Data Consistency and Conflict Avoidance in a Multi-User CAx Environment

Robert A. Moncur[1], C. Greg Jensen[2], Chia-Chi Teng[3] and Ed Red[4]

[1]Brigham Young University, robmonc@gmail.com
[2]Brigham Young University, cjensen@byu.edu
[3]Brigham Young University, ccteng@byu.edu
[4]Brigham Young University, ered@byu.edu

ABSTRACT

This paper presents a new method to partition a model and maintain data consistency in a multi-user CAx system by introducing three new types of constraints specifically designed for a collaborative environment. These constraints work by constraining and controlling both features and users across an entire multi-user CAx platform. The first constraint type includes locking or reserving features to enable only one user at a time to edit a given feature. The second constraint type, collaborative feature constraints, allows flexible constraining of each individual feature in a model, and the data that defines it. The third constraint type, collaborative user constraints, allows the constraining of user permissions and user actions individually or as a group while providing as much flexibility as possible.

To further present this method, mock-ups, and suggested implementation guidelines are presented. To demonstrate the effectiveness of the method, a proof-of-concept implementation was built using the CATIA Connect, a multi-user CAD prototype developed as a plugin to Dassualt Systems CATIA V5. Using this implementation use cases are provided to show how this method provides important tools that increase collaborative capabilities of a multi-user CAx system. By using the suggested method design teams will be able to better control how their data is used and edited, maintaining better data consistency and preventing data conflict and data misuse.

Keywords: Multi-user CAD, data consistency, collaborative design.

DOI: 10.3722/cadaps.2013.727-744

1 INTRODUCTION

Engineering and product development, by nature, is a highly collaborative activity. With the widespread reach of the internet, as well as the ever-increasing network bandwidths and computer

processing speeds, the technologies needed to improve collaboration in modern CAx systems are in place, and have led to the development of multi-user CAx software. Traditionally, PLM applications manage data conflicts by restricting file editing access to a single user, using secured file check-out and check-in methods. Multi-user tools will require simultaneous editing access to those same files by a collaborating group, with new requirements for data flow management and data consistency in a distributed user network. Without an effective method to manage CAx data consistency (maintaining the desired form and intent for the data in the midst of several users working on the same model and possibly the same features within that model) multi-user CAx tools become useless as would any software tool that did not maintain the integrity of a user's data.

The primary research objective of this paper is to develop a data consistency architecture and system to protect data and prevent data conflict in a collaborative CAx environment. First, situations that could compromise data consistency and/or introduce data conflict in a collaborative design session will be introduced. Next a methodology and implementation for avoiding and/or preventing these situations will be presented. This system will include a user interface and procedure that could be adapted to an existing CAx system through its API. The user-interface and procedure for the system will then be created and implemented into the multi-user CAD plugin CATIA Connect although it could be developed using a similar collaborative system working with one of several commercial CAx systems. The basic data structure of these systems would be adaptable to such a system. Finally usage examples and conclusions will be given.

2 BACKGROUND

2.1 Collaboration and CAx

Over the last ten years researchers have implemented collaborative functionality into CAx tools. This development has been done in two ways; the first being several new CAx tools which have been developed from the ground up, creating the software architecture in such a way that collaborative features would be native to the software. Some of these software packages include web-based systems such as WebSPIFF [2], WebCOSMOS [17], CADDAC [13], and NetFeature [12]. In these systems there is a central modeling server where all modeling operations are executed and the central CAD model stored. The clients are responsible for visualization tasks and interaction between the clients and server. The data transmitted on the network is the faceted CAD model data, which leads to heavy network traffic load. Approaches such as incremental faceted model transmission adopted in WebCOSMOS [17], macro-files adopted in WPDSS [10] are proposed to alleviate the data transmission problem, but the transmission problem still exists. Additional collaborative CAx systems include CollabCAD and NXCollaborate. Most of these systems were merely prototypes and never developed into commercial-grade software.

The second method that has been used to develop collaborative CAx applications is called transparent adaptation, which simply means that plugins that provide collaborative capabilities are developed for existing CAx systems. This method simplifies the adding of collaborative capabilities into already popular and industry accepted tools. Transparent adaptation tools are implemented through the API of existing systems and thus no code changes are needed to be made to existing commercial-grade software code to implement collaborative features. However, this approach is limited by the exposed functionality of the given CAx system's API, which in some software packages is not robust enough to effectively implement to support a collaborative plugin. Examples of transparent adaptation implementations include NXConnect [14], a plugin for Siemens NX that synchronizes CAD geometry between multiple users, and CoMaya [1], a plugin for Autodesk Maya that also synchronizes animation geometry between multiple users. Additional examples of transparent adaptation collaborative plugins for the Microsoft office software are developed by CodoxWare [18].

2.2 About Data Conflict

A collaborative CAx system based on a relational database is particularly vulnerable to *update conflicts* and *delete conflicts*. This vulnerability arises as several users are updating and deleting the same data simultaneously in the CAx system. An update conflict occurs when the replication of an update to a

row (of data in a database) conflicts with another update to the same row. Update conflicts can happen when two database transactions originating from different sites update the same row at nearly the same time. An update conflict could also occur if one client has several local modifications which have not been propagated to the database (possibly due to an interrupted network connection) then those changes are all propagated at once when the network connection comes back online. If there are any modifications to existing data that happened while the network connection was interrupted, an update conflict could occur. A delete conflict occurs when two transactions originate from different sites, with one transaction deleting a row and another transaction updating or deleting the same row, because in this case the row does not exist to be either updated or deleted [15].

2.3 Data Consistency Approaches

Two major approaches have been used by researchers in constraining and controlling the flow of data in collaborative software. The first approach, the *optimistic* approach, is an approach where no restriction or locking of the data takes place in the system, thus resulting in a less restricted feel by users, but also creating the potential for more challenges in data consistency management. Such an approach has been effectively used in multi-user document editors such as Google Docs, CodoxWord, and Microsoft SharePoint. To the author's knowledge a truly optimistic approach has never been successfully implemented in a CAx system, likely due to the complexity of the data relationships within a CAx system. Augustina et al. explain how such an approach is not possible using traditional collaborative data transfer technology such as operational transformations because many data objects could have both multiple parents and multiple children, thus not allowing the use of traditional operational transform technology to manage data as is done in the multi-use text editors [1].

The second—and most common—approach for data consistency control in multi-user CAx systems is the realistic (sometimes knows as pessimistic) approach, which involves locking parts of the entire CAx model to prevent problems with data consistency. Advantages of realistic approaches to data consistency in CAx systems are easy implementation (as no special transformations of data are required as data is transferred) and the effective prevention of both update and delete conflicts. The strictest realistic approaches fully lock the collaborative design session allowing only one user to edit the model at a time, and the remaining collaborative users to have a view-only mode of the model. More segmented approaches have been implemented in some of the Microsoft Office 2010 collaborative document editing software, allowing certain segments of the document to be locked for editing, while allowing the rest of the document to be open for editing from other collaborators. Changes to the document are not propagated automatically, rather are broadcast to other users when the document is saved and when the user chooses to broadcast the changes.

2.4 Data Consistency Implementations

Several methods for managing data consistency in a collaborative setting have already been proposed in current research. Jing, et al. use a local locking mechanism, based on operational transforms, which helps avoid conflicts by distributing locks to local features in a model in a replicated collaborative CAx system [7]. Bu, et al. use a semantic locking method to prevent semantic violation. The locks are classified into region lock and object lock for resolving violations at different levels of detail. User negotiation and versioning rules are used to resolve conflicts among the collaborating users [3]. Chen, et al. apply three coordination rules embedded in e-Assembly to satisfy collaborative assembly constraints in a client-server 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) [6]. Lin, et al. explains 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 locking, 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 multiple users manipulating the same object would confuse the constraint relations [9]. Marshall recommends a task-based method that will allow hierarchal administrative control over nearly the entire design process. This system is very intricate and requires significant planning and setup to be effective [11]. Cera, et al. recommends a role-based system that only provides necessary geometry of a collaborative CAD model based on a

user's role. Additionally details of the model can be obscured to protect proprietary information in the model [4]. A more realistic method proposed uses a token-based session, which only allows one user to edit the part at a time [5],[8]. Additionally a softer "traffic light" method which visually warns users when it is ok to edit model features was proposed by Bidarra et al. to lift some strict restrictions imposed by the more rigid token-based systems [2].

3 DATA CONSISTENCY METHODS

To facilitate multiple CAx users' within a collaborative system and environment (and ease the fear of data loss due to system defect), a new method to protect and coordinate the transfer of data as well as to avoid data conflict and data loss has been developed.

This section presents the general method for the improvement of data consistency in a collaborative CAx system. Note that while the method described is largely in the context of a CAD system the method described is intended to be universally applicable to any collaborative CAx system. For example, in the context of a collaborative CAD assembly, rather than CAD part modeling, each part in the assembly could be treated as a feature and this method could be an effective data consistency system for a CAD assembly system as well. The method steps will be demonstrated with many examples and references in the context of a CAD system; refer to section four for implementation details.

3.1 User Interface

3.1.1 Feature List & Feature Reservations

The method for maintaining data consistency uses an information window displaying a representation of a part's feature list which provides several options that users in a collaborative design session can manage when and how the different features in a model can be edited. The appearance of this collaborative feature list form could vary depending on the implementation of the collaborative system (fully integrated vs. transparent adaptation) and the style/design of the software hosting the implementation, but the basic functionality should remain the same. Fig. 1 is a mock-up of a collaborative feature list (see Fig. 11 for the implementation of this form).

The most basic functionality of the feature list is to *reserve* or *lock* a feature. A user can change the 3-state toggle to reserve, lock, or release the selected feature by changing a toggle button or select menu near the feature name in the feature list. When the state of the feature is changed, it is then propagated to all other users currently in the collaborative design session. This switch or button should visually reflect the changed state of the feature (through color, font-style, etc.) and also provide additional information such as the specific collaborating user who has locked or reserved the feature. The mock-up example feature list in Fig. 1 shows this toggle button as white when a feature is released, blue when a feature is reserved, and a lock icon when the feature is locked. The actual implementation of the state toggle should match the style of the CAx application into which it is integrated.

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 with a model while preventing update and delete conflicts.

A locked feature is very similar to a reserved feature; the difference is that when a feature is locked, the user who "owns" the lock must first release the lock before that feature can be reserved, constrained, or locked by other users. Locking allows tighter control over features, such as an interface feature with another model. The user interface could additionally implement convenience features to allow users to request a lock be released via chat message, email, icon transmit state, etc.

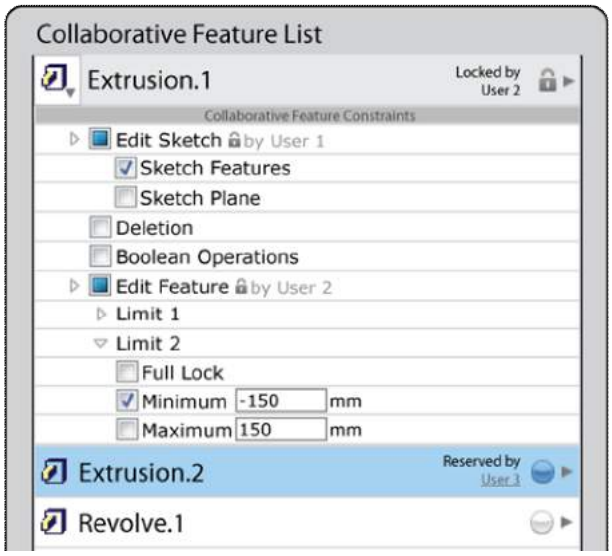

Fig. 1: Mock-up of a collaborative feature list form.

Depending on the current feature state (released/reserved/locked), feature items in the collaborative feature list form should be highlighted with a certain color. The mock-up feature list in Fig. 1 displays released features as white, reserved features as blue, and locked features as gray, although any color scheme could conceivably be used. The color will visually convey the state of the feature to the user. A unique color for each user in the collaborative design session could also be used to convey which user has reserved or locked a given feature. Additionally, the same color scheme could be applied to the rendered features inside the CAx system visual editor to help users see which features are edit-able (Fig. 2). Decals/icons (such as a lock symbol) could also be applied to features in a CAx system's visual editor to further convey its state to the user.

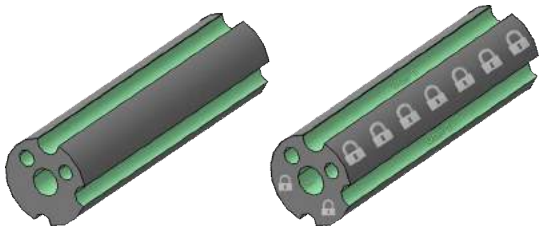

Fig. 2: Colors and/or decals help users visualize the edit-ability of features.

Additional user interface options could be used to toggle the state of features in the CAx visual editor as well. One method would be a right-click context menu as shown in Fig. 3. Another method would use a triple-click of the mouse on that feature, allowing users to quickly reserve and lock features while keeping their cursor in the visual editor.

The ideal implementation of the collaborative feature list functionality would be to implement it directly into a CAx system's existing feature list using the same look and feel of that CAx system. Other menus or buttons should be implemented similarly. This will make users familiar with that particular CAx software more comfortable with the collaborative features, and make it easier for them to adopt the new functionality into their existing workflow.

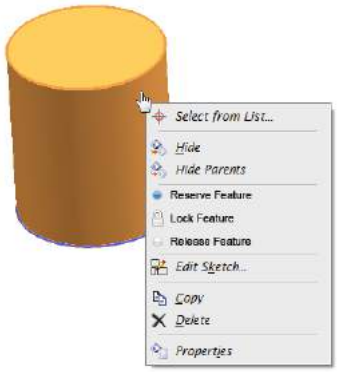

Fig. 3: Mock-up of a right-click context menu being used to toggle the state of a feature.

This feature-wise locking approach provides additional capability for CAx model decomposition. In addition to locking/reserving features individually, they could also be locked or reserved in groups. If features are related in a tree-like structure, an entire branch of features could be locked by simply locking the parent-feature in the tree. As new features are added to a model, they could inherit any lock/reservation properties from their parent feature. This functionality provides a way to effectively partition a CAx model, having individual users working on groups of features in a model. It would also include less overhead than a spatial model decomposition method that decomposes a model using planes and surfaces. Fig. 4 shows a mockup of how features could be locked in a tree-like structure. Feature 3 and all of its children are locked by User 1, and Feature 7 and all of its children are locked by User 2.

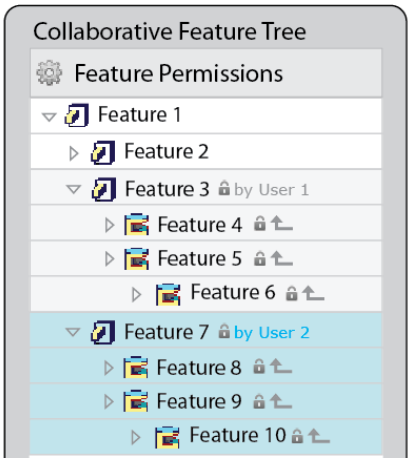

Fig. 4: Mock-up of features locked by a tree branch.

3.1.2 Collaborative Feature Constraints

The advanced functionality of the collaborative feature list allows collaborative feature constraints to be added to the model such as limiting the range in which a CAD extrusion can be defined. Collaborative feature constraints allow exact control over every aspect of the feature, which adds more flexibility to control the manipulation of the feature (compared to simply locking the feature). These constraints could be accessed from a pop-up panel accessed by users from the collaborative feature list using a single click. If a collaborative feature constraint is defined, the name of the user who created the constraint is displayed next to the constraint. Similar to locking, a convenience method

could be implemented for a user to contact the "owner" of the constraint in the event that it needs to be modified or deleted.

Collaborative feature constraints will be specific to the type of feature that they govern and are based on the different data parameters that define that specific feature. For example, in a CAD system an extrusion is defined by a sketch (with its respective features [lines, arcs, etc.] and plane) as well as a starting limit, and an ending limit. Constraints for a CAD extrusion could constrain the "edit-ability" of the extrusion's sketch features or sketch plane, provide for feature deletion or perform Boolean operations on the feature, as well as constrain the starting and ending limits to a certain value, or a certain range of values. Fig. 1 shows collaborative constraints set for a CAD extrusion so that Extrusion.1's sketch feature cannot be edited, and that Limit 2 of the extrusion cannot be less than -150mm (also see Tab. 1). Collaborative feature constraint definitions will be unique for each different type of feature in a given CAx system. The best set of definitions will have to be determined depending on the implementation and the types of features in that CAx system.

3.1.3 User Awareness

Awareness of the other users concurrently editing a document would motivate a method to list the other users participating in the current collaborative design session. Icons next to each user's name could indicate if they are active, inactive, busy, etc. similar to popular internet chat applications. Additionally, a chat bubble icon next to their name would allow users to initiate chat sessions between users as they are working together in the collaborative design session. This section would also be a place that other advanced communication features could be implemented such as audio/video chatting, or automatic translation of a text chat in a multi-lingual collaborative design session [16]. Clicking on a user's name could additionally reveal user design related information such as their expertise, location, position, company, etc.

3.1.4 Administrative Interface & Collaborative User Constraints

The administrative interface allows deeper control over the flow of CAx data in a collaborative system, allowing administrative control over every aspect of the collaborative model. The administrative interface allows an administrator to control visibility and edit-ability of CAx data by user, by feature, by groupings of features (such as a part), by groupings of groupings of features (such as assemblies and subassemblies), and any other combination of these groupings.

The administrative collaborative form mock-up in Fig. 5 allows an administrator to toggle any number of different constraints for all users in the system (see Fig. 21 for an image of the implemented form). The functionality of collaborative user constraints will vary depending on the implementation, but recommended constraints should include model views, feature reservation (and hence edit features), feature locking, and creation of collaborative feature constraints. In Fig. 5, User 1 can view any CAD Model in the collaborative system (shown by the highlighted eye icon), as well as make reservations (shown by the highlighted blue dot icon). This user, however, is not allowed to lock any features (shown by the grayed-out lock icon), and is not allowed to create collaborative constraints (shown by the grayed-out constraint icon).

When a specific user's name is selected (highlighted in yellow) the permissions for this user are displayed and can be set per assembly, per sub-assembly, and per feature. When one of these options is toggled, the option could also be toggled for all children. For example, if locking for a certain user is enabled at an assembly level, then all of the features in all child sub-assemblies and parts could also have the locking enabled. The hierarchy (whether the global or user-specific constraints are enforced over one another) of these collaborative user constraints could be modified to fit the needs of the specific application.

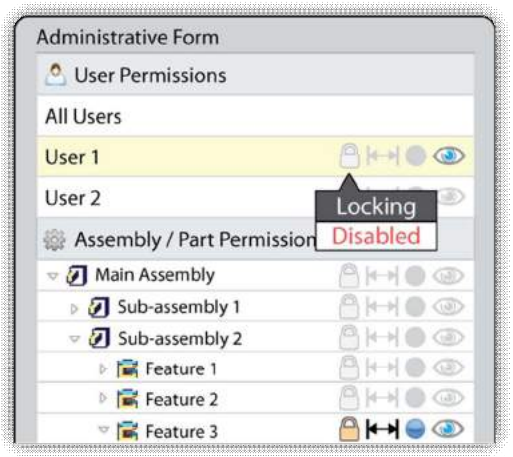

Fig. 5: Mock-up of the administrative form.

When the *All Users* option is selected, then the global collaborative constraints menu becomes active for all features (see Fig. 5). In addition to creating global permissions for each assembly/sub-assembly/part/feature, this will allow the administrator to create or arbitrate any collaborative constraints that need modification. Global permissions for each assembly, sub-assembly, part, and feature will override the user-specific features. The hierarchy of the global constraints could be modified to fit the needs of the specific application. Also, this form could potentially be used to manage task-based constraints or spatial model decomposition as suggested by Marshall [11] if such were also available in the same implementation.

The administrative interface should be integrated directly into a CAx system using the same look and feel as used for other administrative menus and settings windows. A consistent look and feel will make users feel more comfortable with the interface functionality.

3.2 Enforcing Reservations, Locks, and Constraints

The enforcement of reservations, locks, and collaborative constraints could be done in two different ways, depending on the implementation and API access for the given CAx system. For either method to be effective, it will be necessary to propagate changes in collaborative constraints/reservations/locks to each client as quickly as possible. Network latencies could limit the effectiveness of propagating these changes to the clients in a design session.

The best and most intuitive way to enforce these data consistency constraints would be to implement the enforcement directly into the visual editor of the CAx software. If a feature was reserved or locked, or a collaborative constraint was set, then the visual editor would prohibit the user from violating the feature locks or collaborative constraints. To implement such capabilities would require either modification of the source code of the given CAx system, or sufficient API access to prevent editing of features. This method would be able to directly communicate to users from the native CAx visual editor the current state of each feature in the model, making the system more intuitive for users and more usable. A mock-up integrating collaborative feature constraints directly into a CAD software package is shown in Fig. 6.

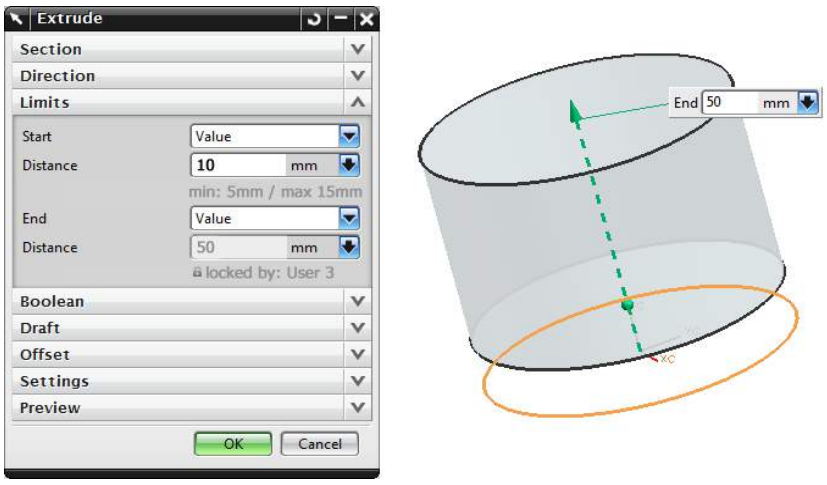

Fig. 6: Mock-up of collaborative feature constraint interface (Siemens NX 7).

If neither the software's source code nor sufficient API access is available to constrain users from editing the model from the standard CAx visual editor (as is the case in many commercial CAx software packages), then another approach could be used to enforce the locks and reservations; see Fig. 7. This approach works by checking the edit permissions of each modified feature before modifications are propagated to the database. If the feature is not reserved or locked by the user trying to modify the feature, or any collaborative constraints are violated then the local modifications are not saved into the database, and the current CAx data is reverted to the pre-modified state. In some cases, it may be necessary to check the permissions of the parent feature as well when creating constraint features, such as constraining the ability to perform Boolean operations on a CAD feature. To help users understand how this works, a series of informative warnings would have to be displayed to notify the user how the system works and when feature locks or constraints have been violated. The biggest drawback of this implementation is that it is not as intuitive for users because the CAx visual editor will allow them to edit the feature locally even if it is locked by another user. While effective technically, this could be considered a UI flaw. Another drawback is the increased amount of network usage and server resources needed to check constraints upon feature edits, although it should only consume a small amount of network and server resources as it would be a read-only query to a database. The most significant benefit of this implementation is that it is completely effective in blocking unauthorized modifications from being broadcast to the database, which is of the utmost importance to maintain data consistency in the model. Another practical benefit is that full API access to the CAx system UI is not needed and the method could be easily developed and implemented as a plugin into existing systems.

It is recommended that this enforcement method also be implemented underneath the first method described in this section as a failsafe to prevent modifications to locked/reserved/constrained features in the event that the collaborative constraint information is not immediately broadcast to all users. The process used to enforce collaborative constraints for feature modification is shown in Fig. 7.

Enforcing collaborative user constraints not directly linked to feature modification can be done in a slightly different way. For example, to enforce a collaborative user constraint that prevents a certain user from viewing or opening a part or assembly, the process in Fig. 8 could be used. Additionally, for the process of enforcing collaborative user constraints that prevent a user from reserving or locking a feature, the process in Fig. 9 could be used. Finally, the process of allowing users to create collaborative users constraints can be found in Fig. 10.

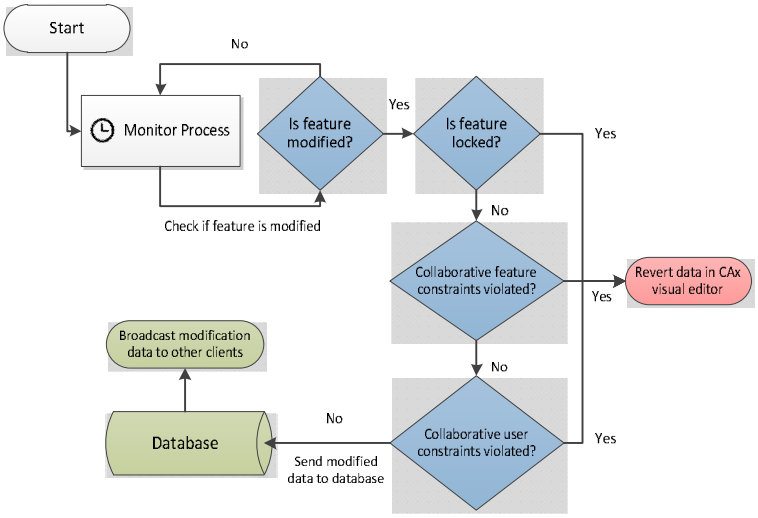

Fig. 7: Feature modification constraint enforcement.

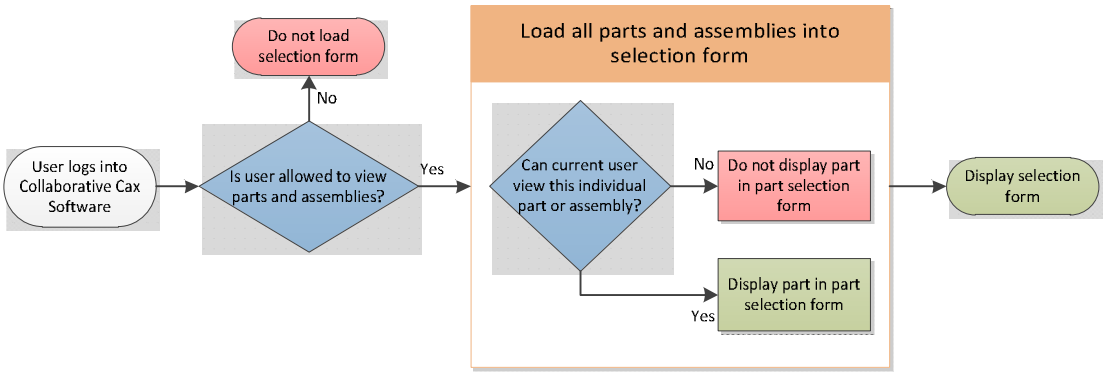

Fig. 8: Enforcement of collaborative model viewing permissions.

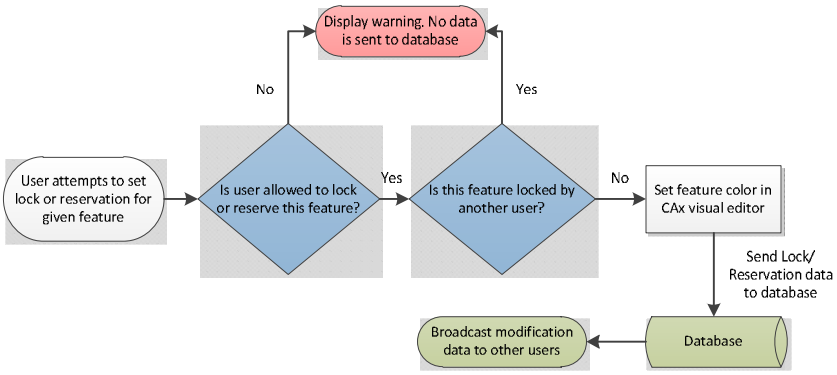

Fig. 9: Enforcing collaborative constraints for locking/reserving features.

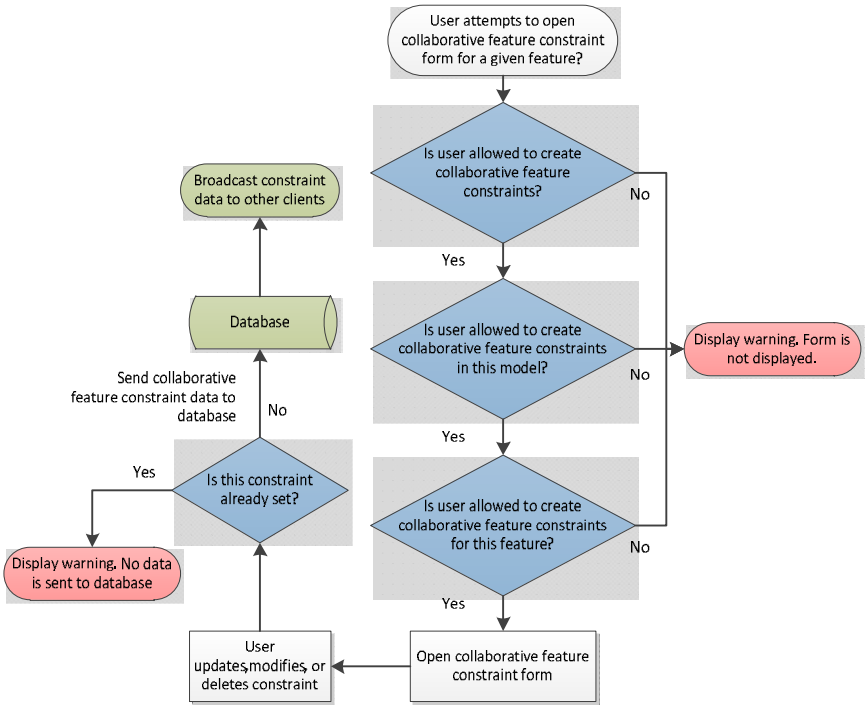

Fig. 10: Enforcement of collaborative feature constraint creation.

4 IMPLEMENTATION & EXAMPLE

4.1 Implementation

4.1.1 Collaborative Feature List

The implementation of the proposed method was integrated into the CATIA Connect software plugin developed at Brigham Young University. CATIA Connect is a transparent adaptation multi-user CAD plugin that synchronizes CAD features between multiple clients. It was built using a client-server architecture allowing several users to participate in a multiple user design session simultaneously.

The implemented collaborative feature list (see Fig. 11) has been built to mirror the native CATIA feature list (a mock-up of this form is also available in Fig. 1). As new features are added locally or on remote sessions, they will be created in the CATIA visual editor and listed in the collaborative feature list. From this list, the lock/reserve/release functionality is very straight-forward to operate. A three-state checkbox can be toggled easily with one click. As the checkbox is clicked, the *state* column will update to display “released”, “reserved”, or “locked”, and the *by* column will display the username of the user who has locked or reserved the feature (it will be left blank if the feature is released).

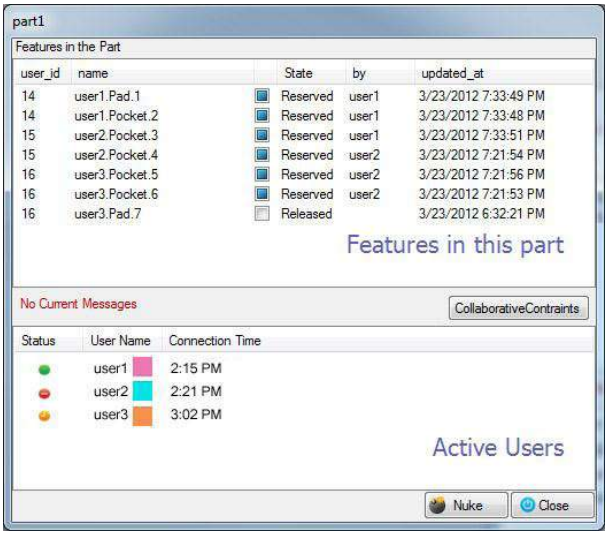

Fig. 11: Implemented feature list form.

As the state of a feature is changed, the color of the feature is updated in the CATIA visual editor to each user's color (see Fig. 12) and changes to the database are propagated. Remote clients will receive an update of the state change each time the update timer process in CATIA Connect retrieves updates.

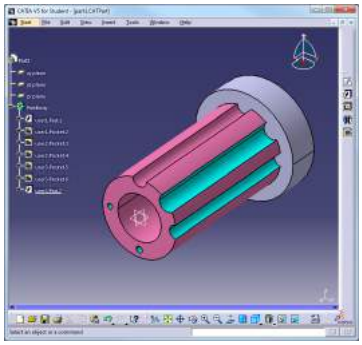

Fig. 12: Colored CATIA CAD model representing the state of its features.

4.1.2 Collaborative Feature Constraints

The collaborative constraints menu can be opened from the feature-tree menu by double-clicking on one of the features or by selecting the feature and then clicking the *Collaborative Constraints* button (Fig. 11). Once open, the collaborative constraints menu (Fig. 13) is populated with the available collaborative constraints for the given features. The collaborative constraints can be toggled by clicking the checkbox next to the label. The values for each constraint can also be edited directly here as well. The set of collaborative constraints is unique for each feature as each feature is defined using different parameters. Full collaborative constraint definitions for a CATIA pad (or extrusion) are found in Tab. 1.

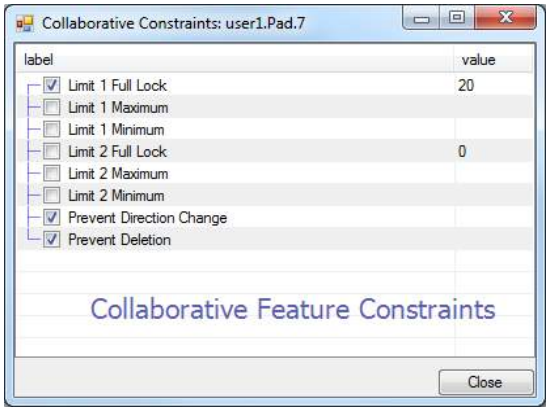

Fig. 13: Collaborative constraint form for a CATIA pad.

Collaborative Feature Constraint Definitions for a CATIA Pad	
Limit 1 Full Lock	Completely locks the ability to change the limit 1 for the extrusion
Limit 1 Maximum	Limits the ability to make the limit 1 of the extrusion greater than specified value
Limit 1 Minimum	Limits the ability to make the limit 1 of the extrusion less than specified value
Limit 2 Full Lock	Completely locks the ability to change the limit 2 for the extrusion
Limit 2 Maximum	Limits the ability to make the limit 2 of the extrusion greater than specified value
Limit 2 Minimum	Limits the ability to make the limit 2 of the extrusion less than specified value
Direction Lock	Prevents the ability to change the direction of the pad

Tab. 1: Collaborative constraint definition for a CATIA pad.

4.2 Usage Examples

The example in this section illustrates how the method is effective and how it could be used by designers to enhance collaboration. The examples are all given in the context of a CAD system. This example will demonstrate the functionality and the utility of the methods:

- Feature reservation/locking
- Collaborative feature constraints
  - Limit locking
  - Limit tolerances
  - Deletion prevention
- Collaborative user constraints
  - Part viewing
  - Feature reservation/locking
  - Permission to create collaborative feature constraints

This example starts with the objective of three designers (lead designer, structural designer, and engine cooling designer) modeling the engine block of Fig. 14 using CATIA Connect.

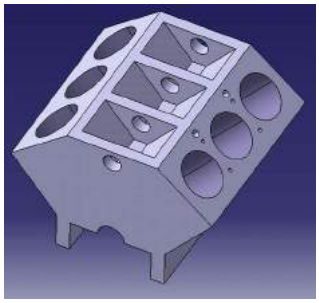

Fig. 14: Engine block to design.

For the first modeling operation, the lead designer creates the main engine block in Fig. 15(a). This user places a *deletion lock* collaborative feature constraint, to prevent accidental deletion of this key feature by other users. Immediately after the block is created, the lead designer and the piston designer are able to simultaneously create several negative extrusions to continue the shaping of the engine block in Fig. 15(b)(c). The lead designer then locks the negative extrusions that will be used for the piston cylinders. These extrusions then immediately turn the lead designer's color, which is blue, communicating to the other users that this particular feature is locked by, and that it cannot and should not be edited (Fig. 15(d)).

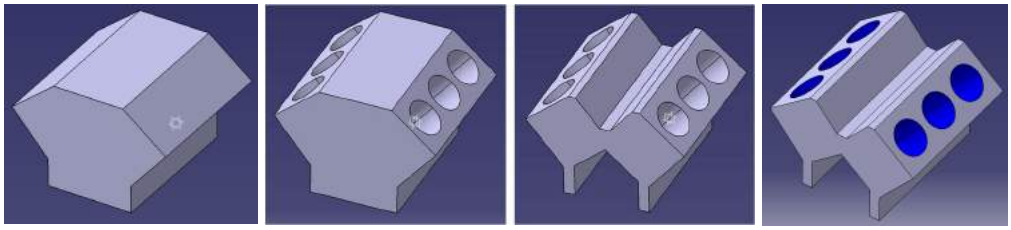

Fig. 15: (a) Main engine block (b) Piston cylinders (c) Shaping extrusions (d) Piston cylinders locked.

As soon as the cylinders for the pistons are created, the engine cooling designer is able to begin the modeling of additional negative extrusions around the piston cylinders to reduce weight and dissipate heat (Fig. 17). After creating this feature, this designer needs to place three collaborative feature constraints on it. The engine cooling designer must ask the lead designer (who is the administrator) to enable the collaborative user constraint to allows this. The lead designer enables this collaborative user constraint using the Administrative Form. After having this collaborative user constraint enabled, the cooling designer is able to create the following collaborative feature constraints: Limit 1 full lock, Limit 2 maximum of 3.5 inches, Limit 2 minimum of 2 inches (Fig. 16).

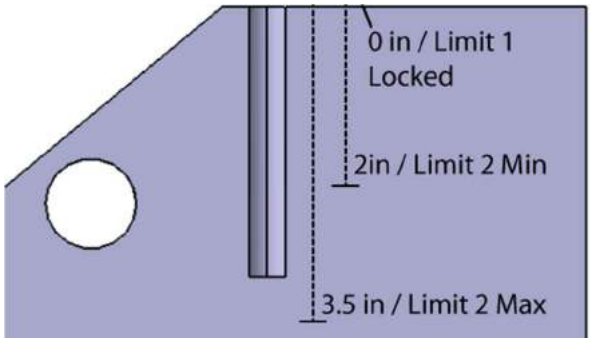

Fig. 16: Collaborative Feature Constraints Visualized.

These collaborative feature constraints are created to keep the first limit of the extrusion flush with the angled surface of the engine block, and keep the depth of the extrusion within an allowable tolerance. These collaborative feature constraints will allow this feature to be adjusted according to heat-transfer and stress optimization analysis results, while preserving other geometry that should not be modified.

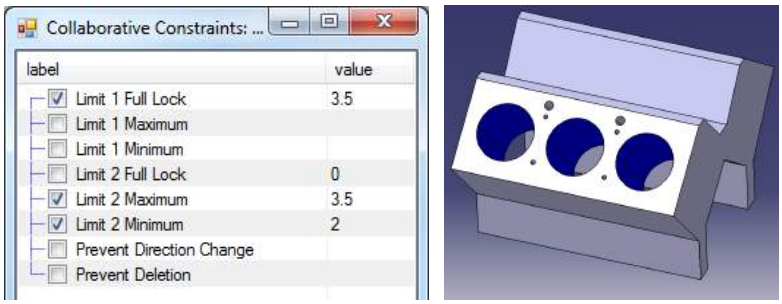

Fig. 17: Creating collaborative feature constraints for negative extrusions for heat dissipation.

The modeling of the engine block continues as the structural designer simultaneously creates 4 ribs which are formed through the entire block (Fig. 18). While this designer is creating the ribs, the cooling designer makes a small mistake, and attempts to edit the piston cylinders. Luckily this feature was previously locked by the lead designer so the change that the cooling designer made was not allowed by CATIA Connect, preserving the geometry of that feature. A warning was also displayed to notify this designer that they do not have permission to edit this feature.

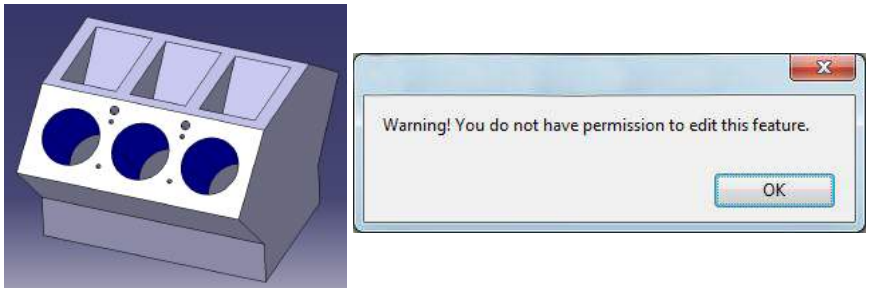

Fig. 18: Creation of rib features and edit permissions warning dialog box.

Once the ribs are created, the lead designer creates additional negative extrusions in the engine block for the camshaft and the crankshaft (Fig. 19).

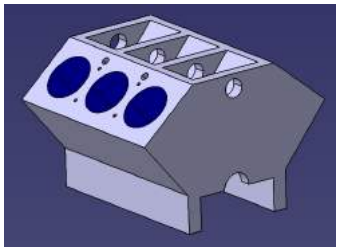

Fig. 19: Negative extrusions for camshaft and crankshaft.

Next, the model is passed off to the analysis group, and shortly after analysis results are received via email sent to all three designers at the same time. These results specify the optimal value for the outer piston hole extrusion limit 2 is 2.8 inches. As these results are received, all three users then update the

model according to these results, and all attempt to reserve the feature to make the appropriate modifications at the exact same time. As each user tries to make the reservation, the lead designer is able to secure the reservation, and ultimately is the user able to make the modification to this feature. As the piston and cooling designers see that the feature is reserved by the lead designer, they immediately know that the lead designer will be making the appropriate changes to that feature. This also spurs a short chat conversation where the lead designer explains to the other two users that the appropriate changes to this feature have been made, and clears up any confusion about who will make the design changes.

After this situation is resolved, the final features are created to complete the engine block design. The larger inner holes for the crankshaft are created by structural designer (Fig. 20(a)), while the cooling designer is able to concurrently create the final rectangular slot to finish the geometry of the engine block (Fig. 20(b)).

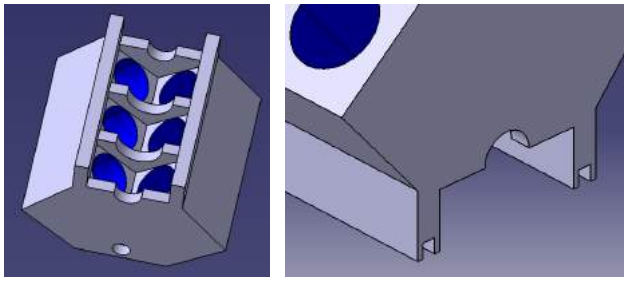

Fig. 20: (a) Additional crankshaft geometry (b) Slots on the lower ribs.

Now that the geometry for the engine block is complete, a design review will take place. For this design review, the geometry for this part will be shared with several reviewers, some which are remotely connected. To allow the reviewers access to the part, User 1, who is also an administrator, enables only the *can_view* collaborative user constraint for them (Fig. 21). Enabling only this constraint will protect the created feature data for this part, allowing reviewers to view the model, but not edit it.

As the design review takes place, two design changes are recommended by the reviewers. It is recommended that the slot on the bottom of the block is extended another two inches, and it is recommended that another similar slot be placed in the other side of the engine block. Still in the design review, the designers then discuss the changes and assign the structural designer to make the first change and the lead designer to make the second change. The structural designer reserves the slot feature—turning it green—then extends it two inches (Fig. 22). The second change is made simultaneously by the lead designer who creates the new slot feature (Fig. 22). These changes are able to be made quickly during the design review and the reviewers are able to approve of the design changes in real-time. The design review is and the design of the part is now complete (Fig. 14).

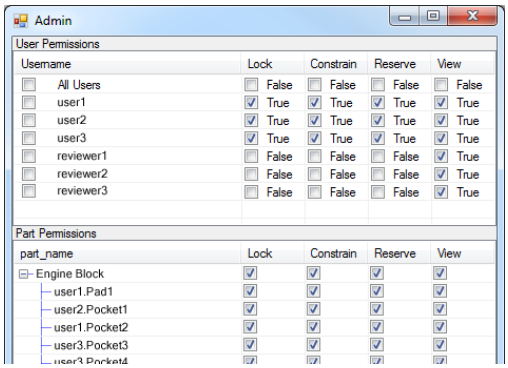

Fig. 21: Admin form showing collaborative user constraints for the engine block.

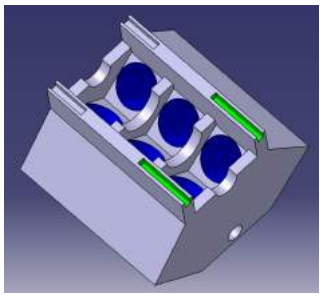

Fig. 22: Changes implemented from design review feedback.

5 CONCLUSION

We have presented a new method and interface for users in a multi-user CAx design environment to partition a CAx model and maintain data consistency by introducing new constraints for collaborative CAx design. The three-state selector to reserve/lock/release features will help users collaborate within shared design space. The visual awareness conveyed by the different coloring of features as they are locked or reserved will serve as an additional avoidance method for data conflicts, as it lets users know where in a CAx model other users are editing, and which features they are allowed to edit. The collaborative feature constraints developed will help protect models as the CAx data is shared with several different users. The administrative interface and collaborative user constraints will help a CAx design team define the roles of individuals on their team and protect their data from being changed inadvertently or by unauthorized individuals when it is shared with others. Overall, this data consistency method allows collaborative CAx software to be used more safely and effectively because it protects the data created by the software, controls how the data is used, who can modify it, and who can access it. The method outlined in this paper could be implemented in many different ways depending on the application, but the ideas will improve the usability and overall quality of collaborative CAx software, and ultimately prevent data conflict and protect data in a multi-user design session.

6 FUTURE WORK

While much has been done to show the effectiveness of this research, additional work is still being added and planned, i.e.:

- Integrating the method directly into a variety of CAx systems such as CAM, FEA, CAD drawings, and CAD assemblies, and analyzing the method's effectiveness in actual design situations.
- Expanding collaborative feature constraint definitions and looking at how collaborative feature constraints could work with other CAx systems.
- Performing further usability testing with different user interface configurations for the locking/reservation method, collaborative feature constraints, and administrative interface to further enhance the ease-of-use of implementations of this method.
- Investigating how feature reservation/locking could integrate with other CAx model decompositions (e.g. spatial decomposition) to create hybrid decomposition methods. This investigation could improve control over the creation of new features in a CAx model.
- Exploring more methods to promote multi-user awareness in a CAx model using labels, colors, etc.
- Investigating ways to improve extensibility to make future integrations and development of this system more scalable.

ACKNOWLEDGEMENT

The authors would like to thank the NSF for their funding and support of the Center for e-Design, BYU site, in which this paper and research were completed.

REFERENCES

[1] Agustina, A.; F. Liu; S. Xia; H. Shen; and C. Sun: CoMaya: incorporating advanced collaboration capabilities into 3d digital media design tools, in Proceedings of the 2008 ACM conference on Computer supported cooperative work, 2008, 5–8, http://dx.doi.org/10.1145/1460563.1460566.

[2] Bidarra, R.; E. van den Berg; and W. F. Bronsvoort: A Collaborative Feature Modeling System, Journal of Computing and Information Science in Engineering, 2(3), 2002, 192, http://dx.doi.org/10.1115/1.1521435.

[3] Bu, J.; B. Jiang; and C. Chen: Maintaining semantic consistency in real-time collaborative graphics editing systems, IJCSNS, 6(4), 2006, 57.

[4] Cera, C.; I. Braude; I. Comer; T. Kim; J. Han; and W. Regli: Hierarchical Role-Based Viewing for Secure Collaborative CAD, in Proceedings of the 2003 ASME International Design Engineering Technical Conferences & The Computer and Information in Engineering Conference (DETC/CIE2003), 2003, 10.

[5] Chan, S.; M. Wong; and V. Ng: Collaborative solid modeling on the WWW, Proceedings of the 1999 ACM symposium on Applied computing - SAC '99, 1999, 598–602, 10.1145/298151.298487.

[6] Chen, L; Song, Zhijie; Feng, L.: Internet-enabled real-time collaborative assembly modeling via an e-Assembly system: status and promise, Computer-Aided Design, 36(9), 2004, 835–847, http://dx.doi.org/10.1016/j.cad.2003.09.010.

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

[8] Li, W. D.; J. Y. H. Fuh; and Y. S. Wong: An Internet-enabled integrated system for co-design and concurrent engineering, Computers in Industry, 55(1), 2004, 87–103, http://dx.doi.org/10.1016/j.compind.2003.10.010.

[9] Lin, K.; D. Chen; C. Sun; and G. Dromey: Maintaining constraints in collaborative graphic systems: the CoGSE approach, in ECSCW 2005, (September), 2005, 185–204.

[10] von Lukas, U.: Collaborative Geometric Modelling using CORBA Services, OOGP'97, 7(2), 1997, 91.

[11] Marshall, F.: Model Decomposition and Constraints to Parametrically Partition Design Space In a Collaborative CAx Environment, 2011.

[12] Qiang, L.; Y. F. Zhang; and a. Y. C. Nee: A Distributive and Collaborative Concurrent Product Design System through the WWW/Internet, The International Journal of Advanced Manufacturing Technology, 17(5), 2001, 315–322, http://dx.doi.org/10.1007/s001700170165.

[13] Ramani, K.; A. Agrawal; M. Babu; and C. Hoffmann: CADDAC: Multi-Client Collaborative Shape Design System with Server-based Geometry Kernel, Journal of Computing and Information Science in Engineering, 3(2), 2003, 170, http://dx.doi.org/10.1115/1.1582882.

[14] Ryskamp, J. D.: Developing Intelligent Engineering Collaboration Tools Through the use of Design Rationale, Brigham Young University, 2010.

[15] Urbano, R.: Oracle Database Advance Replication, 2007.

[16] Xu, Y.: A Flexible Context Architecture for a Multi-User GUI, 2010.

[17] Zhou, X. and J. Li: A Web-based synchronized collaborative solid modeling system, Chinese Journal of Computer Integrated Manufacturing Systems, 2003, 960–965.

[18] Codoxware: Connecting people and documents, 2011. [Online]. Available: http://www.codoxware.com/. [Accessed: 04-Jan-2012].