2014-08-01

Conflict Management and Model Consistency in Multi-user CAD

Ammon Ikaika Hepworth
Brigham Young University - Provo

Conflict Management and Model Consistency in Multi-User CAD

Ammon I. Hepworth

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

C. Greg Jensen, Chair
W. Edward Red
Kevin B. Tew
Joseph J. Ekstrom
Christopher A. Mattson

Department of Mechanical Engineering

Brigham Young University

August 2014

ABSTRACT

Conflict Management and Model Consistency
in Multi-User CAD

Ammon I. Hepworth
Department of Mechanical Engineering, BYU
Doctor of Philosophy

The NSF Center for e-Design, Brigham Young University (BYU) site has re-architected Computer Aided Design (CAD) tools, enabling multiple users to concurrently create, modify and view the same CAD part or assembly. This technology allows engineers, designers and manufacturing personnel to simultaneously contribute to the design of a part or assembly in real time, enabling parallel work environments within the CAD system. Multi-user systems are only as robust and efficient as their methods for managing conflicts and preserving model consistency. Conflicts occur in multi-user CAD when multiple users interoperate with the same or dependent geometry. Some conflicts can lead to model inconsistencies which means that each user's instance of the model are not identical. Other conflicts cause redundant work or waste in the design process. This dissertation presents methods to avoid and resolve conflicts which lead to model inconsistency and waste in the design process.

The automated feature reservation method is presented which prevents multiple users from simultaneously editing the same feature, thus avoiding conflicts. In addition, a method is also presented which ensures that copies of the model stay consistent between distributed CAD clients by enforcing modeling operations to occur in the same order on all the clients. In cases of conflict, the conflicting operations are preserved locally for manual resolution by the user. An efficient model consistency method is presented which provides consistent references to the topological entities in a CAD model, ensuring operations are applied consistently on all models. An integrated task management system is also presented which avoids conflicts related to varying user design intent. Implementations and results of each method are presented. Results show that the methods effectively manage conflicts and ensure model consistency, thus providing a solution for a robust multi-user CAD system.

Keywords: multi-user CAD, synchronous collaboration, data consistency, conflict management

ACKNOWLEDGMENTS

I would like to thank my wife, Monica who has been an incredible support through this effort. Thanks goes to my graduate committee for their support and guidance. The majority of the content in this dissertation comes from several articles in which I am lead author, but many students and faculty contributed. I would like to thank the many co-authors who contributed to the articles: Greg Jensen, Ed Red, Kevin Tew, Tom Nysetvold, Josh Bennett, Keith Halterman, Jared Yarn, Mark Bennett, Daniel Staves, Daniel Ricks, Logan Hill, Devin Shumway, Nathan Fronk, Mark Trent, Glen Phelps, Brett Stone, and Bryce DeFigueiredo. I also appreciate journals and conference which gave permission to include the content of the articles in this dissertation, namely: Computer-Aided Design and Applications, Journal of Computing and Information Science in Engineering, Concurrent Engineering: Research and Applications, and The International Conference on Collaboration Technologies and Systems. Special thanks to the industry sponsors participating in the Center for eDesign BYU Site, who funded this research: Boeing, Pratt and Whitney, Belcan, PCC Airfoils, Spirit AeroSystems and CD-adapco.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Atomic unit	The smallest unit which cannot be divided among more than one user for simultaneous contribution
Conflict avoidance	An approach which attempts to keep conflicts from occurring
Conflict management	An approach to manage conflicts which includes both conflict avoidance and resolution
Conflict resolution	An approach which attempts to sort out conflicts which are unavoidable
Feature	The CAD building block unit consisting of controlling parameters in feature-based CAD which are combined to create a parametric geometry model
Model consistency	The assurance that all replicated instances of the model are identical
Optimistic	An approach to conflict management which assumes conflicts are infrequent enough to be resolved after they occur
Pessimistic	An approach to conflict management which attempts to avoid all conflicts
Semantic conflict	Conflicts which occur when multiple users perform operations which violate the design intent of the model, leading to process waste
Syntactic conflict	Conflicts which cause inconsistency or an invalid state of the model, causing software bugs
Topological entity	An element of the boundary representation model which makes up the geometry of the CAD part, consisting of a face, edge or vertex

CHAPTER 1. INTRODUCTION

Conflict management and model consistency are not problems in today's commercial CAD systems because they are single user modeling and design environments. Conflicts occur in multi-user CAD when multiple users interoperate with the same or dependent geometry. Some conflicts can lead to model inconsistencies which means that each user's instance of the model are not identical. Other conflicts cause redundant work or waste in the design process. Simply stated, in single user environments conflicts and model inconsistency cannot occur because it is inherently a multi-user CAD issue.

The single user limitation in CAD is evidenced within any engineering firm, by the number of designers, modelers, engineers, etc. huddled around a single workstation where one person is directed to make needed changes during a critical moment in a design cycle. Or, watching as one individual spends days, weeks and months to build a complex model. This was not the case post-World War II, with teams of engineers working simultaneously on the same large J-size sheet of mylar or vellum, completing the drawings in a fraction of the time it would take a single engineer. However, today only one technician can work on a sheet in CAD drafting environment regardless of the virtual size of drawing sheet. The days of concurrent parallel workflows have given way to the single user serial CAD workflow. 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 allows teams of users to concurrently contribute to the design of a part in real time, assuming the conflict management and model consistency problems can be solved.

The multi-user CAD systems developed at BYU are plug-ins to existing commercial single user CAD systems. This is accomplished through the CAD systems' application programming interface (API) to extend the original single-user functionality to multi-user. A client-server (CS) architecture is implemented where each client has a replicated instance of the CAD part. As users

model, operational data is extracted from clients and sent through a centralized server to remote clients. Client models are updated as remote operations are received from the server. The operation data is saved in a database on the server for persistent storage (see Fig. 1.1). These systems allow for simultaneous modeling and real-time updates from several remote clients [1–14]. Until now, these systems have not had methods for conflict management and model consistency, making them unusable in a commercial setting.

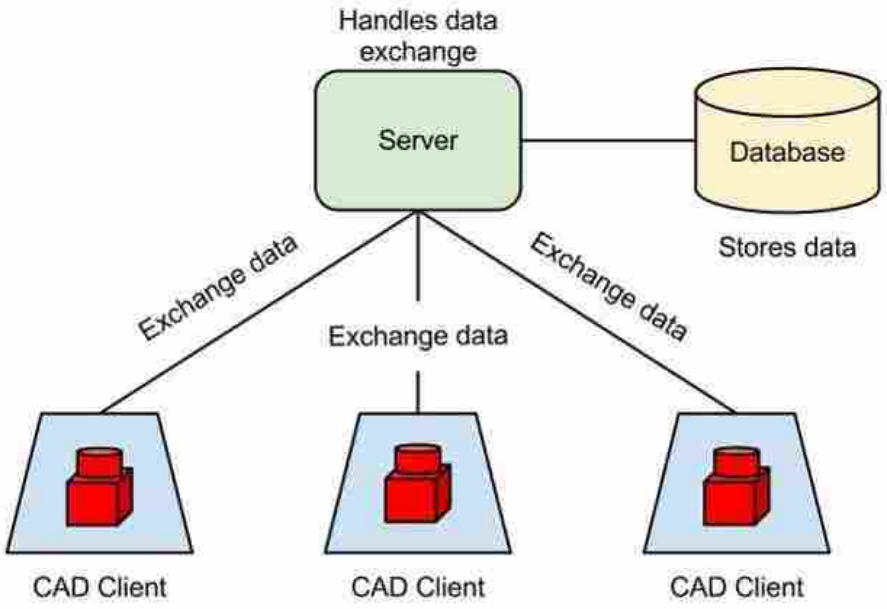


Figure 1.1: Basic replicated multi-user CAD architecture

Conflicts which occur when multiple users edit the same part or assembly are one of the central problems encountered in the development of simultaneous multi-user CAD. For example, simultaneous edits of the same or dependent feature may cause conflicts to arise within the model. Conflicts can lead to user confusion or inconsistencies between distributed user models. These conflicts must be appropriately managed for users to work effectively and to ensure all replicated copies of the model remain consistent on every client. This dissertation will develop methods and principles to manage conflicts by avoiding or resolving them. It will also present methods of model consistency which enforce a global order of operations and persistent naming of topology. The implementations and results of these methods show that they are effective in managing conflicts and providing model consistency in multi-user CAD systems.

1.1 Background

1.1.1 Multi-user CAD

In 2010 Red et al. reviewed the 50 relevant articles pertaining to the research in collaborative CAD that has been ongoing since the mid-90s [11]. This section summarizes the relevant multi-user CAD literature highlighted in [11] as well as after 2010. The research community has settled on two main architectures for collaborative CAD systems: centralized and replicated. A centralized architecture utilizes a central server which executes modeling operations received from remote clients. After performing operations, the server passes the resultant geometric data to clients for visualization and interaction. In this approach, the server performs the computation for all geometric operations and clients are used for visualization and user interaction. WebSPIFF [15], NetFeature [16], CADDAC [17], CollFeature [18] and WebCOSMOS [19,20] are examples of centralized collaborative CAD systems. A downside to this approach is that it results in large amounts of data sent over the network [21].

Conversely, replicated collaborative CAD systems have copies of the model data on each client which are required to stay in sync. Operation data is sent between clients via a network architecture. The respective clients perform the computation for all operations which occur. Examples of these systems include ARCADE [22], CSCW-FeatureM [23], CollIDE [24] and RCCS [21]. The main challenges with a replicated system are consistency and scalability. The advantage of this type of system is that they can be more easily integrated with existing commercial CAD systems. This is important because migrating to a new CAD system can be extremely expensive for engineering companies. A few replicated, multi-user CAD systems have been developed which are based on existing single user CAD systems. These include TOBACO [25], CollabCAD [26], Cocad [27] and those developed at BYU's NSF Center for e-Design site: NXConnect, CATIAConnect and InventorConnect [3, 9, 11, 12]. This research applies directly to replicated multi-user CAD.

1.1.2 Existing Conflict Management Methods

The current literature describes two major approaches in overcoming conflicts: optimistic and pessimistic. The optimistic approach assumes that conflicts are infrequent and are resolved

only after they occur. It accepts data operations that are submitted asynchronously and uses algorithms to merge the data, detecting conflicts that occur [28]. Some examples of existing optimistic synchronous collaborative software systems are Google docs, CoWord and CoPowerPoint. Many of these systems utilize a technique called operational transform to keep data concurrent [29, 30]. Optimistic is more successful in these systems because of operational or localized independence of the operations, whereas in CAD an operation can break the child operational dependencies. The optimistic approach can cause additional work for the user when they are required to manually resolve conflicts. It can also require redundant work if one completed operation is rejected in favor of another. A completely optimistic approach for multi-user CAD is likely not optimal because it introduces extra work (waste) into the design process.

The pessimistic approach assumes conflicts will occur and uses techniques to block concurrent access to certain data so that data stays consistent between users [8, 28]. Pessimistic systems implement various levels of data blocking, ranging from locking small sections of the model, to a complete lock of the entire model allowing only a single user to edit that model at a time [8]. Pessimistic approaches can limit user agility (the ability of the user to respond to change) by restricting users from contributing where and when they are needed.

Several methods have been proposed for managing conflicts and model consistency in collaborative design environments. For organizational purposes, the conflict management methods are divided into the following categories: turn based, model decomposition, on demand locking, and rules based. They are listed roughly in order from the most pessimistic to most optimistic. This section is concluded with a summary which discusses these methods in relation to an agile and uninterrupted multi-user design experience within a parametric, feature-based CAD system, which is the focus of this research. This dissertation adds mostly to the category of rules-based methods, proposing the rules necessary for less restricted conflict management and model consistency. It also adds a method for model decomposition using an integrated task management system. In addition, it indirectly adds to the on demand locking category by proposing a method for on demand lock removal.

Turn Based

Chan presents methods using a token based system allowing only the user who holds the token to edit the part while the other users are just observers [31]. Li utilizes a similar method that uses a turn based system for specific user functionality (i.e. viewing, deleting and adding to the part) [32]. These methods ensure that users do not make conflicting changes to the model since only one user is allowed to edit the part at a time. However, it disallows a fully parallel design workflow, causing one user to wait while another is modeling.

Bidarra et al. created a collaborative design system called WebSpiff where they assume users will coordinate their operations in a collaborative environment over phone or chat. To assist in this effort they implemented a traffic light system which visually warns users when another user is performing an operation but does not strictly lock users from making changes. Essentially, this softly reserves the entire part, warning other users about making contributions while another user is performing an operation. Their paper only discusses one type of conflict, which is where a user tries to edit a feature that no longer exists and mention that a user is notified when this occurs [15]. This method has some advantages over Chan and Li's methods because there is not a strict part lock, but they do not present enough information to determine whether the conflict management methods are sufficient for complex feature-based CAD modeling. Turn based methods are too restrictive and no attempt will be made to add to this category.

Model Decomposition

Cera et al. developed methods to hide specific design data from certain users' view in a collaborative design environment based on each user's role. The user role controls the access of users to view certain geometry through the partitioning of 3D models. Within a partition, multi-resolution techniques are employed to obscure the geometry in that region so that it remains secure from users without the appropriate level of access [33]. Marshall presents a method of model decomposition within a collaborative CAD system that sets boundaries within a part before modeling is performed. Her method enables administrative controls to divide a model into regions or tasks. These divisions limit a user's access to other regions or tasks [7]. The pre-work required to decompose a model may take time away from the actual design work. The partitions of model

decomposition may also inhibit the flexibility of a parallel design workflow. However, the benefits gained from the improved organization of this approach may be worth the additional pre-work and added restrictions when creating complex models. This dissertation proposes an integrated task management system to allow users to decompose the model into tasks, adding a less restricted method to this category.

On Demand Locking

Bu, Jiang and others proposed an approach for object locking within a collaborative graphics design environment to avoid conflicts in user's design intent, also known as semantic preservation. Their methods of locking both regions and objects on demand, give the user an opportunity to attach design intent data or block users from making specific design changes within the region or object. They also presented a method to overcome conflicts in semantic locking operations. They successfully implemented these methods in a collaborative graphics design system called CoDesign [32, 34, 35]. Moncur also presented a method for on demand feature reservation within a commercial 3D CAD systems. His method allows users to reserve a feature or group of features for a specified duration of time so that other collaborative users have limited or no access to the feature(s) [8]. These methods allow for data consistency in a manual way by requiring users to intelligently reserve features as necessary. However, his method requires users to predict when conflicts will occur, which may not always be predictable. This dissertation indirectly adds to this category by proposing a method for on demand lock removal requiring lock owner approval.

Rules Based

Shen et al. introduce a collaborative drafting tool to aid in mechanical engineering design education. This tool handles conflicts by implementing authorization rules and a team manager. The authorization rules allow other users to modify one designer's entities only when they are authorized to do so. In addition, a user who makes changes to entities that he owns will automatically override any changes made by other users. The team manager acts as a team coordinator and mediator between users; however, he acts as a common designer when making changes to the model [36].

Chen et al. present their collaborative assembly modeling system called e-Assembly. This system allows multiple users to jointly build and constrain an assembly model in real time, based on coordination rules to help avoid conflicts. The rules govern who works on which link entity and allow only one collaborator to work on the same atomic component or link entity [37].

Lin et al. acknowledge that if operations in a multi-user collaborative environment are conflicting with each other, one of the operations has to be removed. They suggest that a method which involves eliminating user's operations through blocking/aborting is less desirable than a masking method because when such a method is employed, these operations are lost to the system and cannot be brought back when necessary. The masking method, on the other hand, maintains both resulting variations of the model when conflicts occur, but only displays the version that has the highest priority, thus masking all other variations. Since all variations are stored internally, if the operation with the highest priority is removed, the system displays the operation with the next highest priority. This masking strategy was implemented in Collaborative Genetic Software Engineering to manage tree structure constraints. They suggest that the masking method may also be applied to CAD, spreadsheets, graphical interface toolkits, and simulation systems [38].

Liu et al. suggest that an optimistic approach using operational transform is not possible for a CAD system due to the relational complexity of the features in a CAD part [39]. However, research by Jing et al. has shown that this may be feasible. They utilize operational transforms to merge multiple compatible user operations on uniquely named topology, allowing for a less constrained multi-user interaction within a collaborative CAD system. They do not however, discuss methods for managing conflicts on multiple child features referencing the same topology or handling conflicts between incompatible features [21].

Rules-based approaches have the potential to provide the least overhead and restrictions for a collaborative CAD environment, thus most of the methods proposed in this dissertation fall in this category.

Summary of Conflict Management Methods

Although turn based conflict management systems do avoid all potential conflicts, they are severely limiting for a parallel design workflow because only one user can edit the part at a time. Model decomposition is an improvement on this method because it allows multiple users to

contribute to the same model. However, the pre-work involved in setting up the boundaries may time away from the actual design process. It may also restrict the design process by forcing users to work in a confined area. However, this method may be beneficial in complex models due to the improved organization it provides. On demand feature locking methods allow for multiple users to be in the same model but require users to predict when conflicts will occur, which will unlikely avoid all conflicts. Rules-based approaches have the potential to provide the most user flexibility of all categories in the development of conflict management and model consistency because they do not restrict the user unnecessarily. This dissertation builds on top of these approaches by proposing efficient conflict management methods which avoid and resolve conflicts while ensuring model consistency.

1.2 Principles and Methods for Conflict Management

There are two main types of conflicts in feature-based CAD: syntactic and semantic conflicts. Syntactic conflicts result from multi-user operations which cause incompatible data to coexist and lead to errors in the CAD program. Semantic conflicts originate from users misunderstanding each others design intent which lead to unexpected design results. Within a parametric feature-based CAD system there are various types of potential syntactic conflicts which occur in a part. This dissertation classifies syntactic conflicts into the following sub-types: feature/self, parent/child and child/child.

The feature/self conflict occurs when multiple users edit the same feature or feature parameters at the same time. An example of this is when two users try to edit the same extrude feature and one user inputs 3 mm for the extrude length while the other inputs 5 mm. Since the value for extrude length cannot be both 3 mm and 5 mm, a conflict will occur (see Fig. 1.2).

The parent/child conflict occurs between a parent feature and a child feature which directly depends on that parent. This happens when a simultaneous creation, edit or deletion of a parent or child causes the other to fail. For example, this occurs if one user creates a hole in an extrusion while a second user changes the extrusion length. Since the hole and the extrusion no longer intersect, this renders the hole invalid (see Fig. 1.3).

The child/child conflict is between two features which reference the same parent geometry. An example of this would be if one user creates a chamfer on an edge while a second user creates a

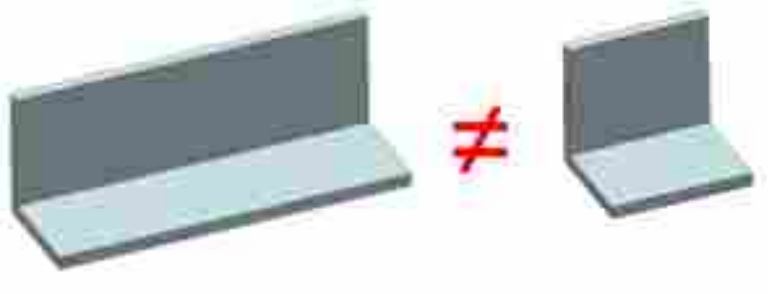


Figure 1.2: Feature/self conflict example

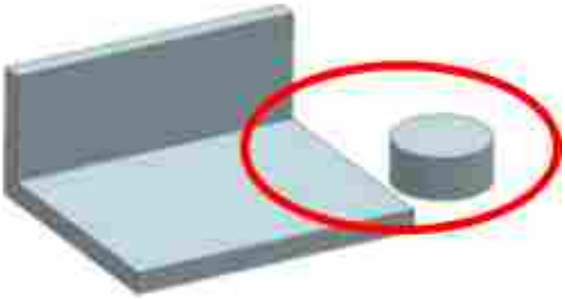


Figure 1.3: Parent/child conflict example

fillet on the same edge. Since these two features reference the same edge and the edge disappears after one of these operations is performed, a conflict results (see Fig. 1.4). If these various types of syntactic conflicts are not appropriately managed, they will lead to errors within the multi-user CAD system.

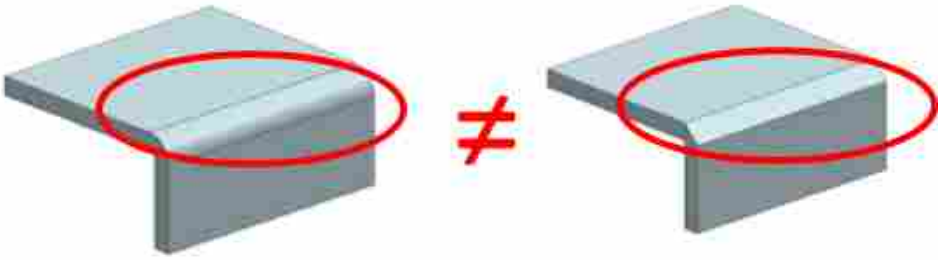


Figure 1.4: Child/child conflict example

In addition to syntactic conflicts, semantic conflicts are also important to manage. These are conflicts which originate from users misunderstanding each other's design intent which result in

operations which cause unintended design results. For example, two designers put a hole on a face at slightly different locations, causing the holes to overlap each other (see Fig. 1.5). This would not cause a syntactic violation because these are both valid operations that can coexist. However, this causes unintended results in the model and thus violates each of the users' design intent. Semantic conflicts reflect lack of communication between users, not a CAD system architectural flaw. However, managing these conflicts will reduce redundancy and overhead in the parallel design process.

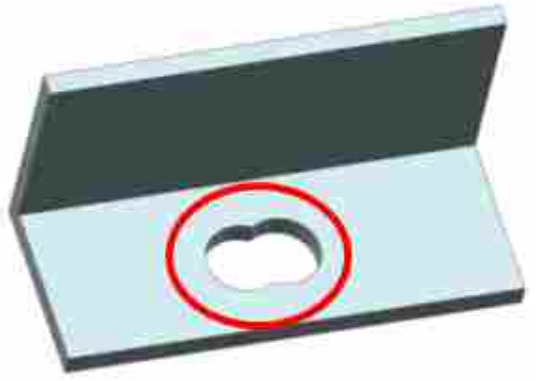


Figure 1.5: Semantic conflict example

For developers of a multi-user CAD system, syntactic conflicts are a major concern because they can lead to model inconsistencies and software bugs. In contract, semantic conflicts do not cause software bugs, but they are a problem because they lead to redundant work.

This dissertation proposes a rules-based method called automated feature reservation which automatically avoids feature-self conflicts (ch. 2). An on demand method for reservation removal is also presented. An implementation of this method is discussed and results show that feature-self conflicts are automatically avoided using this method. Another rules based method for conflict management and model consistency is proposed which resolves other types of syntactic conflicts (ch. 3). This method ensures a global order of operations for all clients while allowing users to resolve conflicting operations.

After a departure in chapters 3 and 4 to discuss model consistency, this dissertation returns to the discussion of semantic conflicts. Ch. 5 discusses how the automated feature reservation method avoids semantic conflicts. A mathematical model is developed and results of a series of

tests are discussed which suggest that semantic edit conflicts are avoided using the automated feature reservation method. In addition, ch. 6 discusses a decomposition method, namely the integrated task management system. This system allows teams of users to coordinate their efforts through a multi-user task list embedded in the multi-user CAD system. Results of several test suggest that teams which are allowed access to the integrated task list have fewer semantic conflicts.

1.3 Principles and Methods for Model Consistency

As mentioned, while dealing with syntactic conflicts it is critical to ensure model consistency. An important principle of model consistency in multi-user CAD is ensuring a global order of operations for all clients. This means that the order in which operations occur on one client must be consistent for all clients in the same model. If a global order of operations is not employed, geometry will not necessarily remain consistent between clients. When operations are allowed to occur on multiple clients simultaneously, they may be applied out of order because operations take time to apply and transmit via the server. This dissertation will discuss methods which ensure that modeling operations will occur in the same order on all the clients, while still allowing multiple users to simultaneously contribute to the model. In cases of conflict, a resolution method preserves conflicting operations locally for later reuse or manual resolution by the user (ch. 3).

Closely aligned to the model consistency material in ch. 3 is the principle of consistently referencing the same topological entities on each client, the subject of ch. 4. Feature-based CAD systems create features that parametrically reference topological entities which include faces, edges and vertices. References to the topological entities in a CAD part on one client must be the same on all clients to ensure dependent features are applied to the same topological entity on all clients. For example, a fillet feature operation applied to an edge in one client needs to be applied to the same edge in all other replicated client models. If the system fails to keep names of features and entities persistent between clients, models will become inconsistent, causing errors to occur. This problem is referred to as the persistent naming problem in the literature [21]. Fig. 1.6 shows a graphical representation of persistent naming between clients.

Persistent naming in a replicated multi-user CAD system is challenging because topological entities of at least one major geometry kernel (namely Siemens Parasolid) are not returned in a predictable order [40]. Since faces and edges cannot be identified by the order in which they are re-

turned, they must be uniquely identified them by their geometric properties. However, identifying each topological entity by its geometric properties can be time consuming for models with thousands of faces and edges. This dissertation will discuss a method which efficiently keeps names of topological entities persist between clients (ch. 4). The method is implemented in multi-user CAD and is shown to be significantly more efficient than previous methods while still providing consistency.

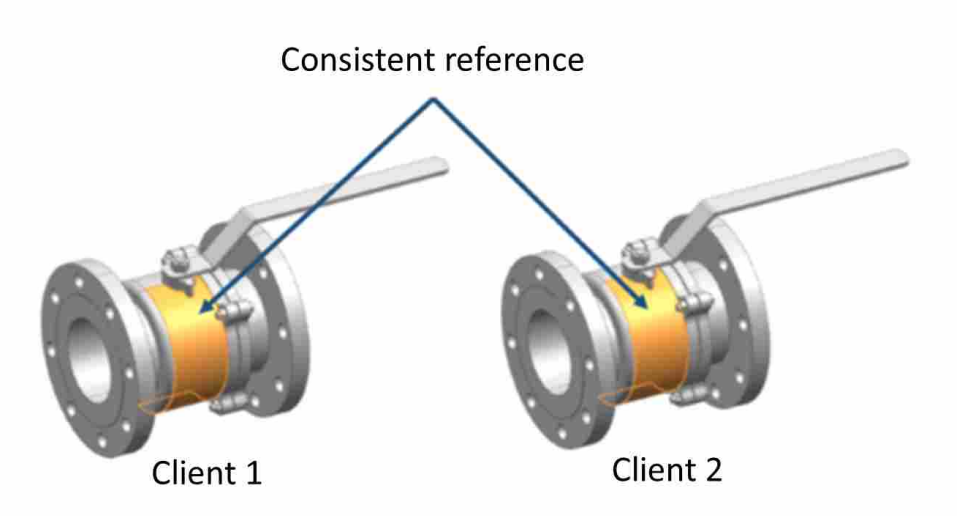


Figure 1.6: The references to the face in these replicated models must be consistent

1.4 Dissertation Objective

The objective of this dissertation is to develop principles and methods for conflict management and model consistency to increase robustness, usability and user efficiency in multi-user CAD.

1.4.1 Robustness

A robust multi-user CAD system is one which provides model consistency, meaning all replicated models are consistent during multi-user modeling. An important element of this is the level in which the system avoids and resolves syntactic conflicts. A robust multi-user CAD system

ensures that the three types of syntactic conflicts (feature-self, parent-child and child-child) do not cause model inconsistencies.

Another important element of model consistency is ensuring a global order of operations. Because the resultant geometry in a CAD model is derived from the order of CAD operations, it is critical that operations remain consistent between clients. In replicated multi-user CAD, operations are allowed to occur on multiple clients simultaneously, which means that they may be applied out of order. This is due to the fact that operations take time to apply and transmit via the server. Ensuring a global order of operations in a replicated multi-user CAD context means that the order of operations is eventually resolved between clients, ensuring a consistent order.

Yet another aspect of a robust, replicated multi-user CAD system is persistent naming. Persistent naming means that all references to topological entities are consistent between replicated models. A robust persistent naming method ensures all dependent features are consistently applied, which is an essential aspect of model consistency.

In the context of this dissertation, robustness refers exclusively to the ensuring of model consistency. This can be solved by eliminating syntactic conflicts, ensuring a global order of operations on replicated clients and providing persistent naming. The principles and methods discussed herein are limited to replicated multi-user CAD systems.

1.4.2 Usability

In the context of this dissertation, usability refers to the level in which the multi-user CAD system can be used by industrial users. The benchmarks for usability are existing commercial CAD systems used by industry. Therefore a perfectly usable multi-user CAD system based on a commercial CAD system is one that has the same performance as the system on which it is based.

A major aspect of performance in CAD is the computation time of system operations. Since persistent naming requires additional steps by the system, this has the potential to significantly reduce performance and thus usability. This dissertation presents a method for efficient persistent naming. The efficiency enhancements presented make the performance of the persistent naming method closer to that of the commercial CAD system, thereby making it more usable.

Although there are many other aspects of usability and performance in multi-user CAD, the discussion of usability in this dissertation is limited to the usability of the persistent naming method presented for model consistency. All other aspects of usability are topics for future research.

1.4.3 User Efficiency

User efficiency of a multi-user CAD system refers to the level in which users efficiently use the system. Semantic conflicts are those which lead to rework and wasted time. Wasted time directly and adversely affects efficiency. This dissertation attempts to increase the efficient use of the multi-user CAD system by reducing semantic conflicts.

A perfectly efficient multi-user system would entirely eliminate all semantic conflicts altogether. Since the elimination of all semantic conflicts would require perfect communication and coordination, it is not the intent of this dissertation to entirely eliminate semantic conflicts. Rather it is to present principles and methods to reduce such conflicts, thereby increasing user efficiency in multi-user CAD.

1.4.4 Summary

The objective of this dissertation is to develop principles and methods to increase robustness multi-user CAD system by ensuring model consistency. This is done by eliminating syntactic conflicts, ensuring a global order of operations on replicated clients and providing persistent naming. Performance enhancements are developed for persistent naming to increase its usability. Additional principles and methods improve user efficiency reducing semantic conflicts by enhancing communication and organization in a multi-user CAD modeling environment.

1.5 Dissertation Organization

The organization of this dissertation is as follows: Chapters 2-6 each begin with a brief introduction that is specific to the content presented in the chapter. Following this, each chapter has a section which discuss the most relevant literature regarding the subjects they discuss. In addition, each chapter has sections which discuss the details of the method, implementation, results, and

conclusions of the their specific subject matter. Chapter 7 draws overall conclusions and suggests future research related to this work as a whole.

CHAPTER 2. SYNTACTIC CONFLICT AVOIDANCE THROUGH AUTOMATED FEATURE RESERVATION

2.1 Introduction

Conflict management is required for a robust multi-user CAD solution. A balance between the extremes of the pessimistic and optimistic approaches has the potential to minimize waste and user restriction. This chapter presents a hybrid approach between the extremes of purely optimistic or pessimistic conflict management, called automated feature reservation. To avoid conflicts that would require manual resolution, the automated feature reservation method automatically places access restrictions on features. 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 to avoid manual merging of conflicts, they are set to a minimum to preserve an agile and uninterrupted multi-user experience. Fig. 2.1 compares the hybrid approach with the extreme optimistic and pessimistic approaches.

The chapter discusses the how the automated feature reservation method prevents multiple users from simultaneously editing the same feature, thus avoiding conflicts. The implementation of the method in a commercial CAD system is also discussed. Results show that this methodology prevents data inconsistency that results from feature/self conflicts. This system prevents CAD modeling conflicts, while providing a uninterrupted user experience within the collaborative environment.

2.2 Background

Chapter 1 categorized the various methods found in the literature for conflict management. Turn based methods only allow one user to contribute at a time [15,31,32]. Turn based approaches are serial and do not enable real-time synchronous collaboration. Model decomposition requires users to partition models geometrically or otherwise [7,33]. Although the partitions avoid conflicts,

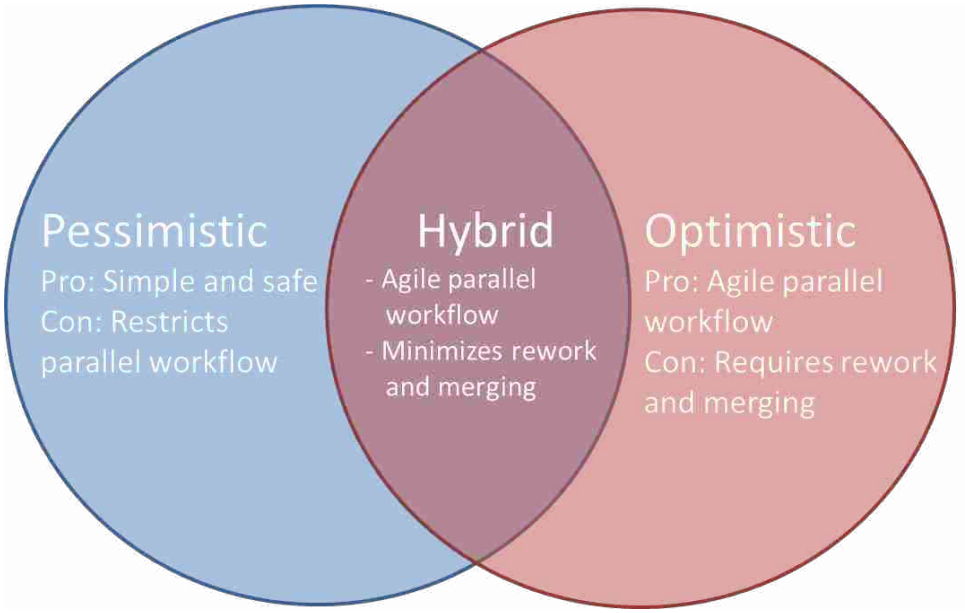


Figure 2.1: Comparison to extreme optimistic and pessimistic approaches

they do not allow users to work in the same modeling space, which may be required for many modeling activities. On demand locking asks users to lock regions or features in the model to avoid conflicts [8, 32, 34, 35]. These methods require users to predict when conflicts will occur, which adds some additional waste to the design process. Rules based methods govern conflict avoidance and resolution by enforcing a set of rules [21, 36–39]. Rules-based approaches have the potential to provide the least waste and restriction for a collaborative CAD environment. The automated feature reservation method, proposed herein, is a rules based method which avoids conflicts through automated feature reservation.

2.3 Conflict Management Methods

This section presents an automated, rules based conflict management system for a multi-user CAD environment. When a user edits a feature, the feature is automatically reserved and blocked from other user edits for the duration of the feature edit. A method is also presented which allows users to remove reservations on demand. These methods allow multiple users to edit the same part simultaneously, without dividing the part into single user regions or manually locking features.

2.3.1 Feature Conflict Avoidance

Commercial CAD systems allow only a single user per part, thus making the part the atomic unit for concurrent CAD interaction with multiple users. When multiple users modify a part simultaneously, a new atomic unit for multi-user interaction must be established. The existing single user CAD building block unit is called a feature. Each feature consists of controlling parameters which are combined to create a parametric geometry model. This dissertation establishes the principle that the atomic unit of a multi-user, feature-based CAD system is the feature. This means that multiple users may not simultaneously modify the same feature within a distributed part. In mathematical terms it can be expressed as follows: Given that F is the set of all features in part P, at a given time interval t, and f is a feature in part P, at time interval t. U is the set of all users in part P, at time interval t and u is a user in part P, at time interval t, the following is true:

$$f \in F(P,t) \tag{2.1}$$

$$u \in U(P,t) \tag{2.2}$$

Axiom 1 states that for a given edit operation there exists a unique set E, in part P, at time interval t, which contains only one feature f and one user u where

$$E(P,t) = f,u \tag{2.3}$$

For interval

$$t_{beginEdit} \leq t \leq t_{endEdit} \tag{2.4}$$

Multi-user, feature atomicity is enforced above by only allowing a single instance of set E to exist in part P, at time interval t, where t is the time interval from the beginning to the end of the edit operation. If this atomicity is not enforced, conflicts of a feature with itself will occur when a user on one client edits a feature while another user is editing the same feature.

The process of editing a feature in a CAD system often takes several seconds to perform because multiple parameters for the feature may need to be modified. During the time it takes to

complete an edit operation by client 1, client 2 could potentially make an edit to that same feature. Once client 1 finishes the edit, the update is sent to client 2 over the server. This update cannot be processed by client 2 who is performing an edit operation and the commercial CAD client can only perform a single operation at a time. Therefore the edit is put into the delayed operation queue. Meanwhile, client 1's update stays consistent until client 2 finishes his edit. When client 2 finishes, the change is sent over the server to client 1 and her feature is modified to the values of client 2's update. After client 2 finishes his edit, she will process client 1's update which was sitting in the delayed operation queue. Thus, client 1 receives client 2's edit update and client 2 receives client 1's edit update. This results in feature data inconsistency between clients after local edits of the same feature (see Fig. 2.2).

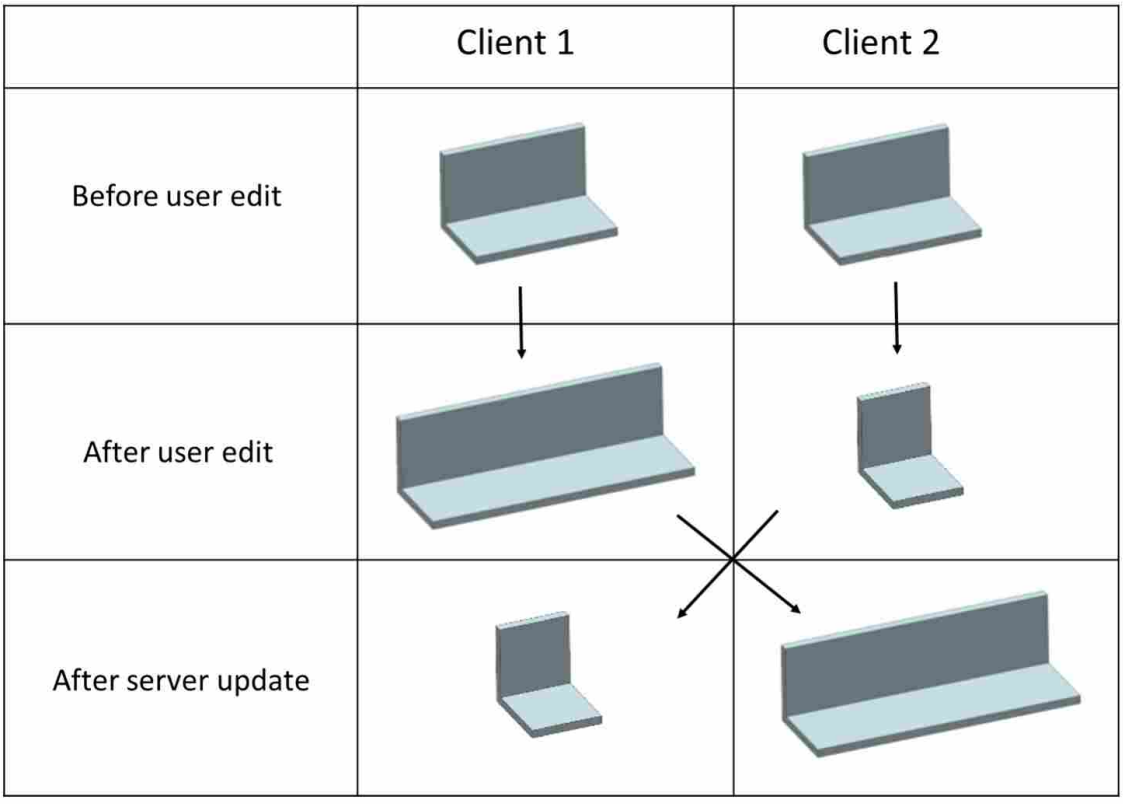


Figure 2.2: Simultaneous feature edit leading to feature data inconsistency

A method for automatic feature reservation and blocking is proposed to prevent simultaneous edits of a single feature by multiple users. 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 prevents other users from editing 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. When the client receives the message the software automatically removes the feature block. In this way, other users are unable to edit the feature for the duration of a user edit. This is called the client blocking method.

This method fulfills Axiom 1 if communication between clients and the server is instantaneous. However, due to network latency, this 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. The set E in eqn. (2.3) is unique only for the interval after the message is received. Eqn. (2.5) accounts for the time it takes to send the block message (t_{addBlock}) and the time it takes the send the remove block message ($t_{\text{removeBlock}}$):

$$t_{beginEdit} + t_{addBlock} \leq t \leq t_{endEdit} + t_{removeBlock} \tag{2.5}$$

Since the interval where set E is unique begins only after the blocking message is received, it is not guaranteed to be unique until after time t_{addBlock}. This violates Axiom 1 and results in a potential data conflict scenario as illustrated in Fig. 2.3. 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. This is done by

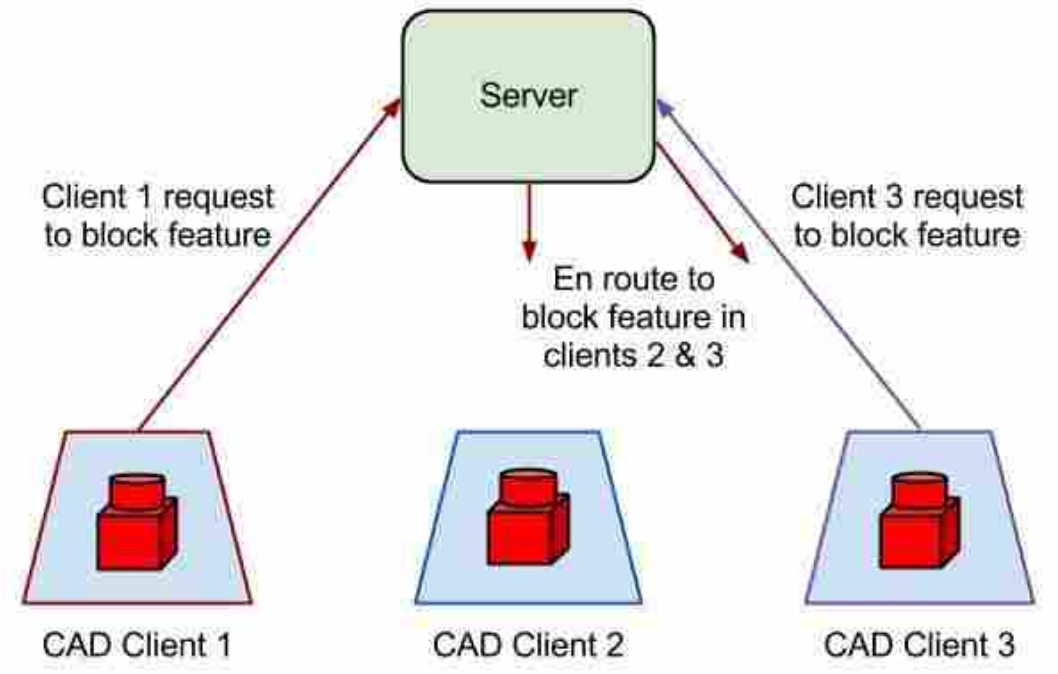


Figure 2.3: Potential issue with client blocking method

setting a Boolean flag associated with that feature to be true. A message is sent to all other clients telling them that the feature is reserved. All clients will then implement a client block on that feature. No message is sent to the originating client, so he continues to edit the feature assuming he is authorized to make the edit. Once the edit is complete, a message is sent to the server to cancel the feature reservation and set the Boolean flag back to false.

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 (see Fig. 2.4).

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. It is shown that since the time interval for set E in eqn. (2.3) exists at the server instead of the clients, the time it takes to send the message is not included in the editing time interval in eqn. (2.4). Assuming that network latency is always less

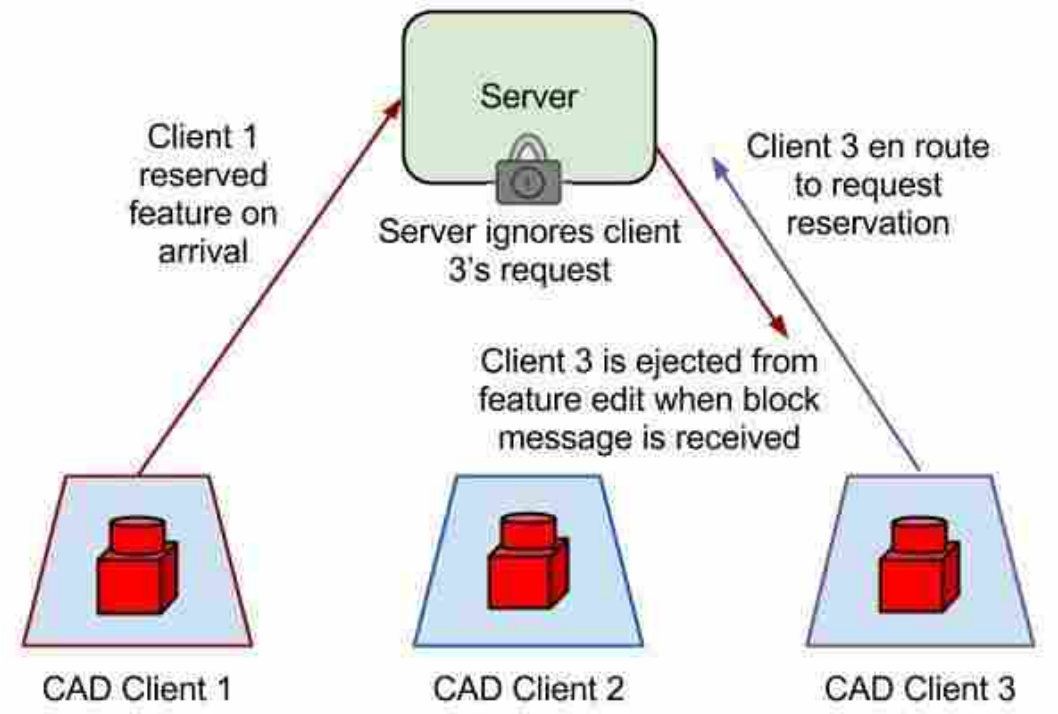


Figure 2.4: Potential issue with client blocking method

than edit time, Axiom 1 holds true because the edit operation does not truly begin until it reaches and is authorized by the server. Therefore the edit time interval is the time it is reserved on the server and is the same as in eqn. (2.4). One enhancement to this approach would be to not allow multi-user editing when latency is too great. This would ensure blocks even when latency is greater than edit time. Chapter 3 presents a method to maintain consistency independent of latency time entirely.

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.

2.3.2 On demand reservation removal

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. For example, if a user leaves for lunch while in a feature edit operation, a second user may need to edit that feature and should not have to wait until the first user returns from lunch to do so. One method to do this is to control on demand reservation removal based on user roles. For example, managers or team leads could have the authority to remove reservations. The problem with this approach is that users must find someone with the authority to remove the reservation or else they can't perform the necessary operation. This approach is very restricted. Alternately, all users could be allowed to remove reservations at any time. The problem with this approach is that users may actually be in an edit when the reservation is removed and their work would be interrupted.

In order to minimize modeling restriction and interruption, the following 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.

All users may request any feature reservation to be removed at any time. Feature reservation removal does not need to involve an authorized user (i.e. manager). On demand reservation removal also minimizes modeling interruption. A user performing an edit is not automatically ejected from their edit. He is notified that a relinquishment request is made, but he retains control of the feature until his time limit to respond expires. Figure 2.5 shows a complete flow chart of the combined client blocking, server reservation and on demand reservation removal methods.

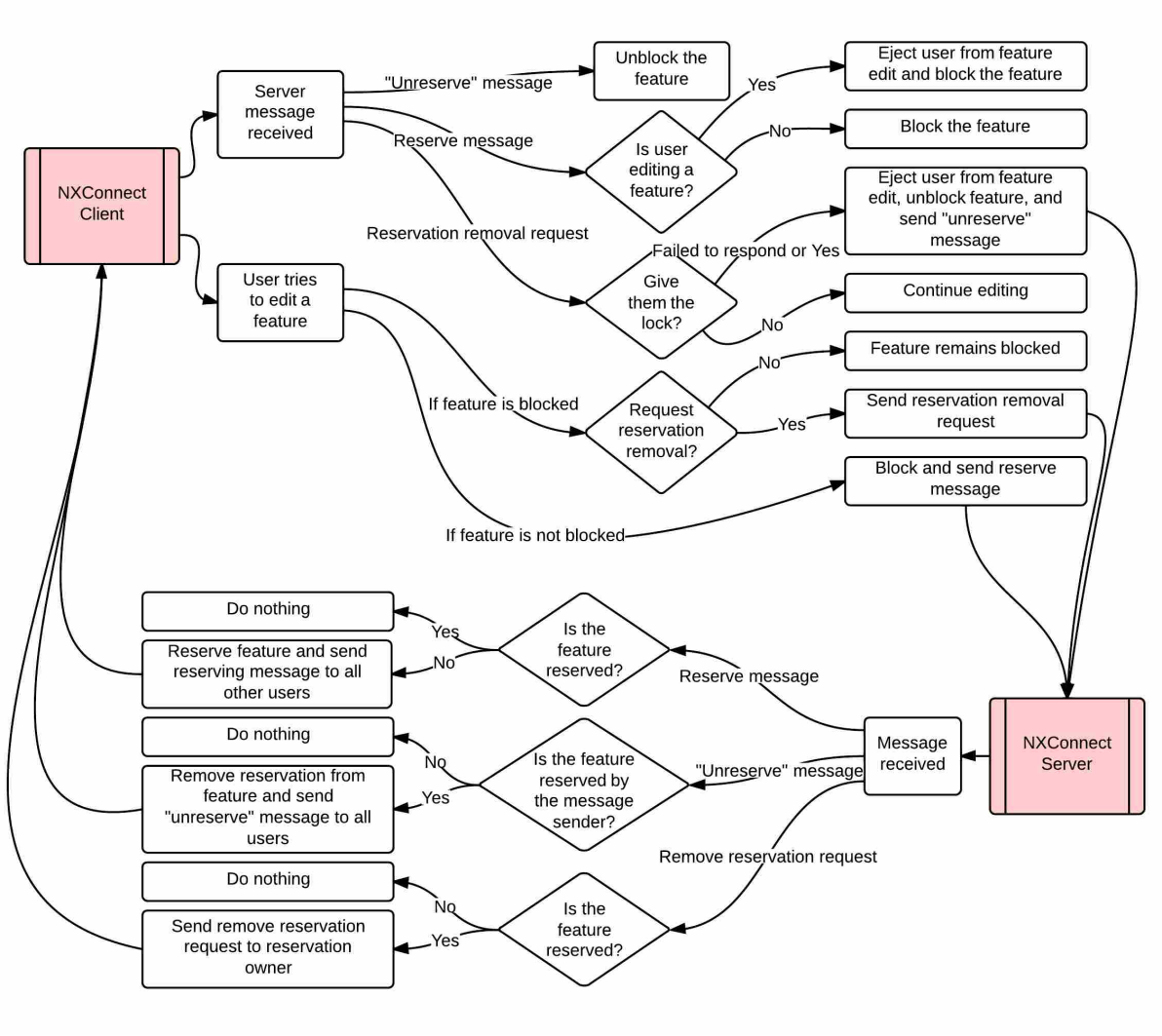


Figure 2.5: Flow chart of combined client blocking, server reservation and on demand reservation removal methods

2.4 Implementation in NXConnect

The automated feature conflict avoidance and on demand removal methods have been implemented into a multi-user CAD system being developed at BYU called NXConnect. The development of this software involves utilizing the Siemens NX application programming interface (API) to extend the current single user functionality of NX to be a simultaneous multi-user application. NXConnect uses a client-server architecture with a thin server and thick client which requires an instance of an NX session running on each client. Each client that is participating in a given model has an identical copy of the NX part that stays in sync with all other clients in that same

model. Data consistency between clients is achieved by propagating model changes to the server and then onto other clients. When clients receive data pertaining to an operation on another client, the NX API function to perform that operation is called on the clients' session so the clients all get changes performed by other users. The data to perform that operation, as well as any geometry reference data, is recorded in a database for future retrieval.

The feature reservation and blocking methods have been implemented in NXConnect as follows: the server has an up-to-date reservation list which contains all the reservations held by users in each part. When a user attempts to edit a feature or features, an event for that feature is activated, which triggers a message to the server requesting access to edit that feature. The server handles the message by checking the reservation list to see whether the specific feature has already been reserved by another user. If the feature is not reserved, the reservation is added to the reservation list to prevent other users from editing the feature. Access is implicitly granted to the requester by not sending a message back. Since the reservation messages are asynchronous, a user does not need to wait for a message back from the server to begin editing a feature. Therefore, if he does not receive a rejection message, he is safe to continue editing. However, if the feature is already reserved, a message is sent to the requester indicating the feature is reserved by another user. When this message is received, the edit dialog is automatically exited to prevent the user from editing the feature. Therefore feature reservation is handled even when multiple users request a reservation at nearly the same time. This is implemented in such a way that reservation can be made for multiple feature edits as well.

When access is granted to edit a feature, a message is sent to all other clients blocking that particular NX feature. When a feature is blocked on the client, it is colored with a color users are trained to recognize as blocked. If the blocked feature is a solid body, all the faces of that body are colored. If the blocked feature is a sketch or curve, the curves are assigned the blocking color. All other features and curves are left to the default color. This communicates that the feature is reserved and not available for editing, while all other features (of the default color) are still available. Fig. 2.6 show the NXConnect reservation implementation: the circle extrusion feature is edited on client 1's session and is reserved for editing in client 2's session.

In addition to coloring the feature, the feature is blocked on the client to prevent users from editing the reserved feature. A client-side status flag, associated with each feature on a given client,

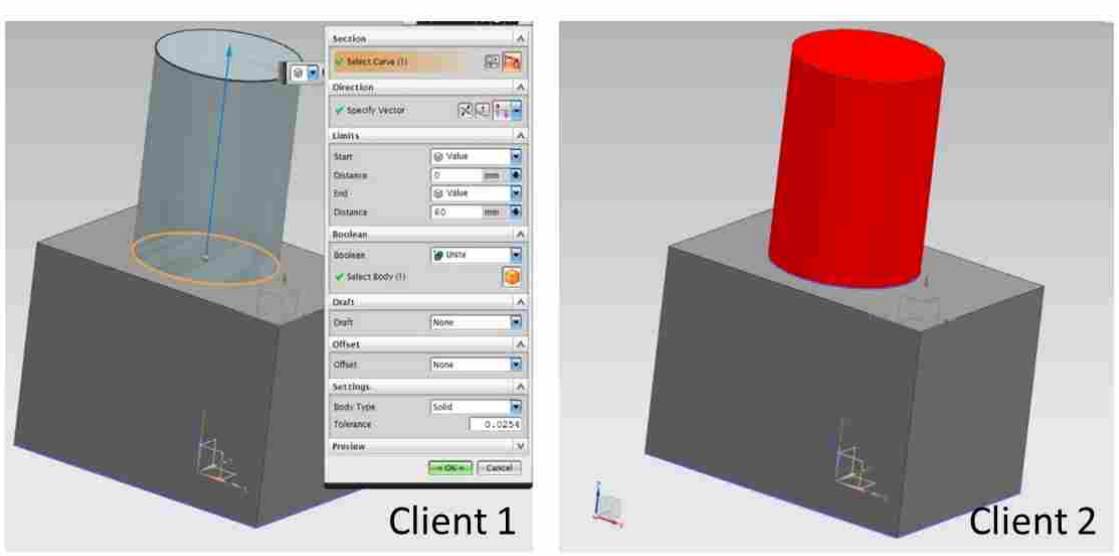


Figure 2.6: Client 1 editing extrusion of circle while client 2 has the circle extrusion reserved (red)

keeps track of whether a feature is “reserved by another user” (RBA), “reserved by me” (RBM) or “not reserved” (NR). The default is set to NR. When the user reserves a feature, the status is set to RBM. It is important to know that a given feature is reserved by the user because when he releases a reservation after an edit, he needs to know which features were reserved in order to remove the reservation. When a reservation message is received from the server, the status flag is set to RBA. Whenever a user attempts to edit a feature, the client first checks to see whether the client-side feature status flag is set to RBA. If this is the case, the edit operation will automatically be canceled, thus preventing users from editing a feature reserved by another user.

Since feature reservations only remain for the duration of the feature edit operation, reservations are removed automatically once the operation is complete. When a user finishes editing a feature, a message is sent to the server requesting the reservation be removed for all features which are set to RBM. When the server receives this message it will remove the reserved feature from the reservation list and send a message to all clients telling them to remove the client block. The clients each handle the message by changing the reservation status to NR and changing the color of the feature back to the default color. In this way the feature becomes available to edit and the client block is removed.

When a user opens a part in which users already have features reserved, it is essential that the client receive all existing reservations in the part. Additionally, when a user logs out or exits a part, all the reservations they have in place must be removed. This is done as follows: when a user loads a given part, the reservation list is queried for all feature reservations in that part. Reservation messages are sent to the user for each feature reservation in the part and the features are blocked according to the client blocking method. When a user logs out or exits a part, messages are sent requesting removal of all reservations for that client. In this way reservations are added and removed by clients who join or exit a modeling session.

On demand feature reservation removal is implemented in NXConnect as well. A user requests reservation removal by double clicking on a feature that is blocked on the client. He receives a message notifying him that the feature is reserved and asks him if he would like to request removal of the reservation. If he clicks yes, a request is sent to the owner via the message server. The reservation owner receives the request message and has 30 seconds to respond, during which time the user may continue to edit the feature. He may also choose to reject the request which allows him to continue editing the feature until another request is received. It also sends a message to the requester notifying him that the request is denied. Conversely, if the reservation owner accepts the removal request, he is automatically ejected from the edit dialog and the reservation is removed. Additionally, if he fails to respond to the message within the 30 second period, he is ejected from the edit dialog and the reservation is removed. Fig. 13 shows the message received when client 1 tries to edit a feature. He decides to request a reservation removal and client 2, who owns the reservation, receives the message as seen in Fig. 2.7. This message explains that client 2 can either accept or reject the reservation removal request and that he must respond in 30 seconds; otherwise the reservation is automatically revoked and he is ejected from the feature edit.

2.5 Results

To show the validity of the automated feature reservation method for preventing feature/self conflicts, two test cases were performed in NXConnect. Both of these tests are performed on a simple angle iron model where two users try to edit the extrusion length at the same time. To simulate simultaneous edits, two users purposely click on the feature edit command at the time. The first case is performed with the feature reservation methods in place and primarily tests the

Figure 2.7: Client 1 requests a reservation removal while client 2 receives a reservation removal request

implementation of the client blocking method. The second case is performed with an added 2.5 second latency on the network to provide enough latency to test the implementation of the server reservation method.

The first test case has two users try to edit the extrusion length at the same time. The result of this test case is that the first user (client 2) to click on the edit feature operation is able to perform the edit. The second user (client 1) is not allowed to edit the feature because the block is already in place by the time the user clicks (see Fig. 2.8). This is due to the fact that network latency is typically less than human reaction time. The chances that two users will send reservation requests within typical message round trip time (RTT) is very small. However, the second test is in place to verify that if this happened, the feature reservation system still remains robust.

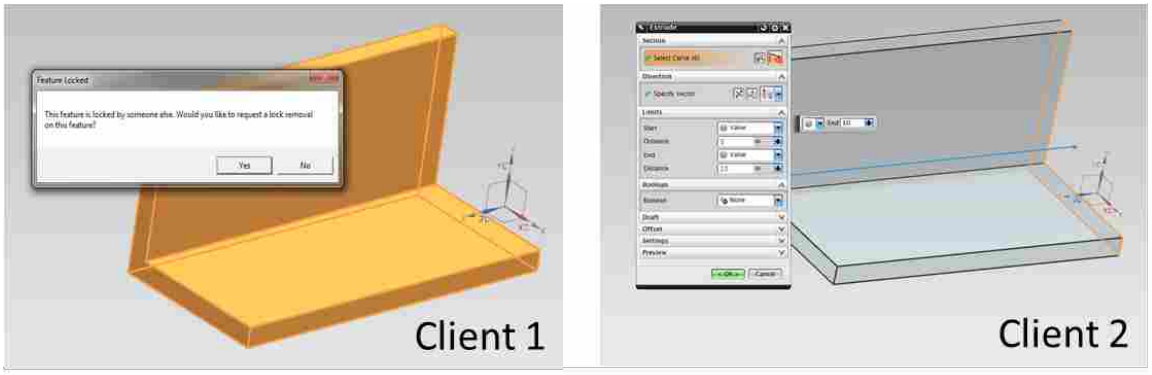


Figure 2.8: Both users tried to edit the feature at the same time and client 2 get to the server first so client 1 receives the block notification from the server

The second test case has two users try to edit the extrusion length at the same time, but has an added 2.5 second latency on the network. This provides sufficient time for both users to send a request message to the server before one receives the reserve message. The result of the second test case is that the first user to click on the edit feature operation is able to perform the edit (client 1 in Fig. 2.9). The second user seems to be allowed to edit the feature but is ejected from the feature edit in less than 2.5 seconds (client 2 in Fig. 2.9).

The upper two images in Fig. 2.9 shows two users being able to simultaneous edit the extrusion for a limited time. However, shortly after, client 2 is ejected from the edit dialog and the feature is blocked on his client (see lower two images in Fig. 2.9). This happens because client 1's reservation request arrived to the server first. When client 2's request was received by the server, it is rejected because the feature was already reserved by client 1. Once he receives a message from the server rejecting his request, he is automatically ejected from the dialog and the feature turns red signifying a feature block. Assuming network latency is always less than edit time, the user whose request is rejected will never be able to complete the edit before he is ejected from the edit dialog.

The results from these two tests show that this methodology prevents the data inconsistency that results from feature/self conflicts as seen in Fig. 2.2. Latency is not only the limiting factor in performance of the system, but it tends to create many cases that can cause data inconsistency between two clients. Maintaining data consistency for a distributed system in a wide-area network is challenging, but this system lays the groundwork for doing so with many users and typical latency [41].

2.6 Conclusions

An automated feature conflict avoidance methodology is proposed which prevents multiple users from simultaneously editing the same feature. This approach automatically reserves a feature for the duration of an edit with priority given to the first client to request the reservation. A method is also proposed for on demand reservation removal. Methods are presented that create an asynchronous feature reservation on the server and a block on the client. Client blocks have a visual cue to communicate the reservation status and forcibly prevent users from editing a reserved feature. Server reservations are in place to prevent feature conflicts when multiple users edit the

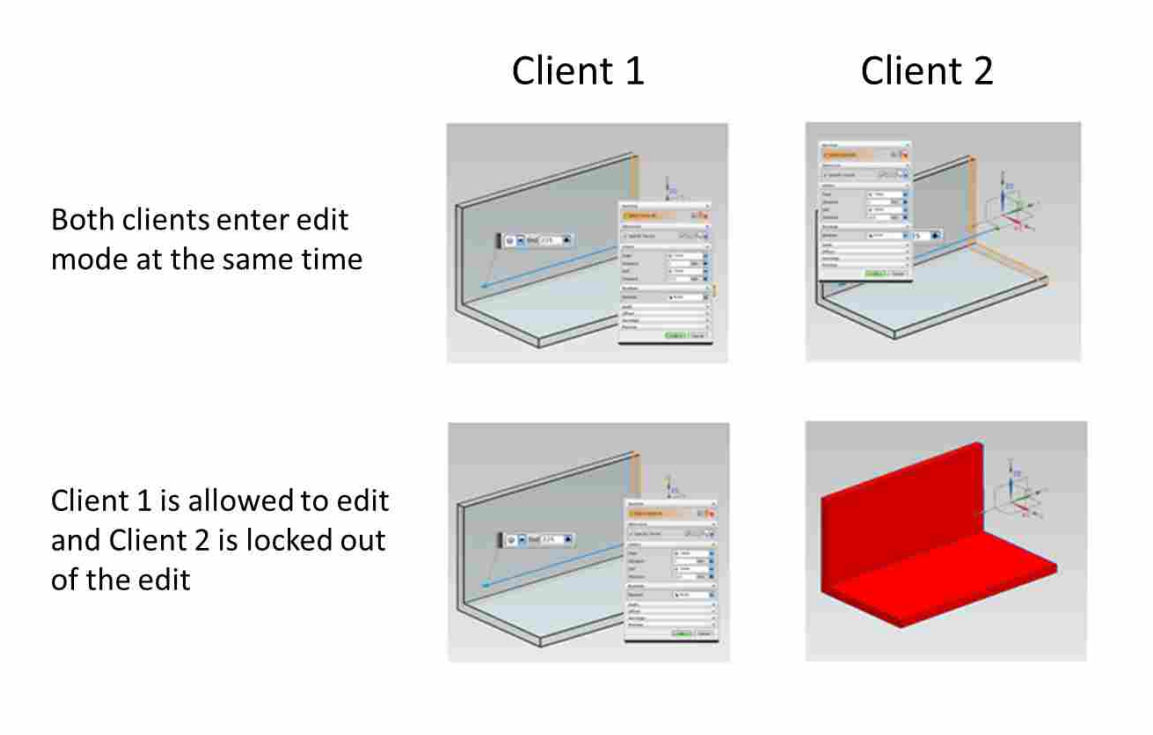


Figure 2.9: Both users are allowed in the edit until one has received a rejection message and then blocked from the feature edit

same feature at nearly the same time. It is shown how this method fulfills Axiom 1 to preserve the atomicity of a single user and feature pair for the duration of a feature edit, assuming that network latency is less than edit time.

This approach shows the ability and functionality of automated feature reservation to prevent conflicts, thus helping to enable a less restricted collaborative environment for concurrent engineering. This approach has less waste 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 avoiding feature/self conflicts.

CHAPTER 3. SYNTACTIC CONFLICT RESOLUTION AND MODEL CONSISTENCY

3.1 Introduction

Chapter 2 discussed methods to avoid feature-self syntactic conflicts. To avoid syntactic issues related to parent-child and child-child conflicts it is critical that a system be in place which ensures model consistency. This chapter presents a method that keeps independent copies of the models consistent between distributed CAD clients by enforcing modeling operations to occur in the same order on all the clients. This is based on the principle that a global order of operations ensures consistency for all replicated models. In cases of conflict, a resolution method preserves conflicting operations locally for later reuse or resolution by the user. This approach ensures conflicts to do cause syntactic issues and provides a mechanism for resolving conflicts in the multi-user CAD system.

The simultaneous multi-user system uses asynchronous communications to allow concurrent communications between CAD user clients and the server. Although this communication style enables multiple clients to work on the same body at the same time, if not carefully coordinated, it can result in model inconsistencies. Commercial CAD systems, such as Siemens NX, are typically single threaded and thus can only perform a single operation at a time. While a client is performing an operation, operations from other users may complete and be received. All operations received from other users must be delayed until the active user's operation is complete. This delay, along with network latency, can cause models to diverge into inconsistent and conflicting states. As more users are added to the multi-user part, the likelihood for model divergence increases.

The example shown in Fig. 3.1 shows the divergence that may occur in a model with compatible (non-conflicting) operations. Two clients start in a consistent state, where each has the same geometry. The clients each begin to work on a new operation at nearly the same time. Updates from the other client are delayed until after each client completes their respective features. Client 1 performs a trim body operation on the cube while client 2 simultaneously performs a

shell operation on the same cube. Regardless of which client completes their operation first, the operation from the remote client is applied to the local client after the local operation. Client 1's order of operations are "trim body" then "shell", while client 2's order of operations is the reverse, "shell" then "trim body".

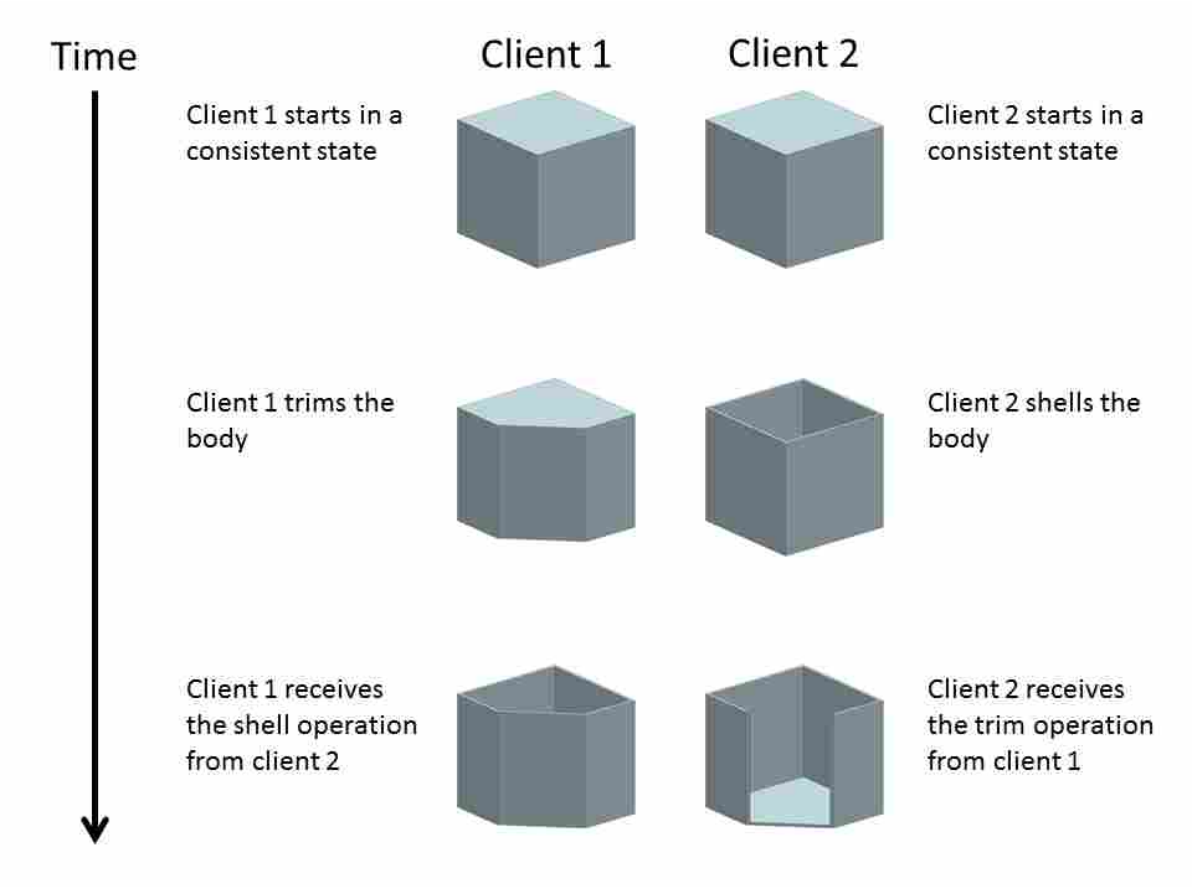


Figure 3.1: Two clients become inconsistent when order of operations is ignored

Forcing a global order of operations resolves conflicting operations in a multi-user system. This is a fundamental principle. The simplest system of enforcing a global order of operations is "first to reach the server wins". This approach verifies operations which reach the server first and rejects any operation from clients that have not applied all remote operations. If conflicts occur with a new operation, the operation is removed. The downside to this approach is that it can result in lost work by the losing user because conflicting operations are lost. A refinement of the "first to reach the server wins" approach which retains some losing operations in a suppressed state is

presented. This allows losing operations to be corrected by the user and reincorporated into the model instead of being lost.

This chapter presents methods which provide model consistency, conflict resolution, and invalid operation preservation for user correction in a replicated, multi-user CAD system by intelligently applying a global order of operations.

3.2 Background

As mentioned in the introduction, mutli-user CAD systems have two main architecture implementations: centralized and replicated. Model consistency for centralized multi-user CAD systems is inherently different than replicated systems because there is only a single instance of the model as opposed to many. RCCS is one replicated system which uses operational transform for model consistency but assumes that conflicting operations do not operate on the same topological entity [21]. ARCADE and CollIDE are two other replicated systems which use a pessimistic locking mechanism to allow one user to modify the model at a time [22, 24]. CSCW-FeatureM uses a token passing system to determine who can operate on the model [23]. Li et al. implement concurrency control for a replicated collaborative design environment consisting of several heterogeneous CAD systems based on a peer-to-peer network. Their concurrency control method does not handle dependency conflicts or preserve conflicting data [42]. None of the aforementioned research discusses applying a global order of operations with data preservation for model consistency in a replicated, multi-user CAD system based on a client/server architecture.

3.3 Model Consistency and Conflict Resolution Methods

Simultaneous multi-user access to a part model, without coordination, results in model inconsistencies. The method called operational consistency controls concurrency by enforcing a global order of operations for all users by ordering operations based on the "first to reach the server" approach. Operations may be performed out of order on distributed clients temporarily, but they are reordered so that they occur in a consistent global order on all clients. Once operations are reapplied in the global order on the client, operations which have not yet been transmitted to the server may be found to conflict with other operations. If this is the case, the feature which

reaches the server first still wins, but preservation of the conflict operation data is attempted so as to minimize data loss and rework.

3.3.1 Operational Consistency

A totally ordered set of verified global operations V, exists on the server for each multi-user part P. This consists of all n verified operations o_v, sent from clients performing operations in the multi-user part P, where o is any operation and o_u is any unverified operation. V is defined as follows:

$$V(P) = \sum_{i=1}^{n} o(v_i) \tag{3.1}$$

A totally ordered set of operations that have been performed by a client and verified by the server for part P is v. A totally ordered set of operations that have been performed by a client but not verified by the server for part P is set u. u and v are related to V, P and o by the following expressions:

$$v(P) \subset V(P), \tag{3.2}$$

$$o_u \in u(P), \tag{3.3}$$

$$o_u \notin V(P) \tag{3.4}$$

A totally ordered set of all operations O, in part P, for a given client also exists in which

$$O(P) = v(P) + u(P), \tag{3.5}$$

And

$$v(P) < u(P), \tag{3.6}$$

In other words, all verified operations v are ordered before u in set O.

Operation Sequence Number

The method to determine order of operations o_{v_i} in the set V lies primarily in keeping track of an operation sequence number (OSN) on the server. The OSN is an integer which represents the current count of verified operations that have taken place in a given model. The server sends the OSN with each operation message to the client. The client keeps a record of the most recently received OSN which is called the ROSN. The ROSN is attached to all messages sent to the server from the client and vice versa.

When an operation occurs on a client, a message is sent to the server containing the client's ROSN and the data to perform that operation. When the message arrives, the client's ROSN is compared to the OSN. If they match, the operation is verified, data is sent to the database and the OSN is incremented. A matching ROSN and OSN means that the totally ordered set of verified operations on the client v matches the totally ordered set of verified operations on the server V.

$$v(P) = V(P) \tag{3.7}$$

If they do not match, this means:

$$v(P) \neq V(P) \tag{3.8}$$

In this case, the operation message is ignored by the server. This occurs because the operation was submitted to the server before the client applied previously verified remote operations sent from the server. In other words, the client is out of date with the server. That client will need to apply remote operations to receive an updated ROSN before the server will accept any operation message from the client.

Fig. 3.2 depicts a client server diagram in which two clients send an operation message to the server at nearly the same time. Client 2's operation B reaches the server first. When this message is received by the server, the ROSN is compared to the OSN. Since the ROSN of operation B (value of 1) matches the OSN (value of 1), this operation becomes verified and the OSN is incremented. When client 1's operation C reaches the server, operation C's ROSN (value of 1) is compared to the new OSN (value of 2). In this case the ROSN is less than the OSN, meaning client 1 is out of date. This operation message is ignored by the server.

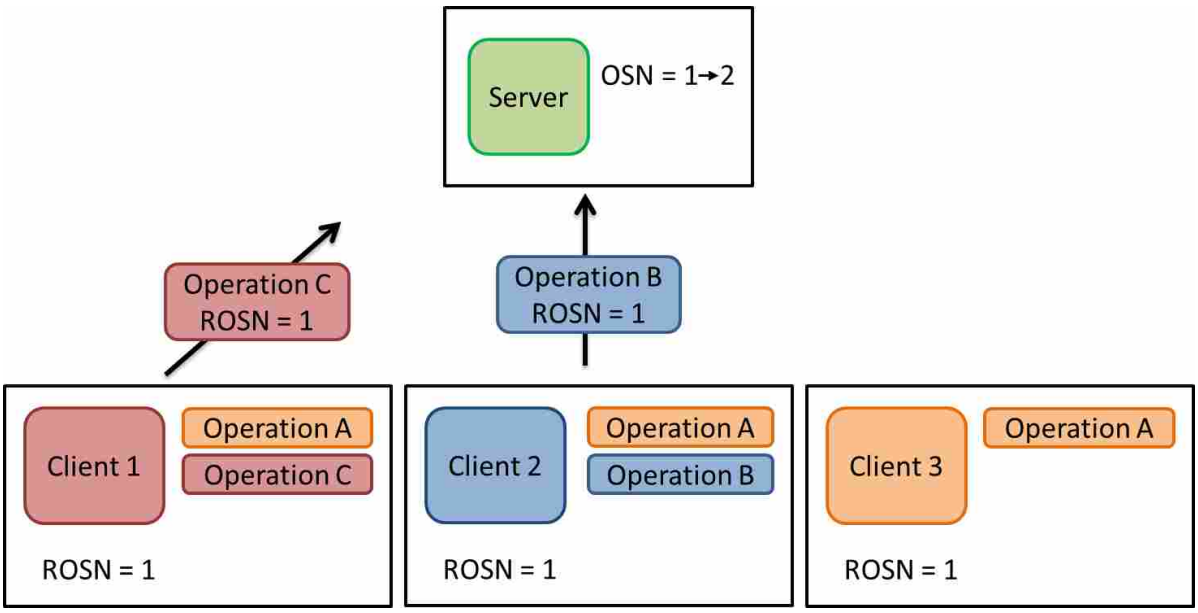


Figure 3.2: Operation B is received by the server first, the OSN is incremented, and operation C is ignored when it arrives

When an operation becomes verified by the server, a verification message is sent from the server to the client to verify that an operation was accepted by the server. The OSN is sent with the verification message so that the client ROSN can be updated upon verification. The operation o_i is added to the set v and removed from set u on the originating client. At the same time that the verification message is sent back to the originating client, operation messages are sent to all other clients in the part. Clients apply the operation o_i and add it to the end of set v. If clients have any unverified operations in set u when an operation message is received, the remote operation is inserted before the unverified operations. This is because eqn. (3.6) demands that all verified operations v, must be ordered before unverified operations u, in the set of operations O.

Fig. 3.3 depicts the operation insertion and ROSN update. It is a continuation of the process started in Fig. 3.2. This shows operation B being sent to client 1 and 3 and a verification message for operation B sent to client 2. It also depicts what happens after the messages are received by the clients. Each of the clients receives a new OSN and updates the ROSN. Operation B is applied to client 1 and 3. In client 1, the local operation C is not yet verified so operation B from the server is inserted before it. Operation C exists only on client 1 because it has not yet been verified.

Once a client has applied remote operations, any local unverified operations are resent to the server with the newly updated ROSN included. When they reach the server and the ROSN matches the OSN, the OSN is updated. Verification and operation messages are sent to the clients, the operations are applied, and the ROSNs are updated. Because each unverified operation u, is re-sent in a message to the server after any remote operations are applied, each client will re-send any unverified, valid operations after either generating new operations or applying operations from the server. This will be done until

$$u(P) = \{\} \tag{3.9}$$

And thus

$$v(P) = O(P) \tag{3.10}$$

Since eqn. (3.7) is also met, the totally ordered set of operations O, on the client, in part P is equal to the totally ordered set of verified operations on the server V

$$O(P) = V(P) \tag{3.11}$$

Fig. 3.4 depicts what happens after re-sending operation C from client 1 to the server. In this case the ROSN and OSN match so the operation becomes verified. This shows verification being sent to client 1 and the operation message being sent to clients 2 and 3. Each client receives a new OSN from the server which is reassigned as the ROSN on the client.

In summary, the global order of operations is determined by the order in which operations are received by the server. All unverified operations, valid or invalid, must be sent to the server before they are forwarded onto other clients and saved in the database for persistent storage. Any operation that arrives with an out of date ROSN will be ignored by the server and must be resent after the sending client has synced up. Any unverified operations stay local to a client until they become verified. All verified operations received by a client are inserted before any unverified operations. Thus, this method ensures that all operations are performed in the same order on all clients and guarantees eventual consistency between clients meeting the condition of eqn. (3.11).

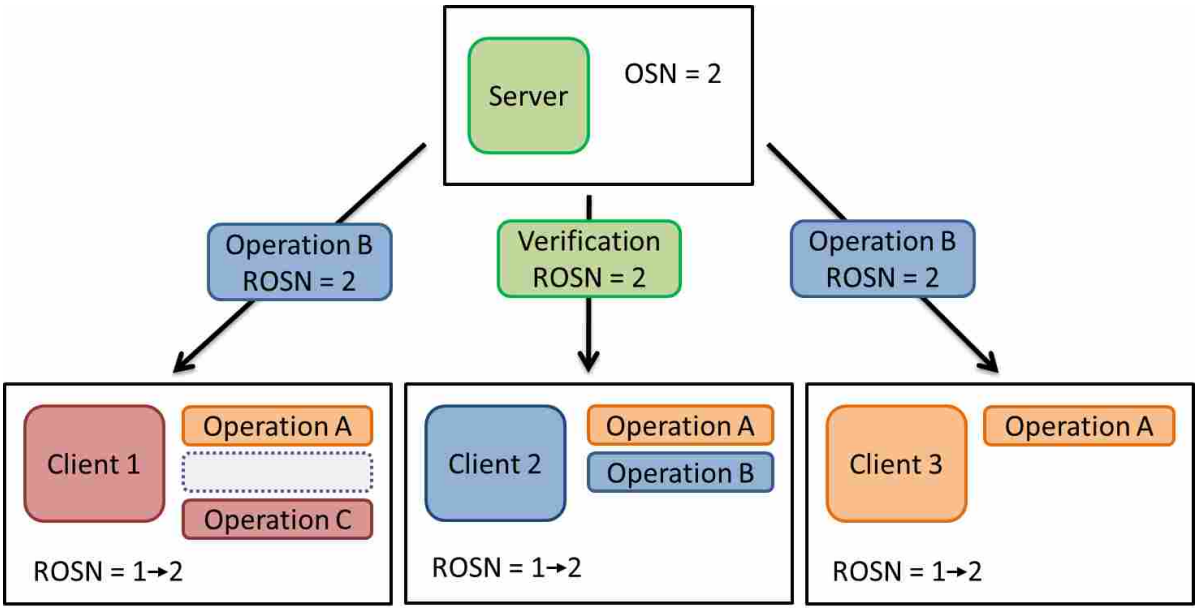


Figure 3.3: Operation B is sent to clients 1 and 3, verification is sent to client 2, the ROSN is updated on all clients and operation B is applied before operation C on client 1

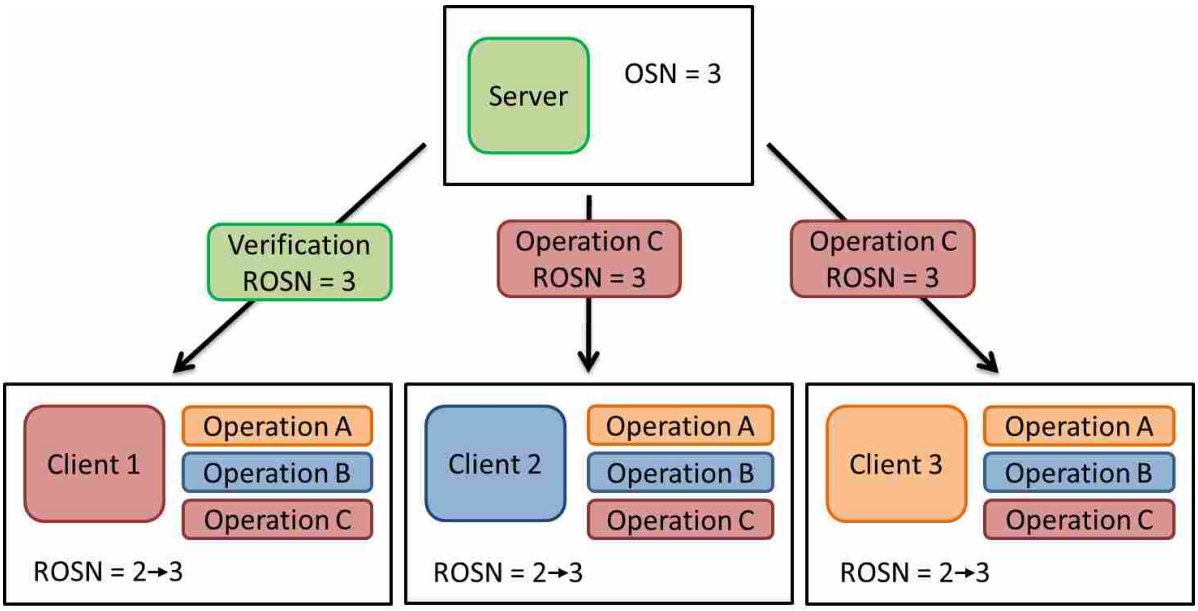


Figure 3.4: After operation C is re-sent to the server, verification and operation messages are sent to clients and the ROSN is reassigned

Client-Side Syncing System

In addition to the logic described above, additional logic is in place to provide a more efficient way for eventual consistency to occur. This logic implements a client-side syncing system

(CSS) which queues the remote operations received from the server while performing an operation on a client. Since no remote operations can occur while a local operation is being performed, remote operations are placed in the queue, set *q(P)*, to be applied after the operation is completed. When an operation is completed on a client, the new, local operation $o_i(P)$ is added to the end of the unverified operation set *u(P)*, resulting in the final unverified operation set $u_f(P)$.

$$u_f(P) = u(P) + o_i(P) \tag{3.12}$$

The operations applied from the queue are added to set of verified operations in like manner.

$$v_f(P) = v(P) + q(P) \tag{3.13}$$

Thus, due to the condition in eqn. (3.6), operations in the queue are ordered before the unverified client operations. Once verified operations are applied, if the new operation which was added to set u is valid, it is sent to the server for verification. Fig. 3.5 shows what happens when a new operation is complete while there are operations in the queue. The left image shows that the new operation is applied after operation B. The right image shows the operations in the queue are then applied before the new operation so that the order becomes consistent globally.

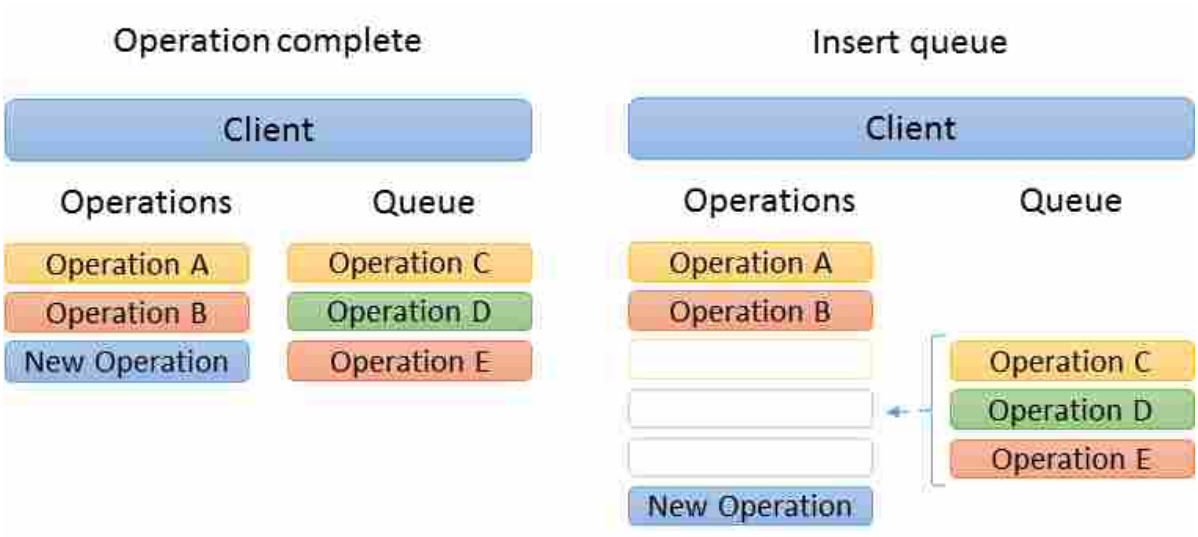


Figure 3.5: Left: The new operation is applied after it is complete; Right: The operations in the queue are applied before the new operation

The CSS makes client operations much more likely to be verified by the server the first time they are sent. This is because remote operations are applied even before the new operation is sent to the server for verification. Since these operations came with a more recent ROSN, the client is more likely to have an operation verified by the server when the operation is sent. This allows for the client to become up to date more quickly and reduces the potential number of messages sent between the clients and server. While the CSS logic makes the system more efficient, the OSN logic is critical to ensure data consistency for all cases. Therefore, the combination of the CSS and OSN logic provides an efficient system for operational consistency.

3.3.2 Conflict Resolution with Data Preservation

The operational consistency method provides the framework to keep conflicting syntactic operations from causing model inconsistency. This is because it guarantees that the first operation to reach the server will trump any conflicting operations which reach it afterwards. Since any remote operations which reached the server first are locally inserted before any unverified operations, this gives the local system a chance to test the operation for conflicts. If after the insertion takes place, the unverified operation fails, the operation is known to be in conflict with remote operations which reached the server first. If this is the case, the operation becomes invalid and is not sent to the server. When operations become invalid due to a conflict, the simplest approach is to revert and dispose the invalid operation. However, if preservation of user operations is possible, invalid operations should be set aside, yet retained so that the user has the opportunity to edit the operation so it becomes valid again. This is preferred because it reduces the amount of rework time by the client who originated the invalid operation.

The decision on whether or not operation data should be preserved depends on the type of operation performed: creation, edit or delete. Invalid creation operations should always be preserved because a creation operation will never interfere with any previous operations, and invalid operations are always ordered after valid operations. In other words, creation operations will not cause other operations to become invalid. However, invalid edit operations can cause other operations to become invalid and thus should be reverted. Invalid delete operations should also be reverted because deletes are a single click operation and thus have no consequential data to pre-

serve. In our implementation, invalid creation operations are the only type of operation which seem practical to preserve.

Invalid creation operations are preserved in the following way: If a creation operation becomes invalid, it is marked with a red X and goes into a suppressed state on the local client who originated the operation. The suppressed state preserves the operation visually in the CAD system for the user to access the operation data. The user has the option to modify the part to make the operation valid. The operation may be edited for compatibility with previously applied operations. Alternatively, the user may choose to delete or revert other operations which cause that operation to be invalid. Once the operation becomes valid, it is treated as a new operation and sent to the server. Fig. 3.6 depicts the process of a user correcting a suppressed, invalid operation. Client 1 corrects the error in operation C and sends it to the server. The operation is received by client 2 and the clients become consistent with each other.

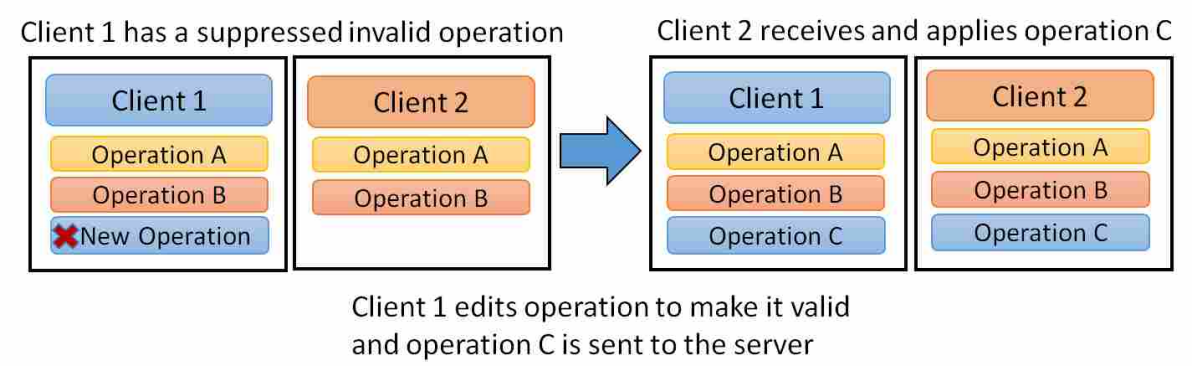


Figure 3.6: Client 1 edits operation C to validate it and it is forwarded to client 2

In summary, the operational consistency method keeps conflicting syntactic operations from causing model inconsistency. Data is preserved by keeping the operation in a suppressed state so that a correct to the model can turn this operation valid. This reduces rework time by the user who created the invalid operation originally.

3.4 Implementation in NXConnect

The operational consistency method on the server side have been implemented in NXConnect as follows: The server has an integer value representing the OSN for each multi-user part it

supports. When the server receives an operation message for a given part, the ROSN that came with the message is compared to the OSN. If the ROSN matches the OSN, the message is forwarded to all other users in the multi-user part and the operation data is stored in the database. A verification message is also sent to the originating client. If the OSN does not match, the message is ignored by the server.

The client side is a bit more involved and is implemented as follows: The client has an integer value representing the ROSN for each part loaded on the client. Multiple parts may be loaded by a client, especially if they are modeling in an assembly. Although a user may have multiple parts loaded, only one part is the work part. The work part is the only part in which operations may be applied. When a user switches between parts, the part is synced with the server and a new ROSN is received from the server for that part.

While actively modeling in the work part, an operation message is sent to the server when a valid operation is completed. The ROSN specific to the work part is included in the message and is used for verification. Once verified by the server, an operation message is sent to all clients who have the part loaded. If a client does not have that part as the work part, the message is ignored by the client. He will receive these updates when he makes the part the work part and it is synced with the server. However, if the client does have the specific part as the work part and they are in an accessible state, they apply the operation. After the operation message is received and the operation is applied by the client, the ROSN for the work part is updated.

A queue exists on each client for every part loaded which contains all operation messages received by the client while in an inaccessible state. To avoid confusion to the user, our implementation puts the client into an inaccessible state while a user is actively creating or editing with a feature dialogue open. This way, no remote operations are applied to the model while an operation is being performed. When the client returns to an accessible state, all creation operations performed while in an inaccessible state are suspended. Edit and deletes performed while in an inaccessible state are undone. The queue is then applied. The creation operations are then reinstated and edits and deletes are redone. If a creation operation is valid upon reinstatement, it is sent to the server and added to the client's unverified operation list. If it is invalid, it means the local operation conflicted with a remote operation and becomes suppressed with a red X, thus preserving the operation locally (see Fig. 3.7). If edits and deletes cause any feature to become invalid, they

are discarded and the user is informed of the discarded operation. Otherwise, they are sent to the server.

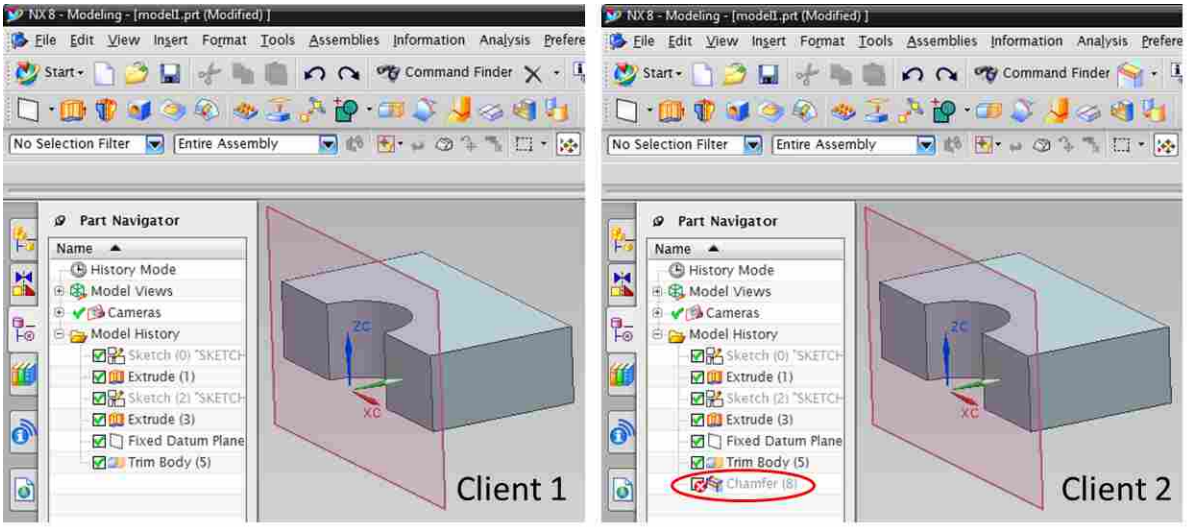


Figure 3.7: The result of conflicting operations leaving an invalid operation on client 2

The client has an unverified operation list for each part which keeps track of all operations which have yet to be verified by the server for a given part. If an operation message is received by the client when there are unverified operations in the unverified operation list, a similar set of actions occurs as when applying the queue. All unverified creation operations are put into a suspended state and unverified edits and deletes are undone. The remote operation is then applied. The creation operations are then reinstated and edits and deletes are redone. If a creation operation is valid upon reinstatement, it is sent to the server. If it is invalid, it becomes suppressed with a red X. If edits and deletes cause any feature to become invalid when they are reapplied, they are discarded and the user is informed of the discarded operation. Otherwise, they are sent to the server. When a verification message is received from the server, the operation which it verifies is removed from the unverified operation list.

Another situation in which the operational consistency method is implemented is when a user opens a multi-user part in NXConnect and joins the multi-user part session. This is applied as follows: the ROSN is set at the time the part is opened. Once the part is opened, it is synchronized with the server. This is a process in which all operation data in the database since the last save

operation is applied to the part one by one (see [3] for details). During synchronization, the part is not in an accessible state, so all remote operations are added to the queue. Once synchronization is complete the queue is applied as described above. This implementation ensures model consistency while users are joining a multi-user part session.

3.5 Results

Several tests were performed to show the validity of the implementation of the operational consistency and conflict resolution with data preservation methods in NXConnect. The first test demonstrates the implementation on a model that was shown in the introduction. The test has two clients simultaneously perform two operations which are not conflicting, but are order dependent. Client 1 performs a shell operation and client 2 performs trim operation on the cube (see Fig. 3.8). The two clients finish their operations at nearly the same time, but the trim body on client 2 reaches the server first. Client 2 receives the shell operation and applies the operation. When client 1 receives the trim, it is reordered before the shell because the shell operation had not been verified (see Fig. 3.9). This test shows that the implementation in NXConnect ensures a global order of operations to ensure model consistency.

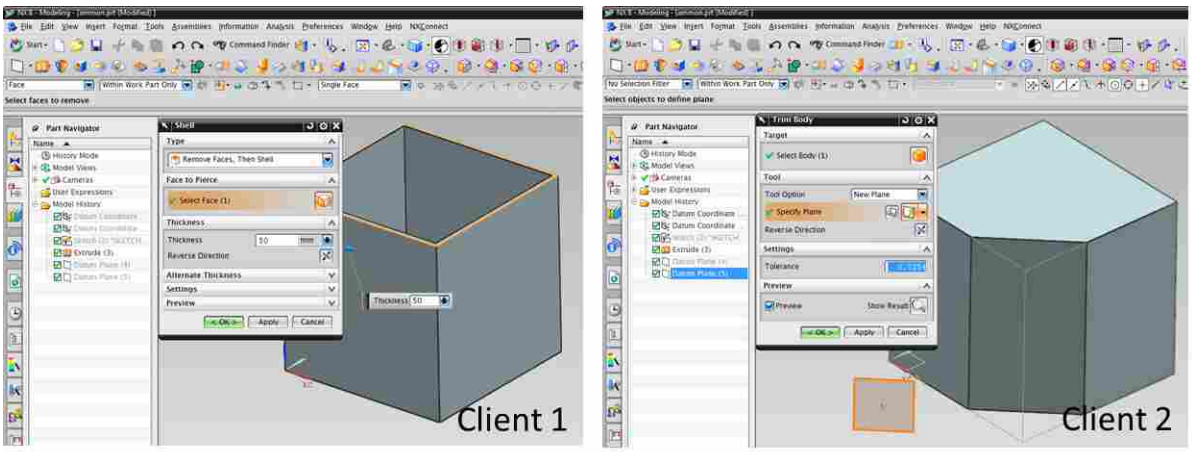


Figure 3.8: Client 1 performs a trim body while client 2 performs a shell on the same body (before)

The next test demonstrates the validity of the implementation of the conflict resolution with data preservation method on a model similar to the second example illustrated in the introduction.

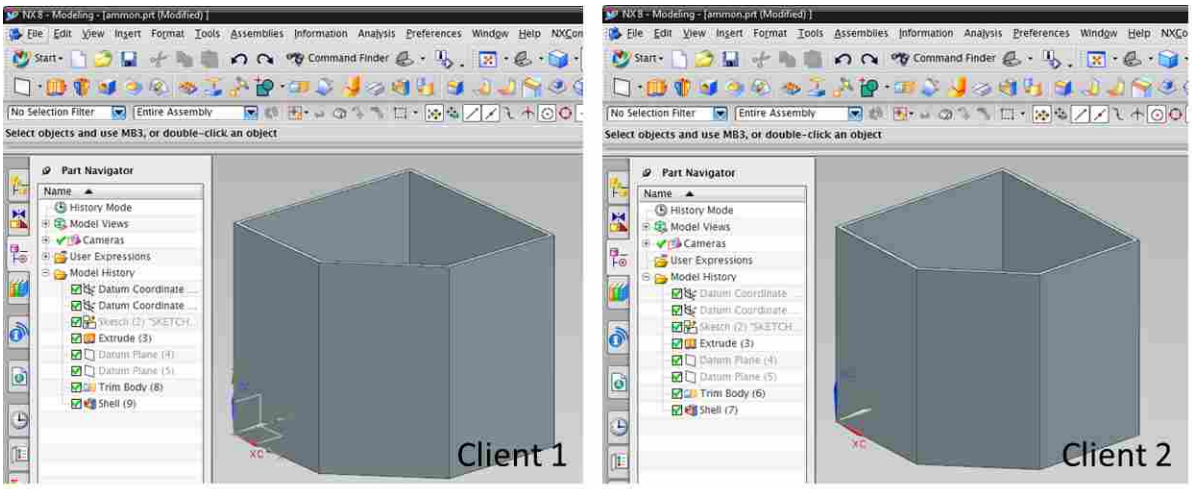


Figure 3.9: The trim body on client 1 is inserted before the shell so the clients become consistent (after)

The test has two clients simultaneously perform conflicting operations on the same edge, resulting in a child/child conflict. Client 1 performs a chamfer while client 2 performs an edge blend both to the top right edge of the cube. The two clients finish their operations at nearly the same time, but the edge blend on client 2 reaches the server first. Since the chamfer on client 1 is not yet verified, the edge blend is applied before the chafer. Since these two operations are mutually exclusive, the chamfer fails and the operation is not sent to the server. This results in the chamfer being suppressed with a red X on client 1 as illustrated in Fig. 3.10. This test shows that the implementation in NXConnect ensures a global order of operations while still preserving invalid feature operations resulting from child/child conflicts.

The third test case demonstrates how the implementation of the operational consistency methods prevents parent/child conflicts. In this case, the clients start with a model containing a cylinder which intersects a cube illustrated in Fig. 3.11. The cylinder and cube are a separate bodies in the model.

In this test, client 1 performs a unite operation of the cylinder and cube. At the same time, client 2 edits the extrusion length of the cube so that it no longer intersects the cylinder. They both finish their operations at nearly the same time. This causes a parent/child conflict because the parent of the unite operation (the extrusion forming the cube) changes, making the unite operation not possible. In the case where client 2's extrusion of the cube reaches the server first, the unite

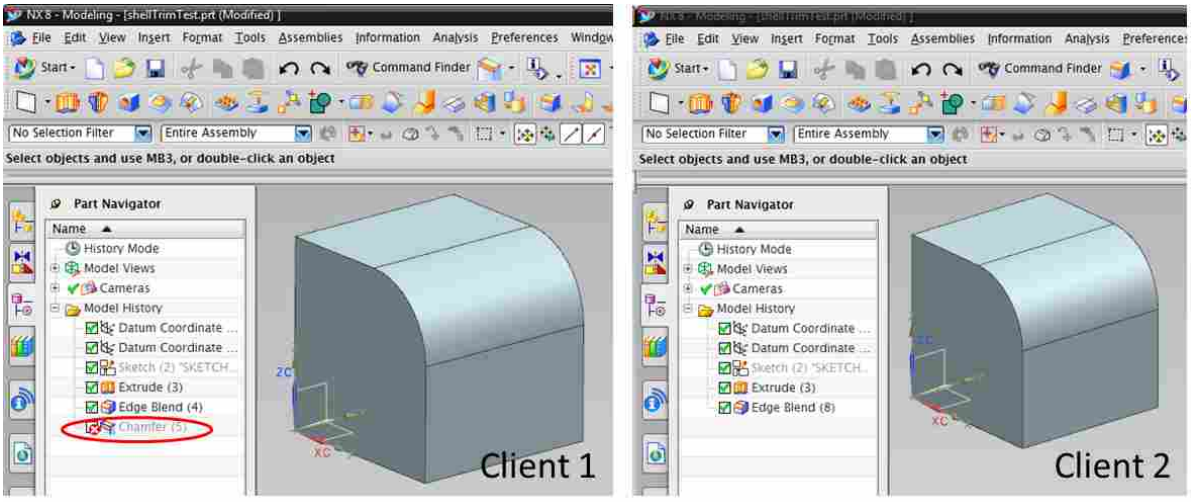


Figure 3.10: Client 1 performs a chamfer on an edge while client 2 performs an edge blend on the same edge

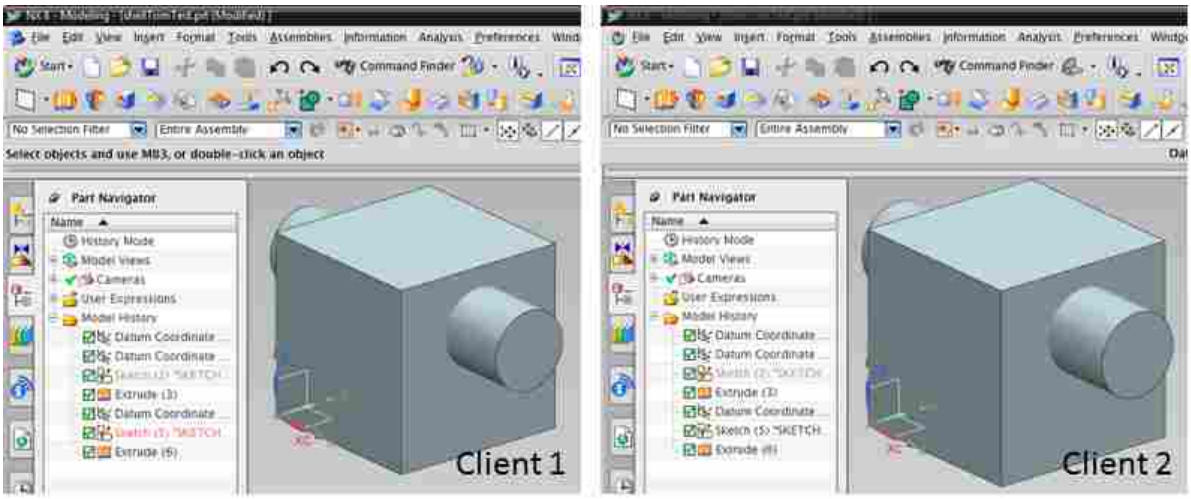


Figure 3.11: Client 1 and client 2 have a consistent model of a cylinder intersecting a cube (not united)

fails and is not sent to the server. The unite operation is preserved on client 1 as a suppressed feature with a red X as shown in Fig. 3.12.

Conversely, in the case where client 1's unite of the cube and cylinder reaches the server first, the edit of the extrusion is undone as shown in Fig. 3.13. The edit operation is not preserved because it would cause the unite operation to fail.

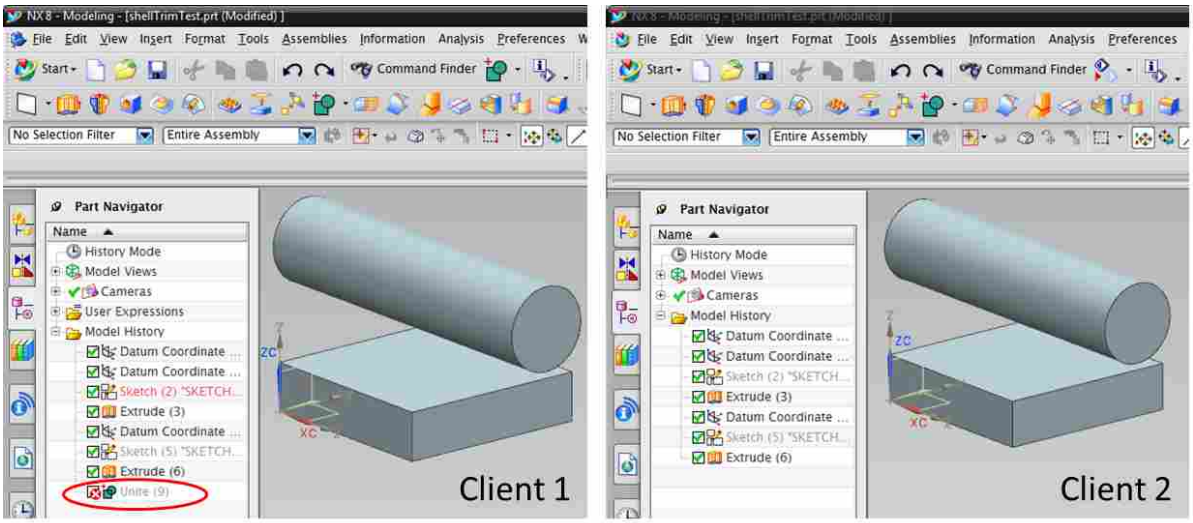


Figure 3.12: Client 1 unites the cylinder and cube while client 2 edits the extrusion; client1 finishes first

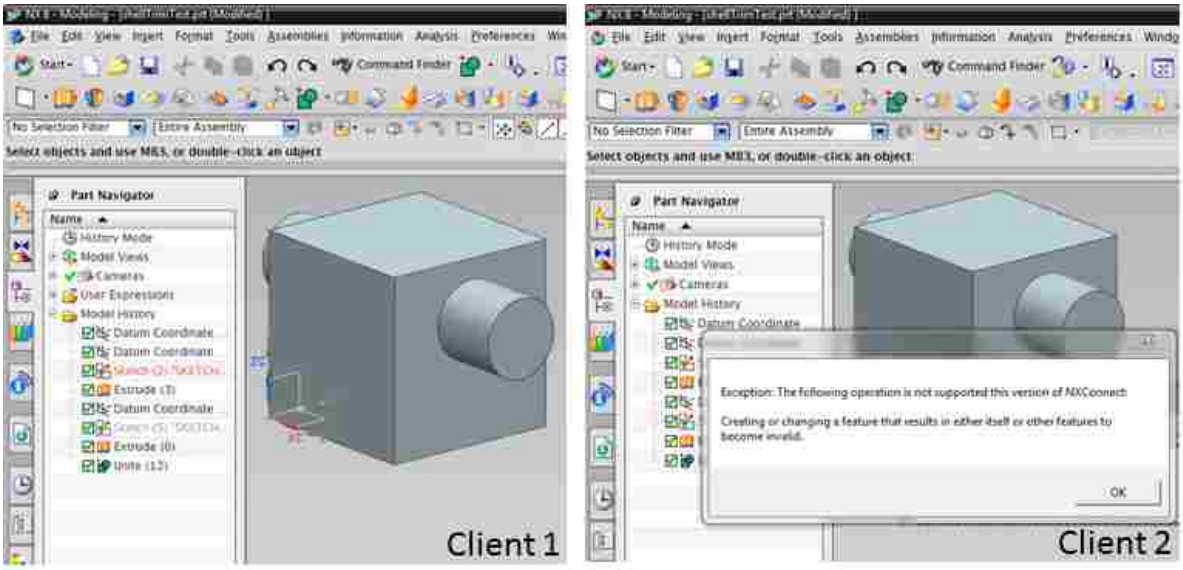


Figure 3.13: Client 2 edits the extrusion while client 1 unites the cylinder and cube; client 2 finishes first

These tests demonstrate the validity of the operational consistency method to ensure model consistency in a simultaneous multi-user CAD system. This is evidenced by the first test which shows that each client performs all operations according to the global order of operations. The other tests show that the operational consistency method keeps child/child and parent/child conflicts from causing syntactic violations and model inconsistencies. These tests also show the

validity of the conflict resolution with data preservation method. Conflicting creation features which don't reach the server first are preserved on the client who originated them. Conflicting edit or delete operations which don't reach the server first are undone to prevent the failure of pre-occurring operations. In addition to these tests, Appendix A shows more complex models which were created in the multi-user CAD system with these methods implemented. These examples qualitatively show the robustness of these methods for industrial scale modeling in multi-user CAD.

3.6 Conclusions

The operational consistency method is presented which ensure a global order of operations for a simultaneous, replicated multi-user CAD system. This relies on an operation sequence number on the server which only verifies operations based on the last operation received. A client-side syncing system utilizes a queue in which remote operations are added while not in an accessible state. The operational consistency method ensures that all verified remote operations are processed before any unverified local operations. This method also ensures that no conflicts will occur globally. The conflict resolution with data preservation method provides that any operations which cause conflicts are preserved. Edit and delete operations which cause other features to become invalid are discarded. Invalid creation operations are preserved locally until the user modifies the operation to become valid or deletes the operation.

The results of the implementation of the methods show that they provide for model consistency in a functional multi-user CAD system, namely NXConnect. The results reflect that these methods prevent models from diverging in the cases where the order of operations is critical in determining the resultant geometry. They also show that both parent/child and child/child conflicts are resolved in the global model. Invalid operations caused by conflicts are preserved for creation operations. Invalid operation preservation reduces the amount of rework due to conflicting operations. In addition, the examples shown in Appendix A qualitatively show the robustness of these methods for industrial scale modeling in multi-user CAD.

CHAPTER 4. EFFICIENT PERSISTENT NAMING FOR MODEL CONSISTENCY

4.1 Introduction

In addition to a global order of operations to ensure consistency, references to the topological entities in a CAD part on one client must be the same on all clients to ensure operations are applied consistency. Feature-based CAD systems create features that parametrically reference topological entities which include faces, edges and vertices. A consistent reference to topological entities is required for dependent features to be applied to the same topological entity on all clients. For example, a fillet feature operation applied to an edge in one client needs to be applied to the same edge in all other replicated client models. If the system fails to keep names of features and entities persistent between clients, models will become inconsistent, causing errors to occur. This issue is referred to in the literature as the persistent naming problem [21].

Because topological entities of the geometry kernel are not necessarily returned in a predictable order, faces and edges they are uniquely identified by their geometric properties. These properties are mapped to a unique identifier and stored in the operational data forwarded to other clients. When the clients receive the remote operation, they identify the bodies, edges, and faces by matching the geometric properties to their corresponding identifier. This method is referred to as eager naming because it identifies all the topological identities as soon as they are created. This chapter will present the eager naming method as a robust solution to the replicated multi-user persistent naming problem. It will also discuss the performance issues with this approach and present an optimized method for efficient persistent naming.

4.2 Background

Persistent naming has been a problem for researchers since the beginning of history-based parametric solid modeling. This is because two different models represent the part. One is the para-

metric model, which consists of modeling operations, and the other is the geometric model [21]. The main issue has to do with keeping the faces, edges and vertices of the geometry model consistent with the parametric model when it is reevaluated from the operational history. Directly using computer memory pointers for identification is not a valid technique because they are transient. Simple enumeration methods do not always work because model edits change topology and the enumeration is no longer legitimate [43]. Several authors have presented solutions to the persistent naming problem in single-user CAD [44–52].

In replicated, collaborative CAD systems, persistent naming is concerned primarily with uniquely identifying topological entities on various remote clients. Jing et al. recognize that naming topological entities directly from modeling history may not be a valid solution if operations on various clients may be performed in a different order. They present methods to solve this by naming topological entities of a given feature in a consistent order [53–55]. One assumption that was made in their implementation is that topological entities are returned in a predictable order. However, this is not the case with all geometry kernels. Siemens Parasolid, for example, does not return faces, edges and vertices in a predictable order [40]. The ordered naming method is also not possible if persistent naming is required across multiple different CAD systems. Jing et al. present methods for persistent naming in a replicated, multi-user CAD system across multiple different CAD systems. This technique is based on geometric properties of the object and are limited to simple geometry and swept features [56].

4.3 Eager Persistent Naming Method

4.3.1 General Naming Methodology

Our method to generate unique identifiers on features and topological entities is as follows:

1. A prefix, '$', identifying the feature as having been named
2. The users unique username, guaranteeing that each users features will have names distinct from other users features
3. The feature type which describes the feature for convenience and readability

4. A sequentially generated integer id, guaranteeing uniqueness from all features on the same client

So, the 57th feature created by a user with username Ivan would receive the name "$Ivan_FeatureType_57," on his client, in the database, and on all other clients. Similar naming conventions apply to all other named entities.

4.3.2 Topology Identification Methods

The general naming methodology discussed in the previous subsection apply trivially to the identification of features, curves, expressions, dimensions and constraints because they can be named directly when the object is created at the client level. In addition to naming these, the identification of bodies, faces and edges must also be consistent across multiple clients because they are referenced in the creation of features, curves, expressions, dimensions and constraints. However, doing this offers some unique challenges. For example, bodies, edges and faces in the Parasolid geometry kernel are "not returned in any predictable order" [40]. Therefore it requires custom methods to identify the same bodies, faces and edges in a part across clients based on their geometric, topological and feature qualities.

Methods have been developed to identify common bodies, faces and edges across clients and are executed in the following order:

1. Edges are identified by querying data on the edge at discrete values along its length with a given tolerance (see Fig. 4.1)
   (a) Linear edges will only require end point data
   (b) Non-linear edges may require more resolution to uniquely identify
2. Faces are identified by querying data at discrete values across the face with given a tolerance (see Fig. 4.2)
   (a) Planar faces will not require discretization if all edges are identified uniquely. This is because they can be inferred from the bounds of the edges
   (b) Non-planar faces may require more resolution to uniquely identify

3. Bodies are identified by using all of its faces and edges which are already identified

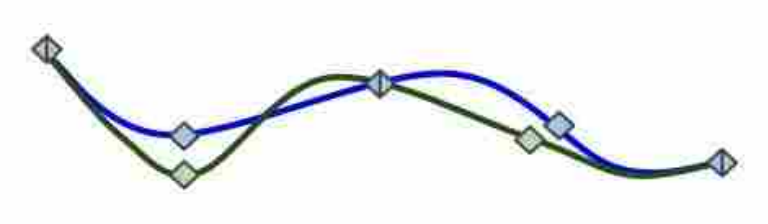


Figure 4.1: Identification of unique edges by discretization

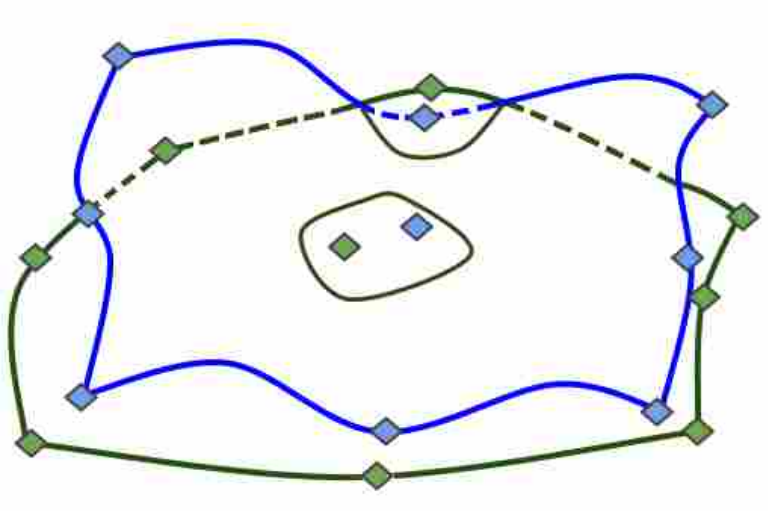


Figure 4.2: Identification of unique faces by discretization

This approach uniquely identifies bodies, faces and edges except for in two cases: 1) identical topological entities occupy the same space in the part, and 2) all data compared on the entities match, but varies between the discrete samples. The first case is solved by inferring the topological entity from the feature it was generated from to identify its uniqueness. For example, two edges which occupy the same geometric space will always be a product of different feature creation operations. In this way, uniqueness is determined for identical topological entities based on the feature that generated it. The second case is solved by having a sufficiently small enough sampling discretization for edges or faces so as to uniquely identify it. However, a fine discretization resolution may take more computation than is desirable. A method optimized to reduce computation time will

provide an iterative approach to refine resolution as required, adding sampling points to compare the geometry as needed. However, this case is so unlikely that it is not be worth implementing.

4.3.3 Eager Naming Implementation

The above mentioned topology identification methods have been implemented into NX-Connect using the NX Open API in three specific ways: the first uses multiple types of geometric data for the object as a whole, the second uses only positional data at multiple positions and the last is a combination of the first two. Each implementation uses the same general flow, in which geometric data from the edges, faces and bodies are saved to the database along with the feature parameters. Just after feature recreation, that data is compared against each edge, face, or body, until a match is found, at which point the object is given the name associated with that data.

The first implementation uses different types of data for different objects. For edges, that data consists of the endpoints of the edge, and the tangent vector and curvature at one of the endpoints. Faces use a face normal and the inverse of the maximum radius of curvature; to reduce the number of times that data must be calculated, only faces with the same number of edges as the target face are compared. Bodies are then identified by the positive identification of all their constituent faces and edges. The API has a bug, however, which prevents this method from being completely accurate. The method used to calculate normal vectors, tangents and curvatures (the NXOpen.GeometricAnalysis. GeometricProperties class of the .NET API) sometimes returns garbage data for objects created using the API (as opposed to through the GUI). Thus, the method returns correct data for features being saved to the database, but incorrect data when attempting to identify them after recreation.

The second implementation uses only positional data, but increases the number of samples, using the somewhat randomly chosen parameter values. For edges, this leads to four points for comparison, including the endpoints. Given the additional complexity of faces, only two points are deemed sufficient to reduce the collision probability to acceptably low levels, and as in the other method, only faces with equal numbers of edges are compared. Bodies are again identified by the constituent faces and edges. This implementation also has an API-related problem, in that the parameterization of edges and faces is sometimes different depending on whether the feature is being created or edited, or using the API vs. the GUI. Thus, two samples taken at the same

parameter values from the same face on different computers may possibly have different values, and the algorithm fails to identify the face.

Since both of these implementations work much of the time, but API bugs cause them each to fail in certain cases, the last implementation is a combination of both these cases. Since both approaches are used together, it offers a safety net so that when one approach fails the other may catch it. This redundancy reduces the total number of times it fails overall and is shown to be a fairly robust approach as shown in the results section of this chapter.

4.3.4 Eager Naming Implementation Results

The persistent topology naming method has allowed complex geometry associations to be produced within NXConnect. Despite the fact that the NX API has known bugs which cause errors to occur in the NXConnect implementation for this method, the combined implementation of both approaches discussed above is shown to be reasonably robust. This is shown by the use of an internal tracking system to observe the robustness of this implementation. Over a three-week period, about 22 active users clicked at least 1771 body-generating commands. Each command could generate multiple bodies, or if the user cancelled the command, no bodies. The data reflect some usage of ordinary NX, some debugging of NXConnect, and significant large-scale modeling, including the tractor demo discussed above. During this period of use, only 22 bodies with identification problems were created. Given reasonable assumptions (most of the usage was in NXConnect, and an average of one body was generated per body-generating modeling command), this suggests the topology naming scheme is on the order of 99% effective. These numbers reflect the effectiveness of the naming methodology despite known API bugs associated with these implementations; resolving these should further reduce issues related to topology identification.

4.4 Performance Problems with Eager Naming

4.4.1 Feature Creation Performance Problem

While the eager naming method does ensure consistent identification for all clients, it can take a considerable time to identify the topological entities of a feature creation operation which

generates many bodies, edges, or faces. Fig. 4.3 shows an extrusion of a grate that creates 406 faces and 1212 edges. The extrude feature creation operation for the grate takes approximately 5 min. to identify all the topological entities using the eager method and is consider excessive. Each client will need to perform this same naming algorithm on their local machine, causing interruption to the user workflow by forcing the user to wait until the naming operation is complete so local client work can continue.

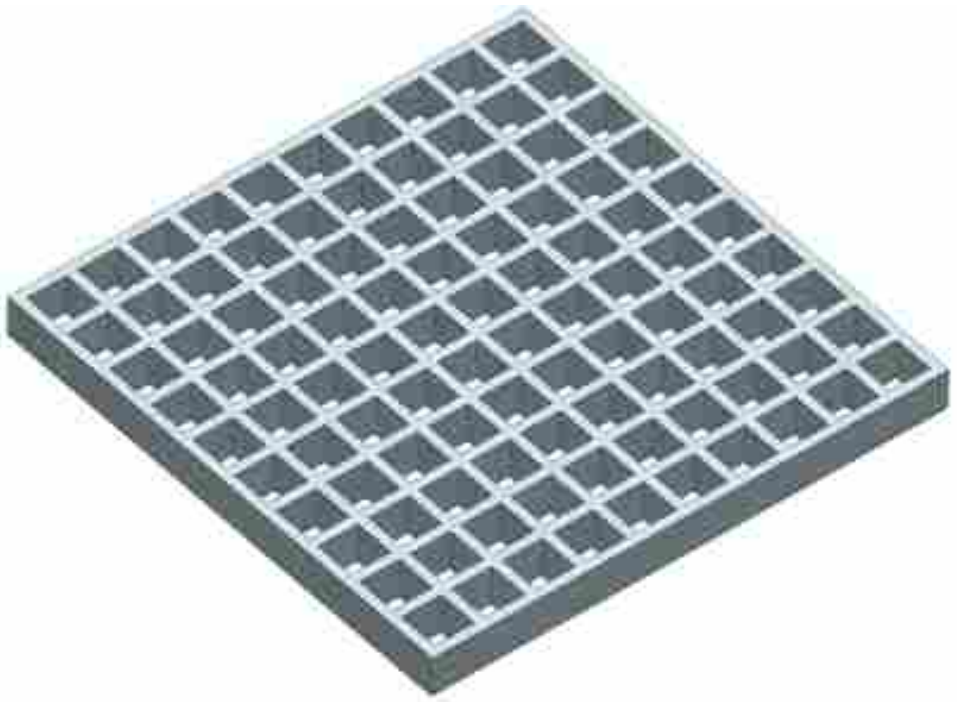


Figure 4.3: The extrusion of the grate takes about 5 min. to identify faces and edges

4.4.2 Feature Edit Performance Problem

The eager naming method further interrupts the modeling process when a feature edit operation is performed. A feature edit operation causes not only the selected feature to change, but also any dependent features to update as well. Each updated feature can cause any number of faces and edges to change shape, size, or location. Because the eager naming method stores topological entity data together with feature operation data, the geometric properties of every topological entity of every dependent feature must be resent to the server with the edit operation data. This results

in a large amount of data being sent between clients, as well as a significant amount of time being spent re-identifying the topological entities of the dependent features.

The examples shown in Fig. 4.4 illustrate the feature edit performance problem well. Image A shows a gear which has 20 teeth and a hole through the center. The gear with 83 faces and 242 edges is created by an extrusion operation which takes about 22 sec. to perform with the eager naming logic. Image B shows fillets applied to all 80 of the vertical edges on the side of the gear creating 80 additional faces and 240 additional edges taking about 37 sec. Image C shows the extrusion edited to thicken the gear, changing the size of all the edges and faces in the model. The extrusion edit takes about 41 seconds to perform, which is longer than the time it takes to perform the fillets. The time to perform this operation includes the time for extracting the geometric data for each face and edge in both the extrude and the dependent fillet operations.

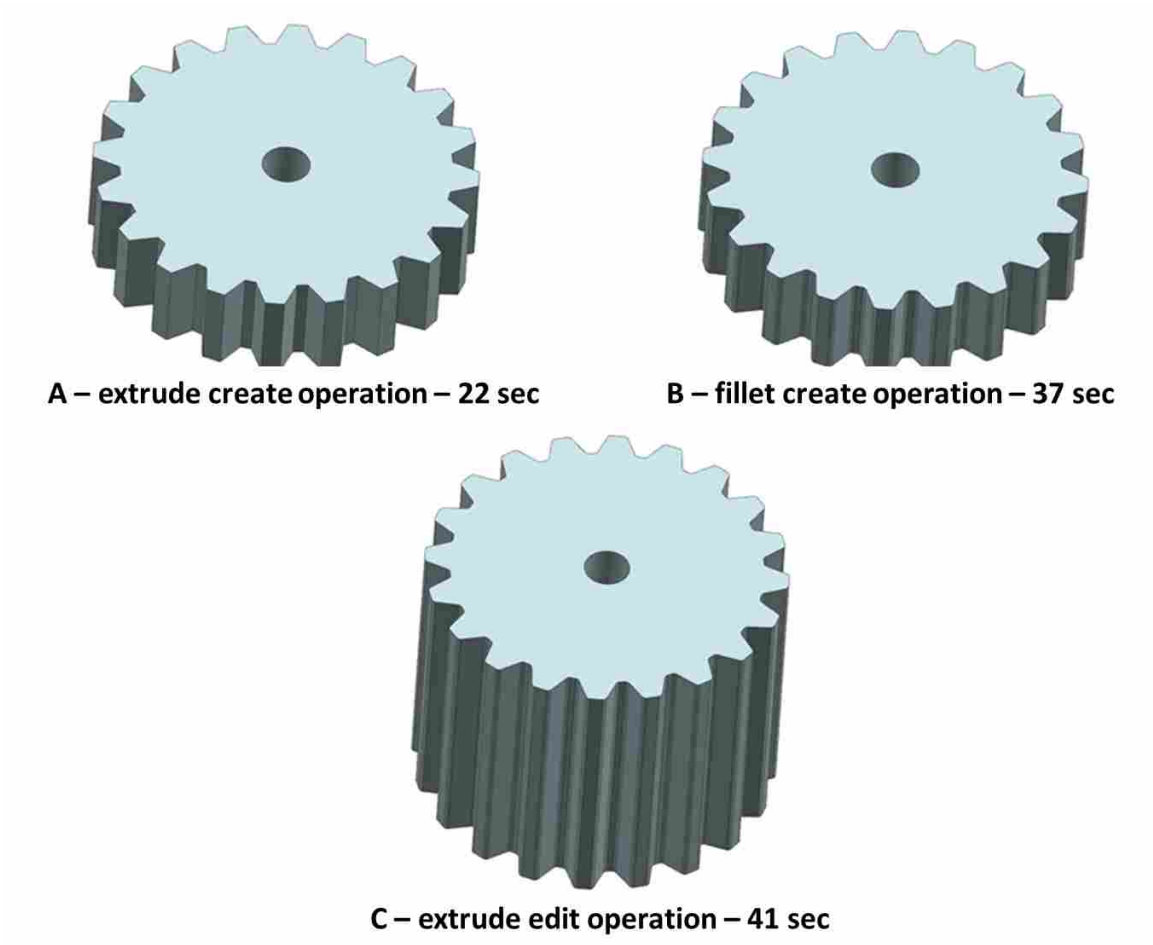


Figure 4.4: A: Gear extrusion; B: Fillets added to gear sides, C: Thickened extrusion

4.4.3 Performance Enhancements

The eager method, though effective, causes unnecessary delay for the most basic of models. For large models or assemblies, the eager naming method severely hinders the multi-user, collaborative CAD experience. Herein is presented an approach based on the geometric identification principles found in the eager naming method, but overcomes its shortcomings with three major enhancements.

The first enhancement solves the feature creation performance problem using lazy naming. Lazy naming shifts the identification paradigm from naming each topological entity immediately after its creation, to a scheme which names topological entities just before they are referenced by dependent features. This relies on the principle that the number of referenced topological entities is always less than the number of topological entities generated in a history-based model. The method is termed lazy because it waits to identify topological entities when they are actually needed. Lazy naming dramatically improves feature creation time because identification does not occur until a topological entity is actually referenced.

The second enhancement solves the feature edit performance problem. It persistently stores the topological entity data separate from the feature data in the database and stores a cache of the same data in a client entity dictionary on each client. This second enhancement eliminates the requirement of the eager method to extract the geometric properties of all dependent features after a feature edit.

The third enhancement additionally helps improve performance for edit operations in the Siemens NX multi-user prototype. This implementation enhancement is called enhanced geometric property extraction and uses a more efficient method to directly query the geometric properties of faces and edges. This is accomplished by taking advantage of the internal NX identification system (Journal IDs) which more quickly extracts the geometric properties of the entities. These three enhancements dramatically decrease the time for naming in feature creation and editing for multi-user CAD, thus improving the overall multi-user CAD experience.

4.5 Lazy Naming to Enhance Feature Creation Performance

The eager naming method identifies and names every face and edge immediately after a feature is created. This method of identifying topological entities is computationally expensive for models with many faces and edges. Conversely, lazy naming identifies and names topological entities when they are used by dependent features. Since the number of referenced entities is far less than the total number of entities that exist in the model, lazy naming saves a significant amount of computation time not having to identify nearly as many topological entities.

For example, when creating fillets on the top edges of a cube (as in Fig. 4.5), only the four edges that are referenced by the fillet operation are identified and named with the lazy naming method. Conversely, in the eager method, each edge and face in the cube are identified and named immediately after creation. In addition, after the fillets are applied, all the newly created edges and faces must be named. This eager process would require the identification and naming of 24 edges and 10 faces.

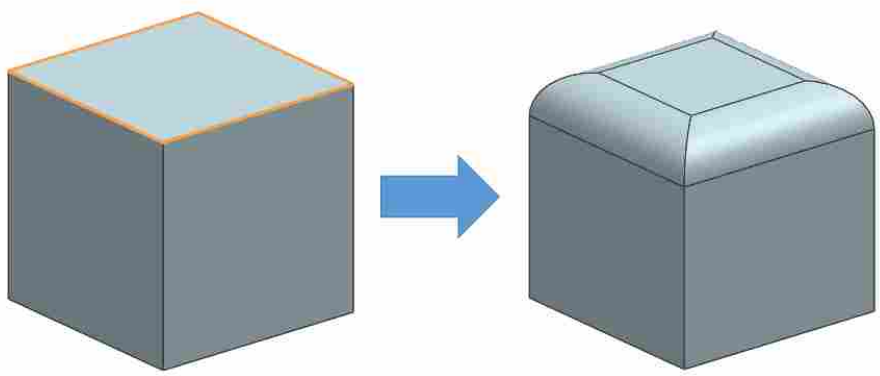


Figure 4.5: The fillet operation with the eager method identifies 34 entities vs. the lazy method which requires identifying only 4

The lazy naming method functions as follows:

1. An operation which references topological entities is performed by a client
2. The referenced topological entities of the operation are given unique names by the client

3. Unique geometric properties are found for the topological entities

4. The body on which the named topological entities reside are identified (see details below)

5. The unique name, geometric properties and body identifier are sent to the server along with the operation data

6. The server forwards the operation data, unique name, geometric properties and body identifier to all remote clients

7. The remote clients identify the topological entities by the body on which they reside and the unique geometric properties and give them the unique names

8. The remote client performs the operation referencing the topological entity by its newly given unique name

In order for the lazy naming method to uniquely identify faces and edges, the body on which these topological entities reside must be uniquely identified (as seen in step 4). The eager naming method identifies bodies by the edges and faces which comprise the body. This is not possible with the lazy naming method because not all the topological entities are identified. Instead, a body is uniquely identified by the feature which originally created it and its bounding box. Bodies can be uniquely identified by the feature which created it, except for in the case where a feature creates multiple bodies. When a feature creates multiple bodies, the bounding box is employed to uniquely identify each body created by the feature. Therefore the combination of the creation feature and bounding box uniquely identifies all bodies in a CAD model, assuming features dont create multiple bodies in the same space.

4.6 Data Cache and Normalization to Enhance Feature Edit Performance

Central to the feature edit enhancement is the entity dictionary. There are two separate implementations of the entity dictionary, one on each client and one in the central database. The client entity dictionary is a cache which stores a relation between the persistent names of topological entities, their geometric properties, and computer memory pointers which reference those entities in the model. The database entity dictionary is the authoritative, persistent copy of the

geometric properties and names of the topological entities. The topological entities are stored separately from the features that use them, allowing updates to entities to take place without having to update all the features which reference them.

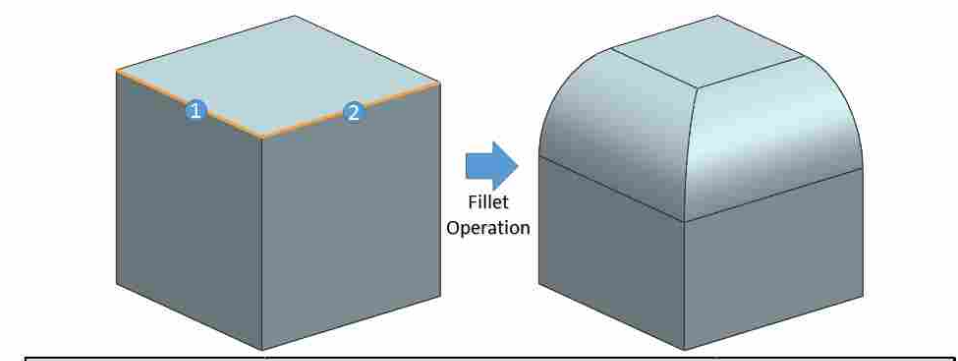
4.6.1 Client Entity Dictionary Cache

In a replicated, multi-user CAD environment, after an operation is performed, the data required to recreate that operation on other client machines is put into a data object. The data object is then forwarded to the server and distributed to other clients. As part of the lazy naming method, all topological entities referenced by the operation undergo the naming process. This process caches the following data in the client entity dictionary for easy access: (1) the unique name of the entity, (2) the geometric properties to uniquely identify the entity on a remote client and (3) the computer memory pointer to the entity on the local client. The first and second elements are included with the operation data sent to other clients when an operation is performed.

Fig. 4.6, illustrates the naming process that occurs after the user performs a fillet operation. The user selects edges 1 and 2 to perform the operation. After the operation is complete, entries are created in the client entity dictionary storing the entity name, Geometric ID (G-ID) and computer memory pointer. The G-ID holds unique geometric properties as well as the body of which the topological entity is attached.

When a remote operation is received from the server, the local entity dictionary is searched to make sure the referenced entities have been previously identified. If a topological entity name is not found in the remote clients dictionary, the geometric properties are used to find the topological entity on a given body and add it to the entity dictionary. This is done using a tolerance to compare the geometric properties of the G-ID of the entity to all of the geometric properties of the topological entities of the body it is attached to. When the topological entity is identified, it is assigned the same unique name forwarded from the remote client. The unique name and local memory pointer is cached in the client entity dictionary for quick look-up when another remote operation is received.

A critical aspect to this method is that feature operation data and topological entity data are isolated. The eager naming method stores topological entity data together with feature operation data. If this data is not normalized, whenever a user performs an edit operation, both the geometric



Client Entity Dictionary

	Entity Name	Geometric ID	Entity Memory Pointer
1	dan_feature_3_edge_1	Edge[{(10,15,20)(5,15,20)(0,15,20)}Body_1]	0x5A884 [Edge 27]
2	dan_feature_3_edge_2	Edge[{(10,0,20)(10,7.5,20)(10,15,20)}Body_1]	0x03D20 [Edge 32]

Figure 4.6: The client entity dictionary contains the entity name, Geometric ID and entity memory pointer for a specific client. The memory pointer is stored for quick access to the entity on a client.

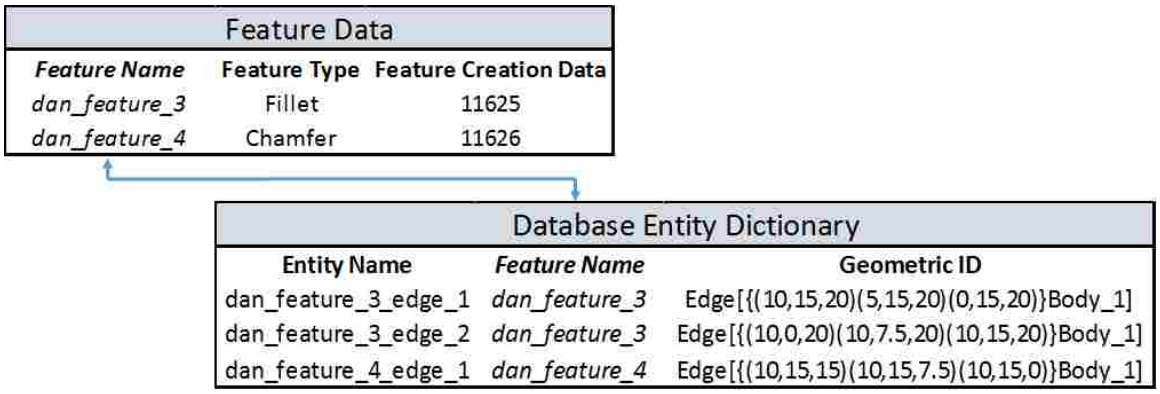
properties of the topological entities and operation data required to recreate the dependent features of the edit operation must be extracted and resent to the server. By storing the entities separate from the features which use them, dependent feature data does not need to be extracted and resent for every edit operation. Only updated geometric data for all changed topological entities in the model are sent to the server with the operation data. The updated geometric data is easily found by querying the client entity dictionary cache for changed entities.

4.6.2 Normalized Database Entity Dictionary

As features are created, edited, or deleted, data used to replicate these operations on other clients are forwarded through a server and stored persistently in a database. This data is the authoritative source of the model. It fully describes how to recreate the feature on another client and includes the geometric properties of topological entities that are referenced by the feature operation. As features are accepted by the server from remote clients, the data is added to the database to allow users who load the model in the future to download the most up-to-date features and topological entity data.

The database entity dictionary stores the topological entity data separate from the feature operation data. The entity dictionary consists of the persistent name of the referenced topological entity, the unique geometric information for identification, and the name of the feature which

references the entity. The database also stores the feature data which consists of the data necessary to create the feature on the client. The feature data references the entity data in the database so that when a feature is applied to a client the appropriate entity data is referenced (see Fig. 4.7.



Feature Data

Feature Name	Feature Type	Feature Creation Data
dan_feature_3	Fillet	11625
dan_feature_4	Chamfer	11626

Database Entity Dictionary

Entity Name	*Feature Name*	Geometric ID
dan_feature_3_edge_1	*dan_feature_3*	Edge[{(10,15,20)(5,15,20)(0,15,20)}Body_1]
dan_feature_3_edge_2	*dan_feature_3*	Edge[{(10,0,20)(10,7.5,20)(10,15,20)}Body_1]
dan_feature_4_edge_1	*dan_feature_4*	Edge[{(10,15,15)(10,15,7.5)(10,15,0)}Body_1]

Figure 4.7: The feature data references the entity dictionary in the database. The database entity dictionary is the global reference for all clients.

By normalizing the data, any changes to the topological entity can be updated in the database entity dictionary without having to update the feature data. Edit and delete operations change the geometric properties of some topological entities. If a dependent feature references one of those changed entities, this data needs to be updated in the database. It is important to have the feature and entity data sets separated to avoid unnecessarily updating the feature data set which requires an update to the dependent features on all clients. This saves a time because all dependent features do not need to be updated when their parent feature changes.

4.6.3 Coordinating the Database and Client Entity Dictionaries

When a part is loaded, operations are performed based on the feature and entity data which are stored in the database. As each operation is performed, a referenced entity is uniquely identified using the unique geometric properties of the entity. Once the entity is found, the local memory pointer, along with the entity name and G-ID are stored in the client entity dictionary. After all the referenced entities for the feature are stored in the client entity dictionary, the feature is created based on the data referenced from the client entity dictionary. Fig. 4.8 shows a representation of

the database entity dictionary (upper table) which directly corresponds to topological entities on the client model. The lower table in Fig. 4.8 depicts the client entity dictionary.

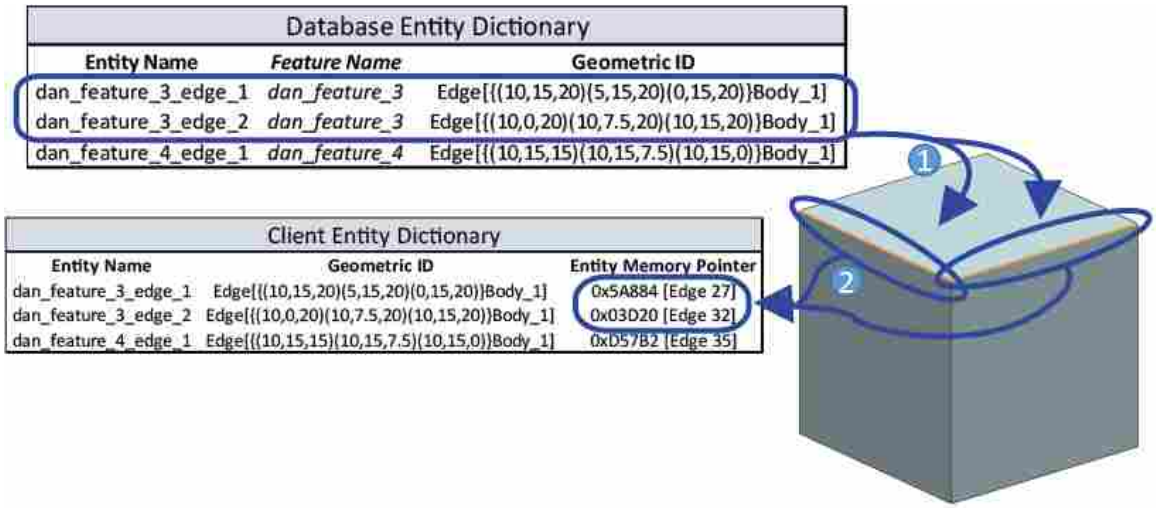


Figure 4.8: Image showing the relationship between feature creation and the client entity dictionary

When create or edit operations are performed on the client it is critical that all the new or updated data be sent to the server. The most recent geometric properties of the changed topological entities are compared against the geometric properties stored in the client entity dictionary to determine which entities have updated. When the most recent topological entity data is different or missing from that found in the client entity dictionary, the geometric properties of these entities are sent to the server with the edit or delete operation data. The operation data together with the updated topological entity data is termed an update message. Although the operation data and updated topological entity data is stored separately, it is critical that the update message contain both pieces of data because the entire message must be either accepted or rejected by the server as a single unit to ensure that it occurs as one operation. The server verification process for model consistency is outlined in Chapter 3.

4.7 Enhancement Implementations in NXConnect

The lazy naming, data cache and normalization enhancements were implemented in NX-Connect to improve feature creation and edit performance. The enhanced geometric property extraction was also implemented to additionally improve feature edit performance.

When an operation is performed on a client in which a topological entity is referenced, uniquely identifying attributes are extracted and placed into the client entity dictionary. The NX API uniquely identifies each topological entity with an identifier called a Journal ID (J-ID). A J-ID is a string value which contains an ephemeral name and geometric properties for an entity. The enhanced geometric property extraction utilizes the J-ID to rapidly extract geometric properties from an entity in a faster way than querying the entity for its geometric properties through the API as done in the eager naming approach. Alone, the J-ID is not necessarily unique. However, it becomes unique if the geometric properties of the J-ID are combined with a uniquely identified body, which is given a Body ID (B-ID). The B-ID is concatenated with the geometric data and put into the client entity dictionary as the G-ID. This G-ID is forwarded with the operation data to the server for verification and subsequent dissemination to other clients.

When an operation is verified by the server, the database entity dictionary is updated by adding or updating a row in the topological entity and feature tables in the SQL database on the server. At the same time, the operation and topological entity data is sent to remote clients. When an operation is received by a remote client, all unverified operations are suppressed to put the model in the same state as when the operation was performed on the originating client. Next, the client entity dictionary is searched by name to see if the topological entity had already been identified. If it is not found in the entity dictionary, the entity is identified in the model. The B-ID forwarded from the originating client is used to identify the body with the same corresponding B-ID in the remote client. Once the body is identified, the J-ID of each edge or face of that body is extracted and compared with the G-ID forwarded from the originating client. This identifies the corresponding face or edge which the operation should reference. The operation is then performed on the client.

When loading an existing part from the database, the client entity dictionary is initialized in the following way: A part file is stored in the database which is uploaded at checkpoints determined by the users, as described by [9]. When a checkpoint is set, a handle (persistent pointer) to each

topological entity is stored with the entity name as a string in the attributes of the part. When the part is loaded, the string is parsed to extract the handles of each topological entity. This handle is used to get the memory pointer and unique name of each topological entity. This allows the client entity dictionary to be initialized without having to re-identify each topological entity, which saves a significant amount of time on load.

4.8 Results

To validate that the three enhancements significantly improve times for feature creation and edits, several timed tests were performed to compare the implementation with these enhancements to the eager naming implementation presented in [9]. Several tests were run on three different implementations of persistent naming methods in NXConnect. The first implementation tested is the eager naming implementation presented in [9]. The second implementation tested incorporates the lazy naming, data cache and normalization enhancements that can be applied to multiple CAD systems. The third implementation tested additionally includes the enhanced geometric property extraction that is only applicable to NX. Three runs for each implementation for several models are performed. The tests were performed on an Intel(R) Xeon(R) CPU E5-1603 0 at 2.80 GHz with 16 GB ram with 64-bit Windows 7.

The first two tests are designed to test the feature creation enhancement of lazy naming. The first test was an extrusion of a rack for a rack and pinion assembly which creates 204 faces and 606 edges (shown in Fig. 4.9). The average computation time of three runs of the rack extrusion using the eager naming implementation was 1 min. 30 sec. on the originating client. Conversely, the average time it took for three tests using lazy naming with was 3 sec., representing a 30X speed-up. The enhanced geometric property extraction had no additional benefit for this creation operation because no geometric property extraction takes place when a create operation is performed (see Tab. 6.1).

The second test was the extrusion of a grate presented in the introduction and shown in Fig. 4.3. This operation creates 406 faces and 1212 edges. The average computation time of three runs of the extrusion operation and naming logic using the implementation in the eager naming method was 4 min. and 39 sec. However, the average time it took using lazy naming was 6 sec. The remaining 6 sec. is the time for NX to perform the operation, extract the operation data and

Table 4.1: Implementation time comparison for the extrusion of rack test

Implementation	Ave. time of 3 tests	Speed-up
Eager naming	1 min. 30 sec.	
Lazy naming, data cache and normalization enhancements	3 sec.	30X
Plus enhanced geometric property extraction	3 sec.	30X

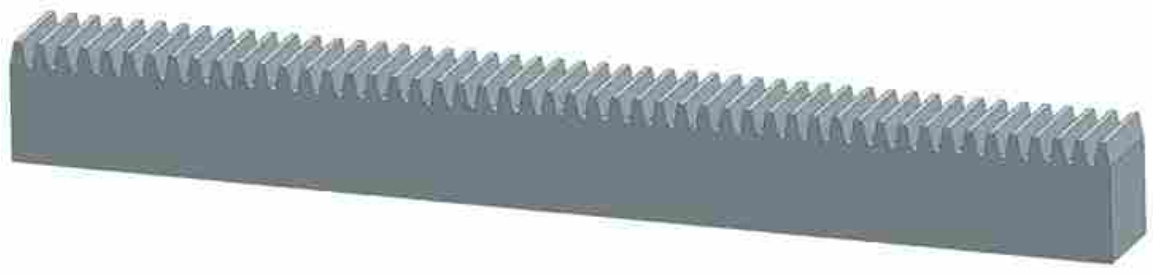


Figure 4.9: Extrusion of rack creating 204 faces and 606 edges

perform error rejection and consistency logic. The pinion and grate tests show a 47X speed-up using the lazy naming method (see Tab. 4.2). Again, the enhanced geometric property extraction had no additional benefit because this is a creation operation.

Table 4.2: Implementation time comparison for the extrusion of the grate

Implementation	Ave. time of 3 tests	Speed-up
Eager naming	4 min. 39 sec.	
Lazy naming, data cache and normalization enhancements	6 sec.	47X
Plus enhanced geometric property extraction	6 sec.	47X

4.8.1 Feature Edit Enhancement Tests

The next tests show the significant time savings with the feature creation and edit enhancements. Since there is a significant difference between the times it takes to perform the operation on the originating client compared to the time it takes to perform the operation on a remote client for the lazy naming implementations, both times are measured in these tests. Since the eager method

performs the same logic on the originating client as on a remote client, only the originating time is reported.

The third test is a gear with a fillet operation creating fillets on the inside and outside edges of the teeth (see Fig. 4.10). Although the extrusion only has one dependent feature (the fillet operation), editing the extrusion requires the identification of 83 faces and 242 edges for the extrusion with an additional 80 faces and 240 edges for the fillet operation. This is a total of 163 faces and 482 faces to identify when the feature and dependent feature are identified using the eager implementation. The average computation time of three tests of the extrusion edit operation was 41 sec. using the eager naming implementation. Conversely, it took 15 sec. to perform the edit operation with the lazy naming, data cache and normalization enhancements on an originating client and 7 sec. on a remote client. It only took 4 seconds to perform with the additional enhanced geometric property extraction on an originating client and only 1 sec. on a remote client (see Tab. 4.3).

Table 4.3: Implementation time comparison for the gear extrusion length edit

Implementation	Ave. time of 3 tests	Speed-up
Eager naming	41 sec.	
Lazy naming, data cache and norm. enhance. (originating)	15 sec.	3X
Lazy naming, data cache and norm. enhance. (remote)	7 sec.	6X
Plus enhanced geometric property extraction (originating)	4 sec.	10X
Plus enhanced geometric property extraction (remote)	1 sec.	40X

The fourth test is the edit of a feature with several child components. This test is the editing of the extrusion height of a cylinder approximated by chamfering a cube several times. Fig. 4.11 shows how this approximated cylinder is created. The leftmost image in Fig. 4.11 shows the extrusion of a square. The middle image shows chamfers around the edges creating an octagon extrusion. Each of those edges are again chamfered. This process is repeated six times, creating an extrusion of a body with 256 sides which approximates a cylinder, shown in the far right image. The resulting cylinder approximation body has a total of 258 faces and 768 edges.

The test performed on this body is to edit the value of the original square extrusion to make it half of the original length (see Fig. 4.12). This operation requires a re-evaluation of

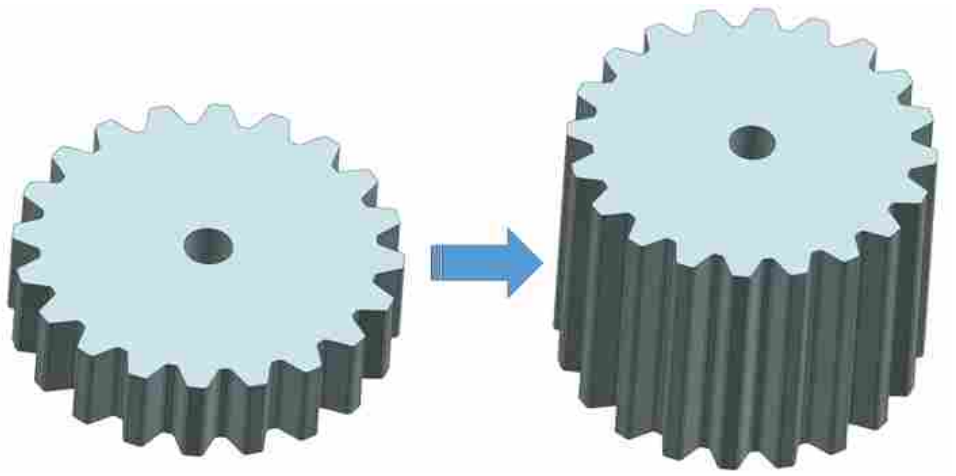


Figure 4.10: Edit of extrusion length of gear with fillets

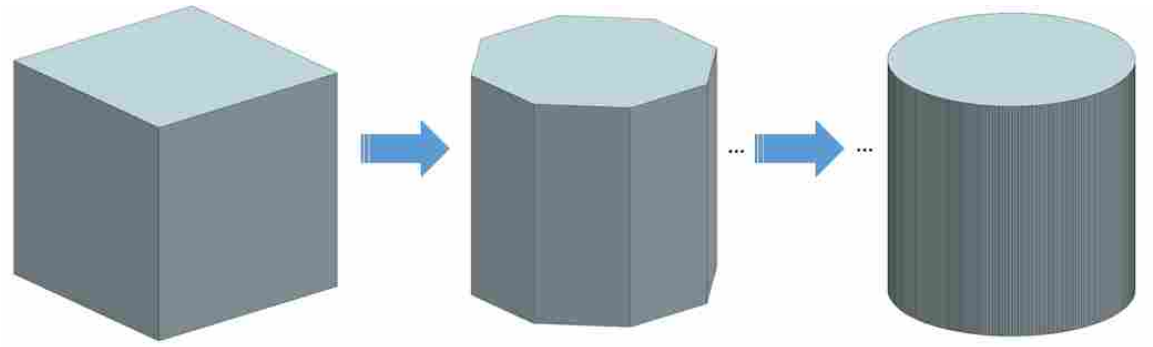


Figure 4.11: Cube with many chamfers added until it approximates a cylinder with 256 sides

all the dependent chamfers so they update as well. The average of three tests using the eager naming implementation is 7 min. and 35 sec. With the eager implementation, the edit operation requires the identification of all the faces and edges after each chamfer operation, totally 522 faces and 1524 edges. The lazy naming, data cache and normalization enhancements only takes an average of 1 min. 43 sec. on an originating client and 30 sec. on a remote client. Conversely, the implementation with the enhanced geometric property extraction took only 8 sec. on an originating client and 6 sec. on a remote client (see Tab. 4.4.

Significant time savings were seen with the enhancements in all four test cases. For the latter two tests, the time to perform the operation on the remote client was significantly less than the time it took on the originating client. The reason it takes more time on the originating client is because the originating client must determine which entities have moved in order to send the

Table 4.4: Implementation time comparison for the extrusion edit of the cylinder

Implementation	Ave. time of 3 tests	Speed-up
Eager naming	7 min. 35 sec.	
Lazy naming, data cache and norm. enhance. (originating)	1 min. 43 sec.	4X
Lazy naming, data cache and norm. enhance. (remote)	30 sec.	15X
Plus enhanced geometric property extraction (originating)	8 sec.	57X
Plus enhanced geometric property extraction (remote)	6 sec.	76X

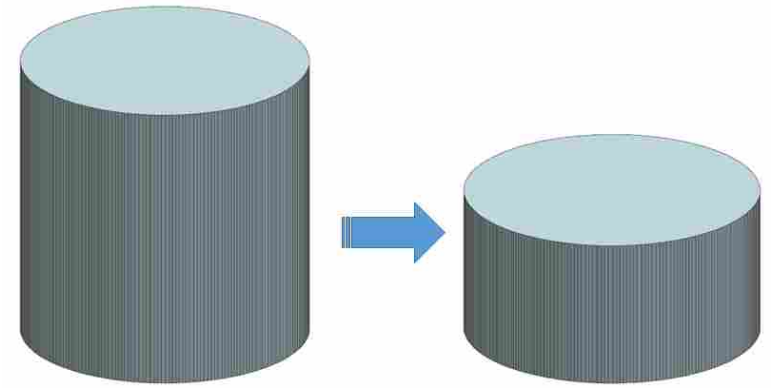


Figure 4.12: Edit extrusion length to half the original length

geometric properties of the updated topological entities to the server. Since the remote clients receive the updated geometric properties of the topological entities, they update the client entity dictionary directly, taking much less time.

The enhanced geometric property extraction implementation significantly improved times for edit operations. The speed-up is even more significant for feature edits with several dependencies. The geometric property extraction enhancement improves times for edit operations because it speeds up the geometric property extraction time in NX. Geometric property extraction is fundamental in determining which entities have moved after an edit operation. Conversely, no geometric property extraction takes place when a create operation is performed since the lazy naming approach delays naming of topological entities until they are referenced by a dependent feature.

In addition to these tests, Appendix A shows more complex models which created in the multi-user CAD system with these methods implemented. These examples qualitatively show the robustness and usability of these methods for industrial scale modeling in multi-user CAD.

4.9 Conclusion

The persistent topology naming methodology theoretically solves the problem of multiple users referencing the same topology entity on separate clients. Due to limitations in the NX API, the implementation has required additional redundancy to be implemented effectively, but has shown to be on the order of 99% robust. The examples shown in Appendix A qualitatively show the robustness of these methods for industrial scale modeling in multi-user CAD may be even more.

Three enhancements dramatically reduce the time required for persistent naming in a multi-user CAD system compared with the eager naming method. The first enhancement improves feature creation performance using lazy naming which defers naming topological entities until they are actually referenced by dependent features. The second method enhancement improves feature edit performance by caching data on the client using an entity dictionary. It also normalizes the topological entity data and operation data to alleviate the need to update dependent features when an edit operation takes place. The third enhancement utilizes the NX Journal ID to rapidly extract geometric properties of topological entities to additionally enhance edit performance. Testing these enhancements in NXConnect shows that these speed-up the naming process in multi-user CAD by at least an order of magnitude over the eager naming method. Significant time savings benefits are also seen on feature edits with dependent features, especially for features with several dependencies. Time savings are even more significant for feature edits with the enhanced geometric property extraction implementation in NXConnect. The examples shown in Appendix A qualitatively show the usability for industrial scale modeling in multi-user CAD.

CHAPTER 5. SEMANTIC CONFLICT AVOIDANCE THROUGH AUTOMATED FEATURE RESERVATION

5.1 Introduction

In addition to syntactic conflicts which lead to inconsistency and software bugs, semantic conflicts occur in multi-user CAD when multiple users perform operations which violate the design intent of the model. Although these conflicts do not cause inconsistency, they are a problem because they lead to redundant work, which adds waste to the overall design process. The automated feature reservation method presented in Chapter 2 prevents syntactic feature/self conflicts. This chapter will show that it also avoids semantic feature/self conflicts which lead to redundant work by multiple users. This is based on the principle that automatically reserving features prevents waste derived from two users performing redundant work on the same feature. This chapter will discuss a mathematical model and the results of experiments which suggest that the automated feature reservation method reduces semantic conflicts in multi-user CAD. Experiments were run with teams of varying sizes comparing the time it takes to edit a multi-user CAD model with and without the automated feature reservation method in place. Results show that there is a reduction in the number of conflicts with automated feature reservation as the number of contributing users in a CAD model increases.

5.2 Background

The conflict management methods found in the literature relating to this chapter were outlined in Chapter 1 [7, 8, 15, 21, 31, 32, 32–39].

5.3 Theory

To show that the automated feature reservation method reduces the number of conflicts in a multi-user CAD model, it is pertinent to develop a mathematical model to predict the number of conflicts and the time wasted due to these conflicts in a multi-user modeling task without the method in place. First, the theoretical effects of a simultaneous multi-user team must be established. When measuring the time it takes for a team to complete a modeling task, there are two aspects to consider: computation time and user time. Computation time is the time for the computer to perform the operations for a modeling task. This time is not scalable with the number of users added to a modeling task. User time is the time it takes for a user or team of users to perform the modeling task. User time is scalable as more users are added to a simultaneous modeling session. The user time t_m, it takes to complete a given task by a team of users, excluding computation time is

$$t_m = t_1/m \tag{5.1}$$

given the time to complete the task for one user t_1, and the number of users m, simultaneously completing the task. Since computation time t_c, does not scale with more users, the total time to complete the task for the multi-user team t_t, can be expressed as follows:

$$t_t = t_m + t_c \tag{5.2}$$

The automated feature reservation method directly affects t_m based on the number of features to be edited in a modeling task and the number of simultaneous users editing at a time. The probability of a conflict during a multi-user modeling task, $P(conflict)$ depends on the number of features to be edited in a modeling task n, the number of simultaneous users m editing, and the state at which the edit takes place i, is

$$P(conflict) = \begin{cases} i/n & \text{for } 0 \leq i \leq (m-1) \\ (m-1)/(n-i+m-1) & \text{for } m \leq i \leq (n-1) \\ 1 & \text{for } n \leq i \leq (n+m-2) \end{cases} \tag{5.3}$$

given the probability of a conflict in a multi-user editing task shown in 5.3, and given t_i as the time it takes a single user to complete at feature operation at i, where:

$$t_m = \sum_{i=0}^{n-1} t_i \tag{5.4}$$

The time wasted due to conflicts t_w, can be approximated as follows:

$$t_w = \sum_{i=0}^{m-1} (i/n)t_i + \sum_{i=m}^{n-1} ((m-1)/(n-i+m-1))t_i + \sum_{i=n}^{n+m-2} t_i \tag{5.5}$$

Thus, the total time it takes to complete a modeling task consisting of n edit operations, with a team of m users without the automated feature reservation method is:

$$t_t = t_m + t_c + t_w \tag{5.6}$$

The derivation for 5.3 and 5.5 is easily seen through a simplified example. Given three users in a CAD model and nine features to edit, the probability that there will be a conflict for the first three choices of a feature to edit is 0 for the first user, 1/9 for the second user and 2/9 for the third user (see Fig. 5.1).

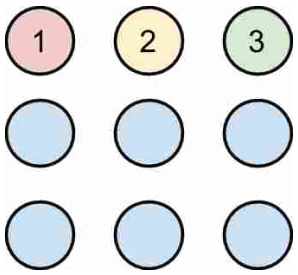


Figure 5.1: The probability for conflict in the first three edits

After all the users choose a feature, the probability of a conflict changes. This scenario operates on two assumptions: first that all feature edits take the same time ti, and that the users do not enter and exit features at identical times. This means that when user 1 finishes the edit operation, user 2 and user 3 will still be editing their features when he chooses the next feature to edit. Thus the probability of a conflict must be recalculated at each new choice of a feature. In

this example, when user 1 chooses the next feature, the probability of a conflict is 2/8 (see Fig. 5.2). When user 2 finishes the edit, the probability of conflict is 2/7 (see Fig. 5.3). This pattern continues for every successive choice of a feature to edit until there are only two left to edit. The probability of conflict for these six choices is $\{2/8, 2/7, 2/6, 2/5, 2/4, 2/3\}$.

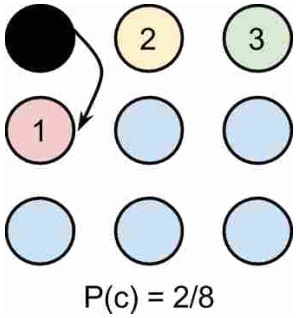


Figure 5.2: The probability for conflict for the fourth edit

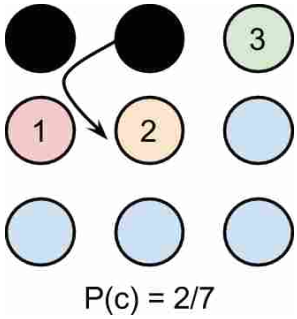


Figure 5.3: The probability for conflict for the fifth edit

Lastly, when all but two of the features have been edited, the probability that there will be a conflict for those last two steps is 1. This means for the last two steps there will certainly be a conflict at both steps (see Fig. 5.4). In summary, for the first set $P(c) = \{0, 1/9, 2/9\}$, for the second set $P(c) = \{2/8, 2/7, 2/6, 2/5, 2/4, 2/3\}$, and for the last set $P(c) = \{1, 1\}$. To get the total number of probable conflicts, sum the probability of a conflict at each step. In this example $P(c)_{total} = 5$. Thus the total amount of time wasted is equal to the number of conflicts multiplied by time it takes to make a single edit at each step as seen in 5.5.

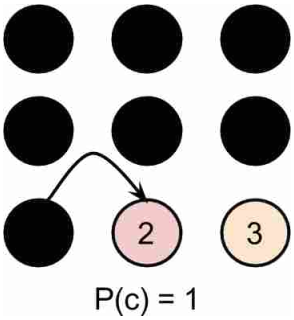


Figure 5.4: The probability for conflict for the second to last edit

It is proposed that with the automated feature reservation in place, t_c in 5.5 is zero. This is because semantic conflicts due to simultaneous edits are avoided in a multi-user editing task. This method can save a significant amount of user time, especially when the ratio of user count to feature edit count and the ratio of user time to computation time is large. In summary, the automated feature reservation method allows the user time to be scaled more closely to the ideal model shown in 5.2 because t_w in 5.6 is zero.

5.4 Experiment and Results

A set of experiments was conducted to verify that the automated feature reservation method reduces the number of conflicts in a multi-user CAD model. Teams of users were given the task of editing several features in CAD parts both with and without the automated feature reservation method enabled. The experiments were set up with groups of 9-14 participants familiar with CAD modeling. Before every run of each experiment, participants were separated into teams of two, three, four, and in one experiment, six persons. The number of participants per team and the specific individuals on each team in each repetition of the experiment were determined by random number generation.

The teams were separated onto computers that were running the two different versions of the multi-user CAD system, one with automated feature reservation method enabled and one with it disabled. Since semantic conflicts result in redundant work and wasted time, it is hypothesized that teams with automated feature reservation will complete the CAD model edit task in a shorter total time than teams without automated feature reservation. To quantify the advantage of using automated feature reservation, the total time to completion is used as a metric. No communication

or planning by the team members was allowed before the beginning of the tests. During the tests, the participants were only allowed to use the collaborative chat feature built into the multi-user CAD system for communication.

5.4.1 Experiment I

For the first experiment, a CAD model was created which consists of 49 circle extrusions of varying radius randomly distributed on the same plane. Random distribution of features ensured that there was no obvious way for users to organize the distribution of workload between users. Since the CAD model is fairly simple, there was not a separation that needed to take place based on skill level. Any bias that came due to participant skill level or computer speed was reduced by the random placement of participants on different teams. Participants were separated into teams and given the task to edit the extrusion length of each circle from 1 to 5 as shown in Fig. 5.5.

A total of eight trials were made on identical models with teams of two, three, four and six users. There were a total of six teams of two users, three teams with the automated feature reservation method and three without. There were eight teams of three users with four teams having the automated feature reservation method in place and four without it. There were a total of six teams of four users and six teams of six users, where in each case there were three teams that had the automated feature reservation method and three that did not.

During the experiment, many users experienced bugs with the multi-user software causing interruptions to the modeling activity. When individuals experienced bugs, the teams continued to work as best they could. Despite these bugs, the experiments were completed and the average results are shown in Fig. 5.6. These results were the first telling sign that using the automated feature reservation method provided a time savings for multi-user modeling. The time savings increases with the number of users contributing to the model, peaking at four users and decreasing somewhat with six users. Since there were so many bugs for this experiment, more experiments were conducted to more fully validate the hypothesis.

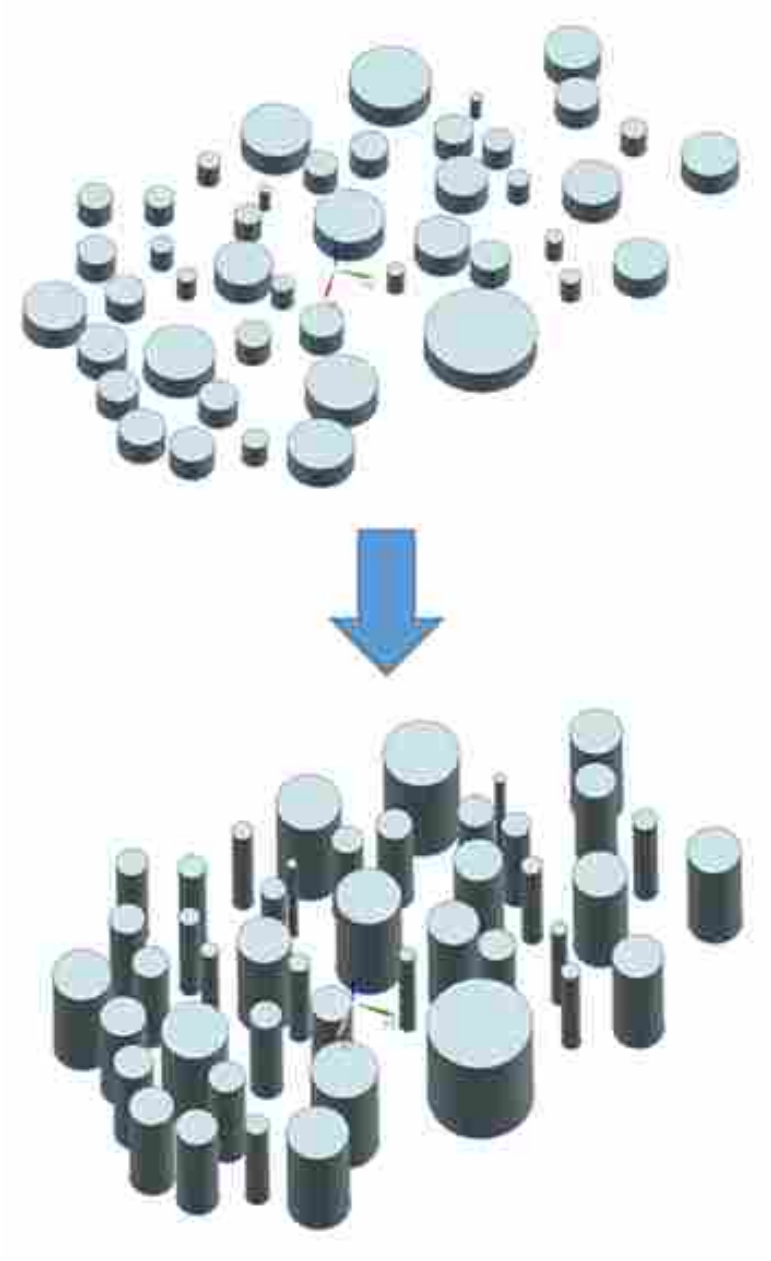


Figure 5.5: The height of each extrusion is edited by each multi-user team

5.4.2 Experiment II

For the second experiment, the majority of the bugs seen in the first experiment were resolved. This experiment used a similar CAD model that was used in the first experiment but consists of 44 circle extrusions instead of 49. A total of six trials were made on identical models with teams of two, three and four users. There were a total of six teams of two users, three teams

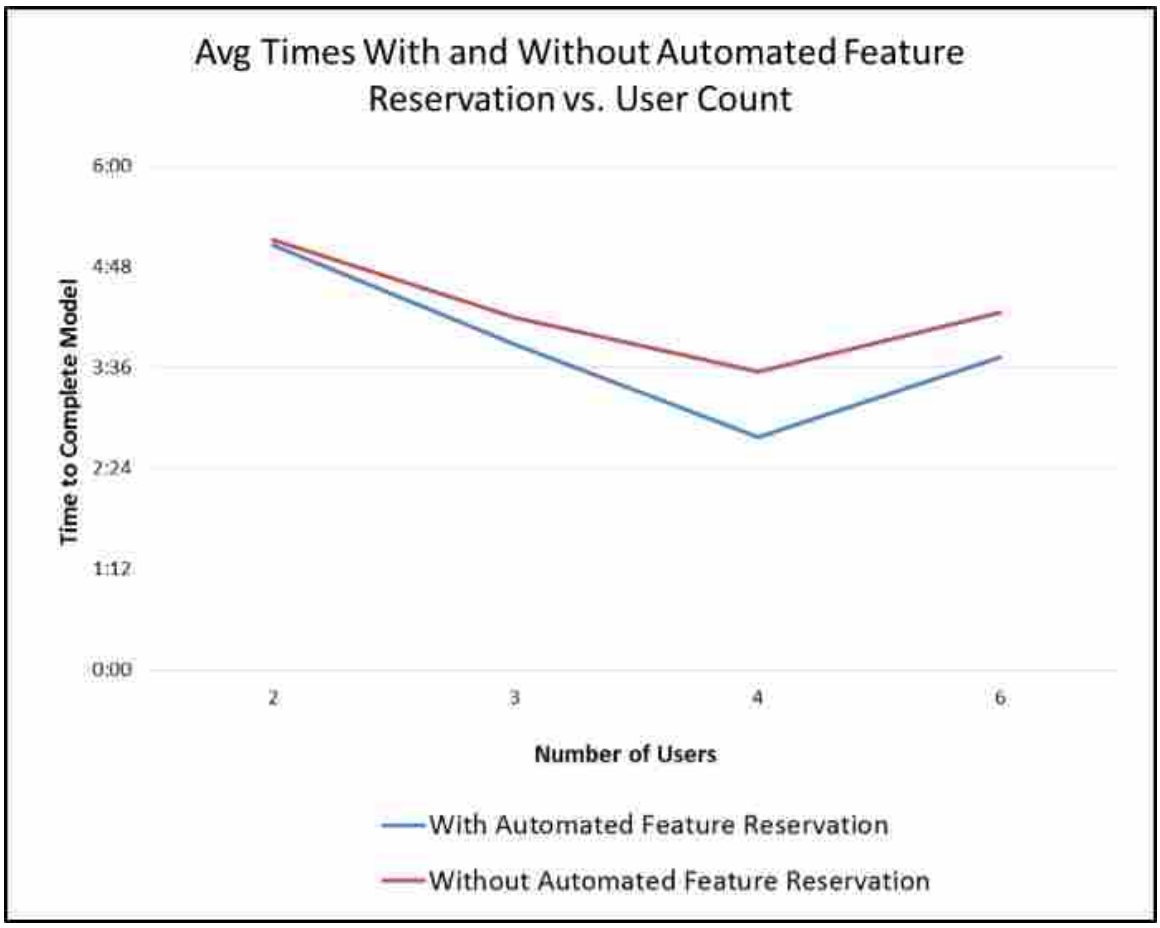


Figure 5.6: Average test results of experiment I

with the automated feature reservation method and three without. There were eight teams of three users with four teams having the automated feature reservation method in place and four without it. There were a total of nine teams of four users, where five teams had the automated feature reservation method and four did not.

Fig. 5.7 shows the raw times of the teams with and without the automated feature reservation method in place. Looking at this data, it is difficult to see the effect for the two-user team, but it becomes more apparent for the three-user and four-user teams. This graph shows an outlier for one team of two with the automated feature reservation. An average of these times with the one outlier removed, as shown in Fig. 5.8, more clearly reveals the effect of the automated feature reservation on the time for teams to complete the model. It can be seen that with a larger multi-user team size, the automated feature reservation increases overall time savings.

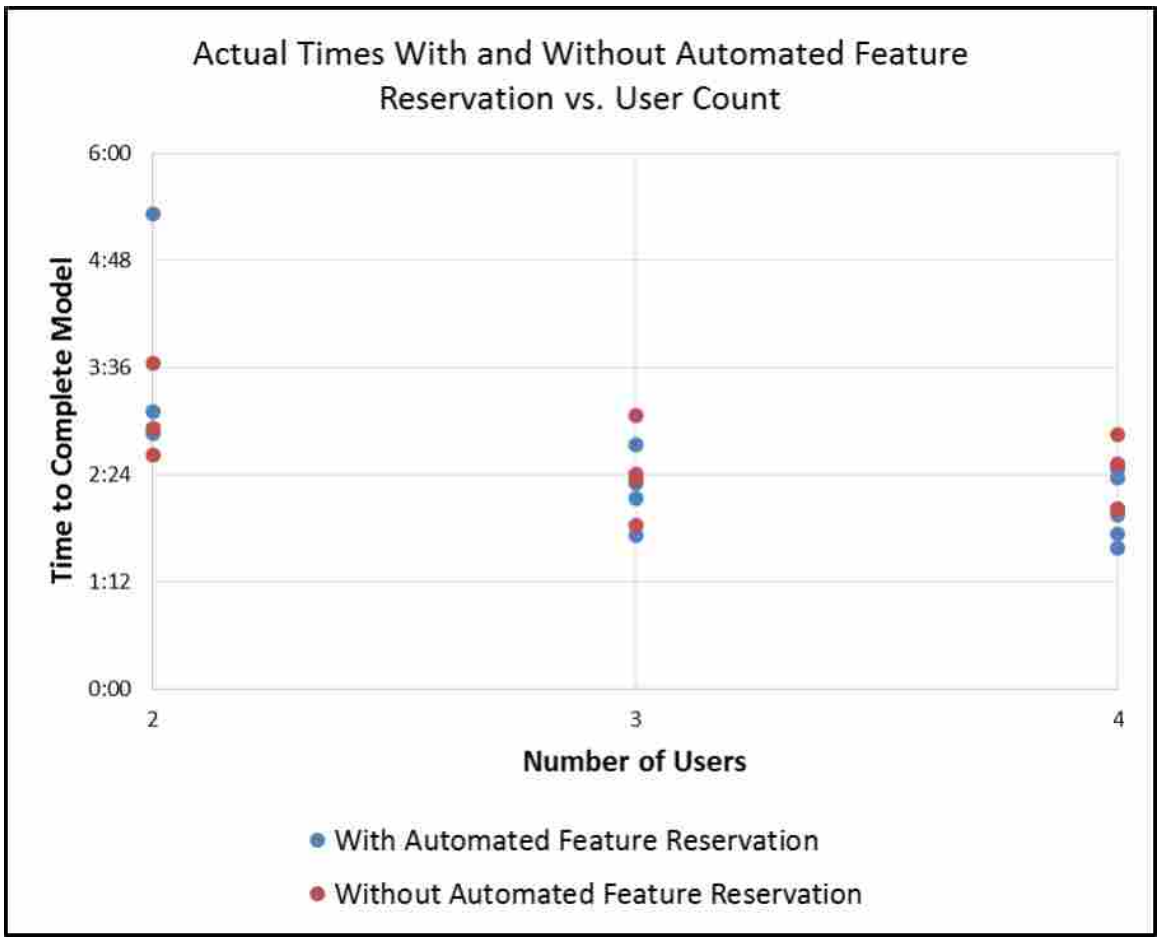


Figure 5.7: Raw times of experiment II with and without automated feature reservation

5.4.3 Experiment III

The third experiment incorporated a model with greater feature complexity in which each feature would require a greater user edit time. A model was developed which was composed of 44 sketches, each with five dimensions to be altered. The purpose of this experiment was to study the effect of automatic feature reservation on the time to edit a model with features with a longer user edit time.

This test showed the opposite of previous tests; that is, teams without automated feature reservation completed the model in less time than teams with automated feature reservation (see Fig. 5.9). Upon analyzing the experiment, it was discovered that hardware contributed greatly to performance for this test. Computers available for this test included some faster models (eight-core processors, 16 GB RAM, solid-state hard drives) and some slower models (single- or dual-core

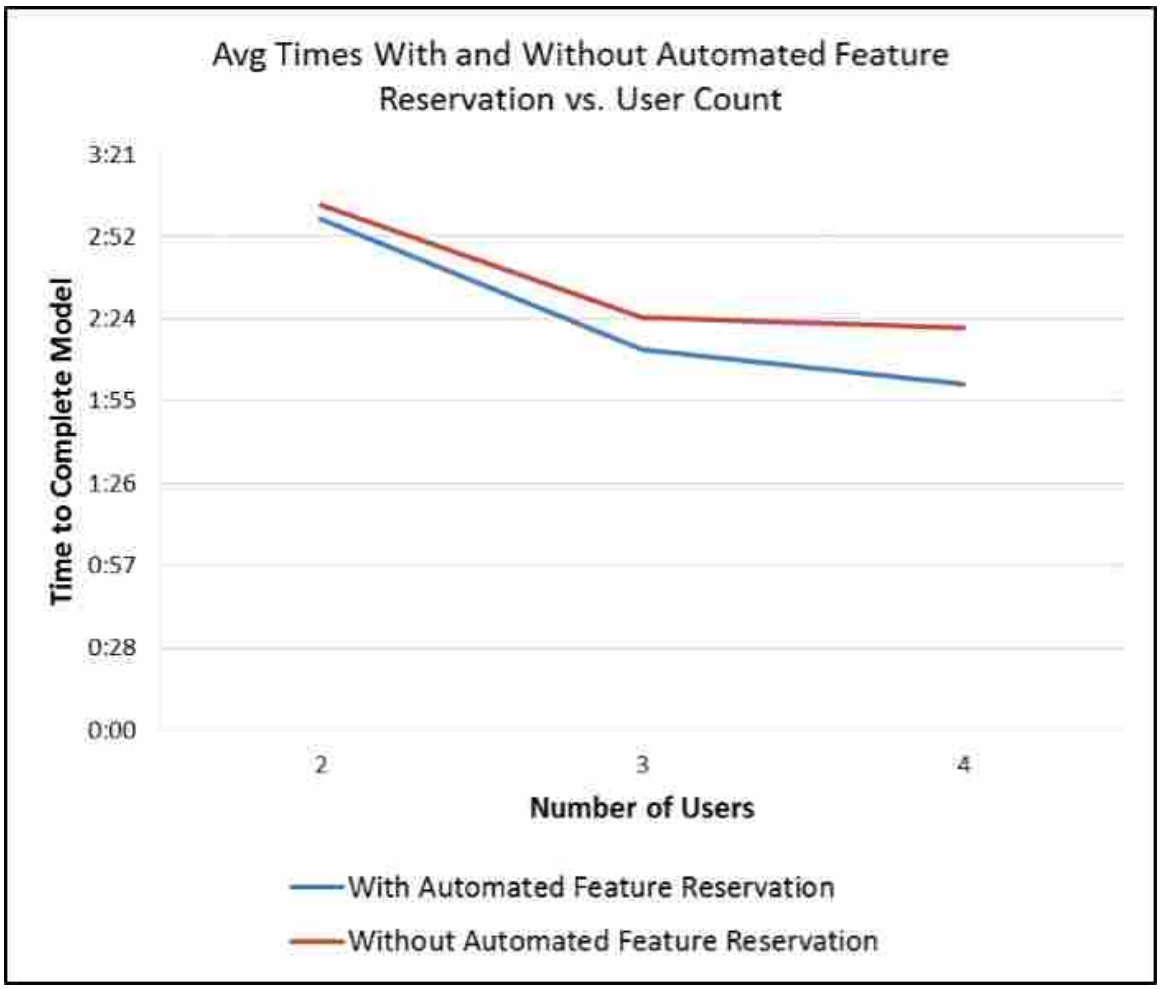


Figure 5.8: Average times of experiment II with one outlier removed on the two user team

processors, 2-8 GB RAM, disk drives). These computers were supposedly randomly distributed among test participants, who changed computers each trial. However, further analysis showed that teams without automated feature reservation had an average of 50% more of the faster computers. Furthermore, the sketch features used on this model took significantly more computation time to create relative to user editing time. The average update time for the slower hardware per feature was 4.78 seconds while the average update time on the faster hardware was 1.78 seconds per feature. The features took an average of 8.3 seconds of user time to edit. Since the teams without the automated feature reservation used more of the faster hardware, they had a distinct advantage over the other teams.

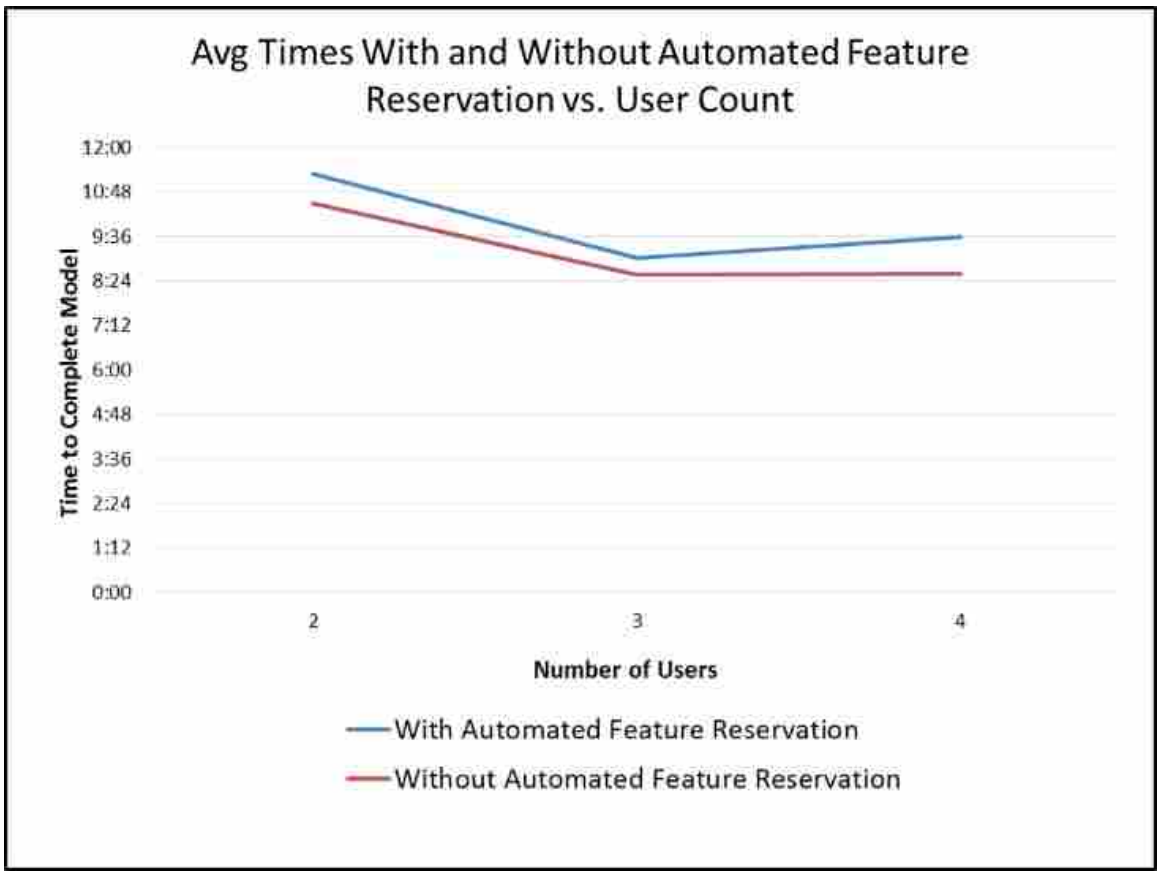


Figure 5.9: Results of experiment III: the greater speed of teams without automated feature reservation is attributed to hardware disparities between teams

These results suggest that for thick client multi-user CAD, hardware speed can have a significant effect on performance, especially with high ratios of computation time to user time. Accordingly, it was decided that a fourth test should be conducted which controlled this variable by giving each participant similar hardware.

5.4.4 Experiment IV

To avoid the issues seen in the third experiment, the fourth experiment attempted to decrease computation time and increase user time for the multi-user editing task. Computation time was decreased by ensuring that every user was using the same fast hardware. The edit of an extrusion was again used because it requires considerably less computation time than the current multi-user implementation of sketch. To increase the user time, the extrusion edit was made more

complicated. Instead of assigning the user to change the extrusion from a 1 extrusion to a 5 extrusion, each user was assigned to change the start limit of the extrusion to 4 and the end limit to 5. Additional features to change within the extrude feature were also included, such as setting the draft option to From Start Limit with an angle of 3 degrees, as well as changing the offset option to Two-sided with start offset of .2 and end of .1 (see Fig. 5.10).

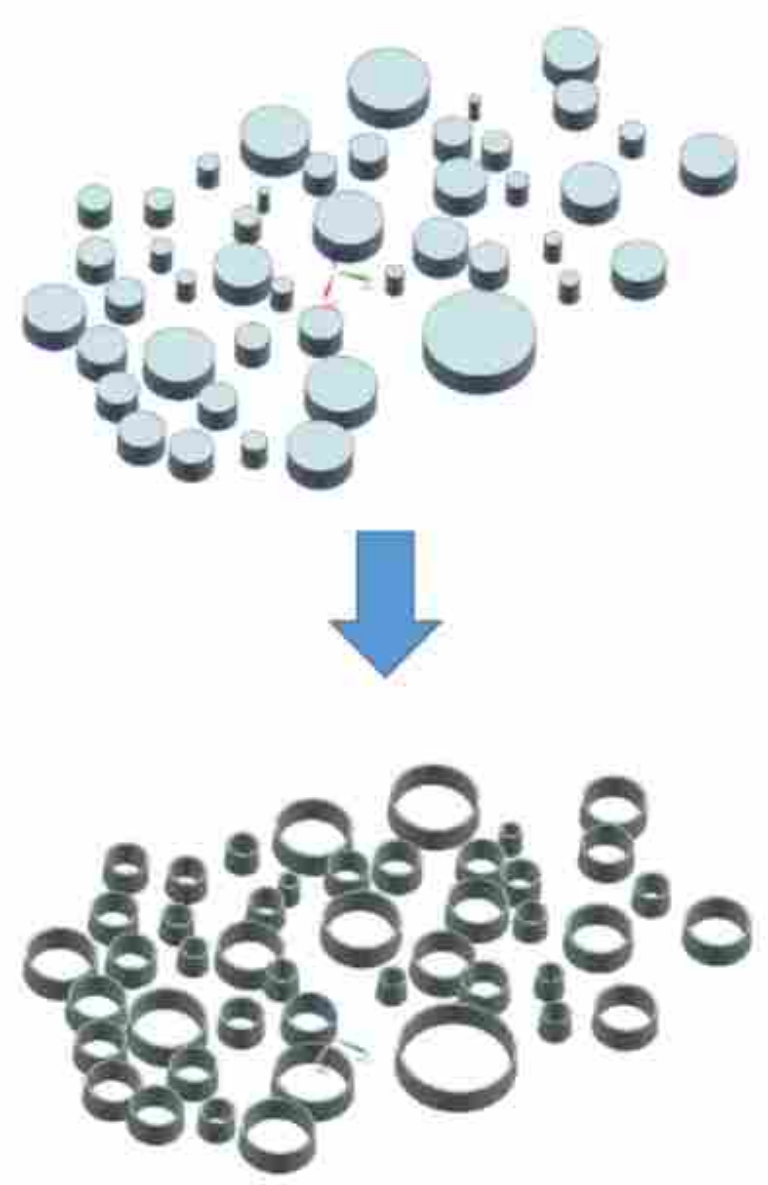


Figure 5.10: Several extrusion parameters are edited by each multi-user team

Six tests of two, three and four users with and without the automated feature reservation were intended to be run. There was a slight mistake that the experiment ended up with seven two user tests without the automated feature reservation and five with it, but the other tests were performed as planned. Once the experiment was run, it was quickly realized that user ability would play a large role in the times that were being received. So, in addition to the multi-user tests, test for each user doing the model alone was also included. This test was to calculate the user speed and use this data to normalize the results that were collected.

To calculate the normalized time the following formula was used, where T is the time recorded during the multi-user test, m is the number of users in the test, T_n is the time of the user m to complete the single user test and T_{avg} is the average single user time.

$$T_{normalized} = (T * \sum_{i=1}^{m} T_{avg}/T_n)/m \tag{5.7}$$

When these normalizations were added to the test results our data more accurately resembled our predicted model. The results using the normalization shown by 5.7 are depicted in Fig. 5.11.

5.4.5 Overall Results Summary

The average results of experiment II and IV are compared to a mathematical model modified from 5.5 which removes the guaranteed conflicts for the last few operations:

$$t_w = \sum_{i=0}^{m-1} (i/n)t_i + \sum_{i=m}^{n-1} ((m-1)/(n-i+m-1))t_i \tag{5.8}$$

The last term in 5.5 is removed for the comparison because the times for the experiments were taken as the first user to complete the test. Equation 5.8 excludes the time it takes for the last few guaranteed conflicting operations.

The predictions are calculated by starting with a two user edit time that is the consistent with the experimental results. Using benchmark tests, the average computation time it took clients to edit an existing feature was calculated. This was determined by recording the time it took for several computers with a variety of system resources to edit a feature both on send and receive.

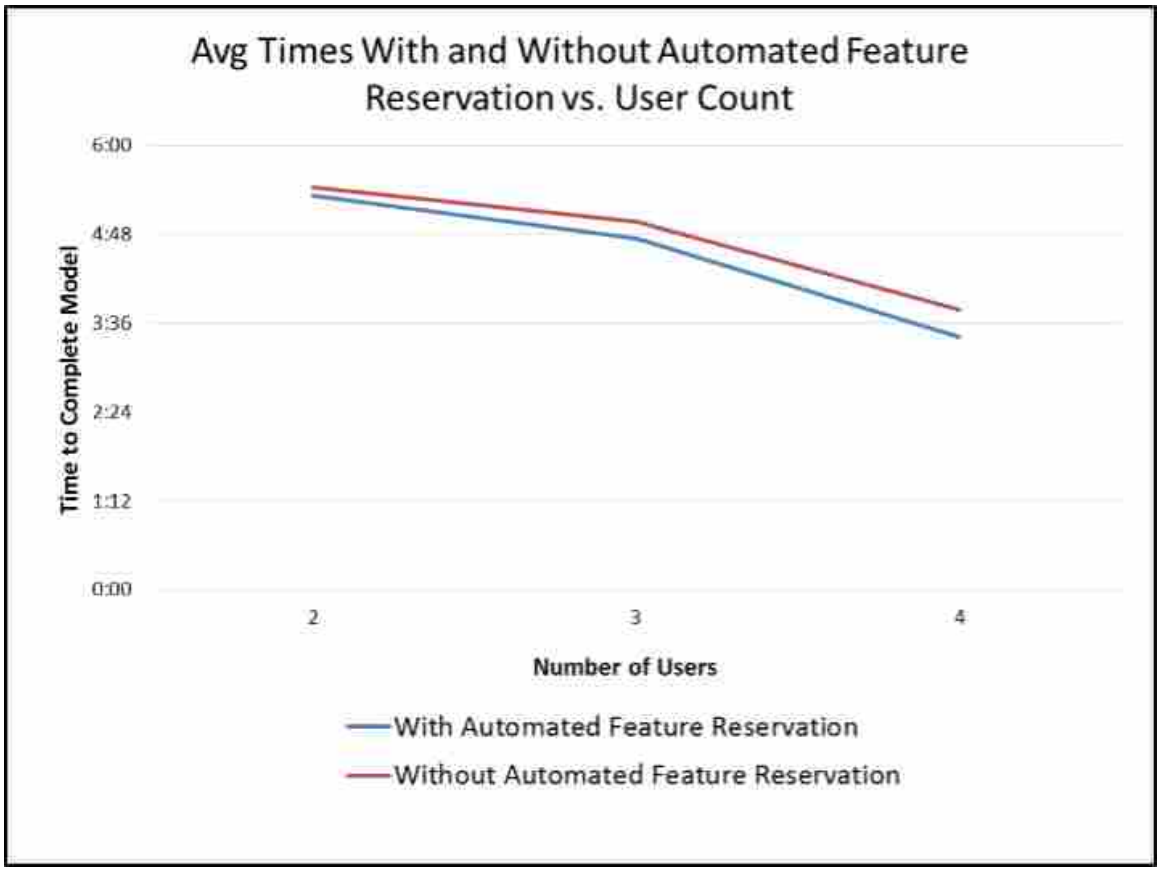


Figure 5.11: The average times for experiment IV, showing results with and without the automated feature reservation

The mean of these times for experiment II was found to be 1.7 seconds. Assuming an average computation time of 1.7 seconds and the average time with the automated feature reservation method is 2:59, the average user edit time is calculated to be 2.4 seconds. For experiment IV the estimated computation time is .53 seconds and thus the average user edit time is calculated to be 6.72 seconds. The predicted results for the experiments with reservation are calculated using 5.2. The predicted results for the experiments without reservation are calculated using 5.8.

The actual results compared to the predicted results using the assumptions of constant user and computation time developed in 5.2 and 5.8, along with the percent error from the experimental data is shown in Table 6.1. The experimental results are reasonably consistent with 5.2 and 5.8. The sources of error are likely due to the following:

- The variation of computer speeds

- The variation of different user skill level effecting the user edit time
- The variation due to multi-user software bugs for different individuals
- Users potentially using a different version of the software implementation as was intended
- The ability of users to compartmentalize tasks and attempt to avoid conflicts...

5.5 Conclusions

The automated feature reservation method helps to reduce the time for a team performing an editing task in a CAD model without dependencies because it reduces the number of semantic conflicts, and thus reduces the amount of waste in the process. A mathematical model has been developed which predicts the times for teams performing simultaneous edits to a model with and without the automated feature reservation method in place with an error of 22% or less. The time wasted due to conflicts depends on the number of simultaneous users, number of features to be edited, time to perform each edit operation, and proportion of user time to computation time. The results suggest that time savings increase for teams using the automated feature reservation as more users are involved in the multi-user editing task. The results also suggest that hardware speed can have a significant effect on multi-user team performance when an activity involves editing features with a high ratio of computation time to user time.

Table 5.1: Predicted vs. actual results for experiments II and IV

		Without Reservation			With Reservation			% Improvement	
	Users	Predicted	Actual	% Error	Predicted	Actual	% Error	Predicted	Actual
Experiment II	2	3:07	3:03	2.19%	2:59	2:59	Baseline	4.28%	2.19%
	3	2:38	2:24	9.72%	2:24	2:13	8.27%	8.86%	7.64%
	4	2:25	2:20	3.57%	2:07	2:01	4.96%	12.41%	13.57%
Experiment IV	2	5:42	5:24	5.56%	5:19	5:19	Baseline	6.73%	1.54%
	3	4:19	4:58	13.09%	3:41	4:44	22.18%	14.67%	4.70%
	4	3:43	3:47	1.76%	2:51	3:24	16.18%	23.32%	10.13%

CHAPTER 6. SEMANTIC CONFLICT AVOIDANCE USING AN INTEGRATED TASK MANAGEMENT SYSTEM

6.1 Introduction

This chapter presents a method to allow designers to quickly communicate and organize their efforts within the multi-user CAD system using an integrated task management system. A system of this nature provides an improved organizational framework to coordinate users efforts. It also allows for multi-user agility by enabling users to contribute when and where they are needed. Using the integrated task management system results in a reduced number of semantic conflicts and thus decreases the overall time to complete models in a simultaneous, multi-user CAD system.

This chapter will discuss the theory of reducing semantic conflicts by enhancing communication and organization while still preserving multi-user agility in a multi-user CAD environment. It presents the details of a task management system, integrated in a multi-user CAD system which provides a vehicle to improve multi-user communication and organization. After that, it discusses the experimental process and results of the experiment to show that the integrated task management system has a significant effect on reducing semantic conflicts in a multi-user CAD modeling session.

6.2 Background

The time it takes a team of users to collectively model a part depends on how well a model can be partitioned and the communication overhead involved. Many authors discuss how these factors affect completion time of software development projects [57–59]. A task that is perfectly partitionable can be subdivided infinitely and shared among an infinite number of co-workers [57]. This type of task has the following relationship:

$$t = e/m \tag{6.1}$$

Where t is the total time to accomplish the task, e is the total effort required, and m represents the number of modelers. When there are communication overheads present, the time it takes to complete a task will still likely decrease with increasing team size, but not at the same rate because the communication of team-members adds waste to the process. If the communication becomes too complex there is a point at which adding additional team members actually increases the time it would take to complete the task. Some tasks are non-partitionable. Non-partitionable tasks cannot be distributed among team members and requires the same amount of effort regardless of team size.

In todays commercial CAD systems, the atomic unit of simultaneous contribution is the part. This means that multiple users cannot simultaneously contribute to a part because it cannot be divided into smaller units of simultaneous contribution. Since some parts can take a significant amount of time to model, having a single user model these parts can slow down the overall design process. In a previous chapter, it is suggested that in a simultaneous multi-user system, the atomic unit be the feature. This means that no two users may simultaneously contribute to the same feature, but may simultaneously contribute to the same part.

Previous methods of reducing semantic conflicts focus on blocking simultaneous access to all or portions of a model [8, 13, 15, 31, 32, 32–34, 36, 37]. These approaches divide the atomic unit of simultaneous contribution into sections of the part, made up of sets of features. When the atomic unit of simultaneous contribution is the feature it allows a higher level of partitioning compared to previous methods. Ultimately, this makes it possible for a model to be partitioned and distributed among a modeling team in the most effective way possible, more closely approximating a fully partitionable task.

6.3 Methodology

Setting the feature as the atomic unit in a multi-user CAD system facilitates the best possible multi-user distribution and thus the potential for the most efficient multi-user modeling. However, modeling in this type of environment can result in many semantic conflicts without a method for users to effectively communicate and organize their various tasks. Semantic conflicts are a type of communication overhead and are caused by miscommunications between designers or a lack of organization to coordinate multi-user efforts. This chapter sets out to give evidence for the princi-

ple that enhancing communication and organization reduces semantic conflicts in multi-user CAD. Di Penta et al. discuss the notion that the number of communication channels, and thus communication complexity, increases as more members are added to a team. This relationship is illustrated in Fig. 6.1 and described by the following equation where C is the number of communication channels:

$$C = m(m-1)/2 \tag{6.2}$$

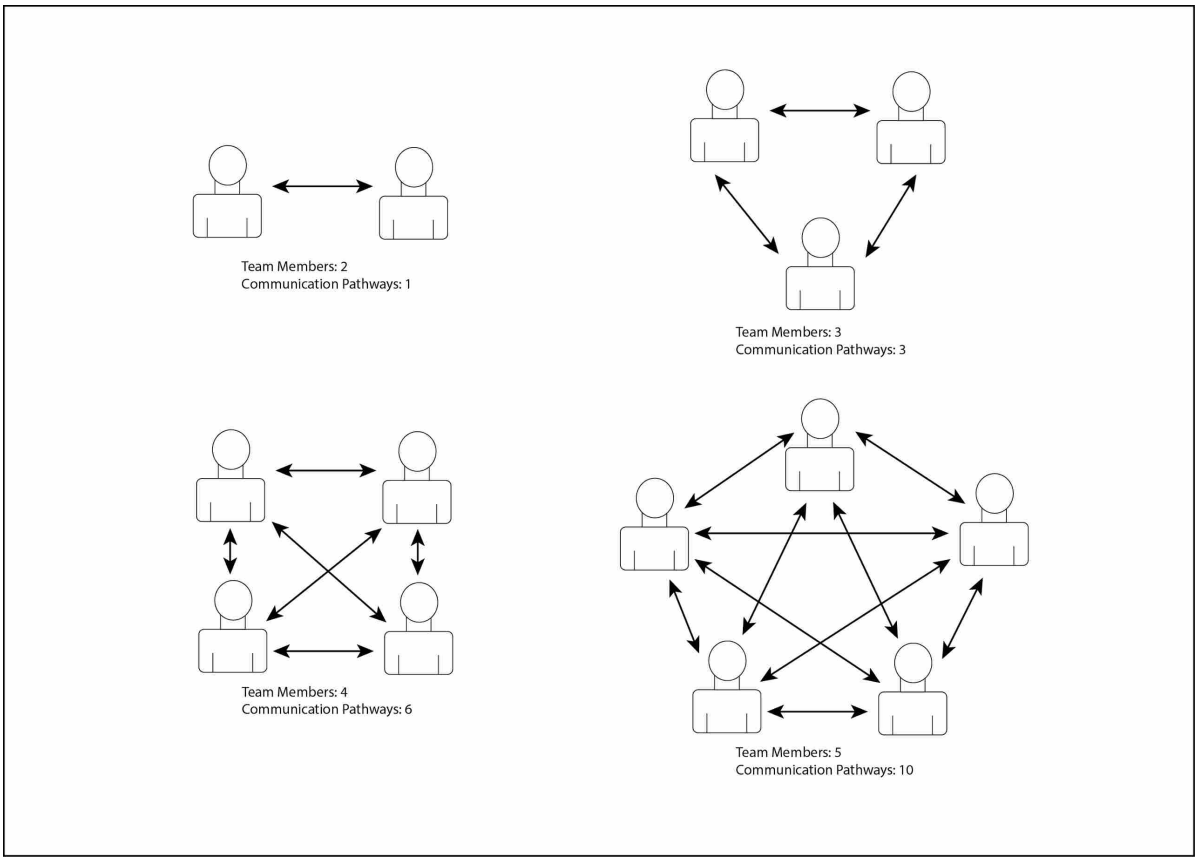


Figure 6.1: Communication complexity increases with the number of team members

A centralized communication system has the potential to significantly reduce the number of communication channels, regardless of the number team members. Since users only have one effective channel of communication, communication complexity is significantly reduced (see Fig. 6.2). Chat rooms and conference calls are two examples of centralized communication systems.

The main benefit of these types of systems is that they allow for the dissemination of information to the entire group at once. Since communicating with all members of the group only requires a team-member to communicate via one channel, the number of pathways required for communicating is reduced to one. This can result in a significant decrease in communication overhead.

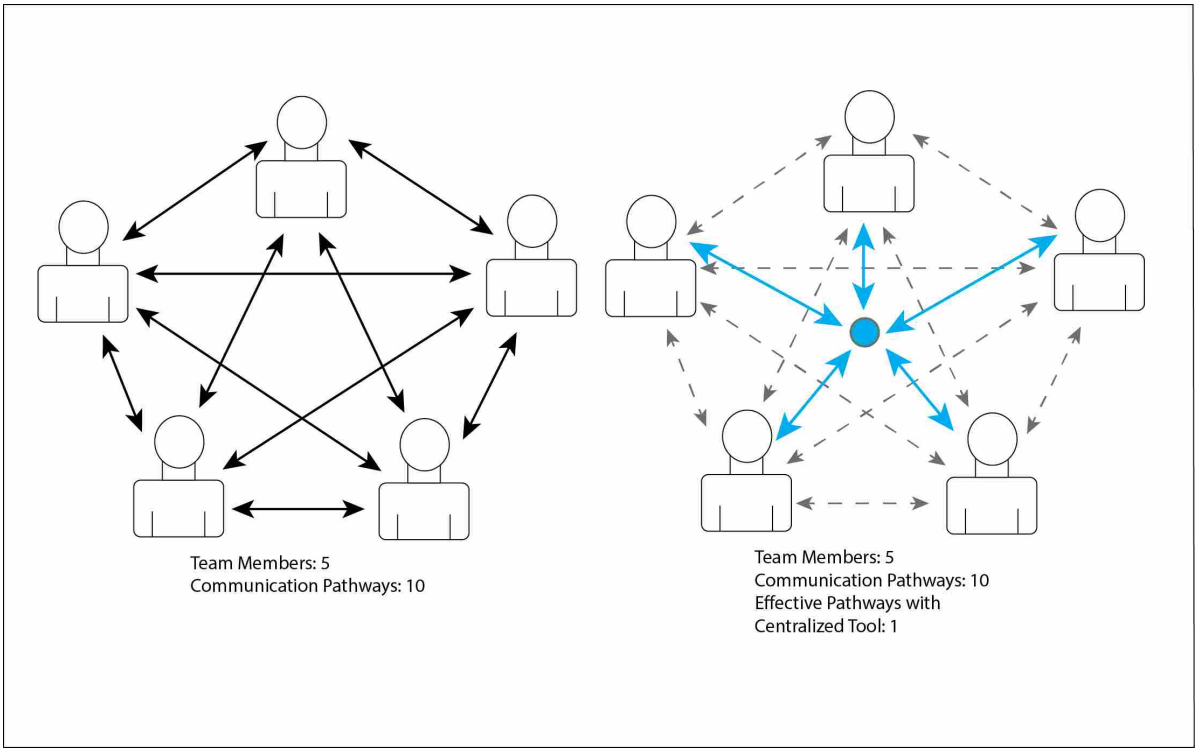


Figure 6.2: A centralized communication system reduces the number of communication channels to one

However, many existing centralized communication tools provide no intrinsic support for organizing the tasks needed to accomplish a design project. Other disciplines, such as software development, have attempted to reduce their teams communication overheads by implementing centralized task creation and management systems for their teams.

A centralized and integrated task management system is proposed to minimize semantic conflicts and preserve agility for modeling in multi-user CAD. This system provides a list of all active users in the model and provides methods for users to add, remove, modify, assign, and mark completion of design tasks. Using this simple tool, the organization of the design project is managed by the group as the model is built.

Beyond enabling team members to communicate through a central hub, an integrated task-list tool allows all users to simultaneously manage design tasks inside the multi-user CAD system. This integrated and centralized tool reduces the number of communication channels and provides an effective framework for users to coordinate and organize their efforts. Since this system operates with the notion that the feature is the atomic unit, it enables multi-user modeling agility by allowing users to contribute wherever and whenever they are needed.

6.4 Integrated Collaboration Tools in NXConnect

To assist users in simultaneous collaboration in NXConnect, two integrated collaboration tools have been developed. One is a chat system designed to allow instant communication between users in the same model. The other is an integrated task management system to organize the tasks needed to complete the model by the multi-user team. Both systems automatically group users which are in the same part model and allow users to create unique groups to communicate. These collaboration tools are designed to assist simultaneous users in organizing a design workflow to complete the model in the most efficient way possible.

6.4.1 Integrated Chat System

The integrated chat system provides users with the ability to communicate via text based messages with any user currently logged into NXConnect. Each user is automatically added to a part-based chat group where modelers working within the same part are able to communicate. If different groups are desired, users can create additional custom groups with a custom set of contributors. All chat groups in which the user is currently a member are displayed to the left of the chat box. The names of each collaborator in the active chat group are displayed to the upper left of the chat box. The user can click on a group name to change the active chat group, updating the chat messages and the list of collaborators (see Fig. 6.3). This functionality is housed within the collaboration tools window.

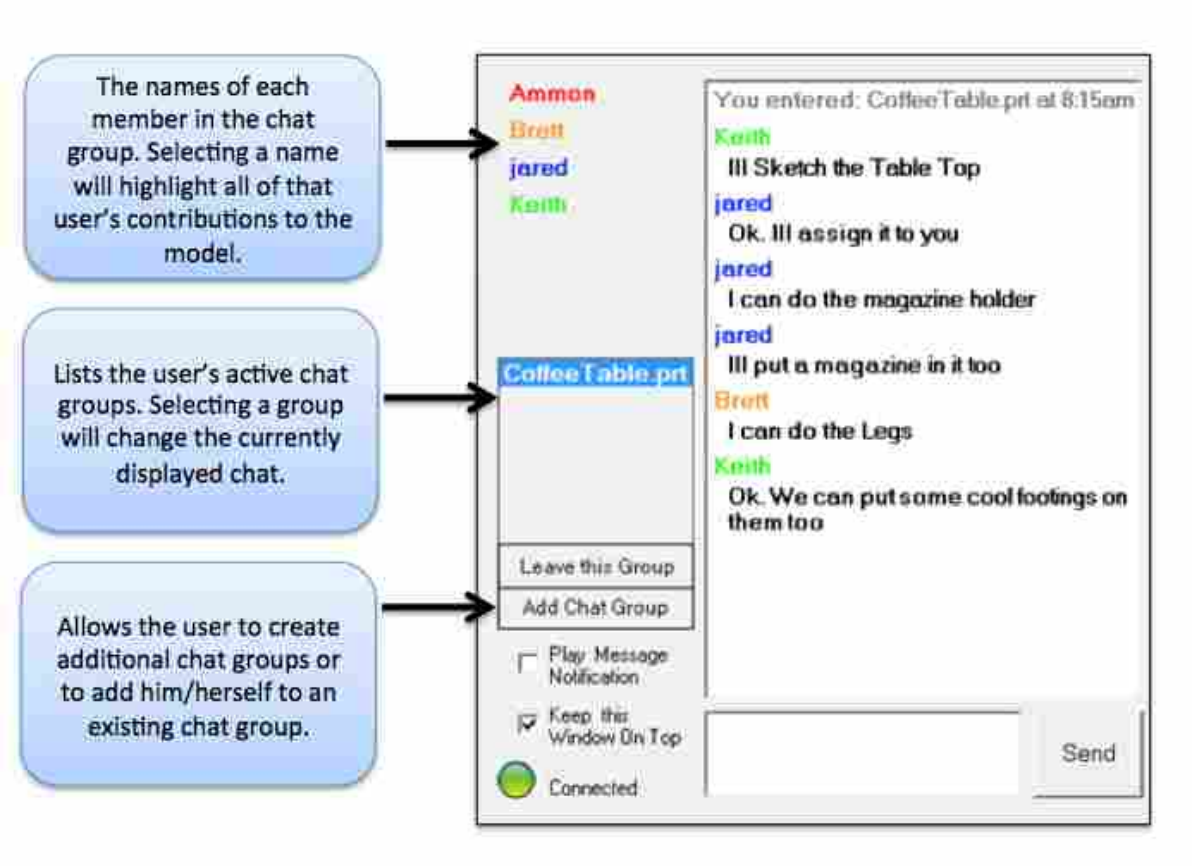


Figure 6.3: Image of the integrated chat system

6.4.2 Integrated Task Management System

The integrated task management system is designed to be simple, lightweight, and user friendly. The simple layout allows users to quickly ascertain task assignments without being overwhelmed with information. Ethnographic and business-oriented work by Bellotti et al. suggests that an optimal task management tool should be as simple as possible and not require large amounts of screen space or extremely detailed descriptions [60]. This simplicity allows for the information to be understood at a glance.

The task management system is nested within the collaboration tools window directly to the right of the chat box. The tool is shared between all collaborators in the active chat group and is updated for all collaborators in real time. When a user changes the active chat group, the tasks corresponding to that group are automatically displayed. All tasks are stored in the central database

and associated with the model to which they apply. This allows the task management system to be persistent between NXConnect modeling sessions.

The task management system displays a table of tasks with an add task button at the bottom. Each row in the table is a task which corresponds to the following columns: a checkbox for task completion, a description of the task to be performed, an assignee, and a delete button. The task complete checkbox allows any user to mark when a task has been completed. A visual indication of completion is given by adding a check to the box and a green background to the task description. This can be checked and unchecked on demand, by any member of the multi-user team. The task description column provides an area for users to describe the details to complete the task. This section can be edited by any users after its initial creation. The assignee drop-down box contains a list of all members of the group whether or not they are actively in the part. Any member of the group can assign a task to himself, another user, or the group as a whole. When a task is assigned to the group, it means that no single user is responsible for completing the task. The final column simply contains a button to delete a task from the list. The task management system is shown in Fig. 6.4.

6.5 Experiment

To verify that the integrated task management system reduces the number of semantic conflicts in a multi-user modeling session, several timed tests were performed by modelers using NXConnect with and without the task management system. Twelve users experienced in NXConnect participated in performing the tests. Fig. 6.5 shows the order of events in the overall experiment. In order to account for varying skill levels between modelers, the experiment began by measuring a baseline speed for each modeler. This was done by giving each tester an identical model specification and timing each of them as they completed the model. This baseline speed test was repeated with a different model at the end of the testing to account for each modelers improvement in modeling through the tests.

After the preliminary individual test, the users were divided into two groups of six. One group was given access to the integrated task management system and one was not. These groups were then both divided in two teams, thereby forming four teams with three members (two teams from each group). For the first two rounds of testing, teams one and two were given access to

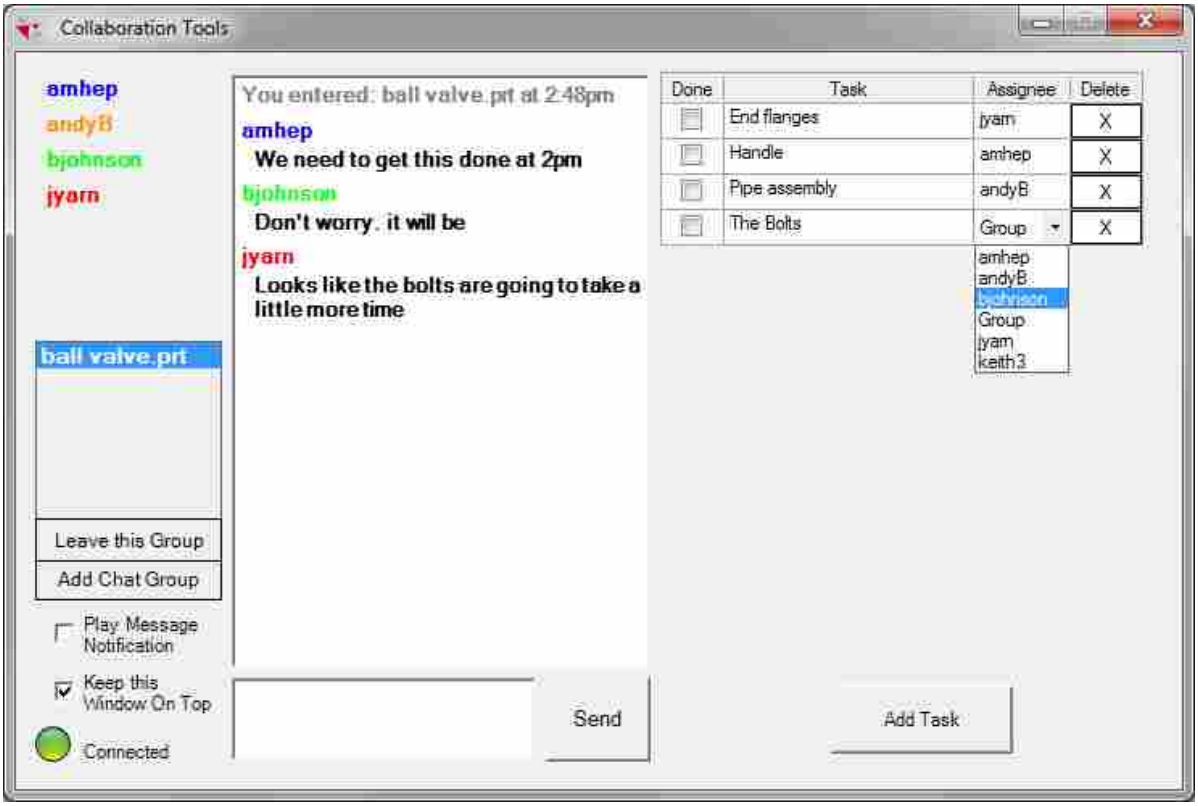


Figure 6.4: The integrated task management system is separated into four columns: task complete checkbox, task description, assignee combo box, and delete button

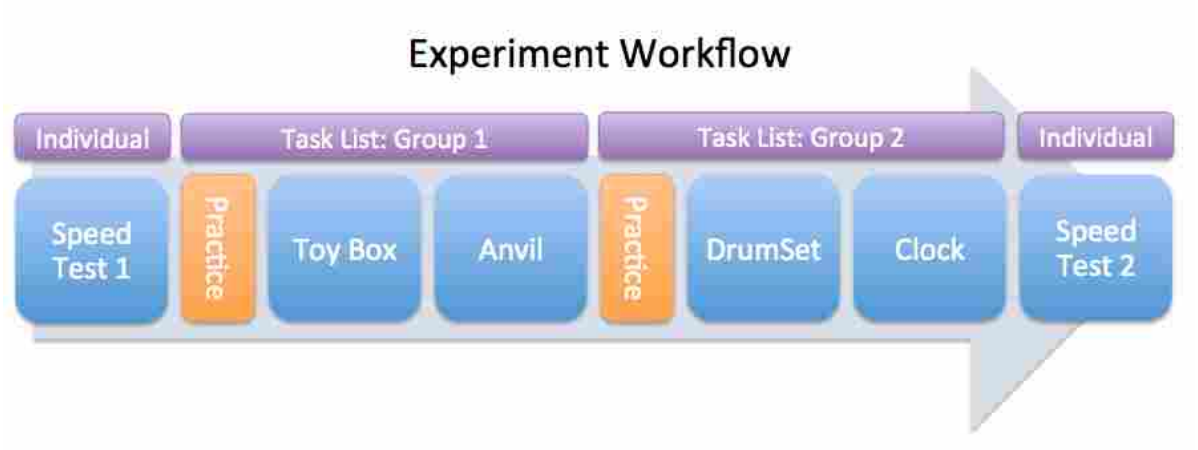


Figure 6.5: The order of tests starting and ending with a speed test evaluation

the integrated task management system, while teams three and four were only given access to the chat tool. A practice round was conducted that allowed users to become familiar using the tools

available to them. Following the practice round, two rounds of testing were conducted. Between rounds, the two teams of each group were randomly reshuffled, creating four more teams. In rounds three and four, the teams were the same as in rounds one and two. However, in these rounds of testing, access to the integrated task management system was switched (see Fig. 6.1). When access to the tool was switched, teams were given another chance to practice using their assigned collaboration tools before the next two rounds of testing. Fig. 6.6 shows an image of each of the models created by the teams.

Table 6.1: Division of modelers for each round of testing

Round	**1**								**2**							
Test	**1 (Toy Box)**				**2 (Anvil)**				**3 (Drum-Set)**				**4 (Clock)**			
Group	**1** (Chat only)		**2** (Chat & Task Tool)		**1** (Chat only)		**2** (Chat & Task Tool)		**2** (Chat only)		**1** (Chat & Task Tool)		**2** (Chat only)		**1** (Chat & Task Tool)	
Modeler	1	2	3	4	5	6	7	8	3	4	1	2	7	8	5	6
Modeler	1	4	7	10	1	2	7	8	7	10	1	4	7	8	1	2
Modeler	2	5	8	11	3	4	9	10	8	11	2	5	9	10	3	4
Modeler	3	6	9	12	5	6	11	12	9	12	3	6	11	12	5	6

For each test, the teams were assigned a single part file. Every modeler was given a unique username for every trial to prevent modelers from knowing the identity of their teammates. Before the tests, both groups received training on how to use the chat tool. Teams with the integrated task management system received additional training on how to use the task management system. All teams were restricted from all forms of voice communication. No restrictions were placed on what external non-voice communication software the groups decided to use. Once the training had finished, the modelers each returned to their workstations and received an instruction packet which included:

1. A unique login and password
2. Name of the part file to open
3. Design specifications for the model
4. Image of completed model for reference

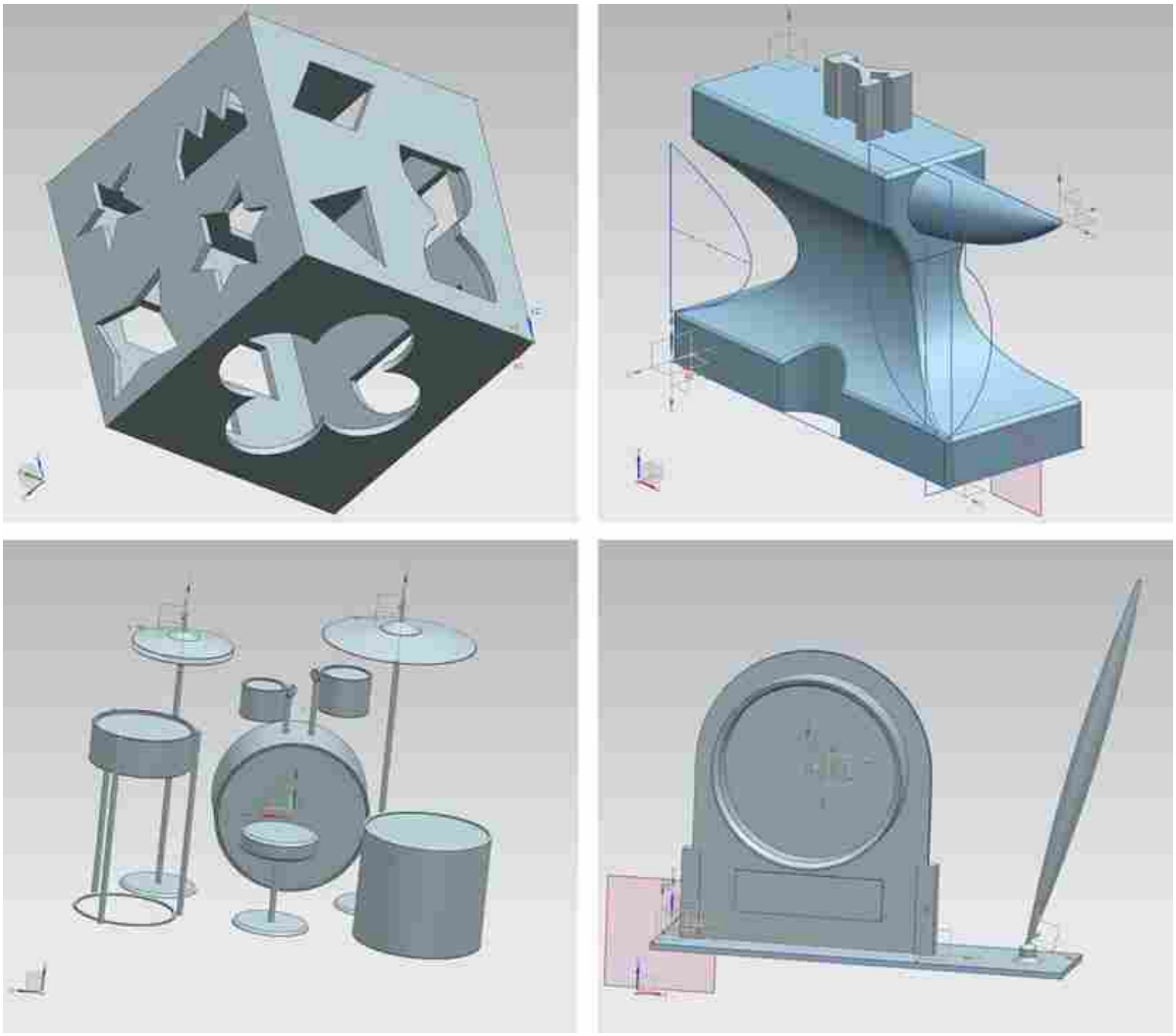


Figure 6.6: Images of the various models modeled by each group

5. A recap of instructions they received during the briefing

This process was repeated with each of the four parts. The time for each team to complete each model was measured and their chats were recorded in a log. In each round, the first model (such as the toy box in round one) was measured and inspected by the researchers to determine if the exact specifications of the model had been met before accepting the model as completed. The second model in each round (such as the anvil in round one) was left more open-ended. Modelers determined when they had completed the model and the researchers confirmed their decision to ensure all necessary features were present.

6.6 Results

The results of our experiment are presented in three distinct ways:

1. Model Time Comparison - Compares the time to complete different models, based on whether or not the teams had the task management system (model held constant, team members varied)

2. Team Time Comparison - Compares each unique teams performance with and without the integrated task management system for the various models they completed (team members held constant, models varied)

3. Chat Message Comparison - Compares the number of messages (coordination and confusion) sent by teams with and without the task management system for each model

6.6.1 Model Time Comparison Results

Fig. 6.7 shows the average raw time taken to complete each model with and without the integrated task management system. The blue columns display the average time of the two teams without the integrated task management system. The red columns display the average time of the two teams with the integrated task management system. This data does not take into account the varying skill levels of the groups members.

The raw time to complete each model was the first telling sign of improvement from the integrated task management system. In all four models, the two groups with the task management system averaged a faster time than the two groups without the integrated task management system. The average time improvement for all models using the integrated task management system was 25%.

In addition to the raw data, the results were corrected to account for the varying skill level among teams and to more accurately compare teams with more highly skilled modelers to teams with fewer highly skilled modelers. This adjustment was performed by giving each participant a modeling score M, based on their individual speed test times. This is calculated by

$$M = t_1 + t_2 \tag{6.3}$$

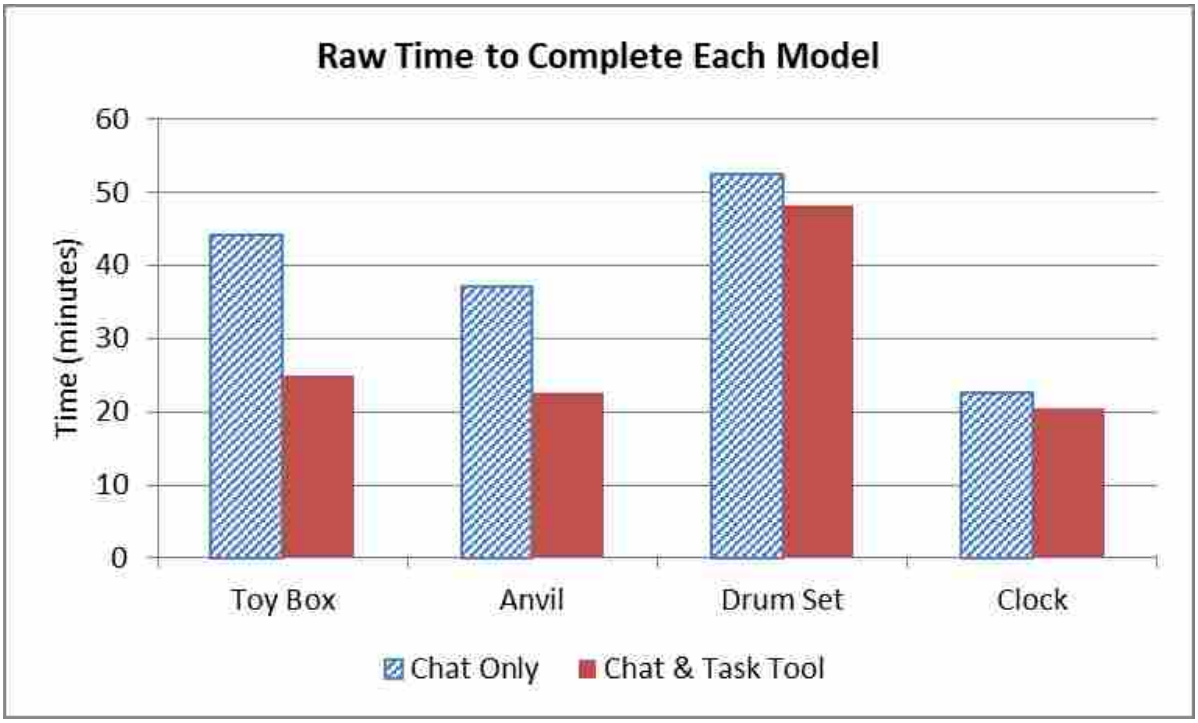


Figure 6.7: Raw time to complete each model

Where t_1 represents the the participants time from the first individual speed test and t_2 represents the participants time from the second individual speed test.

$$C = M_1 + M_2 + M_3 \tag{6.4}$$

Where M_1 ,M_2 ,M_3 represent the respective scores of each modeler on the team.

An average score across all teams, C_A , was calculated for each model

$$C_A = (c_1 + c_2 + c_3 + c_4)/4 \tag{6.5}$$

Where C_1, C_2, C_3, C_4 represent the respective scores of each team participating in the test. The corrected time for a specific model, A_i, for each team, i, is then calculated by

$$A_i = (C_A/C_i)T_i \tag{6.6}$$

Where C_i represents the team score C, for team i, and T_i represents the time it took for team i to complete the model. These corrected times were used to calculate the data represented in Fig. 6.8 in the results section.

Fig. 6.8 shows the average corrected time taken to complete each model with and without the integrated task management system. The times are corrected using Equation 6.7 to take into account the varying skill level of the modelers in each team based on their performance in the pre and post speed tests. The blue columns display average corrected times without the integrated task management system. This is compared to the red columns displaying the average corrected times with the integrated task management system.

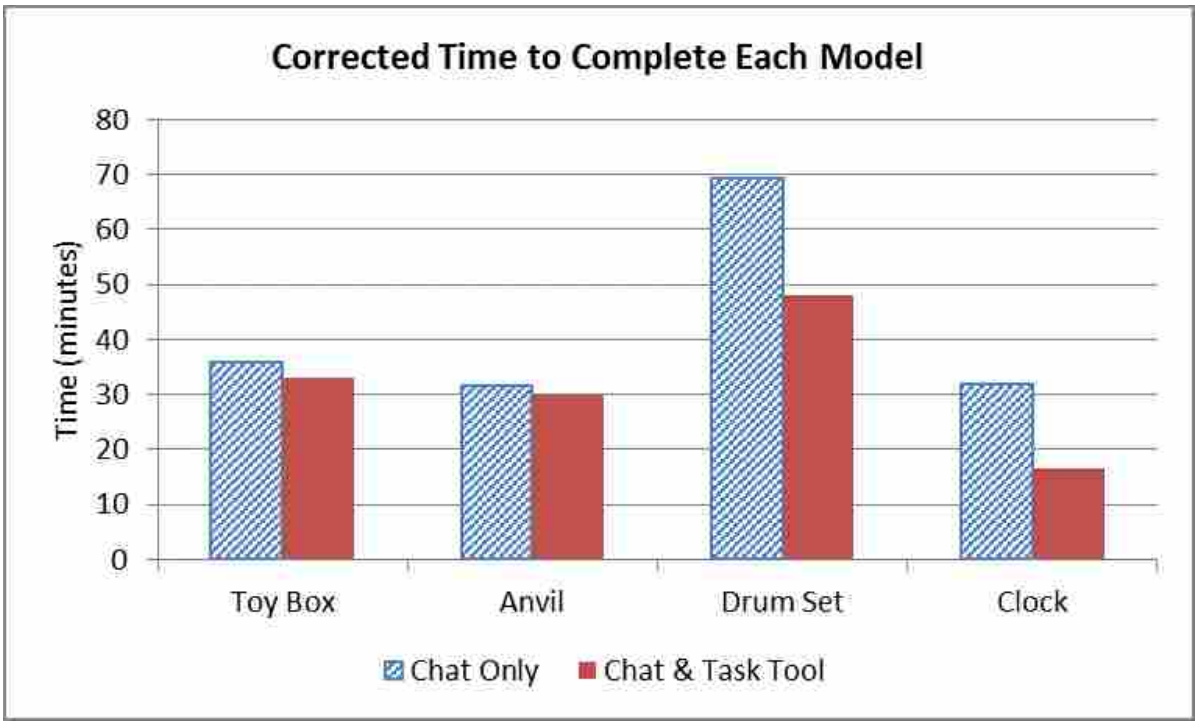


Figure 6.8: Corrected time to complete each model

After correcting the data to account for the skill level of different modelers, the result trends were consistent with the raw times. In every model, the teams with the integrated task management system still averaged a faster time than the teams who only used chat. The average time reduction for the corrected test data was 23%. These results indicate that the time to complete a model or

task will be improved when using an integrated task management system for multi-user modeling activities in industry by a similar time percentage.

6.6.2 Team Time Comparison Results

In addition to comparing the time to complete each model, the times for teams to complete models with and without the integrated task management system were compared. Each of the eight unique teams participated in one model with the integrated task management system and one without (see Fig. 6.6). Since the times for each team to complete the two different models with and without the list are not comparable, the times for model completion are compared against the average for all teams. This is done by finding a Relative Time Percentage, R_i, for each team, i, calculated by

$$R_i = T_i / T_R \tag{6.7}$$

Where T_i is the raw time it took team i to complete the model, and T_a is the average time it took each team to complete the model. Fig. 6.9 shows each teams relative performance with and without the integrated task management system.

From the chart above, six out of the eight teams performed significantly better when using the integrated task management system tool. The average difference between team performances with the task management system vs without (on different models) was 25%. These results indicate that teams using an integrated task management system in a multi-user design environment will experience a significant improvement in their teams performance.

6.6.3 Chat Message Comparison Results

To determine the effect of the integrated task management system on communication, coordination and confusion messages were counted from a chat log that was recorded for each team while creating the model. These messages were compared between the teams who had the integrated task management system and those who only had access to the chat. Coordination messages were defined as all messages having to do with coordination of work effort, including confirmation. Examples of actual coordination messages include, “I’m doing the negative extrude through

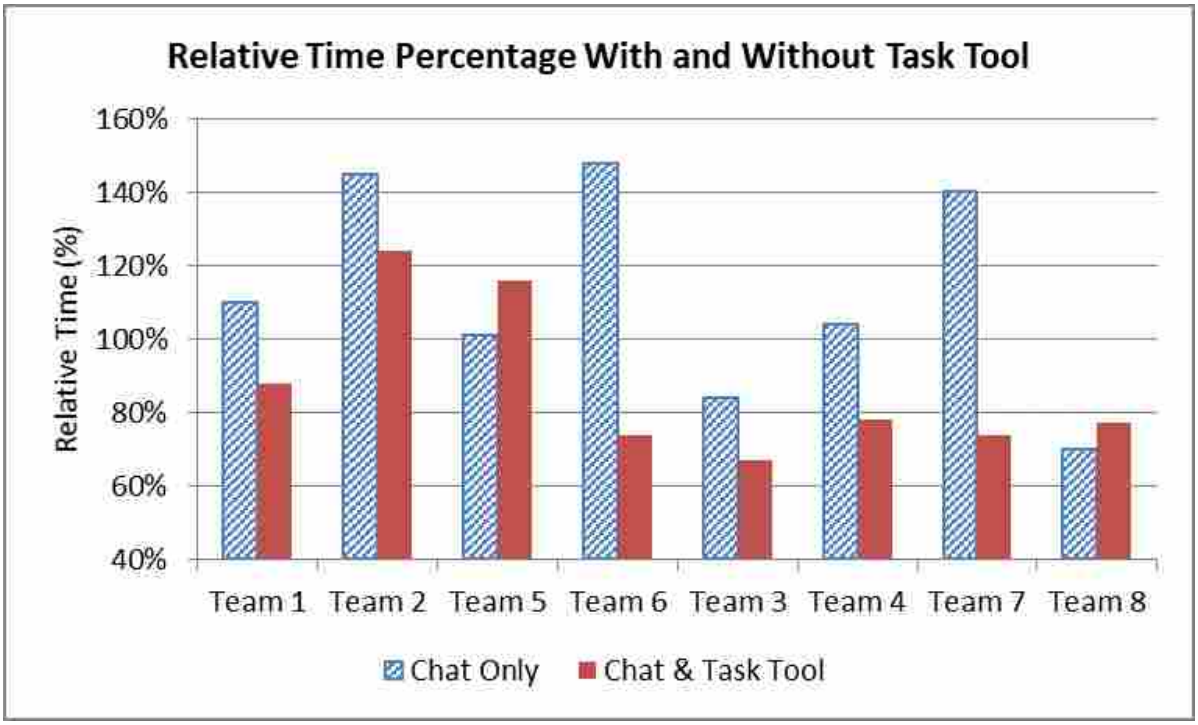


Figure 6.9: Relative time percentage required for completion by teams with and without the task management system

the middle of the block," and, "i'm working on the six-sided star now," and, "k i'll do the throwing star." Fig. 6.10 shows the average coordination messages sent for each model comparing the teams using only chat to those that used chat and the task management system. One caveat of this data is that two chat logs were lost: one chat log for the toy box with chat only, and one chat log for the anvil with chat only were lost. Unfortunately, this means that the average of the chat only for toy box and anvil is only one team each.

The results of the coordination message comparison suggest that the number of coordination messages sent is reduced by teams who are able to utilize the integrated task management system. The overall average reduction in coordination messages for teams using the task management system in these models is 38%. The reduction in coordination messages is attributed to the task management system which allowed users to coordinate their work in an organized manner. Using the task management system gave teams a systematic means by which to communicate user activities. Since coordination information was communicated through the task management system, there was less need to coordinate through chat, thus removing communication overhead.

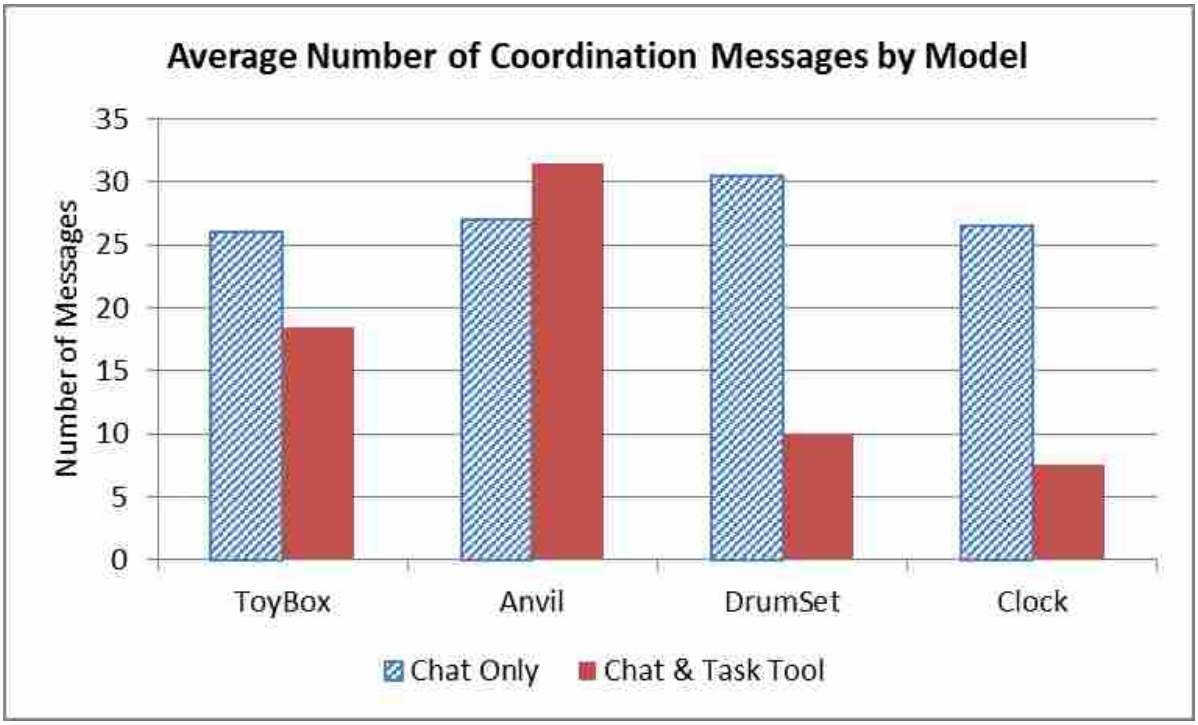


Figure 6.10: Comparison of coordination chat messages sent for each model

Confusion messages were also counted and compared between teams using the integrated task management system and those who did not. Confusion messages were defined as messages which show evidence of a user expecting one thing to happen, but experiencing something else happening and not knowing why. These messages contain elements of surprise, frustration and/or misunderstanding. Confusion messages are a prime example of "communication overhead" for engineering design teams. Examples of confusion messages are, "i said i was doing the base," and, "[He] was supposed to be doing that, and, "someone did my cuts." Fig. 6.11 is a graph of the average number of confusion messages sent for each of the models, comparing teams using the task management system and those who did not.

The results of the confusion message comparison suggest that confusion messages are drastically reduced when using the integrated task management system. The average reduction in confusion messages in these tests for teams using the task management system while creating these models is 76%. This large reduction in the number of confusion messages sent strongly suggests that teams utilizing the task management system had significantly less semantic conflicts than those who did not. The ability to identify tasks, assign team members and mark completion of those tasks

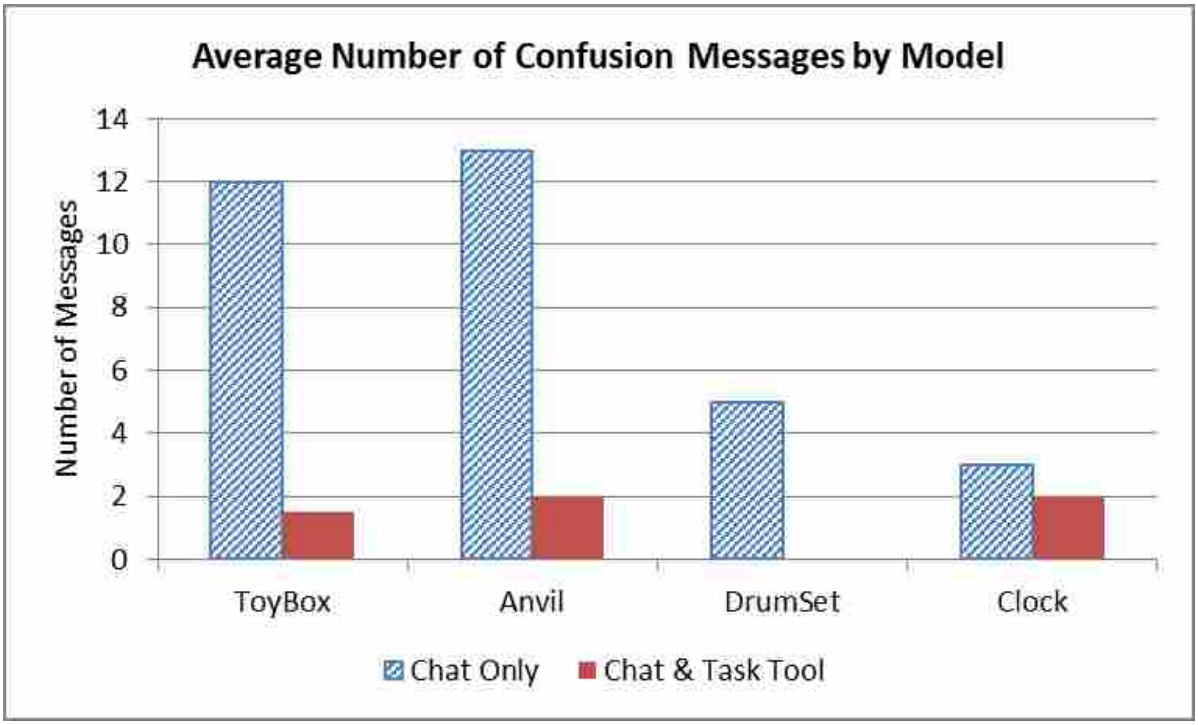


Figure 6.11: Comparison of confusion chat messages sent for each model

is a simple and effective way to organize users assignments. The improved organization and coordination that the task management system provides reduces communication overhead and thus the overall time to complete design models.

6.7 Conclusion

An integrated task management system was implemented in multi-user CAD to prevent semantic conflicts by reducing the number of communication channels required and providing a simple organization framework to coordinate user efforts. This method preserves multi-user agility by giving all users full access to the entire part, except for features which are being edited by another user. The task management system integrated in NXConnect was found to improve modeling speeds by helping users to coordinate responsibilities, thus avoiding semantic conflicts. The integrated task management system was also found to reduce the number of coordination and confusion messages sent by multi-user CAD modeling teams. These results suggest that multi-user teams utilizing an integrated task management system will improve coordination and reduce semantic conflicts while simultaneously contributing to a collaborative design model. This pro-

vides support for the principle that enhancing communication and organization reduces semantic conflicts in multi-user CAD.

CHAPTER 7. CONCLUSIONS

This dissertation presented methods for conflict management and model consistency to enable a robust, usable and efficient implementation of multi-user CAD based on the following principles:

1. The atomic unit of a multi-user, feature-based CAD system is the feature, meaning that no two users may simultaneously edit the same feature
2. A global order of operations in multi-user CAD ensures consistency for all replicated models and resolves conflicting operations in a multi-user system
3. The number of referenced topological entities is always less than the number of topological entities generated in a history-based model
4. Automatically reserving features prevents waste derived from two users performing redundant work on the same feature
5. Enhancing communication and organization reduces semantic conflicts in multi-user CAD

These principles establish the theoretical foundation of the methods developed. Implementations and results of methods derive the following 10 points:

1. Feature reservation automatically prevents syntactic, feature-self conflicts
2. On demand reservation requests allow users to access reserved features without having to wait longer than a few seconds for the owning user
3. Operational consistency automatically prevents all types of syntactic conflicts with model consistency by enforcing a global order of operations
4. The conflict resolution with data preservation method provides that many operations which cause conflicts are preserved

5. The persistent naming method provides model consistency by ensuring consistent references to topological entities between clients

6. The persistent naming performance enhancements improved the speed for topological entity identification by one to two orders of magnitude

7. The automated feature reservation helps to reduce the time for a team performing an editing task in a CAD model because it reduces the number of semantic conflicts

8. A mathematical model has been developed which predicts the times for teams performing simultaneous edits to a model with and without the automated feature reservation method in place with an error of 22% or less

9. The integrated task management system prevents semantic conflicts by providing a simple organization and communication framework to coordinate user efforts

10. Multi-user teams utilizing an integrated task management system will improve coordination and reduce confusion while simultaneously contributing to a collaborative design model

Points 3 and 5 speak to operational consistency and persistent naming, which are the methods absolutely necessary for a replicated multi-user CAD implementation. These methods are critical because they are the mechanisms to avoid all syntactic conflicts, guaranteeing replication of models on separate clients. Operational consistency ensures simultaneous operations by multiple users will be applied in a global order. This prevents all types of syntactic conflicts which would lead to model inconsistency. Persistent naming provides consistent references to topological entities. This ensures features will be consistently applied to all replicated models. The combination of these methods provides the model consistency necessary to enable a robust replicated multi-user CAD system.

Conflict resolution with data preservation, discussed in point 4, is a waste reducing addition to operational consistency. It preserves conflicting data after a conflict occurs and allows users to resolve the conflict manually. This reduces waste, and thus improves multi-user modeling efficiency, by preserving operation data that would otherwise be lost.

The persistent naming method without the performance enhancements in place is not scalable for large models with many topological entities. Point 6 states that the performance enhancements for persistent naming make this method more usable by increasing the performance to more closely resemble that of the commercial CAD system. This increases the usability the persistent naming method in a replicated multi-user CAD system.

Point 1 says that automated feature reservation helps avoid syntactic, feature-self conflicts. That said, with operation consistency in place, automated feature reservation is not absolutely necessary to guarantee model consistency. Since operational consistency ensures a global order of operations, two users may simultaneously edit the same feature without causing consistency problems. However, without the automated feature reservation in place, the last user to finish the edit will override all others. This can cause confusion and waste time. Adding the automated feature reservation prevents users from editing the same feature at the same time, so this confusion doesn't occur. On demand reservation removal, discussed in point 2, provides additional convenience by allowing users to remove reservations to features by requesting access and allowing a user a limited time to respond.

Points 7 and 8 further show how automated feature reservation reduces semantic conflicts by coloring the features which users are editing. The colors instantly communicate the message that users should to avoid editing the given feature. This communication is the mechanism which enables users to organize themselves in a way that avoids feature-self semantic conflicts.

Points 9 and 10, referencing the integrated task management system, further show that communication and organization are the keys to avoid semantic conflicts. The integrated task management system provides users with a simple way to organize and communicate their collaborative tasks, allowing users to intelligently avoid conflicts. Teams using the integrated task management system completed models as a team an average of 25% faster than those that did not use it. Teams using this system also were found to send significantly fewer chat messages for coordination and were found to have much less confusion during the modeling activities.

In summary, operation consistency and persistent naming provide robust mechanisms to ensure model consistency in a replicated, multi-user CAD system. Conflict resolution with data preservation provide additional efficiency to the operational consistency method by preserving work when conflicts occur. The performance enhancements for persistent naming make this

method more usable by increasing the performance to more closely resemble that of the commercial CAD system. Automated feature reservation avoids feature-self syntactic conflicts. Although automated feature reservation is not absolutely necessary for model consistency when operational consistency is in place, it reduces semantic conflicts and thus improves efficiency by communicating where users are working. The integrated task management system additionally provides a simple system for users to communicate and organize their work, thus increasing efficiency. The methods, implementations and results (including those shown in Appendix A) of this dissertation, increase the robustness, usability and user efficiency of multi-user CAD.

7.1 Limitations and Future Work

One limitation to this work is that the mathematical model and study in Chapter 5 only looks at semantic conflict reduction through automated feature reservation for models without feature dependencies. This means the mathematical model only predicts time wasted due to feature-self conflicts. Models without feature dependencies can be distributed among multiple users without any concern for feature order. It also means there is no concern for parent-child and child-child conflicts. Further studies should be done see how the automated feature reservation effects time for modeling when feature dependencies are present.

Another limitation to this work is that the integrated task management system is quite simple. It is no more than a multi-user to-do list that describes tasks to be done and a user it should be assigned to. A more complex integrated system could include dependencies between these tasks as well as projected time for completion. More complexity may provide additional savings by providing additional information to users. It is also possible that the additional complexity may add time. The complexity and resulting time savings may also be dependent on the complexity of the modeling activity. Additional studies should be performed to determine the optimal integrated task management system for a specific modeling activity.

Another limitation to the work in this dissertation is that it is limited to a single commercial CAD system. Additional research should be done to enable multi-user interaction between various commercial CAD systems. This effort will likely be able to use the methods presented in this dissertation for conflict management and model consistency. The major challenge in this effort is the development of a neutral feature-based modeling representation. This is a major challenge

because each CAD system often use a different set of parameters to define each feature. These should be able to be translated with some effort. However, sometimes these features are mathematically different from each other. In addition, each CAD system has unique features which are not implemented in the others. Therefore, a method to represent non-translatable features must be developed.

One big hurdle in developing a commercial solution to multi-user CAD is scalability. As the system supports more users it will be a computational challenge to support them. Since the replicated multi-user architecture requires operations by all users in the multi-user model to be performed on each user machine, the number of operations for given time interval is limited based on the non-scalable resources of the machine. Research should investigate techniques to parallelize the geometry kernel for CAD systems to enable parallel processing of modeling operations. This would improve the scalability of both centralized and replicated implementations of multi-user CAD.

For assembly scalability, research should be done to provide a light weight update of assembly models. The model the user is actively working should be a full weight model, but all other models could be light weight geometric representations. Updates from other parts could be communicated by updating the light weight geometry models in the assembly. Real-time updates could even be provided based on user proximity. Models that are closer where the user is working could update in real-time, but models farther away could update less frequently. This would optimize the number of updates required.

One step beyond a multi-user CAD inter-operable system, is a system which enables multi-user interaction between engineering disciplines. Currently isogeometric analysis enables the geometric integration of CAD and FEA based on NURBS and T-Splines geometry models. CAD currently utilizes trimmed NURBS for geometry representation which is not suitable for isogeometric analysis. Analysis suitable T-Splines allow for the direct analysis of water-tight design models. However, there are currently no feature-based systems which generate analysis suitable T-splines. A feature-based geometry kernel which generates analysis suitable T-splines would allow for a tighter integration of design and analysis. Many years of research are required to develop a suitable system for the integration of all engineering disciplines in a simultaneous multi-user environment.

REFERENCES

[1] Hepworth, A. I., Tew, K., Nysetvold, T., Bennett, M., and Greg Jensen, C., 2014. "Automated conflict avoidance in multi-user cad." *Computer-Aided Design and Applications,* **11**(2), pp. 141–152. 2

[2] Hepworth, A. I., Tew, K., Trent, M., Ricks, D., Jensen, C. G., and Red, E., 2014. "Model consistency and conflict resolution with data preservation in multi-user cad." *Journal of Computing and Information Science in Engineering,* **14**(2). 2

[3] Hepworth, A. I., Nysetvold, T., Bennett, J., Phelps, G., and Jensen, C. G., 2014. "Scalable integration of commercial file types in multi-user cad." *Computer-Aided Design and Applications,* **11**(4), pp. 459–467. 2, 3, 44

[4] Hepworth, A. I., Staves, D., Hill, L., Tew, K., Jensen, C. G., and Red, W. E., 2014. "Enhancements for improved topological entity identification performance in multi-user cad." *accepted for publication in Computer-Aided Design and Applications*. 2

[5] Hepworth, A. I., Halterman, K., Stone, B., Yarn, J., and Jensen, C. G., 2014. "An integrated task management system to reduce semantic conflicts in multi-user cad.". 2

[6] Hepworth, A. I., DeFigueiredo, Bryce, S. D., Fronk, N., and Jensen, C. G., 2014. "Semantic conflict reduction through automated feature reservation in multi-user cad." In *accepted for the 2014 International Conference on Collaboration Technologies and Systems*, American Society of Mechanical Engineers. 2

[7] Marshall, F. D., 2012. "Model decomposition and constraints to parametrically partition design space in a collaborative cax environment." Master's thesis, Brigham Young University. Department of Mechanical Engineering. 2, 5, 16, 71

[8] Moncur, R. A., Jensen, C. G., Teng, C.-C., and Red, E., 2013. "Data consistency and conflict avoidance in a multi-user cax environment." *Computer-Aided Design and Applications,* **10**(5), pp. 727–744. 2, 4, 6, 17, 71, 88

[9] Red, E., French, D., Jensen, G., Walker, S. S., and Madsen, P., 2013. "Emerging design methods and tools in collaborative product development." *Journal of Computing and Information Science in Engineering,* **13**(3), p. 031001. 2, 3

[10] Red, E., Jensen, G., Weerakoon, P., French, D., Benzley, S., and Merkley, K., 2013. "Architectural limitations in multi-user computer-aided engineering applications.." *Computer & Information Science,* **6**(4). 2

[11] Red, E., Holyoak, V., Jensen, C. G., Marshall, F., Ryskamp, J., and Xu, Y., 2010. "ν-cax: A research agenda for collaborative computer-aided applications." *Computer-Aided Design and Applications,* **7**(3), pp. 387–404. 2, 3

[12] Red, E., Jensen, G., French, D., and Weerakoon, P., 2011. "Multi-user architectures for computer-aided engineering collaboration." In *Concurrent Enterprising (ICE), 2011 17th International Conference on*, IEEE, pp. 1–10. 2, 3

[13] Red, E., Marshall, F., Weerakoon, P., and Jensen, C. G., 2013. "Considerations for multi-user decomposition of design spaces." *Computer-Aided Design and Applications,* **10**(5), pp. 803–815. 2, 88

[14] Xu, Y., Red, E., and Jensen, C. G., 2011. "A flexible context architecture for a multi-user gui." *Computer-Aided Design and Applications,* **8**(4), pp. 479–497. 2

[15] Bidarra, R., van den Berg, E., and Bronsvoort, W. F., 2002. "A collaborative feature modeling system." *Journal of Computing and Information Science in Engineering,* **2**(3), pp. 192–198. 3, 5, 16, 71, 88

[16] Qiang, L., Zhang, Y., and Nee, A., 2001. "A distributive and collaborative concurrent product design system through the www/internet." *The International Journal of Advanced Manufacturing Technology,* **17**(5), pp. 315–322. 3

[17] Ramani, K., Hoffmann, C., Agrawal, A., and Babu, M., 2003. "Caddac: Multi-client collaborative shape design system with server-based geometry kernel." *Journal of Computing and Information Science in Engineering,* **3**(2), pp. 170–173. 3

[18] Tang, M., Chou, S.-C., and Dong, J.-X., 2007. "Conflicts classification and solving for collaborative feature modeling." *Advanced Engineering Informatics,* **21**(2), pp. 211–219. 3

[19] Zhou, X., Gao, S., Lie, J., and He, F., 2003. "Flexible concurrency control for synchronized collaborative design." In *ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. 591–598. 3

[20] Zhou, X., Li, J., He, F., and Gao, S., 2003. "A web-based synchronized collaborative solid modeling system." *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing System(China),* **9**(11), pp. 960–965. 3

[21] Jing, S.-x., He, F.-z., Han, S.-h., Cai, X.-t., and Liu, H.-J., 2009. "A method for topological entity correspondence in a replicated collaborative cad system." *Computers in Industry,* **60**(7), pp. 467–475. 3, 7, 11, 17, 33, 49, 50, 71

[22] Stork, A., and Jasnoch, U., 1997. "A collaborative engineering environment." In *Proceedings of the TeamCAD97 Workshop on Collaborative Design*, pp. 25–33. 3, 33

[23] Stork, A., Lukas, U., and Schultz, R., 1998. "Enhancing a commercial 3d cad system by cscw functionality for enabling co-operative modelling via wan." In *Proceedings of the ASME Design Engineering Technical Conferences*. 3, 33

[24] Nam, T.-J., and Wright, D. K., 1998. "Collide: a shared 3d workspace for cad." In *1998 Conference on Network Entities, Leeds, UK*. 3, 33

[25] Dietrich, U., Von Lukas, U., and Morche, I., 1997. "Cooperative modeling with tobaco." In *Proceedings of the TeamCAD97 Workshop on Collaborative Design*, pp. 115–122. 3

[26] Mishra, P., Varshney, A., and Kaufman, A., 1997. "Collabcad: A toolkit for integrated synchronous and asynchronous sharing of cad applications." In *Proceedings TeamCAD: GVU/NIST Workshop on Collaborative Design, Atlanta, GA, USA*, pp. 131–137. 3

[27] Kao, Y.-C., and Lin, G. C., 1998. "Development of a collaborative cad/cam system." *Robotics and Computer-Integrated Manufacturing,* **14**(1), pp. 55–68. 3

[28] Saito, Y., and Shapiro, M., 2005. "Optimistic replication." *ACM Computing Surveys (CSUR),* **37**(1), pp. 42–81. 4

[29] Ellis, C. A., and Gibbs, S. J., 1989. "Concurrency control in groupware systems." *ACM SIGMOD Record,* **18**(2), pp. 399–407. 4

[30] Sun, C., Xia, S., Sun, D., Chen, D., Shen, H., and Cai, W., 2006. "Transparent adaptation of single-user applications for multi-user real-time collaboration." *ACM Transactions on Computer-Human Interaction (TOCHI),* **13**(4), pp. 531–582. 4

[31] Chan, S., Wong, M., and Ng, V., 1999. "Collaborative solid modeling on the www." In *Proceedings of the 1999 ACM symposium on Applied computing*, ACM, pp. 598–602. 5, 16, 71, 88

[32] Li, W., Fuh, J. Y., and Wong, Y., 2004. "An internet-enabled integrated system for co-design and concurrent engineering." *Computers in Industry,* **55**(1), pp. 87–103. 5, 6, 16, 17, 71, 88

[33] Cera, C. D., Braude, I., Comer, I., Kim, T., Han, J., and Regli, W. C., 2003. "Hierarchical role-based viewing for secure collaborative cad." In *ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. 965–974. 5, 16, 71, 88

[34] Bu, J., Jiang, B., and Chen, C., 2006. "Maintaining semantic consistency in real-time collaborative graphics editing systems." *IJCSNS,* **6**(4), p. 57. 6, 17, 71, 88

[35] Jiang, B., Chen, C., and Bu, J., 2001. "Codesign-a collaborative pattern design system based on agent." In *Computer Supported Cooperative Work in Design, The Sixth International Conference on, 2001*, IEEE, pp. 319–323. 6, 17, 71

[36] Shen, L., Hao, Y., Li, M., Zhao, W., and Zheng, J., 2007. "A synchronous collaborative environment for engineering design education." In *Computer Supported Cooperative Work in Design, 2007. CSCWD 2007. 11th International Conference on*, IEEE, pp. 298–303. 6, 17, 71, 88

[37] Chen, L., Song, Z., and Feng, L., 2004. "Internet-enabled real-time collaborative assembly modeling via an e-assembly system: status and promise." *Computer-Aided Design,* **36**(9), pp. 835–847. 7, 17, 71, 88

[38] Lin, K., Chen, D., Sun, C., and Dromey, G., 2005. "Maintaining constraints in collaborative graphic systems: the cogse approach." In *ECSCW 2005*, Springer, pp. 185–204. 7, 17, 71

[39] Liu, F., Xia, S., Shen, H., Sun, C., et al., 2008. "Comaya: incorporating advanced collaboration capabilities into 3d digital media design tools." In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, ACM, pp. 5–8. 7, 17, 71

[40] PLM, S. Parasolid documentation. 11, 50, 51

[41] Commission, . F. C., 2012. A report on consumer wireline broadband performance in the u.s. 29

[42] Li, M., Gao, S., Fuh, J. Y., and Zhang, Y., 2008. "Replicated concurrency control for collaborative feature modelling: A fine granular approach." *Computers in Industry,* **59**(9), pp. 873–881. 33

[43] Kripac, J., 1997. "A mechanism for persistently naming topological entities in history-based parametric solid models." *Computer-Aided Design,* **29**(2), pp. 113–122. 50

[44] Baba-Ali, M., Marcheix, D., and Skapin, X., 2009. "A method to improve matching process by shape characteristics in parametric systems." *Computer-Aided Design and Applications,* **6**(3), pp. 341–350. 50

[45] Bidarra, R., Nyirenda, P. J., and Bronsvoort, W. F., 2005. "A feature-based solution to the persistent naming problem." *Computer-Aided Design and Applications,* **2**(1-4), pp. 517–526. 50

[46] Bidarra, R., and Bronsvoort, W. F., 2002. "Persistent naming through persistent entities." In *Geometric Modeling and Processing, 2002. Proceedings*, IEEE, pp. 233–240. 50

[47] Capoyleas, V., Chen, X., and M Hoffmann, C., 1996. "Generic naming in generative, constraint-based design." *Computer-Aided Design,* **28**(1), pp. 17–26. 50

[48] Chen, X., and Hoffmann, C. M., 1995. "On editability of feature-based design." *Computer-Aided Design,* **27**(12), pp. 905–914. 50

[49] Zhengming, C., Shuming, G., Zhang, F., and Qun-Sheng, P., 2001. "An approach to naming and identifying topological entities." *Chinese Journal of Computers,* **24**(11), pp. 1170–1177. 50

[50] Marcheix, D., and Pierra, G., 2002. "A survey of the persistent naming problem." In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, ACM, pp. 13–22. 50

[51] Wang, Y., and Nnaji, B. O., 2005. "Geometry-based semantic id for persistent and interoperable reference in feature-based parametric modeling." *Computer-Aided Design,* **37**(10), pp. 1081–1093. 50

[52] Wu, J., Zhang, T., Zhang, X., and Zhou, J., 2001. "A face based mechanism for naming, recording and retrieving topological entities." *Computer-Aided Design,* **33**(10), pp. 687–698. 50

[53] Jing, S., He, F., Cai, X., and Liu, H., 2008. "Collaborative naming for replicated collaborative solid modeling system." In *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. 141–150. 50

[54] Liao, B., He, F., and Jing, S., 2005. "Replicated collaborative solid modeling and naming problems." In *Computer Aided Design and Computer Graphics, 2005. Ninth International Conference on*, IEEE, pp. 6–pp. 50

[55] Liao, B., He, F., Jing, S., and Wu, Y., 2006. "A transformation-based method for name converge in quiescent context of replicated solid modeling systems." In *Computer Supported Cooperative Work in Design, 2006. CSCWD'06. 10th International Conference on*, IEEE, pp. 1–4. 50

[56] Jing, S., and Yuan, Q., 2009. "Consistent naming for sweeping features in replicated collaborative modeling system." In *Computer-Aided Industrial Design & Conceptual Design, 2009. CAID & CD 2009. IEEE 10th International Conference on*, IEEE, pp. 899–904. 50

[57] Di Penta, M., Harman, M., Antoniol, G., and Qureshi, F., 2007. "The effect of communication overhead on software maintenance project staffing: a search-based approach." In *Software Maintenance, 2007. ICSM 2007. IEEE International Conference on*, IEEE, pp. 315–324. 87

[58] Putnam, D., Retrived 2013. Team size can be the key to a successful software project. 87

[59] Rodríguez, D., Sicilia, M., García, E., and Harrison, R., 2012. "Empirical findings on team size and productivity in software development." *Journal of Systems and Software,* **85**(3), pp. 562–570. 87

[60] Bellotti, V., Dalal, B., Good, N., Flynn, P., Bobrow, D. G., and Ducheneaut, N., 2004. "What a to-do: studies of task management towards the design of a personal task list manager." In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, pp. 735–742. 92

APPENDIX A. LARGE SCALE MULTI-USER CAD MODELS

The following models were created in NXConnect after the conflict management, model consistency and persistent naming methods were implemented. These examples help to qualitatively show the scalability of these methods.

The example shown in A.1 is an extremely simple model. This is a bulkhead for an unmanned aerial vehicle. The model took three modelers less than 10 min. to model simultaneously in NXConnect.

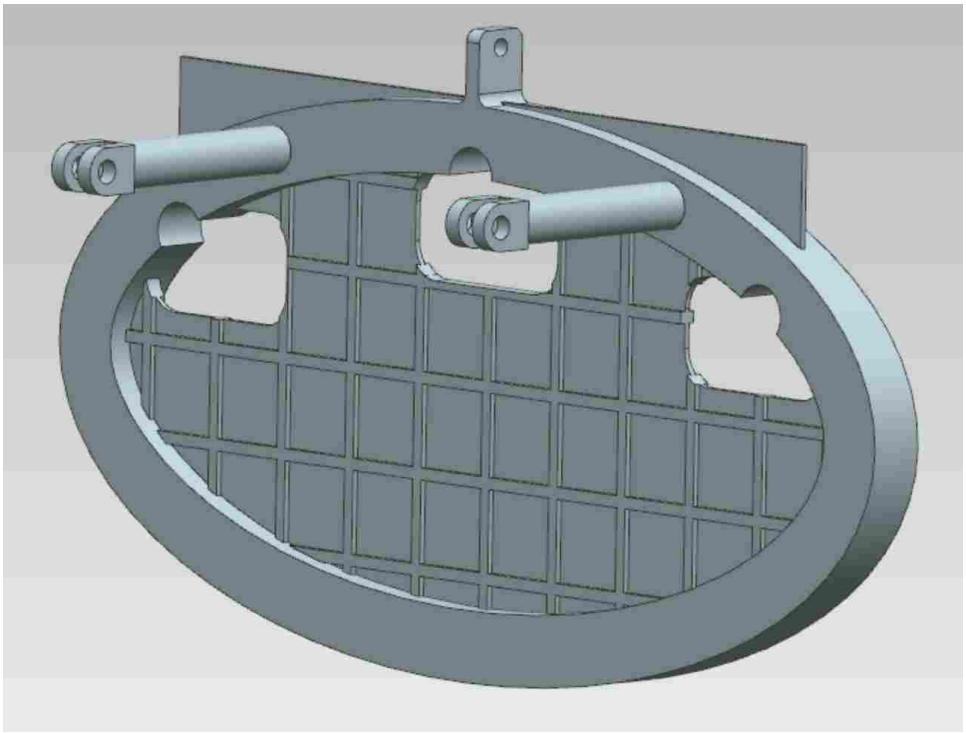


Figure A.1: A bulkhead for an unmanned aerial vehicle modeled in NXConnect

The example shown in A.2 is quite a bit more complex than the previous example. The model is an intermediate case for a jet engine which took four modelers approximately two hours to model simultaneously in NXConnect.

Figure A.2: An intermediate case for a jet engine modeled in NXConnect

The example shown in A.3 much more complex than the previous two examples. The model is an unmanned aerial vehicle which took five modelers approximately two weeks to model simultaneously in NXConnect. This complex example qualitatively shows that the conflict management, model consistency and persistent naming methods implemented provide sufficient robustness and usability for a complex model to be created in multi-user CAD.

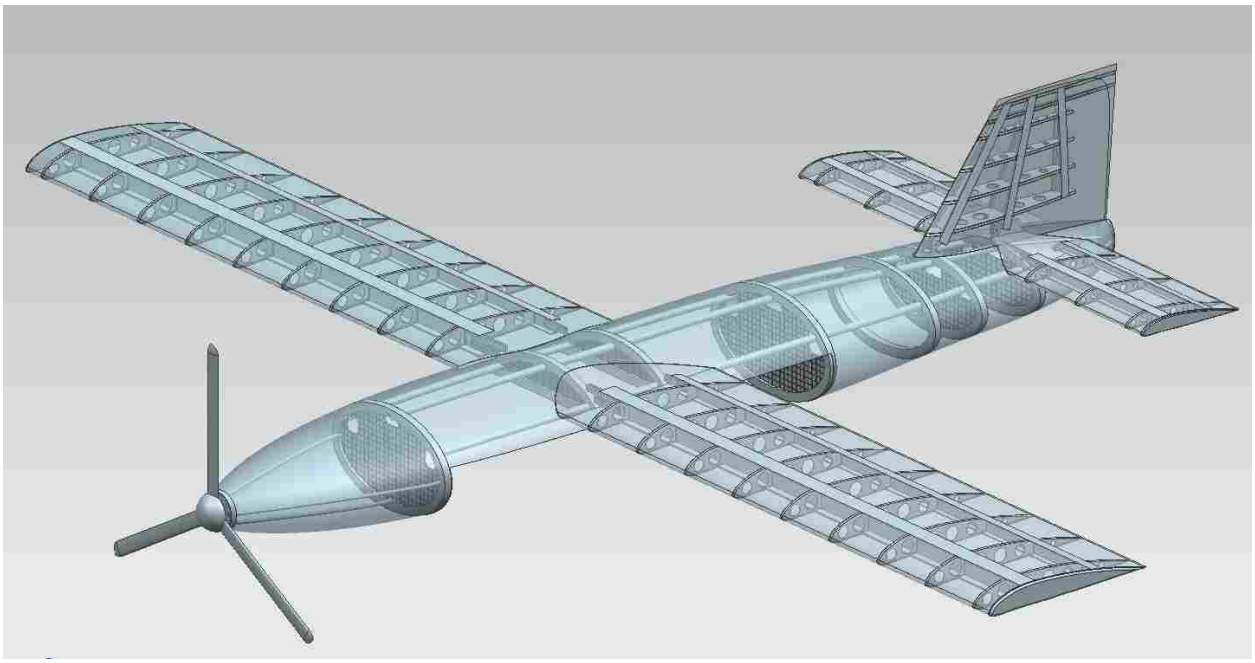


Figure A.3: An unmanned aerial vehicle modeled in NXConnect