



v- CAx: A Research Agenda for Collaborative Computer- Aided Applications

Edward Red¹, Vonn Holyoak², C. Greg Jensen³, Felicia Marshall⁴, Jordan Ryskamp⁵, Yue Xu⁶

¹Brigham Young University, ered@byu.edu

²Brigham Young University, vlholyoak@gmail.com

³Brigham Young University, cjensen@byu.edu

⁴Brigham Young University, feliciamarshall@byu.edu

⁵Brigham Young University, jryskamp@gmail.com

⁶Brigham Young University, yue.xu@pace08.com

ABSTRACT

A dichotomy exists in the engineering design process between 1) product teams organized to engineer products collaboratively, and 2) the single-user architectures inherent in computers and computer-aided design applications (CAx). Selective design authority is typically assigned to individuals within a product team; they become contributors in a serialized design process largely driven by a set of core CAx applications. Unfortunately, single-user serial architectures inhibit concurrent engineering, in spite of the numerous research efforts into product team cooperation, functional constraints, and data/model propagation and transparency. By surveying modern collaborative technologies, including modern internet gaming, and by also demonstrating our own collaborative CAx prototypes, we draw some interesting conclusions as to collaborative limitations. From these conclusions we propose research that will hopefully formalize and focus the multi-user collaborative agenda, including computer and networking architectures, user interfaces, and CAx applications.

Keywords: Collaborative design, multi-user CAx, multi-user computer architectures.

DOI: 10.3722/cadaps.2010.xxx-yyy

1 INTRODUCTION

The late 1970's saw the practical evolution of the personal computer [40], which usage accelerated in the 1980's. This decade also saw the practical emergence of CAD applications as a mostly software 2-D replacement for drafting [41]. It was not till the late 1970's and early 1980's that practical 3-D packages (CADAM, Calma, CADDs, etc.) were used by designers in the aerospace and automotive industries. The CAD transition from mainframe to UNIX workstations provided individual designers with new found flexibility and design independence. Prior to these new CAD applications, engineering designers

and manufacturing personnel gathered freely around large drafting tables to discuss and successfully resolve various design issues.

In the 1990's CAX applications began the transition to personal computers; costs became more reasonable, and modern CAX application software became more accessible to a new community of companies and users, including educational institutions. The use of personal tools such as word processing, graphic design, database management, and network communications exploded during this period.

The 1990's also saw a sharp increase in the global demand for new consumer products as international competition reduced product development timetables, and encouraged a new culture of operational and collaborative diversity. Yet, the basic architectures of both personal computers and computer-aided applications remained fixed - providing the individual with advanced computational tools, while limiting collaboration.

Today, product development practices assign the steps of conceptualization, specification, design, analysis, process planning, and manufacturing to specific departments and ultimately to responsible individuals, all within a highly controlled serial process, with the relevant CAX models securely checked out and managed by PLM or other file management systems. Global CAX collaboration among product team members is primarily observational, with appropriate feedback modification, although it is now normal for CAX models to be "time period checked out" to several team members in a 24 hour "chase-the-sun" work cycle.

One cannot help but wonder about the widespread exposure of intellectual property (IP) that follows such practices. Are companies simply dispersing their technical knowledge among the nations of the world to maintain their competitive standing? Will this short term relevance strategy backfire? It seems obvious that IP exposure is a reaction to global competition and the pressures of shortened development timetables. But we suggest that IP exposure is also the result of not being able to leverage a company's existing talent because of the inherent single-user architectural deficiencies found in modern computer architectures and CAX applications.

This paper will first consider the abundant literature, collaborative prototypes, and existing computer and networking architectures that define and limit multi-user collaboration. Next, we discuss NX Connect, our collaborative client-server (CS) Siemens NX prototype that allows multiple users to simultaneously modify a Siemens NX CAD model. Although our architectural elements are similar to other CS prototypes discussed in the literature, we have chosen to demonstrate collaborative CAD on a major CAX commercial system, and to integrate VOIP communication among the users, along with the automatic capture and distribution of design rationale/intent.

After drawing some general conclusions about the current CAX collaborative state, we propose a collaborative agenda that will hopefully assist researchers, computer companies that develop operating systems, and CAX software companies to converge computer and application architectures towards multi-user environments. We call this environment v-CAX with a focus on computer-aided engineering applications, and with the intent of collapsing engineering design process times.

2 REVIEW: MULTI-USER TECHNOLOGIES

We classify our review into these categories: 1) collaborative architectures; 2) collaborative infrastructure; 3) collaborative interfaces and multi-user tools; 4) model sharing and collaborative security; 5) collaborative constraints and conflict resolution; and 6) gaming.

2.1 Collaborative Architectures

Although Google Docs is not an engineering CAX application, it is one of the first stable and commercially used browser-based, multi-user document and spreadsheet editors [42]. Using client-server (CS) architectures, on-line documents can be imported to client computers and simultaneously edited in an unconstrained manner through the user's browser application. A revision history is automatically maintained for document recovery. The Google Docs application server manages access to a document through an owner, collaborator, and viewer hierarchy, with copy privileges at all levels. Thus, document security is loose and meant for open access environments.

Sun, et al. [32], use M/S API's and Transparent Adaptation (TA) to develop similarly unconstrained collaborative access to Word and PowerPoint documents for multi-user editing. Sun extends operational transform (OT) theory to avoid inconsistencies in editing intent when two or more users are editing the same object. Multiple modes of viewing and editing are implemented, along with multi-user awareness. Sun also provides some limited conflict resolution methods when editing the same element (e.g., a document character or a graphical element), but the final resolution solution for multi-user direct conflicts is unclear, although all modification instances are stored. Each client must import additional software in the form of a collaborative executable that is linked to the Word and PowerPoint applications to record/transmit the document editing operations among the multi-collaborators and the server.

Zheng, et al. [38], extend the TA methods of Sun [32] to AutoCAD as a CoAutoCAD application. Zheng proposes an open unconstrained collaborative architecture, suggesting that conflict resolution among several users can optimistically be resolved. He refers to constrained collaboration such as grid locking as pessimistic approaches to collaboration, yet notes that proprietary information should not be shared, and also notes that collaborative design may need to be organized among the design group individuals. Like Sun he suggests that a multi-user system should provide multi-user awareness at each station.

Ramani, et al. [28], like most researchers in collaborative design, use a CS architecture where the server acts as the master to distribute model changes between collaborating session 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. The ACIS geometry kernel is used in the prototype implementation and models can update locally based on change commands, although the updating frequencies and difficulties are not discussed. The viewing model is stored locally in faceted form.

Cera, et al. [6], have designed a prototype called MUG for group authoring of design semantics and sharing of design models. MUG uses CS architectures, along with Java tools like Java3D and OpenGL. Many multi-modal sharing features of the previous references are demonstrated without detailed discussion into user conflict resolution. Cera states: "The need for common knowledge sharing standards (which go beyond data standards such as STEP) has been often cited as a major obstacle to achieving true concurrent engineering." The paper's main contribution is "designers collaboratively author an assembly design, specifying its structure and layout in 3D while annotating the relationships among all the entities in the assembly with reference to base ontologies."

Jing, et al. [15], appear to use a peer-to-peer (P2P) architecture between replicated collaborative CAD stations. Each peer must also replicate the multi-user software that controls the change inputs and outputs, local locking method, and broadcasting to peers. It is unclear how peers are accepted into the collaborative sessions.

Li, et al. [17], present a 2005 review of the status of collaborative CAD, noting the dominance of CS architectures, lack of data security, and the difficulty of real-time interaction over networks, while proposing that model simplification could adapt modern streaming methods. Li distinguishes thin-server-strong-clients (clients have full model) from strong-server-thin-clients (client model has partial model and mainly used for visualization). **Liu**, et al. [19], recognize the reluctance to use collaborative CAD, which can be related to network latencies, security and cultural issues, and because software companies have not adopted collaborative advances. They propose that independent agent-based methods be deployed to enhance the multi-media communication between collaborating personnel. The intelligent agents can apply rules to communicate among themselves and distribute change information among collaborators.

2.2 Collaborative Infrastructure

This paper does not focus on general product development infrastructure; we are more concerned with the collaborative interaction infrastructure among multi-CAX users. Numerous papers note that engineering collaboration difficulties can be traced to the basic architectural limitations inherent to modern CAD systems, a basic premise of this paper. For example, **Chen**, et al. [8], state: "A collaborative CAD system cannot be simply set-up through equipping a standalone CAD system with IT and

communication facilities. Due to the complexity of collaborative design activities and the specific characteristics/requirements of CAD systems, it needs some innovations or even fundamental changes in many aspects of CAD systems, such as infrastructure design, communication algorithms, geometric computing algorithms, etc.” Chen then demonstrates adaptive modules that can be used to abstract design functionality and pass related data between distributed users, even between different application types, such as AutoCAD and M/S Excel.

Barbosa, et al. [2], note that conventional CAD systems control their distributed models through version control and propagation change. He suggests that core infrastructure changes are imperative for effective collaboration, suggesting that CAD models need an object model foundation. Models should be made up of objects that can be collaboratively interpreted, edited, linked and distributed. These object models, built on a fundamental geometry engine like ACIS, will relate model attributes, provide links to multiple collaborators, record change histories, and can be instantiated into visual forms.

Bonneau, et al. [3], demonstrates, for complex building applications, a hierarchical file structure for breaking up building structures into sub-structure files, then assigning file design responsibility to distributed collaborators. In a sense, the manager artificially breaks the complex structure into an assembly of component parts (and files), such as different floors or structures. Document control and integration, with oversight management, then permit multi-users, assigned appropriate files, to add to or modify the overall structure.

Fan, et al. [12], consider grid computing and hybrid CS/P2P systems for sharing resources among a set of collaborators. Effective CAX collaboration will require all users to access equal resources; else there will be a productivity imbalance.

2.3 Collaborative Interfaces and Multi- User Tools

Researchers have experimented with multi-user interfaces for decades. **Liu**, et al. [19], propose to use intelligent agent software to support interaction among multi-users. Each site must have this software. Agent software can locally capture audio and video streams for each user and pass them over the collaborative network; another agent might capture design changes, and update these changes among users, according to user privileges.

Shen, et al. [29, 30], use Augmented Reality (AR) concepts to display a model among collaborating design teams, where each member can assume a different view point, but only one member has edit rights. Using CS architectures, the model is replicated at each user site and server software manages the change propagation among the users.

To improve multi-user collaboration in fusion control rooms **Wallace**, et al. [34], have developed a multi-cursor X window manager that allows multiple users to concurrently interact with all desktop components, including applications, menus, etc. Multi-cursor support has been added at the systems level, rather than application level, allowing multiple users to simultaneously control the active applications on the screen using different colored cursors. Only one cursor can grab an application at a time, but multiple people can control the same cursor. The architecture uses a single event buffer which simulates multiple event queues by separating the system cursor events from the multi-cursor events. The limitation is that the X window software is essentially time sharing the system cursor among the different users. Another limitation is that user conflicts occur without carefully orchestrated interactions. The authors found that larger displays worked better in the multi-cursor sessions.

Xu, et al. [37], intercept and broadcast Windows GUI events to distributed users, rather than capture CAD API events. This avoids the development of collaborative software based on application API's. All users must be exercising the same application and the event interceptor and broadcasting software must also be replicated at each site. There remain some problems that relate to initialization and synchronization.

Multi-touch research has been conducted over several decades and we now see screen touch gestures integrated into user convenience interfaces on cell phones and PDA's; see [21] - [24]. **Team-Player** [33] is a game that allows you to use multiple mice and keyboards on a Windows based system. TeamPlayer is designed for a group environment where multiple people will be interacting with the

same computer. Each mouse is assigned a unique colored cursor to identify it. It appears to handle the problem of multiple mice by shifting the focus to the "real" Windows cursor for actual actions.

Multi-Cursor ICEWM [22] is a multi-user window manager currently available for Unix X11 environments. It creates a desktop environment that allows for simultaneous input from multiple users. Users can work concurrently on different applications or desktop items. Each user gets a uniquely colored cursor. The multi-cursor window manager works by time-slicing the single system cursor. The state of each user's multi-cursor is maintained and restored just before an XEvent is sent to the system.

Dempski, et al. [9], have developed large user interactive display walls that use camera-based touch solutions and that can resolve finger dynamics at 30 Hz. The system is capable of displaying multiple user interactions, but one touch at a time, although the researchers are experimenting with multi-touch solutions.

2.4 Model Sharing and Collaborative Security

Collaborative security should be a primary concern for industries engaged in global product development. The CS and P2P architectures used in the reviewed research prototypes inherently expose IP to multiple collaborators. **Wang**, et al. [36] address this problem by proposing lean and selective information sharing among collaborators, using encryption means to securely pass the information over the network among collaborators. Based on role-based viewing methods, partial data sharing is used to deter reverse engineering, but must be well integrated with data management systems.

Cera, et al. [5], use role-based viewing control access control through geometric partitioning of 3D models. "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." This technique is used to implement hierarchical security access privileges based on the Bell-La Padula model.

Reference [43] describes the **SoftXpand** product that, by adding several video cards, creates separate Windows XP virtual machines running from each video card. In this brute force method the virtual machines can be displayed either on one monitor, or on several monitors. The obvious advantage is that on each virtual machine you can run different processes. This method could be used to share, yet secure, the design models.

2.5 Collaborative Constraints and Conflict Resolution

Jing, et al. [15], use a local locking mechanism for distributed and replicated collaborative CAD systems to avoid user-to-user conflicts over a network. Topological correspondence conflicts are handled by hierarchical naming conventions and some user assigned responsibilities. The RCCS (Replicated Collaborative CAD System) architecture appears to be P2P so that the ACIS models are replicated locally at each user station, along with the collaborative management software. The RCCS methods use operational transforms (OT) to bind the topological operations with the object parameters when they are transmitted to the other collaborators for model updating.

Bu, et al. [4], use semantic locking to prevent semantic violation, where the locks are classified into region lock and object lock for resolving violations at different levels of granularity. Users can attach comments to the lock explicitly or implicitly using some rules. User negotiation and versioning rules are used to resolve conflicts among the collaborating users.

Chen, et al. [7], apply three coordination rules embedded in e-Assembly to satisfy collaborative assembly constraints in a CS environment. The coordination rules serve to maintain consistency among collaborators working on different aspects of an assembly (*atomic* object, object links, and object interface constraints).

Dempski, et al. [9], note that their display walls resolve multiple user interactions one touch at a time, and that this introduces some conflict among the gathered users, who may be sub-grouped around an application.

Constraint-based parametric design is a useful way of scaling designs where only a few parameters control the design details. Although **Lai**, et al. [16], do not focus on collaborative CAx, they do

demonstrate some hierarchical constraint-based design concepts that could prove useful to decomposing tasks among a set of collaborators. **Lin**, et al. [18], note the difficulty of applying constraint methods when several collaborators are concurrently engaged in graphic design, while noting that constraint methods are proven tools in single user systems. Lin considers collaborative blocking, masking, and time stamped methods when constraints are applied to design operations such as moving a graphical object that is subject to some geometrical or parametric constraint, and where multi-users working the same object would confuse the constraint relations.

Lu, et al. [20], suggest that product collaboration falls into three primary areas: 1) design rationale and decision-making processes; 2) design data, as an information flow process among design activities; and 3) design activity as a collaborative group conducting design operations based on task dependencies. These areas often fail to acknowledge the importance of social interaction among the product team members, leading to the need for guidelines to *social interaction* and *conflict management*. The authors use Petri Nets to represent the collaborative process and to incorporate socio-perspective states that accompany the Petri representations.

Panchal, et al. [25], present an Interval-Based Constraint Satisfaction (IBCS) Method for decentralized, collaborative multifunctional design. The authors develop a strategy of capturing a designer's decision making processes (as represented by design parameters) throughout the collaborative process, then representing the range of the design parameters as design constraints. By comparing each user's constraints, the design space is incrementally reduced, complexity is reduced, and the design converges.

Ram, et al. [26], propose that object-oriented databases (ODBMS) are more appropriate for design activities than relational databases [36]. Although commercially object databases are dominated by relational databases, ODBMS are better designed for the diversity of information found in engineering design, and that can be represented by design objects, with object links to other information such as video and audio. Integrating collaborative design into object databases will require methods for conflict resolution, because the differing views of the participants. Conflicts normally occur when one participant optimizes the design problem at the expense of the other participant problems. Ram shows how a constraint meta-object can be used to propagate the design constraints (e.g., parameters regions or limits) among the participants, allowing them to resolve visually or verbally the various design objects.

Sun, et al. [31], address the problem of maintaining design intent when exchanging CAD data between collaborators on different CAD platforms. Sun notes that feature based techniques, along with the design history/order, can preserve the design geometry on different CAD systems, but that the design intent is lost - see [49]. Sun proposes a constraint conversion method to preserve design intent using, e.g., Hoffman's geometry certificates [14].

2.6 Gaming

Gaming is concerned with network speed and strategies, with reducing network communication latencies between players, and with maintaining data consistency and security for hundreds/thousands of players. Reference [40] presents algorithms for handling these problems where a server can update each player and the numerous AI clients (~100 or so units) with *relevant* (reduced) data consistency at 1 - 2KB/sec. The methods use data compression and grid prediction algorithms to reduce the amount of dynamic data that has to be sent between players.

Reference [45] notes that server centralized gaming architectures suffer from robustness (single point of failure) and scalability limitations, while suggesting that distributing the game architecture among the users, according to available resources, will provide greater speed and robustness. This P2P mode has its own challenges in terms of data consistency and updating latencies, although noting that many games accept weak data consistency. The Colyseus architecture partitions game state among the distributor player servers, dynamically maintains a table of servers, and speculatively pre-fetches objects by predictive techniques.

GauthierDickey, et al. [13], propose a P2P architecture (greater scalability, but more difficult to implement) as alternative to CS architectures (limited throughput and scalability), noting the challenges of maintaining consistency, access rights, and event propagation, while avoiding cheating.

Wang, et al. [35], propose hybrid P2P/CS architectures to collect a set of clients around distributed Game Service Platforms (GSP) in local clusters. Each GSP maintains local cluster states and can query the GSP states of other clusters. The hardware and software implementations are not detailed.

Douglas, et al. [10], apply an agent-based approach to facilitate real time interactions between player entities in a P2P game architecture. A Spatial Data Service (SDS) manages entities within regions; entities register their current region of involvement with the SDS and query the SDS for other entities in the region, thus making connections/interactions through real time agents. Note that regional based approaches reduce significantly the P2P information and dependencies in massive multi-player games. **Assiotis**, et al. [1], also split the virtual world into smaller areas, but use a distributed CS architecture, where server density is a function of player density in virtual regions.

Endo, et al. [1], deal with the P2P problems of cheating, user interaction delays, and network congestion by collecting user machines into a cluster with some acting as site servers. The site servers act to store all user states within the site cluster. Updates are balanced against all site servers and accepted by all user machines. One site server is designated the administrative server to handle the server delegations.

Chu [46 - 48], an IBM researcher, describes a distributed server architecture for MMO games that consists of distributed servers to interact with gaming clients, an application server to integrate the distributed gaming servers, and a database server to persist and retrieve data.

Recognizing that the network is the bottleneck in gaming sophistication, and that gaming resolution continues to increase, **Ramakrishna**, et al. [27], construct a multicast tree connecting the players that have a low average player-to-player latency. "The main goals of our research are to enable the building of self-configuring and self-optimizing structures for the routing of packets between game clients. It is our aim to ensure that the number of packets containing game information transmitted through the network be minimal while also ensuring that the average time taken by a packet to move from one client node to another be as low as possible." The drawback is that this architecture, although user transparent, still has to be deployed over network nodes, and thus requires the cooperation of Internet Service Providers.

3 NX CONNECT - COLLABORATIVE CAD PROTOTYPE

NX Connect, as an add-on to the existing CAD package Siemens NX, allows multiple users to access and make changes simultaneously to a single part file. This prototype software was designed to investigate the limitations of collaborative modeling using a primary commercial CAx application, rather than develop a new application from the ground up. Siemens NX has an extensive Application Programming Interface (API) available with the software.

NX Connect allows multiple users to open and actively make changes to a part file simultaneously. Each user is able to view the part independently and utilize the zoom and rotational functions without affecting the view of the remaining users. This allows each user to focus on the intended task while viewing updates made to the part file by other users.

3.1 NX Connect Architecture

NX Connect utilizes CS with a thin server and strong client. The server will retain all necessary information for the part file and each of the workstations will then utilize that information to create/update identical parts on each of their respective machines. The server also retains and makes available each change made during the design session. Each client generates a local copy of the part file and utilizes information gained during the design session to locally update the part. Therefore most of the computation and processing is done on the local workstation, allowing for a much faster runtime environment. NX Connect relies upon four custom modules, Fig. 2.

The **Information Storage Module** (ISM) in the server stores the part features and related data, including all information relating to part location and multi-user access. The primary qualification for designing an ISM is that it must quickly and efficiently store and recall small and large amounts of data. We applied Microsoft's SQL Server as the RDMS which is easily accessible by calls made in C#.

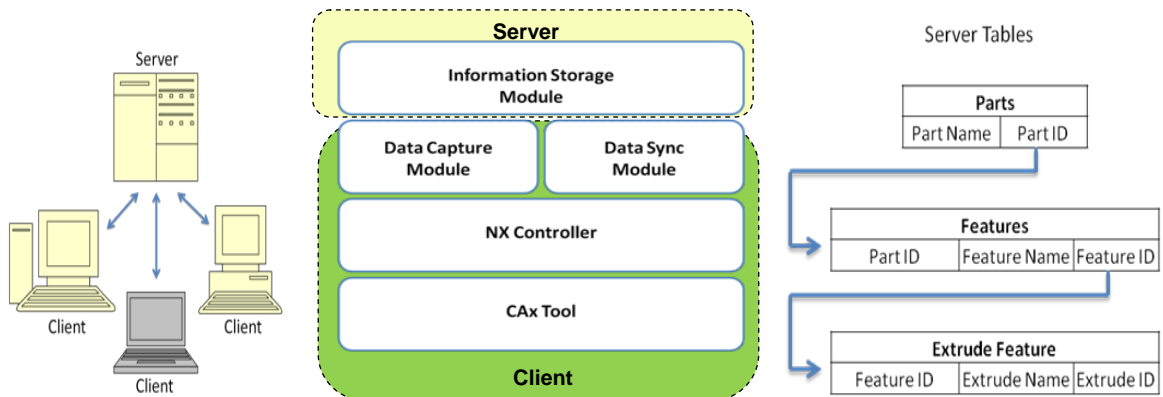


Fig. 1: CS architecture

Fig. 2: NX Connect modules

Fig. 3: Server feature attributes

A series of tables are created and linked together to organize the various data types. Among those created are tables to manage the Parts, Users, Features, and each available feature type. A hierarchical structure was implemented where each part stored in the Parts table is assigned a Part ID which is then used in the child table that contains all of the features, Fig. 3. The features for the previously mentioned part will have as a parameter value the Part ID for which the features apply. Each feature is then assigned a Feature ID that is utilized in the respective child table corresponding to the feature types (extrude, revolve, etc.). Similar relationships are implemented within the Feature Type tables linking those features to other children parameters such as the sketch used.

Data stored in Microsoft SQL Server is limited to a subset of primitive data types and cannot store more complex NX Object types. Primitive values consist of common data types such as *int*, *float*, and *char*. To store the information for each part file it is necessary to first break down each of the part features into parameters that could be described using these primitive types. Fig. 4 illustrates how an NX Object is broken down into its primitive values, where the X, Y, and Z coordinates are stored in the database as either float or double type variables along with the sketch name. By using primitive values a minimal amount of change data can be sent over the network.

The NX Connect multi-user environment requires changes made locally on each users system to be constantly written to the database and the database changes to be continually broadcast to all connected users. Two modules were developed for this purpose: 1) Data Capture Module (DCM) and 2) Data Sync Module (DSM).

Data Capture Module - To eliminate the need for each system to constantly send information to the database the DCM monitors the NX session to determine when changes have been made to the part file. When a change is detected the DCM determines the feature that has been created, modified, or deleted and then alerts the NX Controller of the change, then passing the change information for the modified feature.

Data Sync Module - The DSM monitors the ISM to determine if any changes have been uploaded by another user. Upon finding a change to the database, the NX Controller is alerted to the changes.

NX Controller - At the core of the NX Connect architecture is the NX Controller. This module converts all part information into their primitive values for storage in the database, as well as converting these parameters back into useful data to construct the corresponding features in the NX session on each users machine. As mentioned above, the NX Controller receives data from the DCM and DSM. When the DCM reports a change in the local NX part file, the controller gathers all pertinent information about the modified feature by utilizing the NX API. Using the API commands the controller determines the feature type and then deconstructs the feature into the basic parameters such as extrude start and stop distance, draft angle, and Boolean type. If a sketch is used for the specific feature the controller stores the sketch name for later name reference. All such information is sent to the corres-

ponding table in the database for distribution to each user. This controller function is labeled the push function.

The NX Controller is also responsible for a related pop function, Fig. 5. After a user modifies the existing CAD document, and changes are uploaded to the database, the DSM will then alert the Controller of a change ready to be distributed to each user. The NX Controller will then read the feature data and place each parameter in the appropriate API function to recreate locally the feature or modification on all user workstations.

3.2 Implemented Functionality and Example

Currently, our NX Connect prototype has implemented networked exchange of datum planes, sketches, extrusions, Booleans, and revolutions. Within the sketching function line and arc features can be created. Presently, no geometric or dimensional constraints on a sketch are recorded in the database or transmitted to other workstations. Other more complicated features have not yet been implemented such as sweeps, blends, surfaces, or 3D curves. By testing the NX API with our basic functionality, we are confident that we could generate a commercially viable collaborative CAX under the CS architecture, with some minor API additions, and with some ownership/revision modification of PLM software.

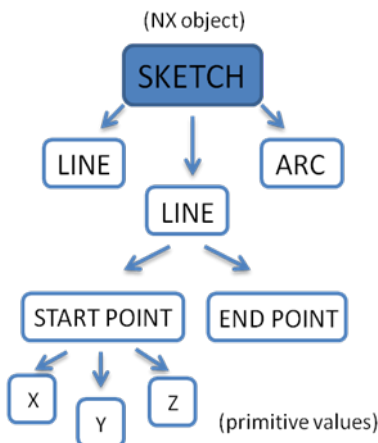


Fig. 4: Object primitives

Fig. 5: For each feature the push and pop functionality must be implemented separately. This code shows the pop function.

```

//Read in variables from Database
ExtrudeFeature extrudeFeatureToPop = (from ef in nxconnectdb.ExtrudeFeatures
where ef.ExtrudeFeatureID == ExtrudeID
select ef).FirstOrDefault();

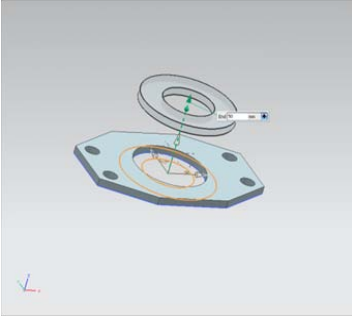
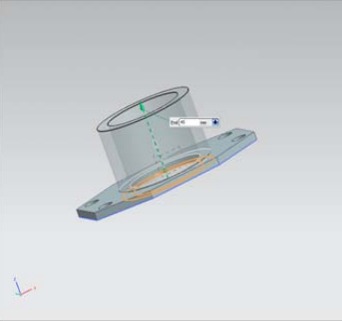
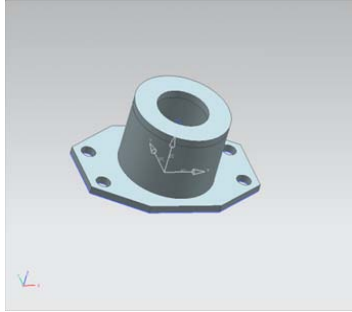
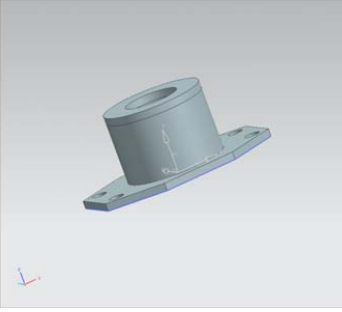
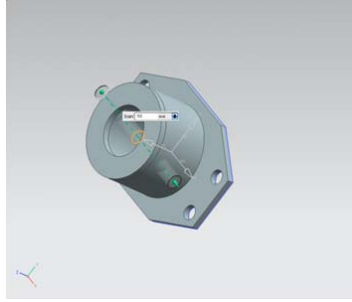
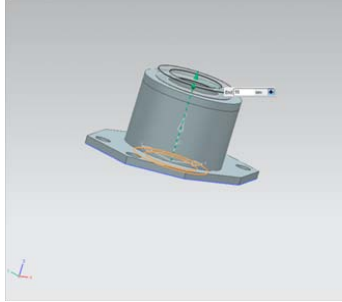
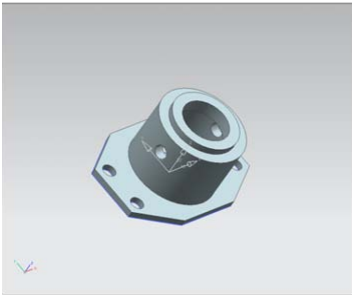
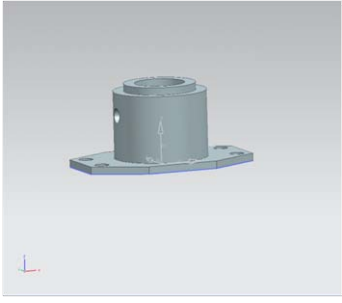
string nameOfSketch = extrudeFeatureToPop.nameOfSketch;
string EndExtendValueRightHandSide =
    extrudeFeatureToPop.endExtendValueRightHandSide;
string EndOffsetRightHandSide =
    extrudeFeatureToPop.endOffsetRightHandSide;
string StartExtendValueRightHandSide =
    extrudeFeatureToPop.startExtendValueRightHandSide;
double helpPointX = (double)extrudeFeatureToPop.helpPointX;
double helpPointY = (double)extrudeFeatureToPop.helpPointY;
double helpPointZ = (double)extrudeFeatureToPop.helpPointZ;

//Set up Extrude Builder
NXOpen.Features.Feature nullFeatures_Feature = null;
NXOpen.Features.ExtrudeBuilder extrudeBuilder1;
extrudeBuilder1 = workPart.Features.CreateExtrudeBuilder(nullFeatures_Feature);

//Read variables into the Extrude Builder
Sketch sketch1 = (Sketch)workPart.Sketches.FindObject(nameOfSketch);
NXOpen.Direction direction1;
direction1 = workPart.Directions.CreateDirection(sketch1,
    dirSense, withinOrWithoutModeling);
extrudeBuilder1.Direction = direction1;
extrudeBuilder1.Limits.StartExtend.Value.RightHandSide =
    StartExtendValueRightHandSide.ToString();
extrudeBuilder1.Limits.EndExtend.Value.RightHandSide =
    EndExtendValueRightHandSide.ToString();
extrudeBuilder1.Offset.EndOffset.RightHandSide = EndOffsetRightHandSide.ToString();

//Create Extrusion
NXOpen.Features.Feature feature1;
feature1 = extrudeBuilder1.CommitFeature();
  
```

In the orchestrated example of Tab. 1 two users collaborate to make a part that requires multiple features. In this particular example one user creates the initial plate by extruding a sketch and is then joined in the design session by user two. Both users work simultaneously to complete the desired part of Step 4 in Tab. 1.

Steps	User 1	User2
<p>Step 1 - User 1 has the task of creating the upper ring of the part, while user two is creating the outer walls of the cylinder. Each user is editing their respective feature using the extrude function. These figures show a screen shot from each user at the same time. Both users are about to finalize their respective extrusions.</p>		
<p>Step 2 - User 2 finishes the extrusion first and the outer cylinder is updated to the screen of user 1 while user 1 is still working within the extrude dialogue box. User 1 then completes the action and the upper ring is sent to the database and out to user 2 with less than a second of delay. These figures show updated screen shots of each screen after the extrude functions are completed.</p>		
<p>Step 3 - A third user might also join the session and be responsible for uniting all the pieces. At the same time User 1 creates a through hole in the side of the cylinder by adding a cylinder to be subtracted from the part, and User two creates a lip on the top surface of the part. These figures show user 1 creating the through hole in the part while user 2 works on the upper lip.</p>		
<p>Step 4 - These figures show the final part after all actions were completed. By working together in a parallel modeling environment, the complex part was created faster than conventional single user approaches.</p>		

Tab. 1: NX Connect example

3.3 User Interfacing

As add-on software to the commercial package NX, NX Connect does not change the modeling environment available through the basic NX software. The NX Connect interface consists of a small pop up window created when the NX Connect DLL is run from within the NX system. Within the window are options to populate the parts list currently stored in the database, to load a selected part from the list, and to upload the user's current part into the database. As of now, no functionality has been added to display other users within a design session or to communicate with others who might be working at the same time. Each user is aware of other users only through updates they might make to the part file.

Integration with other collaborative techniques will be continually investigated. For example, inter-personal communication tools like Voip Buster, Ventrilo, or Skype have been used in the NX environment and could satisfy higher-level communication in real-time collaboration. Currently we have a prototype that utilizes Skype integrated into NX so that multiple users can collaborate through audio. The goal of the v-CAX environment is to provide a high level interface for modeling and viewing by multiple users, without changing dramatically the modeling interface accepted by industrial users of commercial CAX applications.

3.4 Limitations

The Siemens NX system was not created for multi-user capabilities, nor was the API developed with this purpose in mind. One significant issue is determining when the NX part file has undergone a change. Initial attempts included trying to access the event buffer which would contain the most dynamic information about changes in the part file. However, it appears that NX does not allow simple access to the event buffer. An alternative method has been devised utilizing undo marks to mark restore points for the undo function. NX creates undo marks for many different actions performed within a session, including many that have no significance to the multi-user functionality. The DCM poles the NX session for undo marks that signify significant changes to the part and then sends the alert to the NX controller.

Another area of difficulty is found in the way NX Connect handles sketches. Currently NX adds sketches to the history sequence as soon as a sketch is begun, and while a sketch is being edited it waits for all the geometry and constraints to be added before updating the model to contain the updated sketch. Thus, while in sketching mode NX does not commit any other updates. If another user adds a feature to the model history while in sketching mode, the local machine will not receive those updates, including the start of another sketch by another user. Thus two users can not concurrently create a sketch, a major detraction from the overall usefulness of a concurrent modeling system. Other than the sketcher the other NX operations do not update until they are finished. Fixing these deficiencies would not seem to be difficult.

The basic design of a CAX API does not recognize the need to pass design changes among multi-users. For example, in certain cases while executing the command to return the name of a specific parameter such as Datum Axis, NX returns the handle to the parameter object, which is a memory address specific to the user's computer, but invalid on other workstations. It seems that a collaborative API could be introduced to resolve this problem by converting the geometric change data into an appropriate format for Internet exchange. CAX companies are already well versed in providing API's for conversion to neutral forms, e.g. IGES and STEP.

4 COLLABORATIVE CAX OBSERVATIONS

Global product development needs better tools for collaboration. After reviewing the cited and excellent body of research, then comparing it to the single user CAX commercial applications being widely used today, such as NX, Catia, PTC ProE, SolidWorks, Varimetrix, etc., we ask why is there not widespread use of collaborative product design tools? Conversations with CAX architects reveal that modern CAX/computer OS architectures are simply not built for multiple users and have not anticipated (or perhaps even desired) architectural flexibilities like multiple screen events, multiple event buffers, multiple cursors, multi-user GUI's, regional/spatial model decomposition beyond assemblies, applica-

tion layering of CAX models over partially completed models (simultaneous design, analysis, process planning), and others.

4.1 Primary Observation Limitations

If collaborative CAX is to become practical, it will require significant modifications/additions to existing computer OS's and commercial PLM/CAX tools. We emphasize that product development in most companies depends on current commercialized CAX software. Ultimately, the end user will have to demand new collaborative tools from their CAX providers. There is one exception though. With a few additional interface tools and some increased flexibility in PLM version control, a CS adaptation could sustain multi-user CAX collaboration on shared models, but with some IP vulnerability.

4.2 General Observations

We list our general observations in no particular order of preference:

- 1) *Network performance* - Network communication latencies and inconsistencies drive the architectural solutions of many of the successful prototypes, depending on application type. CS is the dominant network architecture used for collaborative CAX investigations, but hybrid P2P and CS networking architectures may regionally replicate servers in response to network traffic and distributed user space densities.
- 2) *User density* - The application type (game, CAX application) will determine user density and participation flexibility. Gaming is particularly limited by CS architectures as user densities grow. P2P architectures, though less structured, less secure, and more difficult to profit from, enable higher user densities through means such as spatial/gridding observation and prediction, and regional control clustering (distributed serving).
- 3) *Replication* - All collaborative approaches replicate something at the multi-user sites. It may be CAX or gaming models, the application itself, or dedicated collaborative control/management software which may include databases. The collaborating tools must be transparent to end-users, other than those designed to promote multi-user awareness and interaction. An important security concern to product industries is that CS collaboration may require CAX model replication at user sites.
- 4) *Security* - Gaming is concerned about cheating and security, and provides methods to minimize cheating problems. There is less focus on security issues in the collaborative CAX engineering prototypes that we reviewed. P2P architectures provide less security than CS architectures where a server application can more easily detect user anomalies.
- 5) *Model/Process Ambiguity* - Gaming can tolerate some model or user action ambiguity, while CAX models/processes must be precisely defined and maintained. CAX activities in the early product conceptualization and specification stages can tolerate some ambiguity.
- 6) *Model editing* - Collaborative model editing is examined at many levels, from the root operations [4,5,6] used to construct parts/features, to model partitioning and spatial gridding, and other constraint methods used to break the model into user assignable design regions. Proponents refer to unconstrained collaboration as optimistic, whereas regional gridding and constraint methods with user assigned access and non-authorized blocking are referred to as pessimistic. Optimistic methods require more user communication and conflict resolution, which may demand time stamps and other operational sequence information.
- 7) *Collaborative agents* - A popular, less structured approach to user interaction is to use software agents to observe, record, and transmit user actions among a set of collaborators. These agents can be configured for application type and user/cultural personalities.
- 8) *Conflict resolution* - Some researchers allow users to edit the same objects, depending on audio/video personal interaction to resolve conflicts, whereas other researchers use grid (regional) locking, semantic and session locking, and model blocking to partition multi-user interactions.
- 9) *User interfaces* - Limitations such as single event buffers and screen cursors have limited GUI collaborative flexibility for CAX applications. Researchers have generated multi-cursor control using software like X windows but the event control is limited to time sharing the OS cursor event. Xu [37] uniquely captures and transmits CAX Windows GUI events among collaborators

rather than program at the CAX API level. Researchers have observed that multi-user collaboration is more effective on larger displays.

5 AN AGENDA FOR PRACTICAL COLLABORATIVE CAX

As noted earlier, CS collaborative prototypes are feasible when both models and CAX application software can be replicated and distributed. Replication leads to some IP vulnerability and PLM systems are not designed for simultaneous ownership of models or processes by multiple personnel. Product development tasks are typically delegated to organizations with defined expertise, and ultimately assigned to individuals. Multiple individuals are not likely to be assigned the same tasks in the same time zone, although they may be asked to cooperate with lesser ownership privileges.

Today, companies use task decomposition and integration to schedule an orderly product flow through the development and production stages, even when activities are distributed globally. It seems unlikely in the near term that companies will champion unconstrained (simultaneous) low-level model editing, because of the intense user interaction needed and the inconsistency of Internet communications. It also seems unlikely that companies will allow more model ambiguity or IP exposure. Thus, collaborative advances need to protect IP in a distributed environment, and yet allow personnel meaningful ways to simultaneously edit models. This appears to favor some modes of partial model exposure, e.g., model decomposition and constraint boundaries.

Companies will be interested in new collaborative methods that blend transparently within existing CAX tools, rather than complex and additional applications that run outside the mainstream CAX tools. This could be satisfied by expanding CAX API's to communicate model changes over the network, while expecting a managing server to update the changes and maintain model integrity. Server clustering poses a potential hybrid P2P/CS collaborative solution in global product development, but poses more of an IP risk. Partial model assignment could work in a clustering architecture.

A logical question is how might CAX GUI's be modified to support multi-user collaboration? Multiple and larger displays are rather self-evident. Multiple displays would expand the viewing window and even allow for the "brute force" compounding of applications, each application assigned to a different user. Of course compound applications would need compound cursors. Multiple cursor inputs in a *single* application window could lead to editing conflicts and also require that the CAX event buffer be redesigned to recognize multi-cursor uniqueness. There are other challenges to overcome, such as converting single event buffers to multi-event event buffers. How would users in the same application window navigate the hierarchically layered menu windows? Some CAX applications do not allow access to the event buffer through the API, but may provide access to the design operations and parameters through a history file sequence and undo markers.

Another strong research component that could be integrated into existing CAX GUI's is agent software. Agents could be deployed to reflect the cultural preferences among a set of collaborating users, and to filter and distribute user events among collaborators. Unique avatars and other dynamic icons could reflect progress, query events, uncertainties, etc., to connected multi-users through the GUI. Current CAX GUI's could be extended to configure multi-user agents on each user's interface.

Model decomposition and constraint boundaries (regional blocking) could be implemented without major changes in current GUI architecture, and allow users to collaborate on a single model without spatial or GUI event conflicts. Viewing strategies could permit collaborating users to view or mask the partially completed regional models of the other users. This could resolve unanticipated design issues sooner, and promote teamwork and learning.

Agenda Item 1: Collaborative Understanding - Initial investigations should examine current tools for development collaboration, and how industries are applying these tools, particularly examining business practices when distributing IP through globally distributed product groups. The investigation should also consider expanding API access to the CAX kernel, identifying where the API could be modified for a multi-user environment. This would include direct access to the event buffer, and new API's that would extract the core parameters for the various design objects, rather than handles to proprietary modelling objects.

Agenda Item 2: Practical CS Collaborative Architecture - Numerous prototypes are working on CS architectures, where the model is replicated at each site. If model security can be relaxed among a set of trusted product engineers, then the development of a few more tools will make CS collaboration practical. These include CS for localized collaboration and distributed server clustering.

CS for localized collaboration - In more localized environments where the network proves reliable and consistent, and assuming that several CAx companies are willing to expand/modify their API set to resolve some of the minor deficiencies mentioned earlier, a strict CS architecture could support multi-user collaboration. New multi-user software, such as the NC Connect prototype in this paper, or some of the other notable prototypes reviewed in this paper, could be used to generalize an API set for multi-user collaboration, with capabilities to expand current CAx GUI's for multi-user observation and model activity updating.

Distributed server clustering - Gaming technologies have made strategic advances in networked server distribution/clustering that could prove useful to collaborative CAx. Network latency is a critical dynamic to both successful gaming and CAx collaboration. Global companies will have development offices distributed among several global sites, and thus need to organize their facility servers by user density and development tasks. PLM systems need to incorporate a simultaneous access multi-user file mentality different from current versions, allowing a manager to assign simultaneous user file read/write access. Other PLM procedures will need to be developed to dynamically update part files in a simultaneous multi-user environment, with some type of manager alert, review and approval process.

Agenda Item 3: Model Decomposition and Multi-user Constraints - CAx and PLM systems understand part assemblies and divided ownership. It seems that expanding these concepts to regional model decomposition would be practical using many of these same features. Whereas boundary conditions are used to segregate and join assemblies, constraint equations and other functional relationships could be used to decompose continuous parts regionally, by volume, surface, perimeter, functional relationships, etc. - see Fig. 6. New methods would assign users to a specific model region, divide the regions by constraint features or functional relationships, monitor the screen cursor against the region boundaries, and expose/hide/lock a decomposed model region from other collaborators.

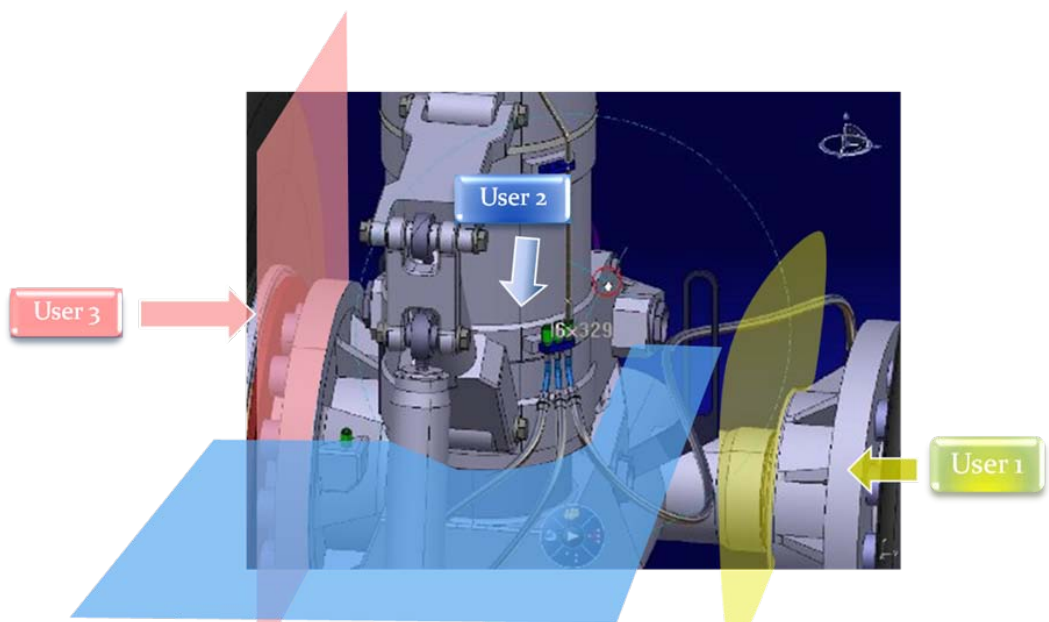


Fig. 6: Regional decomposition and constraint/boundary planes

The CAX GUI should enable only those menu options appropriate to the features within the user's editing region.

Agenda Item 4: Multi-User Screen Dynamics - Generally, CAX API's provide for customizing GUI menus/panels for individual user needs, but are not designed for multi-cursor-user collaboration because computer OS's presume a single event cursor per computer screen and do not have API's to communicate part changes among a multi-user group. Newer OS's now incorporate multi-cursor capabilities that could show other user design edits, but for viewing purposes only. Regional blocking will not allow removed users to change a locally owned model. A strong need for new collaborative GUI's would be the interpretation of multi-user edit intent among collaborators, and its timely transmission. Multi-user awareness is critical to those engaged in collaborative design, along with appropriate data filtering. Agent software has been shown to be effective in capturing and transmitting local changes like these, and could be designed to capture the various cultural elements in global product development. Agent software could be incorporated into existing GUI's with appropriate filters to pass only the data relevant to the current user activities. We propose that agent software be strongly investigated as an add-on capability to existing CAX GUI's. Many gaming features such as avatars, lexicons, predictive actions, and regional dynamics, could prove useful in interpreting another user's design intent among several users.

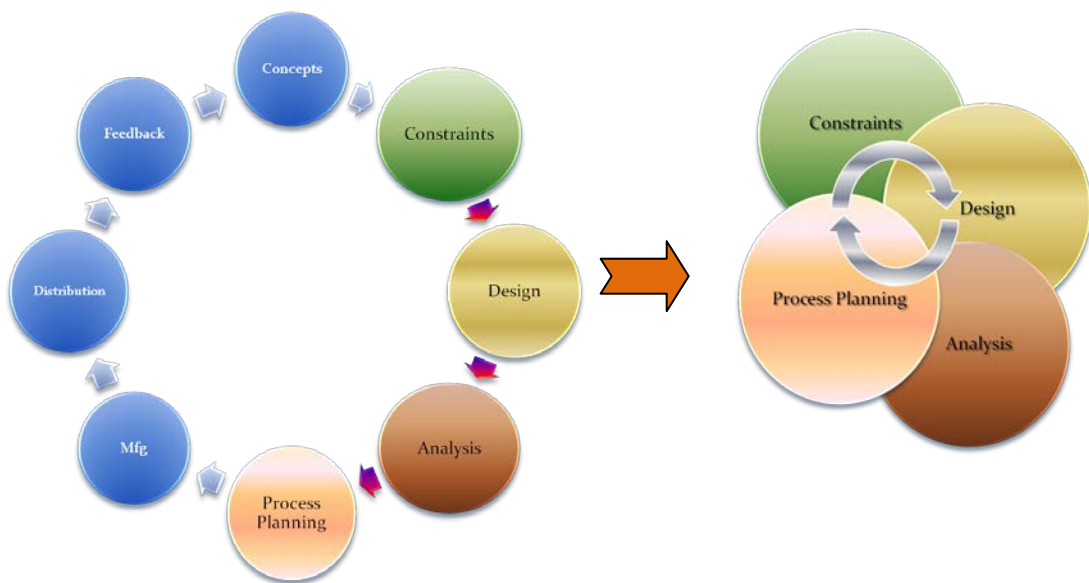


Fig. 7: CAX process layering

Agenda Item 5: Collaborative Layering - Figure 7 shows the conventional product development process on the left side. Historically, the steps of specification (constraints), design, analysis, and process planning consume the *Engineering Design Process* activities. These activities are conducted serially, with the related tasks spread among dedicated departments and personnel. When downstream design or manufacturing errors are discovered, feedback modifications move the product/part back upstream. CAX applications are designed for this serial streaming and feedback process, and perhaps even encourage its serial nature. We would suggest that existing CAX architectures can be changed to encourage the process layering shown by the Venn diagram of Fig. 7. Regional decomposition, when carefully considered, could encourage several CAX processes to be conducted simultaneously in partially or fully completed regions, such as laying down finite element models in completed regions, or tool paths. Dimensioning and manufacturing features, tolerances, tool and machine selection are examples of activities that could be applied immediately after the part geometry is reasonably

well defined. This level of collaboration would encourage personnel with different expertise to provide more timely feedback and reduce the development cycle.

Agenda Item 6: Collaborative Security - Product models contain sensitive information (trade secrets, patentable ideas) that must be protected to preserve competitive advantage. The new collaborative modes that we have investigated introduce new patterns of data sharing that must provide appropriate access control so that only authorized individuals and processes gain access to information and are authorized to make changes.

It is necessary to explore fine-grained, scalable, easy-to-use, and flexible access control methods to ensure that all changes are consistent according to policy and only available to authorized entities. New methods must be suitable for access control in open systems where the participants are not in the same security domain. Traditional authentication approaches based on identity may not be suitable for an open environment. New policy-driven approaches that rely on trust management approaches for sharing across organizational boundaries should be considered, along with audio and video recognition.

6 SUMMARY

We recommend that researchers, end-users, CAx software companies and vendors recognize that multi-user CAx is feasible and practical, with minor changes and additions to existing API's and design procedures. Using the existing body of research and our own prototypes, we have proposed a practical agenda for development that will lead to commercially viable applications and that will assuredly reduce product development cycle times. We have also suggested that the changes will most likely be successful if they are minor revisions to existing and implanted commercial CAx software.

7 REFERENCES

- [1] Assiotis, M.; Tzanov, V.: A distributed architecture for massive multiplayer online role-playing games," <http://pdos.csail.mit.edu/6.824-2005/reports/assiotis.pdf>
- [2] Barbosa, C.; Feijo, B.; Dreux, M.; Melo, R.; Scheere, S.: Distributed object model for collaborative CAD environments based on design history, *Advances in Engineering Software*, Elsevier, 34(10), 2003, 621-631.
- [3] Bonneau, P.; Gabrielaitis, I.: Applying multi-user technology for modeling complex CAD objects," *Statybines Konstrukcijos Ir Technologijos Engineering Structures and Technologies*, 1(1), 2009, 89-94.
- [4] Bu, J.; Jiang, B.; Chen, C.: Maintaining semantic consistency in real-time collaborative graphics editing systems," *IJCSNS International Journal of Computer Science and Network Security*, 6(4), 2006, 57-61.
- [5] Cera, C.; Braude, I.; Comer, I.; Kim, T.; Han, J.; Regli, W.: Hierarchical role-based viewing for secure collaborative CAD, *Proceedings of DETC'03 2003 ASME Design Engineering Technical Conferences*, September 2-6.
- [6] Cera, C.; Regli, W.; Braude, I.; Shapirstein, Y.; Foster, C.: A collaborative 3D environment for authoring of design semantics, *Drexel University Technical Report DU-CS-01-06*, September 2001
- [7] Chen, L.; Song, Z.; Feng, L.: Internet-enabled real-time collaborative assembly modeling via an e-assembly system: status and promise, *Computer-Aided Design*, 36, 2004, 835-847.
- [8] Chen, X.; Fuh, J.; Wong, Y.; Lu, Y.; Li, W.; Qiu, Z.: An adaptable model for distributed collaborative design, *Computer-Aided Design & Applications*, 2(1-4), 2005, 47-55.
- [9] Dempski, K.; Harvey, B.; Korytkowski, B.: Multi-User affordances for rooms with very large, interactive, high resolution screens, http://www.accenture.com/NR/rdonlyres/EB097C88-C0F9-484E-890A-E5E7571A7154/0/multiuser_affordances.pdf
- [10] Douglas, S.; Tanin, E.; Harwood, A.; Karunasekera, S.: Enabling massively multi-player online gaming applications on a P2P architecture, *Proceedings of the International Conference on Information and Automation*, December 15-18, 2005, Colombo, Sri Lanka, 7-12.
- [11] Endo, K.; Kawahara, M.; Takahashi, Y.: A distributed architecture for massively multiplayer online services with peer-to-peer support," *IFIP International Federation for Information Processing*,

- 229, Network Control and Engineering for QoS, Security, and Mobility, IV, ed. Gaiti, D., (Boston: Springer), 147-158.
- [12] Fan, L.; Kumar, A.; Jagdish, B.; Bok, S.: Development of a distributed collaborative design framework within peer-to-peer environment, *Computer-Aided Design*, 40, 2008, 891-904.
- [13] GauthierDickey, C.; Zappala, D.; Lo, V.: A fully distributed architecture for massively multiplayer online games," *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, Draft paper, 2004.
- [14] Hoffman, C.; Juan, R.: Erep-An editable high-level representation for geometric design, *Geometric Modeling for Production Realization*, P.Wilson, M. Wozny, M. Pratt (eds), North Holland, 1993:129-164.
- [15] Jing, S.; He, F.; Han, S.; Cai, X.; Liu, H.: A method for topological entity correspondence in a replicated collaborative CAD system, *Computers in Industry*, 60(7), 2009, 467-475.
- [16] Lai, Y.: "A constraint-based system for product design and manufacturing," *Robotics and Computer-Integrated Manufacturing*, 25(1), 2009, 246-258.
- [17] Li, W.; Lu, W.; Fuh, J.; Wong, Y.: Collaborative computer-aided design-research and development status, *Computer-Aided Design*, 37(9), 2005, 931-940.
- [18] Lin, K.; Chen, D.; Sun, C.; Dromey, G.: Maintaining constraints in collaborative graphic systems: The CoGSE approach, *ECSCW 2005: Proceedings of the Ninth European Conference on Computer-Supported Cooperative Work*, 18-22 September 2005, Paris, France, 185-204.
- [19] Liu, Q.; Cui, X.; Hu, X.: An agent-based intelligent CAD platform for collaborative design," *ICIC 2008*, CCIS 15, pp. 501-508.
- [20] Lu, S.; Jian Cai, J.: A collaborative design process model in the socio-technical engineering design framework, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 15(1), 2001, 3-20.
- [21] MagicMouse: <http://www.apple.com/magicmouse/>
- [22] Multi-Cursor ICEWM: <http://sourceforge.net/projects/multicursor-wm/#>
- [23] Multi-touch: <http://venturebeat.com/2009/09/30/bumptop-adds-multitouch-interface-for-windows-7-computers/#>
- [24] Multiple Windows cursors: <http://weblogs.asp.net/cfranklin/archive/2004/02/21/77787.aspx>
- [25] Panchal, J.; Fernandez, M.; Paredis, C.; Allen, J.; Mistree, F.: An interval-based constraint satisfaction (IBCS) method for decentralized, collaborative multifunctional design, *Concurrent Engineering*, 15(3), 2007, 309-323.
- [26] Ram, D.; Vivekananda, N.; Rao, C.; Mohan, N.: Constraint meta-object: a new object model for distributed collaborative designing, *IEEE Transactions On Systems, Man, And Cybernetics—Part A: Systems And Humans*, 27(2), 1997, 208 - 221.
- [27] Ramakrishna, V.; Robinson, M.; Eustice, K.; Reiher, P.: An active self-optimizing multiplayer gaming architecture," *Cluster Computing*, 9(2), 2006, 201 - 215.
- [28] Ramani, K.; Agrawal, A.; Babu, M.: CADDAC: Multi-client collaborative shape design system with server-based geometry kernel, *Transactions of the ASME*, 3, June 2003, 170-173.
- [29] Shen, Y.; Ong, S.; Nee, A.: Collaborative design in 3D space, *VRCAI 2008*, Singapore, Dec. 8 -9.
- [30] Shen, Y.; Ong, S.; Nee, A.: Product information visualization and augmentation in collaborative design," *Computer-Aided Design*, 40(9), 2008, 963-974.
- [31] Sun, W.; MA, T.; Huang, Y.: Research on method of constraint conversion in feature-based data exchange between heterogeneous CAD systems, *Journal of Mechanical Science and Technology*, 23 2009, 246-253.
- [32] Sun, C.; Xia, S.; Sun, D.; Chen, D.; Shen, H.; Cai, W.: Transparent Adaptation of Single-User Applications for Multi-User Real-Time Collaboration, *ACM Transactions on Computer-Human Interaction*, 13(4), 2006, 531-582.
- [33] TeamPlayer: <http://lifehacker.com/5080196/teamplayer-enables-multiple-input-device>
- [34] Wallace, G.; Bi, P.; Li, K.; Anshus, O.: A multi-cursor X Window manager supporting control room collaboration, <http://www.cs.princeton.edu/omnimedia/papers/multicursor.pdf>
- [35] Wang, Y.; Tan, E.; Li, W.; Xu, Z.: An architecture of game grid based on resource router, *Grid and Cooperative Computing*, Springer Berlin/Heidelberg, 3032, 2004.

- [36] Wang, Y.; Ajoku, P.; Brustoloni, J.; Nnaji, B.: Intellectual property protection in collaborative design through lean information modeling and sharing," ASME Transactions Journal of Computing and Information Science in Engineering, 6(2), 2006, 149-159.
- [37] Xu, B.; Gao, Q.; Li, C.: Reusing single-user applications to create collaborative multi-member applications, Advances in Engineering Software, Elsevier, 40, 2009, 618-622.
- [38] Zheng, Y.; Shen, H.; Sun, C.: Leveraging single-user AutoCAD for collaboration by transparent adaptation, 2009 13th International Conference on Computer Supported Cooperative Work in Design, Santiago, Chile, April 22-24
- [39] <http://msdn.microsoft.com/en-us/magazine/ee336016.aspx> (multi-touch Windows 7)
- [40] http://en.wikipedia.org/wiki/Personal_computer (history of personal computer)
- [41] <http://www.cadazz.com/cad-software-history.htm> (history of CAD)
- [42] http://en.wikipedia.org/wiki/Google_Docs#cite_note-17
- [43] http://www.miniframe.com/solutions_page.asp?id=5
- [44] http://www.gamasutra.com/features/20050613/amir_01.shtml
- [45] <http://www.cs.cmu.edu/~ashu/gamearch.html>
- [46] <http://www.ibm.com/developerworks/architecture/library/ar-powerup1/>
- [47] <http://www.ibm.com/developerworks/library/ar-powerup2/index.html>
- [48] <http://www.ibm.com/developerworks/library/ar-powerup3/ar-powerup3.html>
- [49] <http://www.thefreelibrary.com/Multi-CAD+environments:+new+software+technologies+are+focusing+on...-a0200560685>
- [50] http://en.wikipedia.org/wiki/Object_database