. Experiential data will not be restricted to suppliers, organizations, or sites, or other grouping types, but reflect a different granularity where a particular group of individuals from a variety of organizations might be collected into design teams for optimal collaboration.

Currently, there are no process-specific or model-specific multi-user decomposition principles generally accepted or in standard form, although a few have been suggested in this chapter, such as decomposing a model based on geometric or feature independencies. But spatial or feature independence recognition is of no help if you lack multi-user expertise for the observable independencies.

Consider the following steps employed in current product development practices:

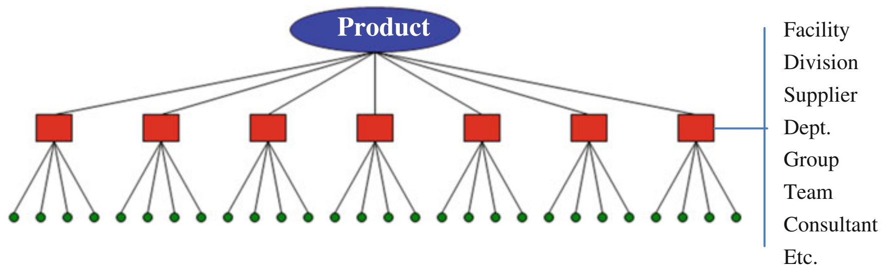


Fig. 23 Product decomposition

Step 1 *Product idea*. A product idea is conceived; requirements are specified, along with related specifications

Step 2 *System decomposition*. Product development is decomposed into elemental forms, like systems, subsystems, components, parts, etc., with more detailed requirements and specifications.

Step 3 *Organizational decomposition*. These elements are then assigned and developed according to a set of decomposed tasks (and related schedules), which recognize organizational capabilities (company, division, contractor, supplier, group, team, etc.), resource availability (raw materials, OTS/off-the-shelf components, tools, workstations, facility, personnel, etc.), schedule, logistics (moving things around), and sometime local politics (unions, laws, etc.) which may affect all the previous. Much of the decision making involves experienced personnel, similar or competitive products, and past development histories.

Step 4 *Resource decomposition*. Assignment of tasks is usually based on experiential information and resource availability, and subject to other constraints, such as government regulations, or competitive response from competing companies.

Step 5 *Completion*. Decomposed product and related development assignments are then completed according to assignments, resources, and schedule. Because current CAx applications are single user, the process is serial with corrections made as development errors are discovered in the stage gate process, which requires time-consuming iterations to the CAx models.

Figure 23 shows the typical decomposition hierarchy used in product development industries. Different organizational principles may be applied to manage the product team, such as matrix, pyramid, group, or even virtual teaming where team members are not necessarily co-located.

Now, consider the emerging paradigm of multi-user CAx. From the viewpoint of multi-user decomposition the same five steps are amended to consider possible multi-user administrative principles in Table 5. An associated asterisk (*) denotes that the tool/principle needed is poorly developed or not formalized.

Table 5 Postulated multi-user administrative principles

Step	Name	Multi-user implementation
1	Product idea	Collaborative principles* are used to review and tag the requirements and specifications for potential multi-user teaming opportunities, according to personnel and supplier availability and capability.
2	System decomposition	Conventional practice decomposes a product based on performance histories of organizations making similar products, by group or supplier capabilities, or by location, logistics, government regulations, etc. Using personnel and supplier multi-user experiential data*, and multi-user possibilities from Step 1, decomposed elements can be directed towards multiple organizational entities, for potential virtual teaming.
3	Organizational decomposition	Multi-user decomposition requires comparison of available user backgrounds against model specifications and features to determine whether personnel are available and suitable for multi-user mode. It would seem that new databases* would maintain relevant experiential data about personnel based on their educational background, technical and model space experiences, and expertise both in technical matters and in human-interaction and management. It is likely that department or group level resources, i.e., experiential databases*, could be searched to establish expertise and experience, and existing schedules can also be used to determine personnel availability.
4	Resource decomposition	Multi-user decomposition tools* would examine the model design space to determine model complexity and whether there several regions that are independent or mildly independent, based on proposed model features, or similar products previously developed. If personnel resources are available, and model is sufficiently complex, multi-user decomposition of the design model among several users may be desirable. In the case where multi-user training* of novice users by more experienced personnel is desired, model complexity may be relaxed.
5	Completion	Multi-user sessions are scheduled and managed by a team leader*, using integrated and networked multi-user CAx GUI's* and client-server architectures to store user actions and design rationale histories in a central database*.

Figure 24 shows how multi-user virtual teams can be organized from a community of personnel, suppliers, or other organizational entities, given access to experiential histories and capabilities. Also note that one individual could be assigned to sit on several multi-user teams, as could personnel from different functional areas, like designers, analysts, testers, marketers, production personnel, etc.

Not all decomposition tasks are compatible with multi-user development. Tang et al. (2000) notes three types of relationships between design tasks: uncoupled, coupled, and decoupled. Uncoupled tasks have low interdependency and can be performed in parallel. These qualify for multi-user collaboration. Coupled tasks are highly dependent and utilize iterative cycles to solve conflicts.

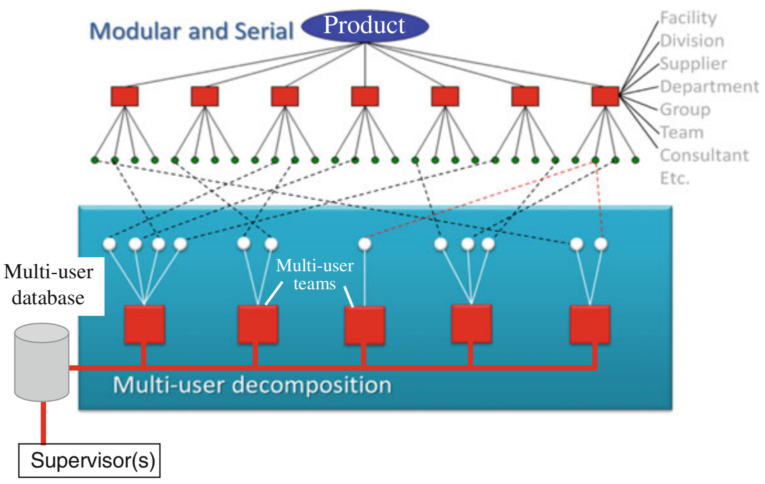


Fig. 24 Product decomposition: *top* is conventional; *bottom* shows organized multi-user teams (*red squares*)

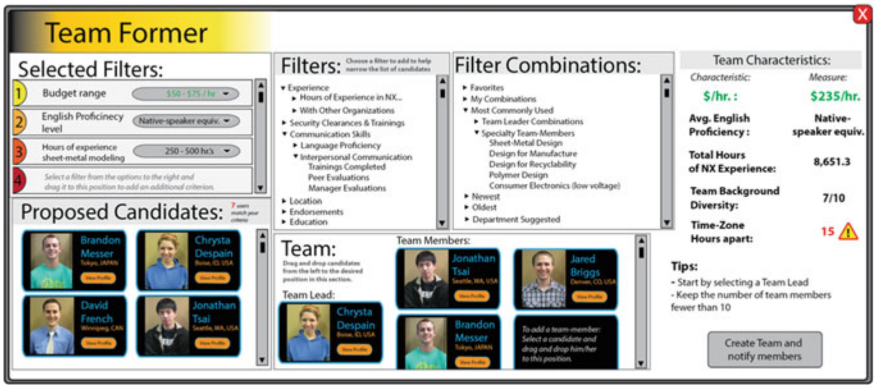
Tasks that are decoupled can be performed sequentially and contain only one way dependencies. Holyoak (2012) extended the methods of Tang to apply a Design Structure Matrix (DSM) to correlate design tasks specifications and determine task dependencies. From these multi-user assignments can be appropriately made.

3.3 *Human Elements in Multi-user Team Forming*

Step 3 of Table 5 presents an opportunity to more optimally allocate human resources within product development communities. By adapting forming techniques based on personnel experiences and capabilities, product organizations can organize employees into multi-user teams that take full advantage of their experiential backgrounds.

"Customer Relationship Management" and "Data Mining" (Rygielski et al. 2002) are techniques that identify and extract information about individual customers. Using this data, an organization can apply algorithms to predict future behavior and improve customer relationships. Customer data is parsed by automated algorithms to predict customer behavior, needs, and desires. Vendors like Amazon.com participate in a Customer Relationship Management interaction.

When an Amazon webpage suggests another purchase, it does so automatically based on data it has gathered about that specific customer's preferences using "collaborative filtering" algorithms and Search-Based Methods. Additional information such as customer age, gender, geographic location, and more (gathered through surveys, coupon offers, etc.) allow the vendor to compare an individual


Team Former
Selected Filters:
Proposed Candidates:
7 users match your criteria
Brandon Messer Tokyo, JAPAN
Chrysta Despain Boise, ID, USA
David French Winnipeg, CAN
Jonathan Tsai Seattle, WA, USA
View Profile
Filters: Choose a filter to add to help narrow the list of candidates
Experience
Hours of Experience in NX...
With Other Organizations
Security Clearances & Trainings
Communication Skills
Language Proficiency
Interpersonal Communication
Trainings Completed
Peer Evaluations
Manager Evaluations
Location
Endorsements
Education
Filter Combinations:
Favorites
My Combinations
Most Commonly Used
Team Leader Combinations
Specialty Team-Members
Sheet-Metal Design
Design for Manufacture
Design for Recyclability
Polymer Design
Consumer Electronics (low voltage)
Newest
Oldest
Department Suggested
Team:
Drag and drop candidates from the left to the desired position in this section.
Team Lead:
Team Members:
Jared Briggs Denver, CO, USA
To add a team-member: Select a candidate and drag and drop him/her to this position.
Tips:
- Start by selecting a Team Lead
- Keep the number of team members fewer than 10
Create Team and notify members

1. Budget range: $50 - $75 / hr
2. English Proficiency level: Native-speaker equiv.
3. Hours of experience sheet-metal modeling: 250 - 500 hr's
4. Select a filter from the options to the right and drag it to this position to add an additional criterion.

Team Characteristics:

Characteristic:	Measure:
$/hr. :	$235/hr.
Avg. English Proficiency :	Native-speaker equiv.
Total Hours of NX Experience:	8,651.3
Team Background Diversity:	7/10
Time-Zone Hours apart:	15 ⚠

Fig. 25 Artistic rendition of a multi-user experiential search interface

customer to other similar customers. Cluster algorithms (Linden et al. 2003) can then be used to display advertisements and product suggestions that will attract a specific customer.

Now consider the gathering of user experiential data for multi-user CAx environments. Multi-user architectures require methods to organize personnel into local (or co-located) or virtual project teams. To do this optimally, project leaders, or supervisors, need dynamic data representing an individual's capability and experience in applying CAx applications, as appropriately mapped to project requirements and specifications. We suggest that both a user's CAx experiences and their social/cultural markers could be mapped into experiential databases; Fig. 25 is an interface rendition. This data could then be searched by applying search algorithms and automated suggestions similar to the marketing algorithms discussed earlier.

It would not be necessary for a manager to personally know or have interacted with an employee to accurately predict personnel skills, or motivation to work on a given project. Managers can quickly filter employees from multiple organizations for a project group, allowing organizations to more fully utilize the abilities and talents of their employees. Managers will access a much larger pool of candidates. Employees will work on projects for which they are most qualified. Other potential benefits include a decrease in need to relocate employees or travel long distances to work as part of a project team.

4 Multi-user CAx Standards

Multi-user architectures, by providing uniform methods for interacting with distributed suppliers (or even within company divisions using different CAx vendor applications), will spawn standards. These standards will seek to make CAx data interoperable, regardless of vendor application.

Brunnermeier and Martin (1999) note interoperability deficiencies for the automotive industry in the executive summary of a 1999 report prepared for National Institute of Standards and Technology (NIST): “This study estimates that imperfect interoperability imposes at least $1 billion per year on the members of the U.S. automotive supply chain. By far, the greatest costs relate to the resources devoted to repairing or reentering data files that are not usable for downstream applications. This estimate is conservative because we could not quantify all sources of interoperability costs.”

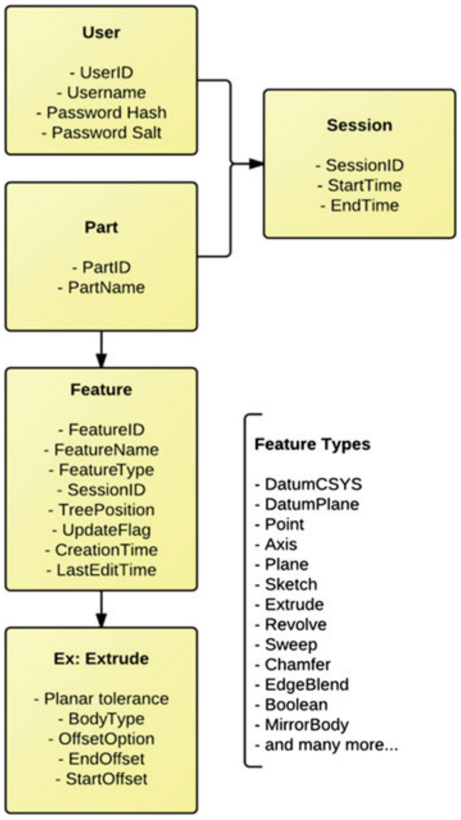
Another interesting statement can be found in the Committee (2008) summary of the report *Pre-Milestone A and Early-Phase Systems Engineering: A Retrospective Review and Benefits for Future Air Force Acquisition*: “There has been about a *threefold increase in delivery time* for most major systems…puzzling given enormous productivity advantages conferred by…Internet and e-mail…advances in knowledge-management and collaboration tools such as…(CAD)…(CAM)…”

It is not difficult to see that “imperfect interoperability” is extending schedules and costs for complex products. Simply consider the increasing supplier chains for large product companies. For example, the Boeing 787 Dreamliner schedule slipped several years, attributed 67 % to outsourcing to suppliers. Boeing uses approximately 6,450 suppliers; Goodrich (now UTAS) and PPG Aerospace are two Boeing Tier 1 suppliers each with supply chains listed in the 1,000’s. Ultimately the *leaves* of the Boeing supply chain touch small manufacturing firms with minimal CAx tools. Model file conversions in the millions and schedule delays are required to get suppliers access to Boeing (or UTAS or PPG) data. File-based *uniformity* translations like IGES, STEP, JT, etc., are architecturally not the solution, because vendor model representations and their data files are not standardized (Choi et al. 2002; Li et al. 2012).

Researchers have targeted solving this problem as early as the mid-1990s where Hoffman et al. (1993) proposed Erep, an editable representation which serves as a global schema for data exchange. Anderson (1998) proposed the Enabling Next GENeration (ENGEN) project to develop a data model to exchange design intent information. He later proposed Solid Model Construction History to exchange history-based information (Anderson 2002). Several papers have discussed the macro-parametrics approach in which a macro records the modeling commands of the parametric feature history. Data is exchanged through a standard set of commands used as a neutral format, which acts as an intermediary between the various commercial CAD macro files (Choi 2002; Mun 2003; Yang 2004; Kim 2007; Han 2010).

Rappoport (2003) recognized the incompatibilities between CAD systems so he proposed Universal Product Representation (UPR) architecture as a potential solution. This is a union of all common data and operation types between CAD systems. Data types not compatible are rewritten as another data item. In this manner, compatible data is transferred between CAD systems parametrically. Li et al. (2009) use a two stage process to recover the entire modeling process of a user and use this to simulate a real human in the target system. Although the

Fig. 26 Entity record



parametric data exchange problem has yet to be completely solved, much progress has been made in this area.

Translation aside, multi-user architectures (e.g., client-server) naturally provide for standardization, because model data can be stored in a neutral, easily accessible format on server databases. Replacing proprietary model files with database representations provides an architectural framework that inherently supports data interoperability. If the database representations can be normalized and standardized, something like that in Fig. 26, then any compatible CAx application can edit the model data regardless of location, assuming that security provisions only allow authorized clients to view and edit the data. Neutral CAx databases will permit organizations up and down the supply chain to collaboratively interact in the same design session to avoid errors caused by passing information "over the wall". In the proposed system data translation is automatic, the supply chain process becomes more collaborative, and data integrity is maintained.

4.1 Data Security

Model sharing security has not been a primary research area (Red et al. 2013a, CIS), but it is a major concern to industries engaged in global product development. Because larger product development industries use many distributed suppliers, their intellectual property (IP) is always at risk, particularly when the model data must be transferred and translated between CAx applications. But even client-server and P2P architectures used in the reviewed research prototypes expose IP to multiple collaborators.

Wang et al. (2006) propose to restrict model sharing by providing 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.

Mensah and Teng (2013) performed an initial investigation into security principles and solutions for multi-user CAx applications. They proposed solutions for solving three of the primary security challenges, namely (1) authentication, (2) authorization, and (3) confidentiality. Authentication involves validating the client computer's identity with the server using the Transport Layer Security (TLS) protocol, and then validating the user's login credentials. An authorization mechanism then ensures that the user is only permitted to see or modify the appropriate CAx data based on the user's security clearance and other permissions. Confidentiality is necessary to ensure that messages being passed between network nodes (e.g., clients, servers) cannot be intercepted/deciphered or modified by unauthorized entities. Mensah and Teng propose that the TLS security protocol would provide confidentiality by encrypting messages that can only be deciphered by clients who have successfully authenticated. They asserted that further research is required to address the security challenges of nonrepudiation, auditing, and message integrity checks.

An authorization mechanism has been implemented into the NX Connect prototype. It consists of a dialog that allows users to manage permissions for the active part, with client-side logic that enforces those permissions, and additional columns in the database for storing each part's permission settings. Upon creating a part, the user who created it is defaulted to be the part manager. In addition to manager level permission, there are also contributor, viewer, and no-access permission levels. Registered users can be assigned to any of these levels for a given part. Users with no access to a part cannot even see that the part exists. Users with viewing access can only view the part; they cannot contribute to, modify, or delete the part. Users with contributor-level access can modify and contribute to the part, but only managers (who have full access) can delete the part, modify the part's permission settings, or perform other part-management types of operations. View-only restrictions are enforced by rejecting all part modification commands from the restricted user via the NX API.

As new multi-user collaborative methods evolve, security architectures will also evolve (Kushwaha and Roy 2010). Cloud serving architectures offer an alternative to placing CAx applications and model data on local workstations. It seems inevitable that CAx applications will be deployed on company secured cloud servers, and thus offer some security advantages, if client access can be managed and intrusions blocked. When models or editing deltas must be moved between mirrored servers to promote global teaming efficiencies, security becomes more challenging (Reddy and Reddy 2011).

5 Multi-user CAx Cloud Service

By implementing a cloud-based CAx agnostic database, CAx cloud serving becomes perfectly compatible with the multi-user CAx architectures presented in this chapter. Cloud computing is growing in popularity for a number of reasons: (1) less need for local deployment of expensive computing technologies; (2) organizational IT infrastructures can be reduced; (3) smaller businesses (e.g., suppliers) can access sophisticated CAx applications that were previously too expensive to maintain; and 4) cloud servers can be scaled according to user demand (Lynn 2012).

Of course, there are hurdles to overcome, some unique to a product development environment where IP protection is critical and where downtime is not tolerated. One of the advantages of widespread and local deployment of CAx applications is redundancy. Depending on the PDM implementation and permission system, design models may be accessible locally even if PDM systems or networks go down, so that work is not stopped. If that is not the case, a cloud server or network outage could shut down an entire development division or suppliers dependent on a particular set of CAx applications. Another solution to this is to replicate (mirror) cloud servers so that if one server goes down, clients can simply access the desired data from another server.

TeamPlatform and AutoDesk 360 are two commercial CAx applications that have placed CAx on the cloud (Dobrzynski 2013). Both applications offer the collaborative access and on-line advantages expected for cloud serving, but remain file-based solutions. Thus, these early applications seem to be more compatible with simpler models, and thus smaller files. In the near future, cloud serving of multi-user CAx applications will provide much greater efficiency and reduce product development costs.

Acknowledgments The National Science Foundation, the Center for e-Design, and BYU's industrial members and research students are acknowledged for their funding and conducting of this research and Center Site. Most importantly, our member companies, by providing access to facilities and personnel, have helped us assess their current collaborative environments and the potential for multi-user CAx improvements.

References

Anderson B (1998) ENGEN data model: a neutral model to capture design intent, PROLAMAT98

Anderson B (2002) Implementor's guide—solid model construction history

Bharambe A, Pang J, Seshan S (2006) Colyseus: a distributed architecture for online multiplayer games. NDSI '06: 3rd Symposium on Networked Systems Design & Implementation 155–168

Brunnermeier SB, Martin SA (1999) Interoperability cost analysis of the U.S. automotive supply chain final report, Research triangle Institute Center for Economics Research, RTI project number 7007–03

Bu J, Jiang B, Chen C (2006) Maintaining semantic consistency in real-time collaborative graphics editing systems. Int J Comput Sci Netw Secur 6(4):57–61

Cera CD, Braude I, Comer I, Kim T, Han JH, William CR (2003) Hierarchical role-based viewing for secure collaborative CAD. In: Proceedings of DETC'03 2003 ASME design engineering technical conferences

Chen L, Zhijie SZ, Feng L (2004) Internet-enabled real-time collaborative assembly modeling via an e-Assembly system: status and promise. Comput Aided Des 36:835–847

Choi G, Mun D, Han S (2002) Exchange of CAD part models based on the macro parametric approach. Int J CAD/CAM 2:13–21

Committee (2008) Pre-milestone A and early-phase systems engineering: a retrospective review and benefits for future Air Force acquisition, National research council of the National Academy of Science

Contero M, Company P, Vila C, Aleixos N (2002) Product data quality and collaborative engineering. IEEE Comput Graph Appl 22:32–42

Dobrzynski D (2013) Comparison of cloud-based CAD collaboration services: TeamPlatform versus Autodesk 360, CADdigest. http://www.caddigest.com/exclusive/MCAD/teamplatform/050213_teamplatform_vs_autodesk_360.htm

Douglas S, Tanin E, Harwood A, Karunasekera S (2005) Enabling massively multi-player online gaming applications on a P2P architecture. In: Proceedings of the international conference on information and automation, Colombo, Sri Lanka, pp 7–12

Endo K, Kawahara M, Takahashi Y (2007) A distributed architecture for massively multiplayer online services with peer-to-peer support. Int Fed Inf Proces 229:147–158

Fan LQ, Kumar AS A, Jagdish BN, Bok SH (2008) Development of a distributed collaborative design framework within peer-to-peer environment. Comput Aided Des 40:891–904

Gao L, Lu T, Gu N (2009) Supporting semantic maintenance of complex undo operations in replicated Co-AutoCAD environments. In: Proceedings of the 2009 13th international conference on computer supported cooperative work in design, Santiago, Chile

GauthierDickey C, Zappala D, Lo V (2004) A fully distributed architecture for massively multiplayer online games, ACM Netgames workshop draft

Han S (2010) Macro-parametric—an approach for the history-based parametrics. In: Han S (ed) Int J Prod Lifecycle Manage 4(4):321–325

Hannan MT, Freeman J (1984) Structural inertia and organizational change. Am Soc Rev 49:149–164

Hepworth A, Tew K, Nysetvold T, Bennett M, Jensen G (2013a) Automated conflict avoidance in multi-user CAD. Comput-Aided Des Appl. accepted for publication, CAD & A and presentation at CAD14, 25 June 2014, Hong Kong

Hepworth A, Nysetvold T, Bennett J, Phelps G, Jensen G (2013b) Scalable integration of commercial file types in multi-user CAD. Comput-Aided Des Appl. accepted for publication, CAD & A and presentation at CAD14, 25 June 2014, Hong Kong

Hoffmann C, Juan R (1993) Erep, an editable, high-level representation for geometric design and analysis. Geometric and product modeling pp 129–164

Holyoak VL (2012) Effective collaboration through multi-user CAx by implementing new methods of product specification and management, Master's thesis, Brigham Young University, Dec

Jensen G (2012) Collaborative multi-user synchronous and asynchronous modeling, analysis and design, Keynote presentation, Defense manufacturing conference (DMC), 26–29 Nov, 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. Comput Ind 60(7):467–475

Kim B, Han S (2007) Integration of history-based parametric translators using the automation APIs'. Int J Prod Lifecycle Manage 2(1):18–29

Kim S, Kuester F, Kim HK (2002) A global timestamp-based scalable framework for multi-player online games. In: Proceedings of the IEEE fourth international symposium multimedia software engineering, pp 2–10

Kushwaha J, Roy BN (2010) Secure image data by double encryption. Int J Comput-Aided Appl 5(10):28–32

Lai YL (2009) A constraint-based system for product design and manufacturing. Robot Comput-Integr Manuf 25(1):246–258

Li WD, Ong SK, Fuh JYH, Wong YS, Lu YQ, Nee AYC (2004) Feature-based design in a distributed and collaborative environment. Comput Aided Des 36:775–797

Li WD, Lu WF, Fuh JYH, Wong YS (2005) Collaborative computer-aided design - research development status. Comput Aided Des 37:931–940

Li M, Gao S, Fuh JYH, Yang YF (2008) Replicated concurrency control for collaborative feature modeling: a fine granular approach. Comput Ind 59:873–881

Li X, He F, Cai X, Chen Y, Liu H (2009), Using procedure recovery approach to exchange feature-based data among heterogeneous CAD systems, CSCW in design

Li J, Kim B, Han S (2012) Parametric exchange of round shapes between a mechanical CAD system and a ship CAD system. Comput Aided Des 44(2):154–161

Linden G, Smith B, York J (2003) Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Computer Society, Jan–Feb, pp 76–80

Liu Q, Cui X, Hu X (2008) An agent-based intelligent CAD platform for collaborative design. ICIC CCIS 15:501–508

Lynn S (2012) 20 Top cloud services for small businesses. PCMAG.com. http://www.pcmag.com/article2/0,2817,2361500,00.asp

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

Mensah F, Teng C (2013) Security mechanisms for multi-user collaborative CAx. In: Proceedings of the 2nd annual conference research information technology (RIIT '13), 59-60

Moncur RA (2012) Data consistency and conflict avoidance in a multi-user CAx environment, Master's thesis, Department of Mechanical Engineering, Brigham Young University

Moncur R, Jensen C, Teng C, Red E (2013) Data Consistency and Conflict Avoidance in a Multi-User CAx Environment. Comput Aided Des Appl 10:1–19

Mun D, Han S, Kim J, Oh Y (2003) A set of standard modeling commands for the history-based parametric approach. Comput Aided Des 35:1171–1179

Nysetvold T, Teng C (2013) Collaboration tools for multi-user CAD. In: IEEE 17th international conference on supported cooperative work in design (CSCWD), 2013, pp 1–5

Panchal J, Fernandez M, Paredis C, Allen J, Mistree F (2007) An interval-based constraint satisfaction (IBCS) method for decentralized, collaborative multifunctional design. Concurrent Eng 15(3):309–323

Ram DJ, Vivekananda N, Rao CS, Mohan NK (1997) Constraint meta-object: a new object model for distributed collaborative designing. IEEE Trans Syst Man Cybern Part A Syst Hum 27(2):208–221

Ramakrishna V, Robinson M, Eustice K, Reiher P (2006) An active self-optimizing multiplayer gaming architecture". Cluster Comput 9(2):201–215

Rappoport A (2003) An architecture for universal CAD data exchange. In: Proceeding of ACM symposium on solid modeling and applications 2003. ACM Press, pp 266–269

Red E, Holyoak V, Jensen G, Marshall F, Ryskamp J, Xu Y (2010) A research agenda for collaborative computer-aided applications. Comput-Aided Des Appl 7(3):387–404

Red E, Jensen G, French F, Weerakoon P (2011) Multi-User architectures for computer-aided engineering collaboration. In: 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, 10(5))

Red E, French D, Jensen G, Walker S, Madsen P (2013a) Emerging design methods and tools in collaborative product development (to be published in J. Computing and Information Science in Engineering, 13(3))

Red E, Jensen G, Weerakoon P, French D, Benzley S, Merkley K (2013b) Architectural limitations in multi-user computer-aided engineering applications (to be published in the Journal of Computer and Information Science, 6(4))

Reddy VK, Reddy LSS (2011) Security architecture of cloud computing. Int J Eng Sci Technol 3(9):7149–7155

Rygielski C, Wang JC, Yen DC (2002) Data mining techniques for customer relationship management. Technol Soc 24(4):483–502

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 Trans Comput-Hum Interact 13(4):531–582

Tang D, Zheng L, Li Z, Li D, Zhang S (2000) Re-engineering the design process for concurrent engineering. Comput Ind Eng 38:479–491

Wang Y, Tan E, Li W, Xu Z (2004) An architecture of game grid based on resource router.Grid and cooperative computing. Springer, Berlin 3032

Wang Y, Ajoku PN, Brustoloni JC, Nnaji BO (2006) Intellectual property protection in collaborative design through lean information modeling and sharing. J Comput Inf Sci Eng 6(2):149–159

Wu D, Thames JL, Rosen DW, Schaefer D (2012) Towards a cloud-based design and manufacturing paradigm: looking backward, looking forward. In: Proceedings of the ASME 2012 International design engineering technical conference and computers and information in engineering conference (IDETC/CIE12), paper number: DETC2012-70780, Chicago, US

Xu B, Gao Q, Li C (2009) Reusing single-user applications to create collaborative multi-member applications. Elsevier, Adv Eng are 40:618–622

Xu Y, Red E, Jensen G (2011) A flexible context architecture for a multi-user GUI. Comput-Aided Des Appl 8(4):479–497

Yang J, Han S, Kim B, Cho J, Lee H (2004) An XML-based macro data representation for a parametric CAD model exchange. Comput-Aided Des Appl 1(1):153–162

Zheng Y, Shen H, Sun (2009) Leveraging single-user AutoCAD for collaboration by transparent adaptation. In: 13th international conference on computer supported cooperative work in design, Santiago, Chile, 22–24 April. ISBN:978-1-4244-3534-0