Title:
Automated Conflict Avoidance in Multi-user CAD

Authors:
Ammon I. Hepworth, ammon.hepworth@byu.edu, Brigham Young University
Kevin Tew, kevin_tew@byu.edu, Brigham Young University
Thomas Nysetvold, tom.nysetvold@gmail.com, Brigham Young University
Mark Bennett, marktb1@gmail.com, Brigham Young University
C. Greg Jensen, cjensen@byu.edu, Brigham Young University

Keywords:
Collaborative design, concurrent engineering, multi-user CAD, CAE

DOI: 10.14733/cadconfP.2014.1-3

Introduction:
Today's commercial Computer Aided Design (CAD) systems are single user modeling and design environments. The National Science Foundation (NSF) Center for e-Design, Brigham Young University (BYU) site is currently developing multi-user CAD tools which enable teams of users to simultaneously create, modify and view the same CAD part. This effort leverages commercial CAD system APIs to build plug-ins which extend existing CAD tool functionality to become multi-user. This allows teams of users to concurrently contribute to the design of a part in real time, enhancing collaboration and enabling a parallel work environment within the CAD system [1, 3-4].

Interferences that occur when multiple users edit the same part or assembly are one of the central problems encountered in the development of simultaneous multi-user collaborative CAD. Varying simultaneous edits of the same or dependent geometric parameters may cause conflicts to arise within the model. When various users simultaneously input different values for the same feature, a feature/self conflict occurs and the distributed multi-user CAD system becomes inconsistent. For example, simultaneous editing the same height value, without conflict management, can result in a value of 10 mm on one user's workstation and a conflicting value of 2 mm on a second user's workstation. A conflict management system must be in place to ensure data consistency between distributed users [2].

The current literature describes two major approaches in overcoming the data conflict conundrum: optimistic and pessimistic. The optimistic approach assumes that conflicts are infrequent and are resolved after they occur. The pessimistic approach uses techniques to block concurrent access to certain data so that data stays consistent between users [3, 5]. This paper presents a hybrid approach between the extremes of purely optimistic or pessimistic conflict management. To avoid conflicts that would require manual resolution, the hybrid approach automatically places restrictions on model access (i.e. feature locks). It communicates user's design intent and avoids potential modeling conflicts through the use of intelligent visual cues and selection limitations. While this approach applies some restrictions and warnings to avoid manual merging of conflicts, they are set to a minimum to preserve an agile and uninterrupted multi-user experience.

Main Idea:
In order to prevent simultaneous edits of a single feature by multiple users, we propose a method for automatic feature reservation and blocking. If a user is editing a feature, this method blocks any other user from editing that feature until the first user finishes his edit. This reservation is communicated by a user sending a message telling the server that the feature is blocked and the server sending a message to all the clients telling them to block the feature. This block makes it so that other users are

unable to edit the feature and creates a visual indicator (i.e. color) showing them that it is unavailable. Once the feature edit is complete, a message is sent to the server reporting completion of the edit. The server then sends a message to all clients telling them to remove the feature block. In this way, other users are unable to edit the feature for the duration of a user edit. This is what we call the *client blocking method*.

The client blocking method alone breaks down if multiple users attempt to edit the feature at nearly the same time. Within the time it takes for the client to tell the server to block the feature and for the server to tell all the clients that the feature is blocked, a second client could attempt to make an edit. This scenario becomes more likely with higher network latency and as the quantity of concurrent users increases. Additional logic is added to the client blocking method so that multiple simultaneous edits of a feature remain consistent, even if they edit it at approximately the same time. This logic, the *server reservation method*, asynchronously reserves features on the server. An asynchronous approach is used so that a user does not need to wait for a response from the server to begin editing a feature.

This method functions as follows: A user attempts to make an edit on a feature that is not yet blocked on the client. A message is sent to the server requesting reservation of that feature. If the feature is not reserved, the server will automatically reserve that feature. Once the edit is complete, a message is sent to the server to cancel the feature reservation. If a client requests reservation on a feature that is already reserved on the server it means that another user is currently editing that feature and the requesting client has not yet received the blocking message. The server will ignore this request. When the requester receives the blocking message he will be ejected from editing the feature.

The server reservation method provides that the first client to have their request received by the server will be the one authorized to edit a feature. This is because the feature becomes reserved once the message is received, assuming it wasn't reserved previously by another client. Data conflict between a feature and itself does not occur because all other users without the reservation will be rejected from editing the reserved feature.

The combination of the client blocking method with the server reservation method provides the benefits of both approaches. The client blocking method provides that most clients will not even attempt to edit the feature because it is blocked for them both visually and interactively. The server reservation method provides for a safety net in the slight chance that multiple users edit a feature at approximately the same time. These methods provide that only one user is allowed to edit a feature at a time, thus avoiding data consistency problems for a feature with itself.

Besides the ability to reserve and remove reservation of a feature automatically, it is important to have the ability to remove a reservation on demand if a user is taking an inordinate amount of time editing a feature. In order to maximize modeling agility and minimize modeling interruption, a method is presented for on demand reservation removal. All users are allowed to request reservation removal on any feature that is currently reserved. When a user reservation removal request is initiated, a message is sent to the owner of the feature reservation notifying them that another user would like to remove the feature reservation. The owner has the option to accept or reject the request. If the request is rejected, the reservation remains with the owner and a message is sent to the requester stating that the request is denied. If the request is accepted, the owner is automatically ejected from the edit dialog and the reservation is removed. Alternatively, the owner may choose to ignore the message and continue his edit but is given a limited amount of time to respond to the request. If the owner fails to respond to a request within the allotted time, he is automatically ejected from the edit dialog and the reservation is removed.

The client blocking, server reservation and on-demand removal methods have been implemented into a multi-user CAD system being developed at BYU called NXConnect. This functionality was tested and functions as specified. Even with artificially high network latency, users were unable to cause conflicts by simultaneously attempting to edit the same feature. The results from these tests show that these methods prevent the data inconsistency that results from feature/self conflicts in a multi-user CAD system.

Conclusions:

The implementation of these methods validates the ability and functionality of automated feature reservation to prevent conflicts, thus helping to enable an agile collaborative environment for concurrent engineering. This approach has less overhead than other approaches because users do not

need to manually manage or merge conflicts, as is the case with other approaches. The user is not required to intelligently manage conflicts on their own (potentially allowing inconsistencies); rather, the CAD system handles them automatically with limited user interruption. Reservation information is communicated to users via non-intrusive visual cues. Additionally, users are not constrained by artificial boundaries that may prevent them from contributing where they are needed. This approach preserves the agility of a truly parallel design workflow while still eliminating feature/self conflicts.

References:

[1] Cannon, L.; Nysetvold, T.; Phelps, G.; Winn, J.; Jensen C. G.: How Can NX Advanced Simulation Support Multi-User Design?, Computer-Aided Design & Applications, PACE Vol. 2, 2012, 21-32.

[2] Hepworth, A. I.; Tew, K.; Nysetvold, T.; Bennett, M.; Jensen, C. G.; Automated Conflict Avoidance in Multi-User CAD, Computer-Aided Design & Applications, 11(2), 2014, 141-152. http://dx.doi.org/10.1080/16864360.2014.846070

[3] Moncur, R.; Jensen, C.; Teng, C.; Red, E.: Data Consistency and Conflict Avoidance in a Multi-User CAx Environment, Computer-Aided Design & Applications, 10(5), 2013, 727-744. http://dx.doi.org/10.3722/cadaps.2013.727-744

[4] Red, E.; Jensen, C.; French, D.; Weerakoon, P.: Multi-User Architectures for Computer-Aided Engineering Collaboration, International Conference on Concurrent Enterprising, 2011.

[5] Saito, Y.; Shapiro, M.: Optimistic Replication, ACM Computing Surveys, 37 (1), 2005, 42-81. http://dx.doi.org/10.1145/1057977.1057980