2016-03-01

Associative CAD References in the Neutral Parametric Canonical Form

Daniel Robert Staves
Brigham Young University

Associative CAD References

in the Neutral Parametric Canonical Form

Daniel Robert Staves

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

Master of Science

Walter E. Red, Chair
Steven E. Gorrell
John L. Salmon

Department of Mechanical Engineering

Brigham Young University

March 2016

ABSTRACT

Associative CAD References
in the Neutral Parametric Canonical Form

Daniel Robert Staves
Department of Mechanical Engineering, BYU
Master of Science

Due to the multiplicity of computer-aided engineering applications present in industry today, interoperability between programs has become increasingly important. A survey conducted among top engineering companies found that 82% of respondents reported using 3 or more CAD formats during the design process. A 1999 study by the National Institute for Standards and Technology (NIST) estimated that inadequate interoperability between the OEM and its suppliers cost the US automotive industry over $1 billion per year, with the majority spent fixing data after translations. The Neutral Parametric Canonical Form (NPCF) prototype standard developed by the NSF Center for e-Design, BYU Site offers a solution to the translation problem by storing feature data in a CAD-neutral format to offer higher-fidelity parametric transfer between CAD systems. This research has focused on expanding the definitions of the NPCF to enforce data integrity and to support associativity between features to preserved design intent through the neutralization process. The NPCF data structure schema was defined to support associativity while maintaining data integrity. Neutral definitions of new features was added including multiple types of coordinate systems, planes and axes. Previously defined neutral features were expanded to support new functionality and the software architecture was redefined to support new CAD systems. Complex models have successfully been created and exchanged by multiple people in real-time to validated the approach of preserving associativity and support for a new CAD system, PTC Creo, was added.

Keywords: interoperability, heterogeneous CAD, CAD Features, neutral format, design history, multi-user CAD, collaboration, collaborative CAD, Neutral Parametric Canonical Form, client-server architecture, associativity, design intent

ACKNOWLEDGMENTS

I express thanks to my wife, Rachel Staves, for the motivation and support she gave throughout this long process. I also give thanks to my professors, Dr. Ed Red, Dr. Steve Gorrell, and Dr. John Salmon for their much-needed guidance to ensure a successful completion of my thesis. Finally, I give thanks to the entire interoperability team in the BYU CADLab, Eric Bowman, Scott Christensen, Devin Shumway, Josh Coburn, and Logan Hill for making this research possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

2D	Two-Dimensional
3D	Three-Dimensional
API	Application Programming Interface
BYU	Brigham Young University
C#	Object-Oriented Programming Language by Microsoft
CAD	Computer-Aided Design
CSYS	Coordinate System
GUID	Global Unique Identifier
IAB	Industrial Advisory Board
NSF	National Science Foundation
NIST	National Institute for Standards and Technology
IGES	Inital Graphics Exchange Specification
JT	Lightweight Neutral Data Format by Siemens PLM Software
STEP	Standard for the Exchange of Product model data
ORM	Object Relational Mapper
Serialize	Process of translating data structures to a format that can be transferred
Deserialize	Parse serialized data to reconstruct data structures

CHAPTER 1. INTRODUCTION

Iterative engineering processes have long been integral to engineering design. Before computer technology assisted with the design process, designers and engineers gathered around the drafting table to coordinate their work on products. Detailed drafting standards were employed to ensure accurate interpretation of drawings when transferred to other designers and manufacturers. Even with these standards, however, close personal contact between the designer and manufacturer was usually necessary [1]. With advances in CAD and communications via the Internet, computers are now a recognized necessity in the design process. While E-mail has allowed designers and manufacturers to be geographically separated, mutual and interactive communication of design information is no less vital now than in the past [2]. Part models and drawings must be frequently exchanged, translated, and checked between designers and manufacturers at each iteration of the design.

While many methods of the design process have remained the same during the transition from the drafting tables of the past to the modern computer aided engineering processes of today, the tools of engineering have changed dramatically. A wide variety of CAD tools are available which specialize in the many different aspects of design, like finite element analysis (FEA), computational fluid dynamics (CFD), and solid or surface modeling. Companies, engineers, and designers choose these tools based on their strengths, ease of use, and familiarity. A 2010 study performed by the Aberdeen Group surveyed best-in-class companies and found that 82% of respondents reported using 3 or more CAD formats in their design process. Included with the study was a list of reasons varying from keeping aligned with supply chain, supporting legacy data, to user preference [3].

Translation processes and standards have been developed for designers and engineers to work with the multiplicity of file types in order to carry out their work. The translation process, however, often strips the original model of its design intent, or intelligence embedded into the

model. Translating also slows the design process and hinders meaningful collaboration between designers working in separate systems.

1.1 The Cost of Poor Interoperability

Due to a shift in production practices in the 1980's, U.S. automakers increased their market share and became more competitive in the US automotive market [4]. A large part of this increase in production is due to a move toward concurrent engineering and design outsourcing. This shift resulted in the sharing of design data between a greater number of people and organizations, both within the company, and between the company and its suppliers. While overall productivity increased, the move highlighted many difficulties related to using heterogeneous CAD packages. Estimates found that imperfect interoperability, or model transfer between CAD packages, during this time cost between $1.02 billion and $1.05 billion within the US Automotive supply chain alone [4]. The vast majority of this cost is due to the time spent fixing data resulting from poor CAD model translations between both the OEM and suppliers. Interaction with Industry Advisory Board (IAB) members of the National Science Foundation (NSF) Center for e-Design has revealed a similar trend in the U.S. aerospace industry.

Difficulties in the translation process arise because of differences in the way each CAD system represents the part model. Because there is no standard, modern CAD packages store feature and design data in proprietary formats, even though each system represents the same type of data: three-dimensional geometric models. These systems often only support interoperability by translating geometric Boundary REPresentation (BREP) data through formats such as IGES and STEP. While these translation methods have been extensively and successfully used in industry, the variances in the translation process between applications can cause geometric errors amongst transferred models. In addition, by only translating BREP data, crucial design intent stored within feature data is lost which greatly hinders meaningful collaboration. Any modifications or adjustments to the model must either be made on the originating system and re-translated, or completely redesigned in a new system. On top of these limitations, this method inherently follows a serial, single-user work-flow; only allowing one user at a time to design or update the model. Before a design can be shared, it must be exported in a neutral format and sent to the end-user. This significantly impedes the advantages of concurrent engineering.

Design intent captures the purpose of the modeling order and geometry chosen by the designer. In modern parametric-based CAD, this is often stored within CAD features, such as axis systems, planes, sketches, extrudes, revolves, and sweeps. It is also stored in the associations relating the CAD features together. By selecting features during the modeling process, an experienced designer implicitly defines the important parameters of the model, which is extremely important to manufacturing and future updates to the design. Because these parallel design processes are often utilized to facilitate collaboration between geographically dispersed designers, these implicit definitions set by the designer are especially important to preserve. When a CAD model is translated into neutral formats such as IGES and STEP, this design intent is lost as all features are replaced by geometric representations. Without post-processing, engineers, designers, and manufacturers are unable to identify the important parameters set by the original designer.

The principles of iterative design call for a constant update process between various groups in the supply chain. Dealing with poor interoperability, however, significantly increases the number of feedback loops required to ensure high-fidelity CAD models as models must be fixed each time the design is passed between groups. These feedback loops extend the design time and can be extremely costly.

Figure 1.1 depicts a simplified design process to illustrate this challenge. Arrows in the figure represent the flow of the design as it is passed between designers and groups, both within a company and between company and suppliers. Part fidelity checks and conflict resolution steps are performed when models created by different designers are merged and before the design is passed to a different group. If the different group uses a different CAD system, the model must be translated and checked again. The red arrows represent the feedback loops through which the design must pass if any of these tests are failed. As the number of designers and CAD systems used increases, the time spent resolving conflicts in the feedback loops increases, which can significantly delay the product launch and increase costs.

1.2 Proposed Solution

By investigating the needs of its industrial members and implementing solutions, the BYU Site of the NSF Center for e-Design has sought to reduce the design cycle time by developing synchronous, multi-user collaborative design tools. The most notable development is the NX Con-

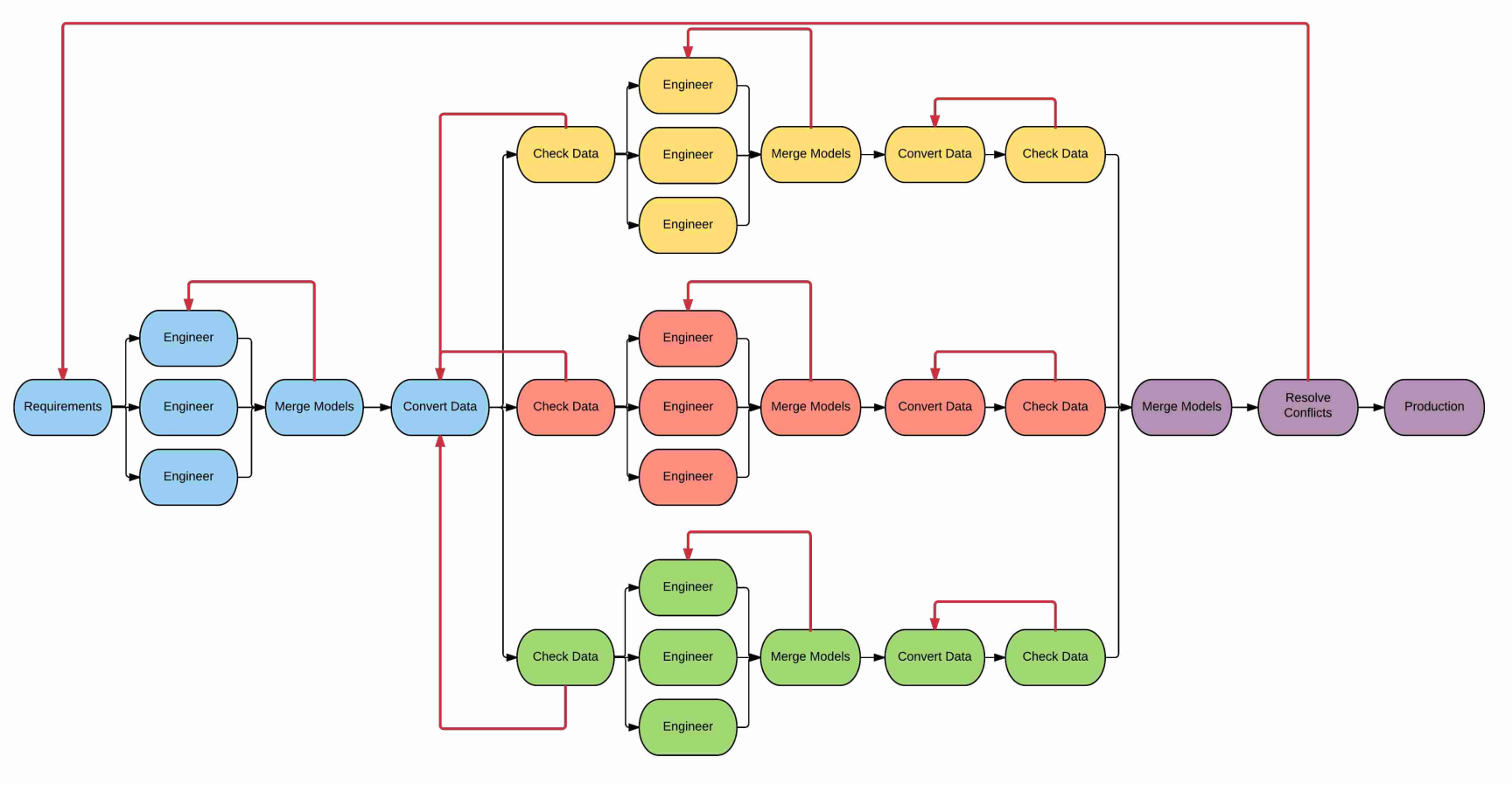


Figure 1.1: A typical serial design process requires feedback loops to correct poorly translated CAD data

nect multi-user prototype, a thin-server, thick-client system which supports cooperative modeling between geographically dispersed clients running Siemens NX. NXConnect aims to reduce the feedback loops in the design cycle by promoting awareness between different modelers working on the same model. This work has paved the way for recent work developing the Neutral Parametric Canonical Form (NPCF), which seeks to solve the interoperability problem by formulating neutral standards for representing CAD features [5]. While still supporting the multi-user objectives of the site, this neutral format enables simultaneous design among heterogeneous CAD systems. A complete neutral format will greatly aid in the translation of both geometry and design intent.

A product development process supported by an interoperable design environment, as illustrated in Figure 1.2, enables users working with a variety of CAD formats to collaborate. This figure depicts an engineering design process where designers, both within the same group and between groups, can simultaneously contribute to a model. Because data is neutralized and accessible in real time between all systems, model awareness between clients in promoted and conflicts are able to be caught sooner in the design process. This has the potential to eliminate the costly feedback loops illustrated in Figure 1.1.

Current work into the NPCF standards have focused on developing neutral formats for the basic geometry used in 2D sketches - point, line, arc, and spline. Standards for the extrude by limit feature and the revolve by limit feature have also been defined [5]. Neutralizing features is the first step in creating a neutral format which preserves design intent. These features have been used to successfully design a variety of part models and assemblies to study its effectiveness at promoting awareness amongst heterogeneous CAD users. The NPCF functionality is limited, however, and further research is needed to expand the NPCF standards to better support associativity. While designers express, to some extent, design intent though their choice of features and the parameters used to define them, the design intent is more often implicitly stored within the associative relationships between CAD features. This occurs whenever a designer chooses between different feature creation methods. Though they will both result in the same geometry, differences in the definitions will cause the geometry to update differently during an edit. The choice of feature used implicitly defines how the model should update during future edits.

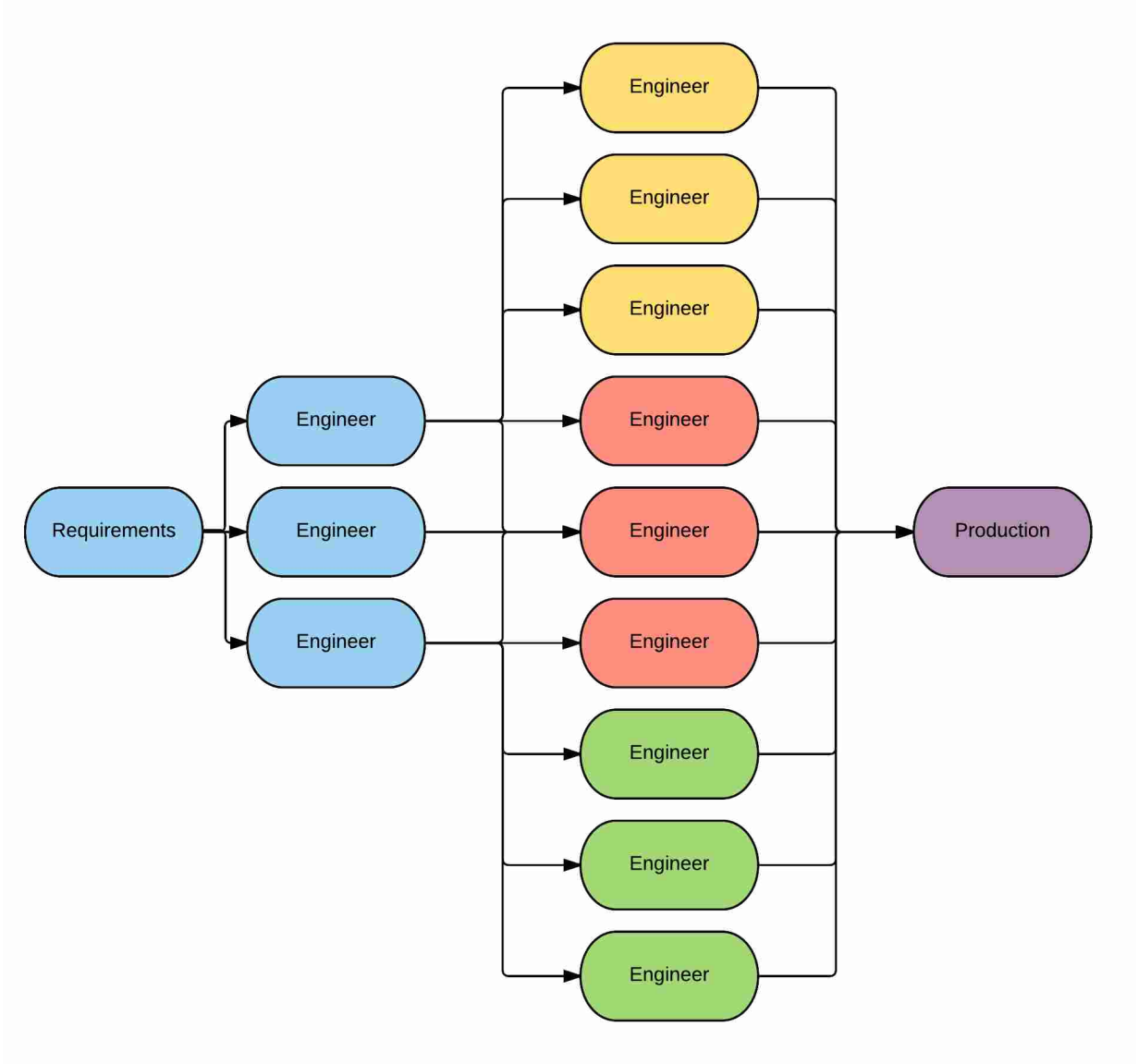


Figure 1.2: The ideal interoperable environment, when applied to a product development process, allows users in different groups and different systems to work in parallel.

Associativity in CAD systems is one of the most powerful ways for a designer to express design intent. By building a 2D sketch feature on a datum plane, the designer implicitly signifies to others that the sketch, and any child features of the sketch, are associated with the location of the plane. Any updates or edits to the part model must preserve this original association. Any neutral format used to translate CAD features from one CAD system to another must retain these associations to preserve design intent.

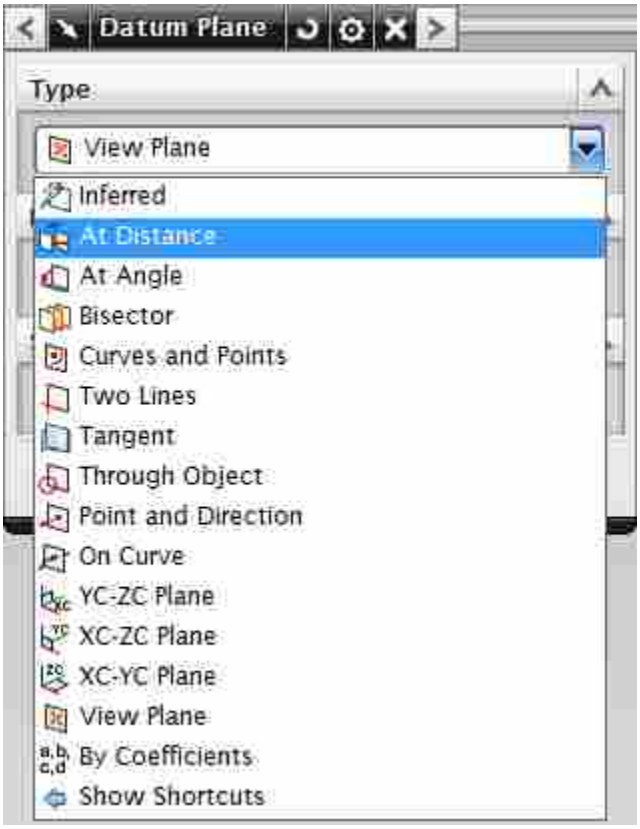


Figure 1.3: Siemens NX employs a variety of feature creation methods for the datum plane feature to enable designers to express critical design intent.

A designer's ability to express design intent can be seen, for example, in the case of a creating plane feature. Most CAD systems support a variety of methods that a designer can use when creating a plane. Figure 1.3 shows the different feature variations that Siemens NX supports when creating a datum plane. Other major CAD systems support a similar list of feature creation options. A designer may signify design intent by creating a plane that runs parallel and equidistant from two CAD surfaces, often called a bisect plane. When any of the surfaces are moved, this plane is moved automatically to support the constraint. This plane could be translated to other CAD systems as a fixed 3D plane in space. The translation, in this case, preserved the plane feature, its location, and modeling history of the part. It did not, however, transfer design intent as when the surface are moved, the plane would no longer bisect the two faces.

1.3 Objectives

The main objective of this research is to enhance the neutral data structure of the NPCF with methods to better support feature associativity within an interoperable CAD modeling session. A secondary objective was to research and develop a system to ensure the data integrity of the neutral format. Both of these objectives drove the selection and order of development for the processes found in Table 1.1.

Table 1.1: The main objectives of this research drove the development of key processes

	Key Processes
1	Support multiple definitions of features
2	Enable Interface-like CAD references
3	Structure Client software to support new systems
4	Support PTC Creo CAD System
5	Support Coordinate System Features
6	Support Plane Features
7	Support Axis Features
8	Extend Support for Extrude Features
9	Support Revolve Features

As a part of this research, the structure of the neutral format was redesigned to enable multiple definitions of features to be utilized during the modeling process. This, for example, enables a designer to either generate a blind extrude to a limit, or extrude up to a plane. The NPCF neutral definition for features have also been updated to support inter-feature references, allowing features to reference other features, or other types of features. This redesign was critical in supporting associativity between neutral CAD features as well as preserving the referential integrity of the neutral data.

Additionally, as part of this research, the number of supported CAD systems was increased from Dassault CATIA and Siemens NX to include PTC Creo at the request of research sponsors. This enabled testing of the NPCF with all three top-tier CAD systems widely used in industry. During the implementation process of all previously implemented NPCF features for Creo, the format was again tested to ensure the neutralization was valid for all systems.

Finally, the new features chosen to be supported serve as a key test to ensure the objectives of this research have been met. Three of the new features are datum features which are extensively used to define the position and orientation of geometry and are key to a parametric CAD model. Multiple feature definitions will be implemented for the 3D features of extrude and revolve.

CHAPTER 2. RELATED WORK

2.1 Multi-User CAD Solutions

Multi-user CAD implementations seek to reduce time spent during the modeling process by enabling collaboration amongst designers and engineers at separate workstations. Users are able to simultaneously interact with and edit the same part model or assembly in an attempt to reduce design time as modeling processes can be performed as parallel tasks. An important principle of a multi-user CAD solution is an awareness of where other users are working.

2.1.1 Thick Server - Thin Client

Research in collaborative CAD solutions have been ongoing since the mid-90s and has focused primarily on two architectures: centralized and replicated. A centralized architecture, where a central server executes and performs modeling operations received from remote clients, enables consistency among clients as there is only one copy of the model. After the server has generated the geometry described by the modeling operation, only visualization data is transmitted to the clients. With only one copy of the model, each client remains at all times in the same state as the others. WebSPIFF [6], NetFeature [7], CADDAC [8], CollFeature [9] and WebCOSMOS [10] [11] are all examples of this central architecture. A downside to this architectural approach is that a large amount of data needs to be sent over the network [12].

In addition, this approach isn't easy to implement with existing commercial CAD systems because most commercial CAD systems API's are single threaded [13]. CAD data is most often divided into parts and assemblies. A true multi-user approach must allow clients to perform CAD operations in any model, or multiple models, simultaneously. A single-threaded API significantly limits the ability to interact with multiple models as operations must be performed serially. For this reason, this architecture isn't ideal for a heterogeneous CAD approach.

2.1.2 Thin Server - Thick Client

In a replicated collaborative CAD system, each client has a copy of the model data which is required to stay in sync. As each client performs an operation, the data is sent to other clients via a network architecture. ARCADE [14], CSCW-FeatureM [15], CollIDE [16] and RCCS [12] are examples of replicated CAD systems built at the kernel level. Because these systems have been designed with multi-user processes in mind, they are very capable of handling the parallel nature of collaborative CAD. Because they have been developed from the kernel level, however, they aren't compatible with the commercial CAD systems currently used today and are unable to improve the state of CAD interoperability.

There are examples of systems that interface with commercial single user systems. These include TOBACO [17], CallabCAD [18], and COCAD [19]. In addition, multi-user systems developed at the NSF Center for e-Design, BYU site are built as direct plugins to major commercial single user systems including Siemens NX, Dassault Systemes CATIA and Autodesk Inventor. These plugins are respectively named NXConnect, CATIAConnect and InventorConnect [20] [21] [13]. The replicated approach is easier to implement with existing commercial CAD systems because each command can be applied the the model serially, however data consistency can be a major challenge. While these systems enable collaboration between commercial CAD systems, they don't support translations between heterogeneous systems.

2.2 Heterogeneous CAD Solutions

Due to the multiplicity of CAD applications available commercially, companies often interface with a wide variety of CAD file formats through interactions with their supply chain. To open and use these formats, CAD applications must translate part model and assembly files between heterogeneous CAD systems. Translation inconsistencies arise because different CAD systems generate different features, or may represent similar features differently. For example, a line segment can be represented either as two endpoints, or as one endpoint, a direction vector, and a line length. The data formats are not always minimized, for reasons of feature associativity. Consider the example of a 2D Arc. This feature in Siemens NX is defined using a center point, a radius, and a start and end angle, and is represented in 3D coordinates. An arc is expressed in PTC Creo

using the same parameters, but expressed in 2D coordinates. The same feature in Dassault CATIA is defined using a center, start and end point in 2D coordinates. Due to the different definitions, translating data between programs is non-trivial. There are a variety of methods which have been developed attempting to solve this problem, though further work is needed to address limitations.

The International Graphics Exchange Standard (IGES) format was first developed in 1980 to facilitate the translation between heterogeneous CAD systems [22]. IGES was the first attempt at resolving the data exchange challenge between CAD systems. It works by translating the CAD model of each system to its basic geometric data and is the most widely used neutral format used today. While IGES has been very successful in allowing models developed in different systems to be exchanged, it falls short in that only BREP data is translated. All associative links between features are broken, and design intent is lost.

Created in 1984, the Standard for the Exchange of Product Model Data (STEP) is a redesign of IGES, aiming at a more advanced, database-oriented, and integrated solution based on product lifestyle data [23]. It fixes some shortcomings of IGES, including standardizing the processors and uses a formal language to define the data structure to avoid ambiguities during interpretation, which could result in as much as 50% failure rates [24]. Work has continued on the STEP standard, most recently by the Solid Model Construction History (SMCH) group which is seeking to preserve design intent through the translation process by recording the history of the model as it is created. SMCH adds an implicit, or history based, representation of the model to the explicit BREP data. This effectively creates solids that maintain their original relationships with other solids after the translation process, preserving design intent and allowing them to be edited and manipulated accordingly [25]. A drawback of SMCH is that the file size is large due to both the B-rep and construction history data being stored, making it difficult for collaborative, multi-user applications in which data is transmitted frequently between clients.

Many commercial CAD companies have developed their own format for compressing and representing models. JT by UGS, U3D by Intel and HP, 3DXML by Dassault are examples of these open formats, which are the preferred open formats for exporting from each respective system [26]. These formats can be licensed to other systems for import. JT, for example, differs from IGES and STEP standards by storing multiple Levels Of Detail (LOD) of tessellated geometry, as well as product structure information of parts and assemblies like texture, color, and surface quality [27].

Like STEP and IGES, however, these formats don't preserve feature information, and therefore lose design intent.

A 2010 concept to integrate feature information into the JT open data format has been put forth by Eigner et al in an effort to enhance the product development process through the semantic interpretation of the features [27]. The features of the model are classified by type and procedural description regarding its usage scenarios. This feature meta-information is collected and stored in an XML format alongside the JT model representation and PLM data while maintaining a neutral format and compact file size. While this method does transfer feature information which helps support many more operations in the development cycle that non feature-based formats, it does not transfer parametric data, limiting applications to post-modeling processes.

The Macro-Parametric Approach, described by Choi et al [28] in 2002, preserves design intent through CAD translation by utilizing each CAD system's built-in macro functionality. During the modeling process, the macro file records and saves each modeling command. During translation, this file passes through a pre-processor, which outputs a standard, or neutral, macro file. In the event where a CAD system doesn't record enough data in the macro file, the pre-processor uses the program's API to extract the required information. This standard macro file created by the pre-processor is then translated into the target system's native format by the post-processor which the CAD system is able to interpret to reconstruct the model. This approach was continued and extended by Mun, et al by defining a set of standard modeling commands [29]. Using this approach, the team successfully translated simple parts from SolidWorks to CATIA while keeping the feature trees intact. This method is not able to be used for multi-user CAD, however, because of the need of an external program to perform the pre and post processing to translate the macro files to and from the standard format. It is also limited because not every CAD system records macro files which are accessible to the user.

The Neutral Modeling Command method addresses the need for a collaborative heterogeneous CAD solution through the use of each CAD system's application programming interface (API) [30]. Each client runs an add-on program which translates system modeling operations (SMO) into neutral modeling commands (NMC) in real time and synchronized between clients through a thin server [31]. The database of supported modeling commands is a super-set of all modeling commands in each supported CAD system. While this method supports real-time edit-

ing of models by geographically dispersed users, it does so on a turn-by-turn basis which significantly limits collaboration. Like the Macro-Parametric approach, this method focuses on modeling commands, rather than feature data which prevents it from defining a neutral format.

While the approaches described previously to enhance interoperability between CAD packages have greatly improved the data transferred during the heterogeneous translation, they do not address the problem of collaboration between these systems. Prior work performed developing the Neutral Parametric Canonical Form sought to address this limitation by enabling synchronous, multi-user modeling between heterogeneous CAD systems by neutralizing feature data in real time. This earlier research focused on developing processes to neutralize feature data. Further work is required to ensure the neutralized feature data retains its associations with other features and CAD parameters, thus preserving design intent between CAD systems.

CHAPTER 3. METHODS FOR INTEROPERABILITY

The methods described in this chapter facilitate heterogeneous, multi-user CAD and are used to improve associativity methods and capture design intent in the Neutral Parametric Canonical Form (NPCF) data structures. The interoperability of the NPCF is tested using the CAD Interop prototype developed by the BYU Site of the NSF Center for e-Design, which is built on the foundation of a client-server architecture. Plug-in software packages, written for each supported CAD system, capture CAD events and translate resulting CAD features into the neutral format defined by the NPCF. This neutral data is transmitted to other clients via the server. The server is also responsible for saving a persistent copy to the database to facilitate other clients loading after geometry has been created.

These methods enable modern, commercially available CAD packages to work in a heterogeneous environment. The reasons for supporting existing CAD packages become clear when considering the various reasons companies select a CAD system. Companies often choose a CAD system for its strengths in certain modeling areas, its support for the company's legacy data, or its familiarity and ease of use with designers. While a completely new CAD system could be developed to meet the simultaneous modeling needs of a company, it would not address the concerns which drive a company to choose a specific CAD system. By developing methods to synchronize models within different CAD packages, companies can retain a certain program used for its advantages toward their application, while being able to collaborate with other companies or groups who have chosen other programs.

Improvements enhancing the associativity between neutral CAD features has been chosen as the first upgrade to the original standards defined by the NPCF due to feedback given by industry sponsors of the Center for e-Design. Associativity in a CAD system is the foundation of design intent, through which a designer can parametrically define features that, when edited, continue to meet the design requirements. This is especially important for multi-user CAD, where designers

may be geographically dispersed but still need to communicate design intent to facilitate meaningful collaboration. This research focused on developing and designing the method and processes for storing neutral feature data within a database to preserve CAD associations. In addition to developing these methods, the NPCF neutral database schema and client software were updated to utilize the new methods.

3.1 Determining a Feature Set

Table 3.1: Most used CAD features in BYU Senior Capstone project using NXConnect

Feature	Count
Spline	3011
Datum Plane	1842
Mirror	447
Datum Csys	364
Offset Curve	326
Thru Curve	295
Point	255
Sketch	229
Extrude	220
Thru Curve Mesh	194
Sew	193
Trim Body	187
Section Curves	141
Skin	125
Blend	110
Absolute Datum Plane	74
Subtract	39

The order of feature additions to the Neutral Parametric Canonical Form are derived from data collected from a Boeing sponsored, BYU senior capstone project listing the most used CAD features in the project. A list of these features and their use count can be seen in Table 3.1. The original set of supported features were the 2D sketch features of point, line, arc, circle, and spline. Extrudes and Revolves were also supported with limited functionality. As part of this research, these features were implemented into a plug-in application for PTC Creo following the standards

set by the NPCF. The neutral definitions 3D features of sketch, extrude, and revolve were also expanded to include functionality for associative references.

In addition to upgrading the initial features defined by the NPCF, neutral definitions of coordinate system features, plane features, and axis features were developed. These datum features are particularly important to associative modeling processes because they are used to define the locations of curves, faces, and solid bodies in part models or assemblies. They also are included in the list of most common features used by the BYU senior capstone team.

3.2 Defining a Neutral Format

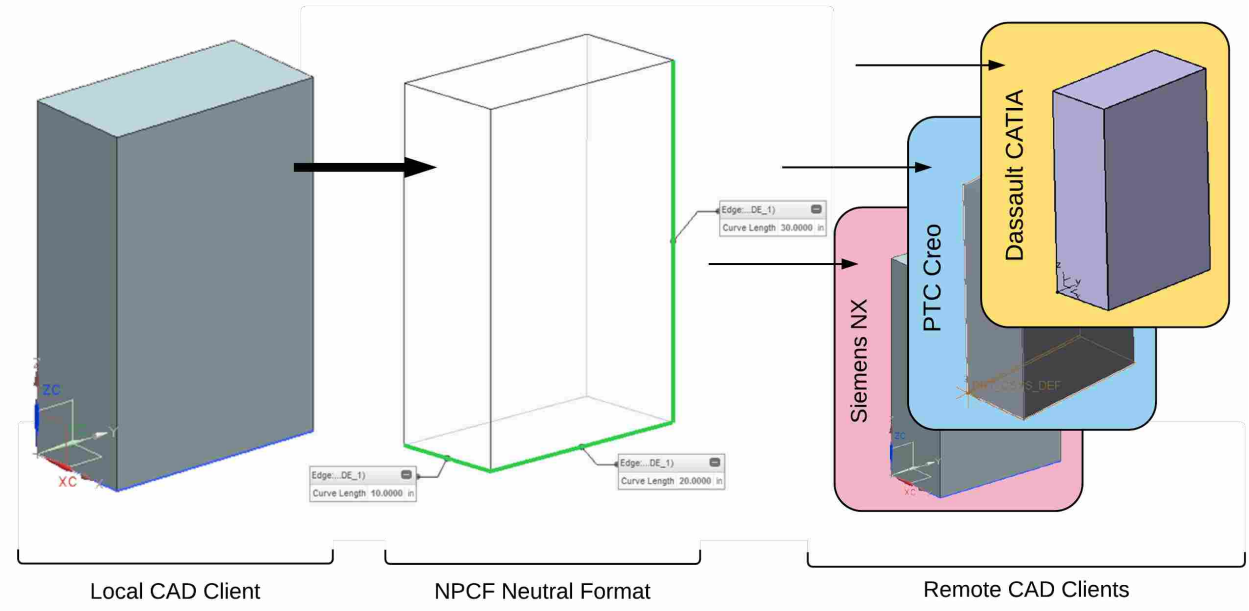


Figure 3.1: Local CAD features and operations are converted to a parametric, neutral format which is accessed by remote clients and incorporated into their respective models

An important distinction between the way the Neutral Parametric Canonical Form enables interoperability between heterogeneous CAD systems, and methods employed by previous CAD interoperability formats are the principles of neutralization versus translation. An interoperability process that uses a neutralization method is illustrated in Figure 3.1 showing how a CAD feature can be represented by it's neutral parameters. Software utilizing this method will convert CAD

data from an originating system to a neutral format after a CAD process is performed. The neutral format, illustrated in the center of the figure, completely defines the feature by its parameters and stored within a database. It is then converted to the CAD format of the destination system and constructed in a local part model for interaction by the other user. The neutral file remains as the master copy of the model from which all systems load data. This single neutral file ensures model consistency between clients and sessions.

A major advantage of this method is its support for multi-user processes. This is because all data is stored in an open format easily accessible and readable by any interested party with the proper access credentials. When support for new CAD systems is to be integrated into the neutralization method, software is required only to convert the CAD specific data to the neutral format, and from the neutral format to the CAD specific format, without any need for consideration of the other supported CAD systems. Access to only one CAD system's API is required to perform these conversions, eliminating the need for a company to hold costly CAD licenses for every format their suppliers may use.

This is in contrast to the translation method. When CAD data is translated from one system to another, methods convert data from the original format directly to the destination format. In this method, the original CAD file is the master copy of the model and must be converted to every other system type. While this method reduces the number of steps required to support interoperability, it doesn't readily support multi-user processes as all CAD features must be translated each time a new format is used. This conversion process also requires access to all involved CAD system's API's which could be very expensive.

An important part generalizing the Neutral Parametric Canonical Form was the addition of a third tier-one CAD system, PTC Creo. When a new feature was added to the NPCF, the creation methods for that feature were compared between Siemens NX, Dassault CATIA, and PTC Creo to determine which methods were functionally identical and would therefore map with minimal conversion. This process is vital in determining the neutral format because it ensures that the feature can be neutralized from any of its creation methods without being converted and potentially losing valuable design intent information.

With a set of features and associated creation methods identified, the parameters required for each creation method were compared between CAD systems to determine the minimum ideal

set which fully defined the feature. With the multitude of CAD systems used today, there are many ways to represent the same feature. In this architecture, each CAD system's feature representation must be neutralized to a single format. This architectural choice ensures that single neutral feature is represented by one neutral data structure. Mun, et al utilized a method for accomplishing a similar goal when formulating a set of standard modeling commands between CAD systems [29]. In this process, feature parameters were determined by journaling a modeling command, as was the case with the previous NPCF paper [5]. Then, because Creo lacks journaling functionality, its API was compared with the results of each journal file to determine crucial parameters. An ideal neutral format is determined by selecting the minimum number of parameters required to completely represent the feature.

3.3 Client-Server Architecture

The multi-user aspects of the Neutral Parametric Canonical Form interoperability prototype require constant communication between clients to ensure model accuracy and consistency. As shown in Figure 3.2, this communication is facilitated via a client-server architecture in which each CAD client routes all messages about CAD operations through a central server. The server is responsible for maintaining a list of active clients and the models that each client has open. As messages are sent to the server, they are forwarded to all other clients associated with the corresponding model. The messages are also saved in a SQL database ordered by time-stamp for later access. When a client first opens a model, all modeling operations associated with the model are sent to the client to ensure the client's local model is up-to-date before new messages are received.

The messages sent by the clients to the server contain the neutral data about CAD features and operations. Each message contains information about the client that sent it, the corresponding part model it is associated with, and a neutral class object containing the feature data. The server is able to save this object to the database using an Object Relational Modeler which automatically converts the object to SQL code. The relationship between the server, ORM, and database can been seen in Figure 3.3, where the multi-user, heterogeneous prototype is divided into 5 major layers. The client layer, at the bottom of the figure, consists of the user and the CAD package. The business layer functionality is written into software plug-in for the CAD package to neutralize and

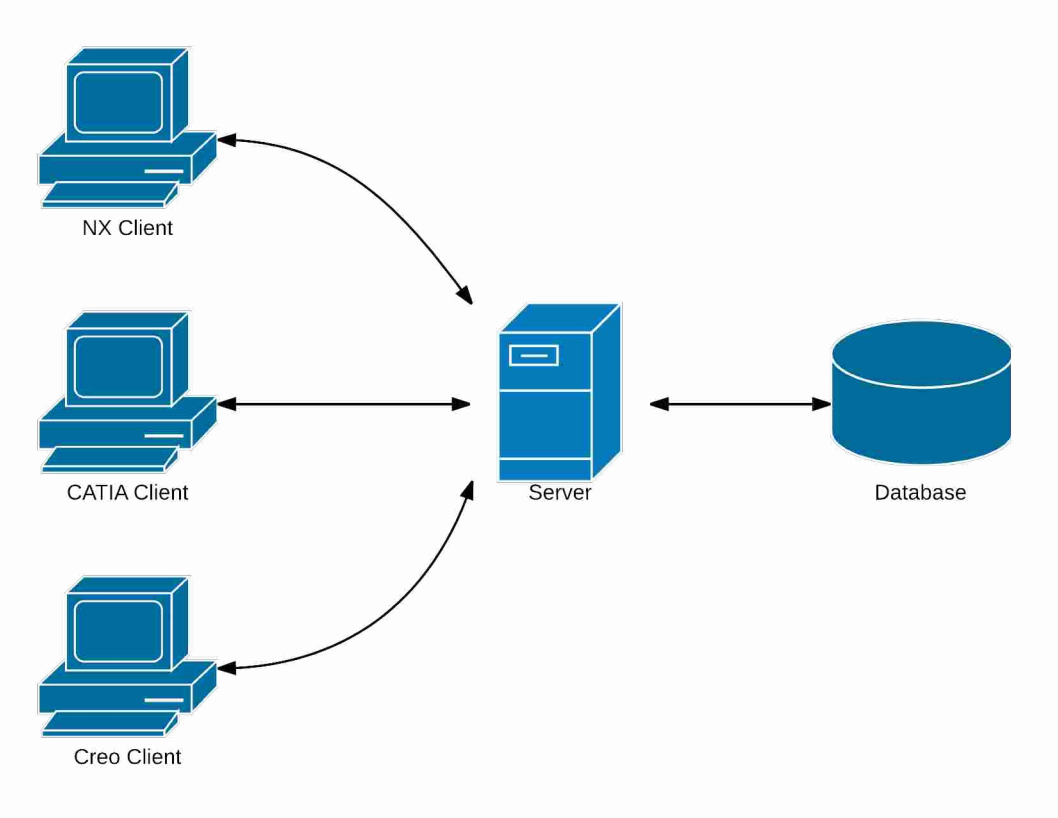


Figure 3.2: Client-Server Architecture employed to facilitate the multi-user capabilities of the NPCF

transfer CAD operations from the client to the server for distribution to other clients. The ORM acts as the communication layer between the server and the database. Finally the database, or the data storage layer, acts as the persistent storage for all CAD operations which enables modeling between sessions.

This client-server architecture helps maintain model consistency by keeping track of operational order between all clients in all models. This consistency ensures that as the number of users working in a model increases, models are synchronized between users. This is especially important during assembly operations which may affect multiple part models. Timestamps on each CAD operation sent to the server are used to set operational priority when being performed on remote clients. Conflicts are resolved by ignoring lower priority operations that result in failed geometry. Server consistency management, as described by Hepworth et al, can be implemented to further prevent conflicts [32] [33].

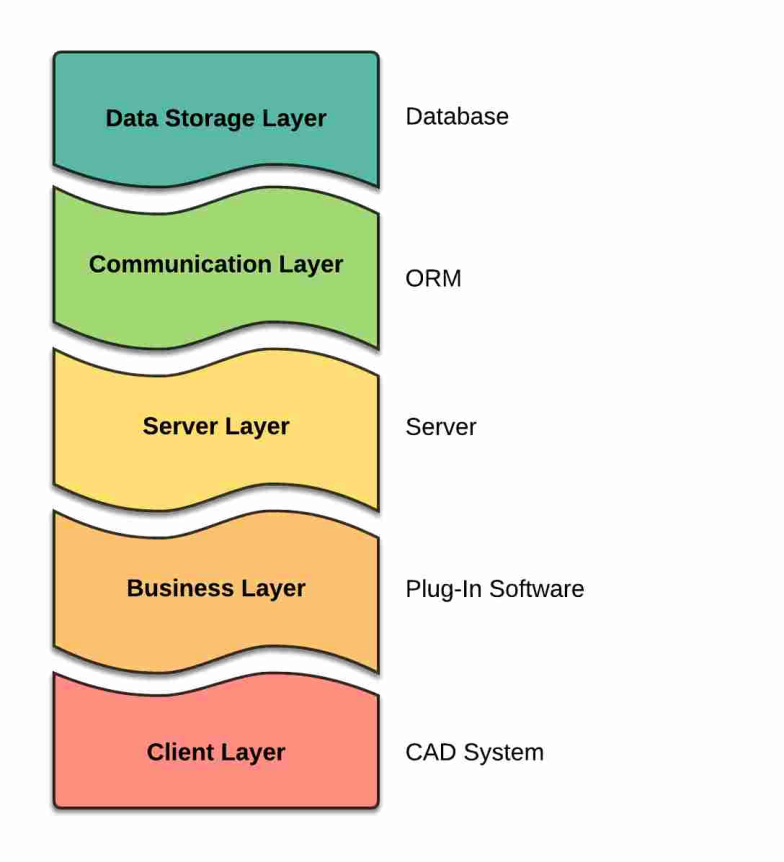


Figure 3.3: The client-server architecture of the Heterogeneous CAD prototype is part of a 5-tier structure in which data is passed from the user at the client level up to the server and database for storage and distribution

3.4 Database

During the multi-user, collaborative modeling process, CAD operation data is passed between clients through a central server which tracks operation order. As CAD operations are received by clients, the geometric calculations defined by the operations are performed in order, and the geometry is incorporated into the local model. Part models are created and remain synchronized because these remote operations happen concurrently with the local operations. When a client first enters an previously-existing model, however, operation data performed before entering the model must be applied before any local modeling operations are performed.

A database is used to record CAD operations as they are performed by clients and routed through the server. The database acts as a persistent storage for the assembly and part models because it stores all operations used to create the model. During the initial load of a model by a

client, the server loads all stored operations pertaining to the model from the database and sends them to the client to be performed locally. The database is only accessed during this initial loading event. All other remote operations are directly forwarded by the server from remote clients to increase the transfer rate of modeling messages.

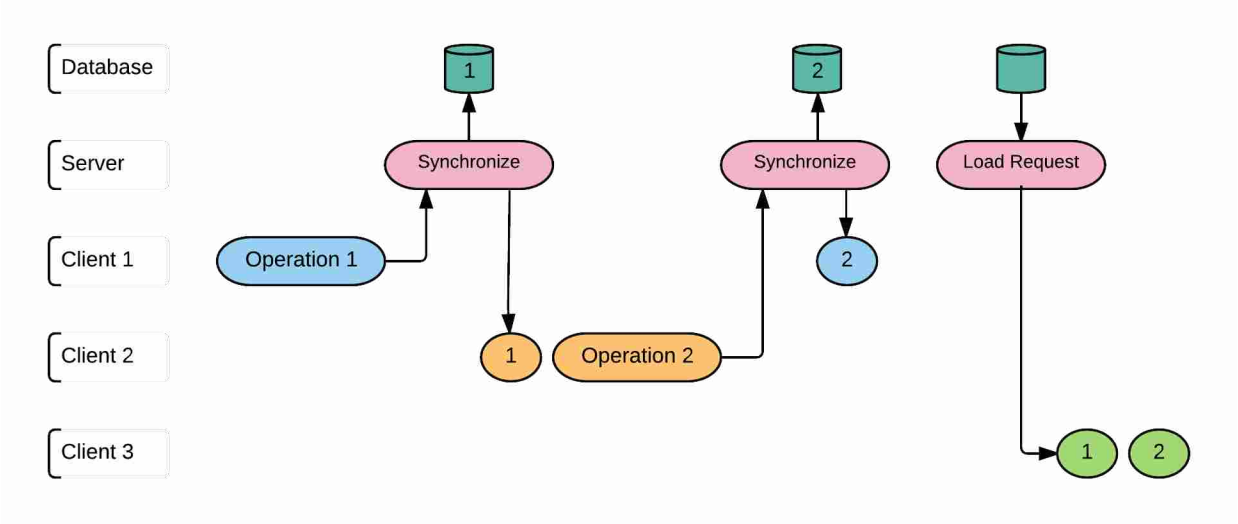


Figure 3.4: Modeling operations received by the server for distribution are recorded in the database for persistent storage. During part model loading, the server accesses modeling operation data from the database for part model synchronization of new client

An example of how database transactions are used during the multi-user modeling process is illustrated in Figure 3.4. In this example, clients 1 and 2 are working collaboratively in a part model. As client 1 performs operation 1, operation data is sent to the server to be forwarded to client 2. While the server is forwarding the operation data, it also writes that data to the database for persistent storage. This work flow enables quick response times between operations because the model synchronization doesn't need to wait on the database save operation. This process is repeated when client 2 performs operation 2. When client 3 joins the part modeling session some time later, before they are able to perform any modeling commands, the server queries the database for all modeling operations associated with the part model and sends them to be applied to the local model. In this manner clients models are synchronized while keeping database queries at a minimum.

Figure 3.5: Inheritance structure for an extrude feature in Siemens NX 8.0 API

For this heterogeneous CAD, multi-user application, the database is designed to store CAD data at its most basic level. In addition, the database must enforce referential integrity to reject invalid data. Modern CAD utilizes feature operations to build parametric models. Features in a CAD system are defined using various parameters, and depending on the function of the feature, may accept different types of parameters. In the CAD system's API, these features, and variations of features, are arranged in a hierarchical order. Figure 3.5 shows an example of the hierarchical structure for the NX 8.0 API, where an extrude feature is a child object of other CAD objects. Features that share similar characteristics and parameters are grouped together. To facilitate this kind of data storage while enforcing data integrity in the database, parameters relating to each CAD feature are distributed between database tables of varying generality. These tables are shared by other features which use the same parameters. For example, both a 2D sketch feature and a plane feature have a feature name and time stamp.

Figure 3.6 depicts the hierarchy of sketch feature and plane feature variations. All first-order features reference the top CAD feature table, which contains basic parameters such as feature name, owning CAD part, and time stamp. First-order features are features which are not variations of another object. This includes sketch features and plane features. Each of these first-order feature tables contains unique parameters used to define that feature. The feature variations of the plane feature reference the first-order plane feature to gain access to the normal vector parameter. The fixed plane feature, for example, references the plane feature table and adds a 3D point parameter for defining location in 3D space, while the offset plane and angled plane features add reference parameters and offset values. Using this hierarchy, a feature or feature variation at the bottom of

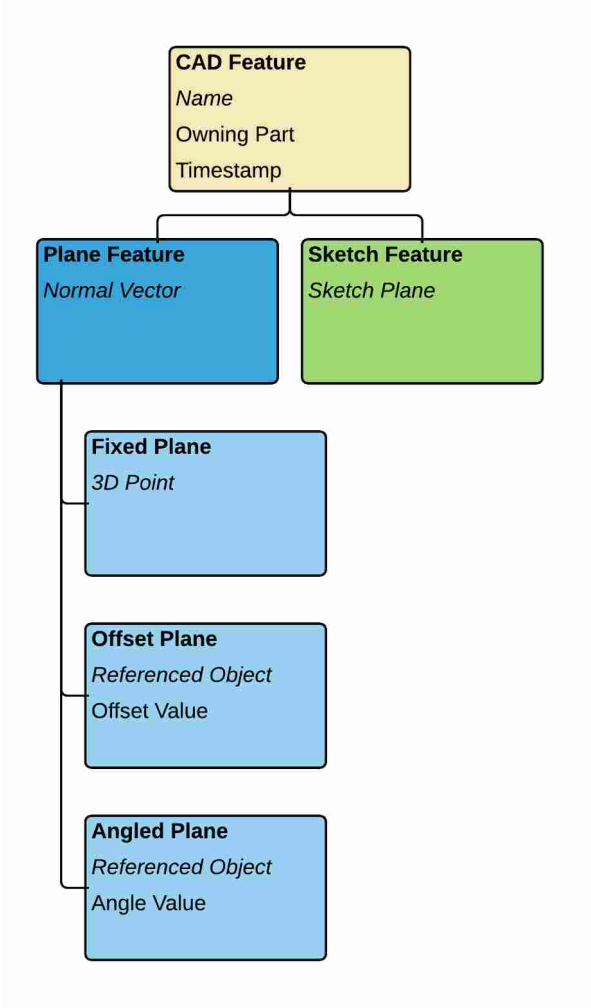


Figure 3.6: Parameters of feature variaitons are divided amongst a hierarchy of tables which groups variations by similarities

the hierarchy is fully defined when it combines the parameter values contained in its table with all those it references. The offset plane table when combined with the parameters of the plane feature table and the CAD feature table contains all parameters required to define an offset plane feature.

Table 3.2 describes the process used to incorporate new features into the neutral database. When a new feature is to be added, feature creation methods are identified and compared between CAD systems to identify commonalities. For example, both a blind extrude in Creo, and an extrude-by-values in NX expect a sketch feature on which the operation is to be performed and numerical limits to identify the start and end points of the extrude. Though these feature variations are given different names, and treat the limits differently, both can be neutralized into the

same format without loss of data or semantic meaning. Contrasting this feature variation with an extrude-to-plane operation, both methods utilize the same sketch parameter, but accept different objects to define limits. When splitting parameters between common tables, as described in Table 3.2, a generic extrude table will be created which stores the referenced sketch parameter, and two feature variation tables will be created to hold the parameters required by the blind extrude and planar extrude respectively.

Table 3.2: Process used to distribute feature parameters between tables when adding new features to database

NPCF Database Feature Addition Process
Identify feature variations on all systems
Identify commonalities between systems
Determine neutral format for all feature variations
Split parameters between common tables
Add associations representing inheritance relationships
Save database

When arranged in this manner, each variation of a feature will share parameters with other features but will have a table containing the parameters which make it unique. As shown in Figure 3.6, a plane feature can be divided into feature variations, with both an offset plane feature and a fixed plane feature sharing parameters defining normal direction. Both features also have parameters which make them unique. An offset plane feature, for example, has an offset distance parameter and an offset object, while a fixed plane has a 3D point defining its location. By arranging the database tables in this manner, data integrity is enforced because only the parameters used to define a feature variation are stored in a table. This allows the database designer to enforce all parameters as required with a defined data type. Invalid database save operations will automatically be rejected.

Besides enforcing data integrity, the arrangement of database tables into a CAD feature hierarchy also enables database parameters to store a variety of feature references. Designers use this type of reference to signify important relationships in a CAD model, implicitly defining design intent. A single parameter used to define a feature can be used to reference a wide variation of features. For example, A 2D sketch feature requires a Datum Plane as a parameter to define its

location in space. Using this hierarchical method, the parameter in the sketch database table can reference the more general plane table, which contains parameters used by both the offset plane and fixed plane tables. This effectively allows a sketch to reference any of these features.

3.5 Object Relational Manager

To facilitate reading from and writing to the database, an Object Relational Manager (ORM) is used to convert data between programming objects and the database. It is most often used as a familiar method in which software developers can interact with a SQL database. The ORM generates programming class code based on the database schema, effectively defining a neutral data structure in which CAD feature parameters are stored. When the proper methods are called, the ORM saves the parameters stored in the neutral data structure to their corresponding tables and columns in the database. These data structures defined by the ORM make up the foundation of the messages sent between the clients and server.

Because the ORM data structure is based on the database schema, it automatically captures all parameters and associations defined in the database tables. Since many of the associations in the database are used to define the hierarchical nature of CAD features and objects, these associations are modified within the ORM to represent inheritance relationships for the neutral data structure. Inheritance is a principle often employed in object oriented programing to define instances where one object derives methods and properties from another object. In the API of a typical CAD system, CAD feature objects derive certain methods and properties from a base Feature class. In our dispersed representation of CAD features in the database, where feature parameters for a single feature may be distributed throughout multiple tables, a feature variation inherits from a base class of that feature. By modifying the associations of the database tables imported by the ORM to represent inheritance, single programming objects are created which contain all the parameters of the features from which it is derived.

The ORM, as shown in Figure 3.7, converts the features and feature variations distributed through the various database tables into single CAD objects which contain all the parameters required to define each feature or feature variation in a neutral format. The database tables for sketch feature and for the plane feature variations are combined into their own data structures. These structures retain their relationships with the corresponding database parameters from which they

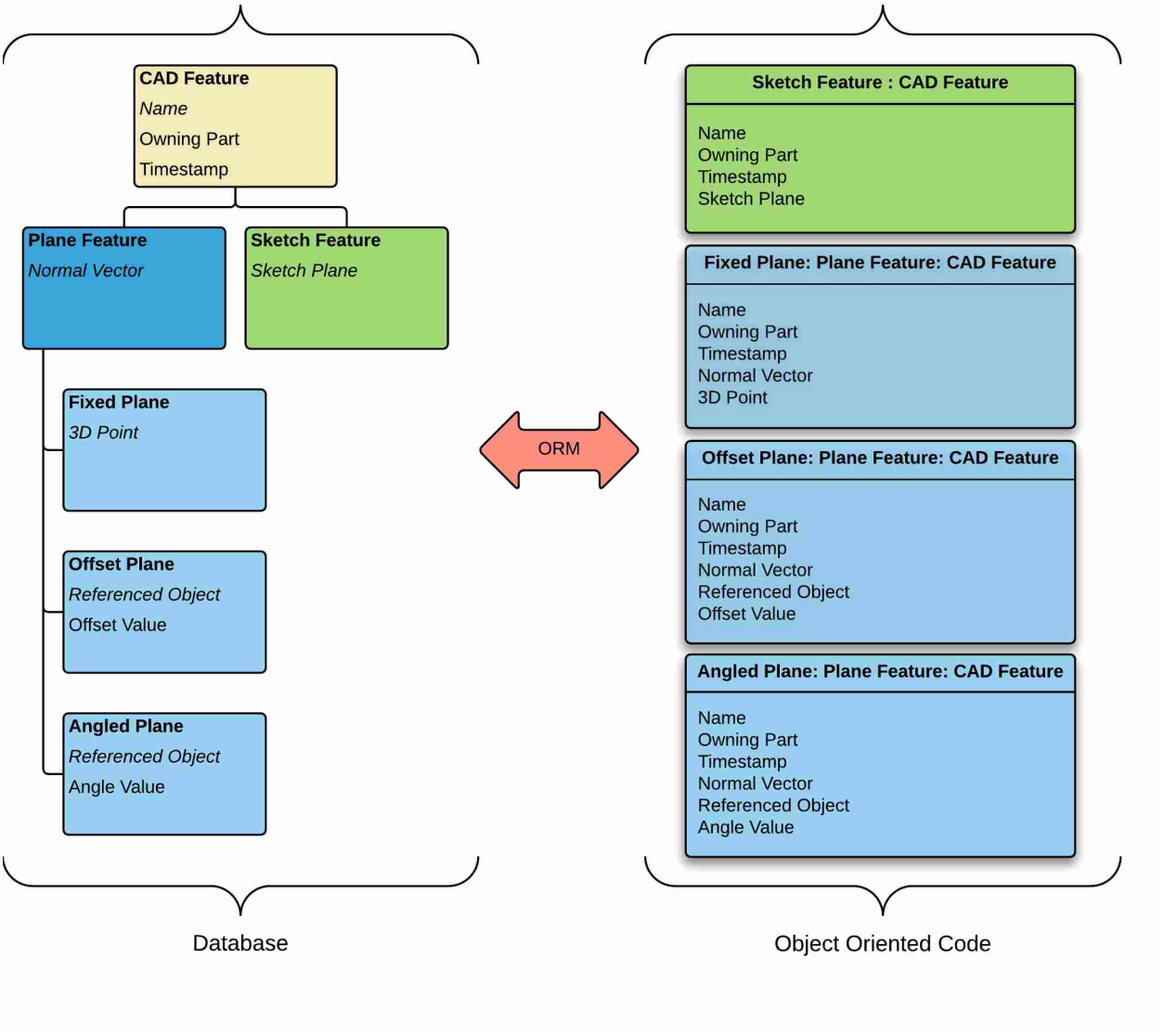


Figure 3.7: The Object Relational Manager converts the database tables and schema into programming data structures

are derived, and are used for accessing and writing data within the database. The hierarchy established within the database between tables is also retained in the data structures created by the ORM as object inheritance.

Apart from combining feature parameters into a single feature object, inheritance is used to define data types. Since each feature variation is represented in the database as a separate table and modified by the ORM to be represented as separate objects inheriting properties from the base feature class, each feature variation object is of the same type as the base feature object. This is an important principle which enables feature references. A 2D sketch feature in CAD, for example,

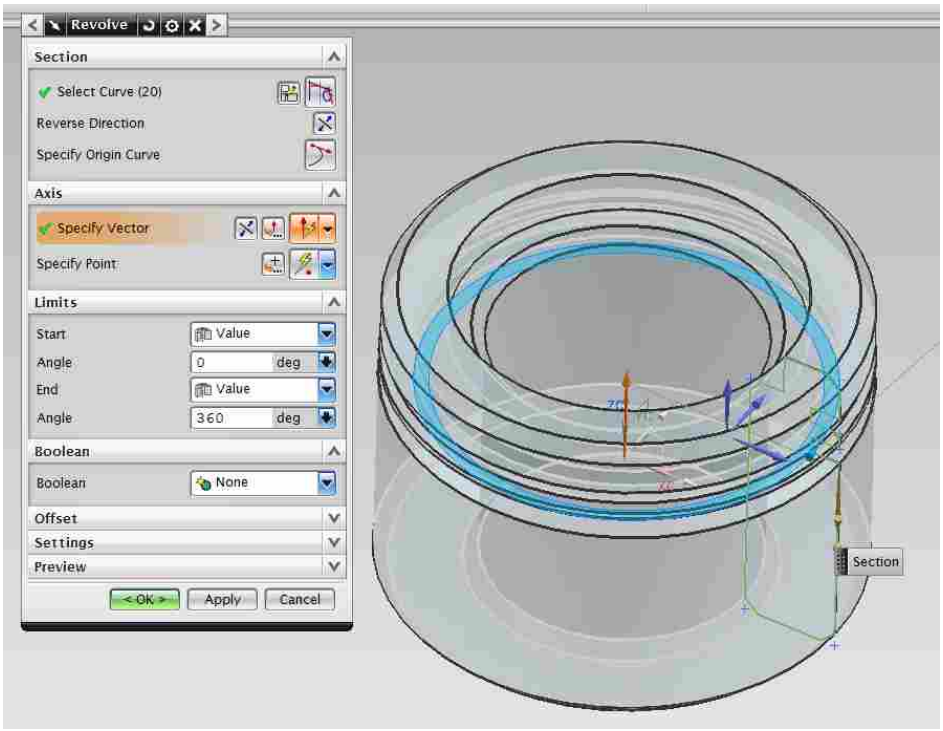
references a plane feature to define location. A plane feature has multiple feature variations from which it can be defined. Using the method describe above, the ORM will generate feature objects for each feature variation. However, since each plane feature variation inherits the base plane feature, the parameter used to reference a plane feature in the sketch feature object needs only to be of type plane object, and any variation of the plane object can be stored. This method both simplifies development and helps to preserve data integrity because only one reference parameter is needed to reference multiple feature definitions.

3.6 Interface Inheritance

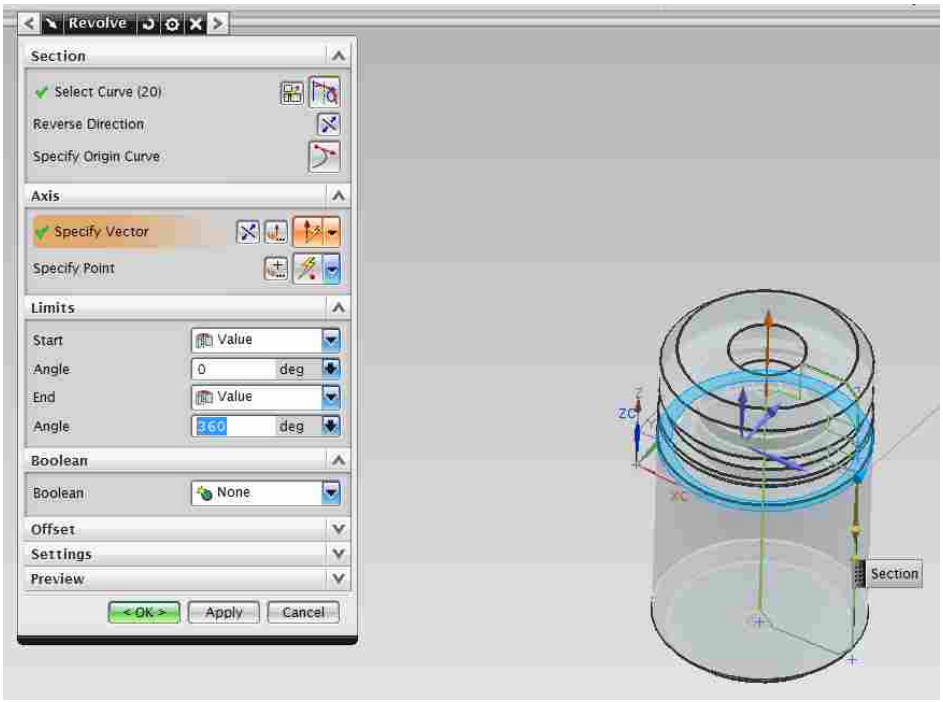
While the method described above enables multiple feature definitions to be referenced, many CAD features can reference completely different types of objects in a single parameter. This is best illustrated by the example in Figure 3.8. A revolve feature revolves a two dimensional sketch around an axis to generate three dimensional geometry. The axis of the revolve could be an axis feature, defined by a 3D point and vector, or could be a line object in a 2D sketch. These two objects in no way share an inheritance structure but must be able to be referenced in a single parameter to maintain data integrity. Further modifications to the ORM are necessary to support this functionality.

This is accomplished by modifying the code generated by the ORM to implement an interface system. Interfaces are a commonly used paradigm in object oriented programming to declare functional similarities between different object types. In essence, it allows unrelated objects in code to interact with other objects in the same way. In the case of the revolve feature example, both the axis feature and the 2D line object implement an interface which denote that both objects can be used as a reference for an axis. In this case, the revolve feature axis parameter does not require a specific feature type, but instead requires an object which implements an axis interface. This is done by modifying the part of the ORM which automatically generates the neutral data structure to include the interface code.

Database tables and data structures which reference an interface rather than a feature need to be modified as well. Because each association in the database requires a single table to represent to represent the association, any feature table using a parameter to reference an interface must create the association with the most basic database table from which all feature tables associate.



(a)



(b)

Figure 3.8: A revolve feature can revolve a 2D sketch around (a) an axis of a coordinate system or (b) a line in the sketch

The parameter used for the reference is given a reserved name which represents the type of interface used. For the case of an axis reference, the reserved name is ReferencedAxis while other reserved names can be seen in Table 3.3. The ORM is modified to identify these reserved names and replace all associated references to the basic CAD object data structure with the interface type corresponding to the reserved name.

Table 3.3: List of reserved names for database columns used to define parameters which reference interfaces

Reserved Name	Function
ReferencedAxis	Declare an object can be used as an Axis
ReferencedPlane	Declare an object can be used as a Plane
ReferencedDirection	Declare an object can be used as a Direction
ReferencedPoint	Declare an object can be used as a Point

While creating a database association with the most basic database table lowers the data integrity of the database by allowing any feature, including invalid features, to be referenced; invalid data is prevented from being added to the database by the modified ORM data structures. The data structures allow only features which implement the correct interface to be referenced. All read/write operations to the database are abstracted as far away from the user as possible to reduce the risk of data corruption. The ORM is as close a possible to the database that it ensures future developers aren't able to make any errors and corrupt the data.

3.7 Client Plug-in Software Packages

To simultaneously support multiple CAD packages within the heterogeneous environment, a client program is integrated as a plug-in to each supported system. This plug-in software is responsible for identifying when new operations are performed, converting information about the operation into the corresponding neutral format, and then packaging the neutral operation data to a format used by the ORM to fill the corresponding database tables. The client then sends this data to the server to be distributed to other clients and to the database.

To facilitate the wide variety of APIs associated with CAD programs the original client-side architecture was modified. As a result of this research, each client now consists of two different

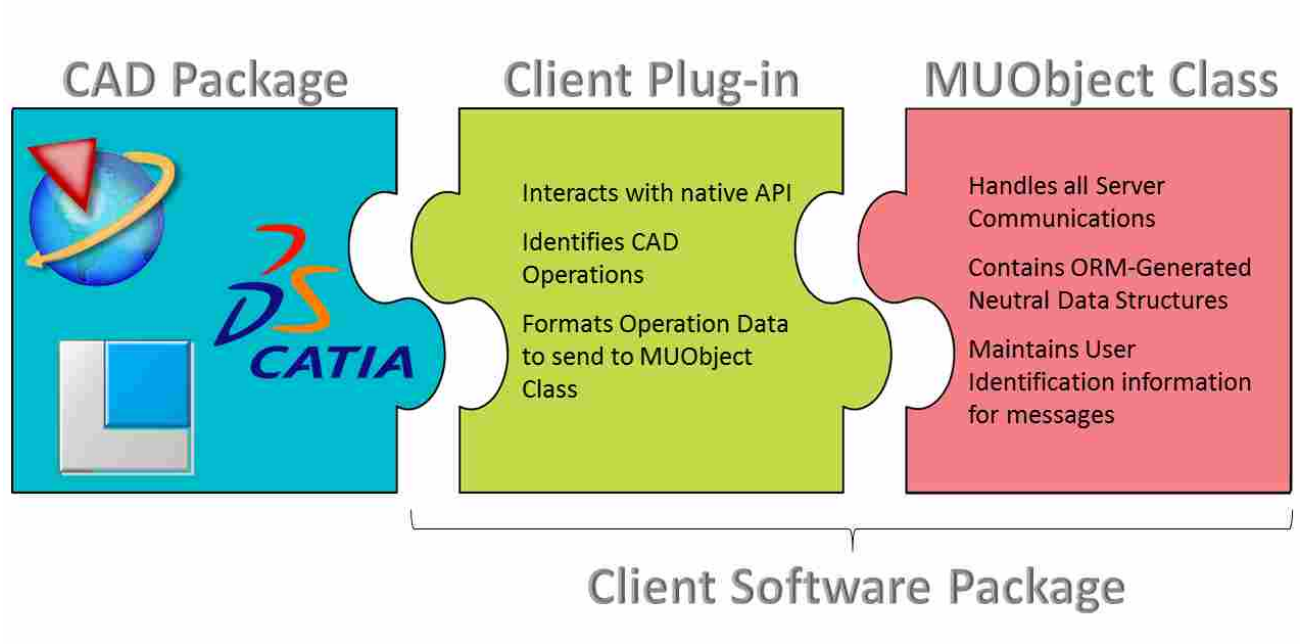


Figure 3.9: The software for the client is split into two sections. A plug-in interacts directly with the CAD program to identify and capture CAD operations, while the MUObject class neutralizes and transmits the data to the server

parts, as seen in Figure 3.9. The multi-user object (MUObject) classes are responsible for handling communications with the server and for neutralizing CAD data. These MUObject classes are all written in the C# programming language. These objects contain the code that was generated by the ORM to read from, and write to the database. When a CAD operation is performed, the MUObject classes convert the operation data to a neutral format and fill the ORM-generated classes. A message is then sent to the server containing the neutral data along with information pertaining to the originating user, the part model that the message is associated with, and a time stamp to be used with ordering operations. The MUObject classes also receive and deserialize messages from the server. Since all clients perform these same basic functions, the MUObject classes can be identical for all CAD packages.

The second part of the client interacts directly with the CAD system's API, and is written in the language best supported by the system. For the case of NX and CATIA, the client plug-in is written the the C# programming language for simplicity of interacting with the MUObject classes. This is possible for these systems because their respective CAD API's fully support the .NET framework. The client plug-in for Creo, however, is written in the C++ programming language because its .NET API lacks important functionality required to support interoperable CAD. The C++ API, however, is complete, fully functional, and able to support the CAD functionality required for interoperability.

The plug-in section of the client utilizes event methods within each CAD system to determine when feature data needs to be synchronized between clients, such as in the case of a feature creation, edit, or delete event. The plug-in then collects information about the updated feature and formats the parameters in a way readable by the MUObject class. When data is received from the server from remote clients or the database, the plug-in calls the proper methods within the API of the CAD system to create, edit, or delete a feature, or perform a part model or assembly level command.

By separating the client software package into two parts, developers working on integrating new CAD packages into the interoperable software are, in essence, further abstracted away from the communication aspects of the multi-user program. Instead, developers are only concerned about translating between the CAD specific data and the neutral format defined in the MUObject

class. This work flow has significantly accelerated development time as new features are supported in an incremental process between CAD systems.

3.8 Neutral Reference Verification

After each referential addition to the NPCF was completed, the newly supported features and methods were tested to validate the approach and verify that the method worked between all three systems. For multi-user CAD modeling to be effective, users must be able to create and edit features on all systems. For the heterogeneous CAD environment, this means that each system must support each feature and define the geometry in the same way. To ensure this happened, testing occurred incrementally as new features were added to each system.

CHAPTER 4. IMPLEMENTATION AND RESULTS

The Neutral Parametric Canonical Form is a set of prototype standards for representing CAD features and other CAD data in a neutral format that supports multi-user, simultaneous CAD. The NPCF has been implemented and tested using Interop, a client-server application utilizing plug-in software to interface with commercial CAD systems. Interop is developed by the BYU Site of the NSF Center for e-Design to understand the requirements necessary to support multi-user CAD processes between these heterogeneous CAD systems. The objective of this research is to extend the NPCF format and the Interop prototype to support associativity and enforce referential integrity during the modeling process in an effort to preserve design intent.

The sections in this chapter detail the implementation process and results for the methods described in chapter 3.

4.1 SQL Database Implementation

The Neutral Parametric Canonical Form defines a set of parameters and relationships which express CAD features in a neutral format. To test their viability, the NPCF was implemented in a SQL database, with a database table for each feature variation, and a database column within the tables for each neutral feature parameter. A SQL database is an ideal format to store neutral CAD data because it is optimized for basic data types. The main objective for neutralization is to convert CAD specific data to a set of these basic data types which are then able to be read and interpreted by any other system.

In this implementation, each table contains a Globally Unique IDentifier (GUID) database column which is used to uniquely identify neutral CAD features stored in the database. It is generated by the client from which the feature originated using the Microsoft Windows API to guarantee uniqueness between all clients with a high-degree of certainty. A flowchart detailing the order of operations after a new feature is created is shown in Figure 4.1. After a new feature is

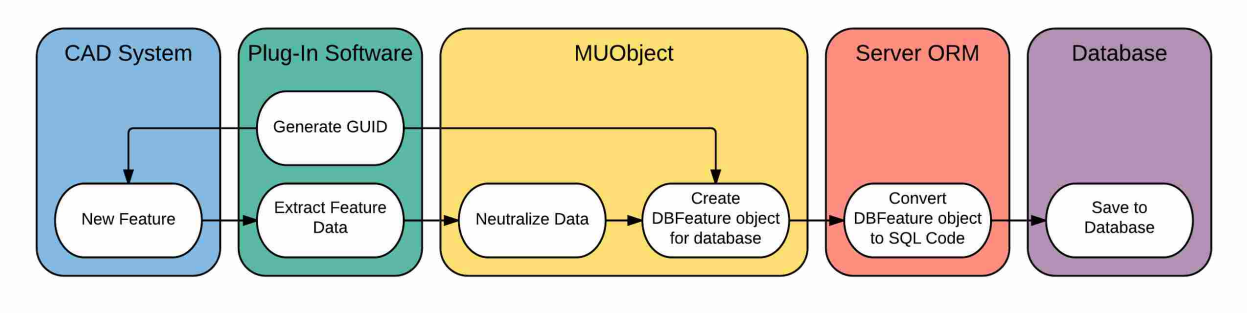
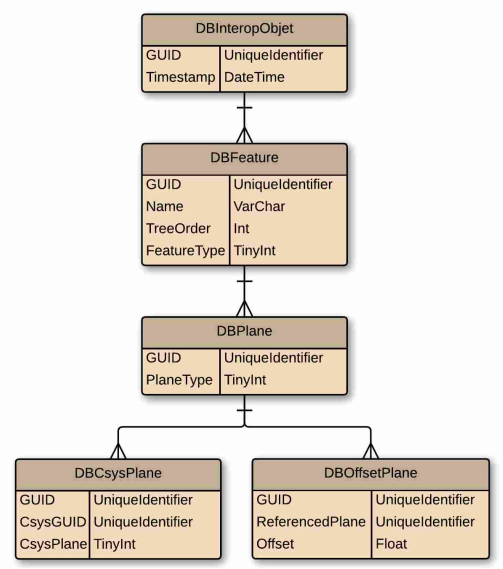


Figure 4.1: Order of operations after a new feature is created in the Interop multi-userprototype

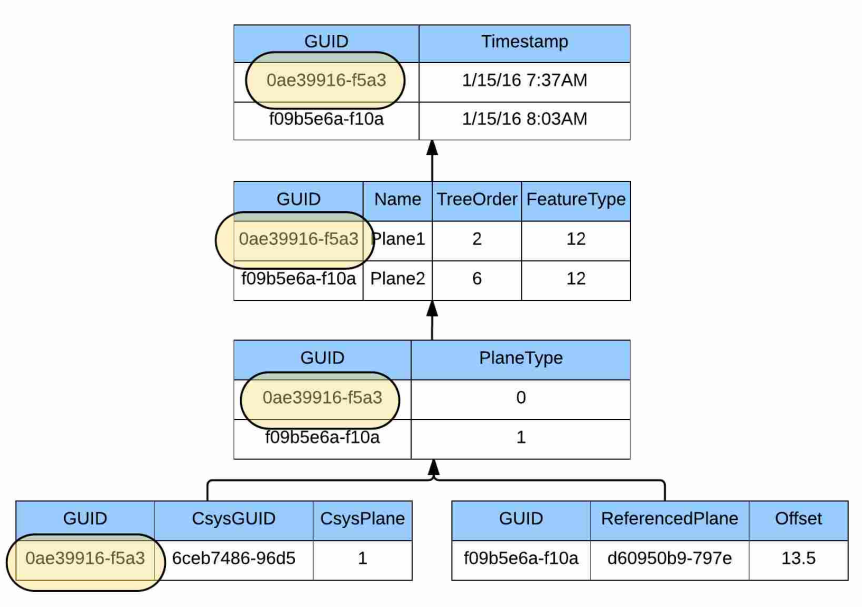
created in the CAD system, a GUID is generated by the plug-in software and stored with both the neutral data in the MUObject class and with the CAD feature in the CAD system. The GUID is used to identify features on all clients, update feature data in the database when edits are performed in the CAD system, and link feature data in the database.

Because the GUID is unique between all clients, it can be used to allow a single feature to be partially represented by multiple database tables in a hierarchical format, similar to the way CAD features would be represented in a CAD system's API. For example, data for a coordinate system plane feature is stored in four separate tables as seen in Figure 4.2. The figure shows both the database schema (a) which describes how the database tables relate to each other as well as a representation of the data stored within each table (b). A GUID and time-stamp for the feature are stored in the InteropObject table. The same GUID, feature name, tree order, and feature type is stored in a Feature table. The GUID and plane type is stored in the DatumPlane table and the GUID, associated coordinate system, and plane type is stored in the DBCsysPlane table. Each table is linked together using a primary-key/foreign-key reference which automatically matches the correct data when querying the database for a feature.

Data integrity is enforced by ensuring each table contains the largest set of parameters not shared with other features or feature variations. This ensures the greatest number of unique feature variations and all columns of each variation table must be completely filled in order to be valid. In order to accomplish this, features must be grouped according to similarities when determining the neutral format. For example, to define a coordinate system plane, the associated coordinate system must be referenced. Since this is not required for an offset plane, the CsysGUID and CsysPlane parameters are included within the DBCsysPlane table, and not the DBPlane table. The resulting



(a)



(b)

Figure 4.2: (a) Neutral data for a single feature is partially represented by multiple tables. (b) GUIDs are used to link partial data in each table together to completely define a neutral feature.

table hierarchy is mirrored during the development of the MUObject classes so each MUObject is responsible for extracting the data required to fill its associated table.

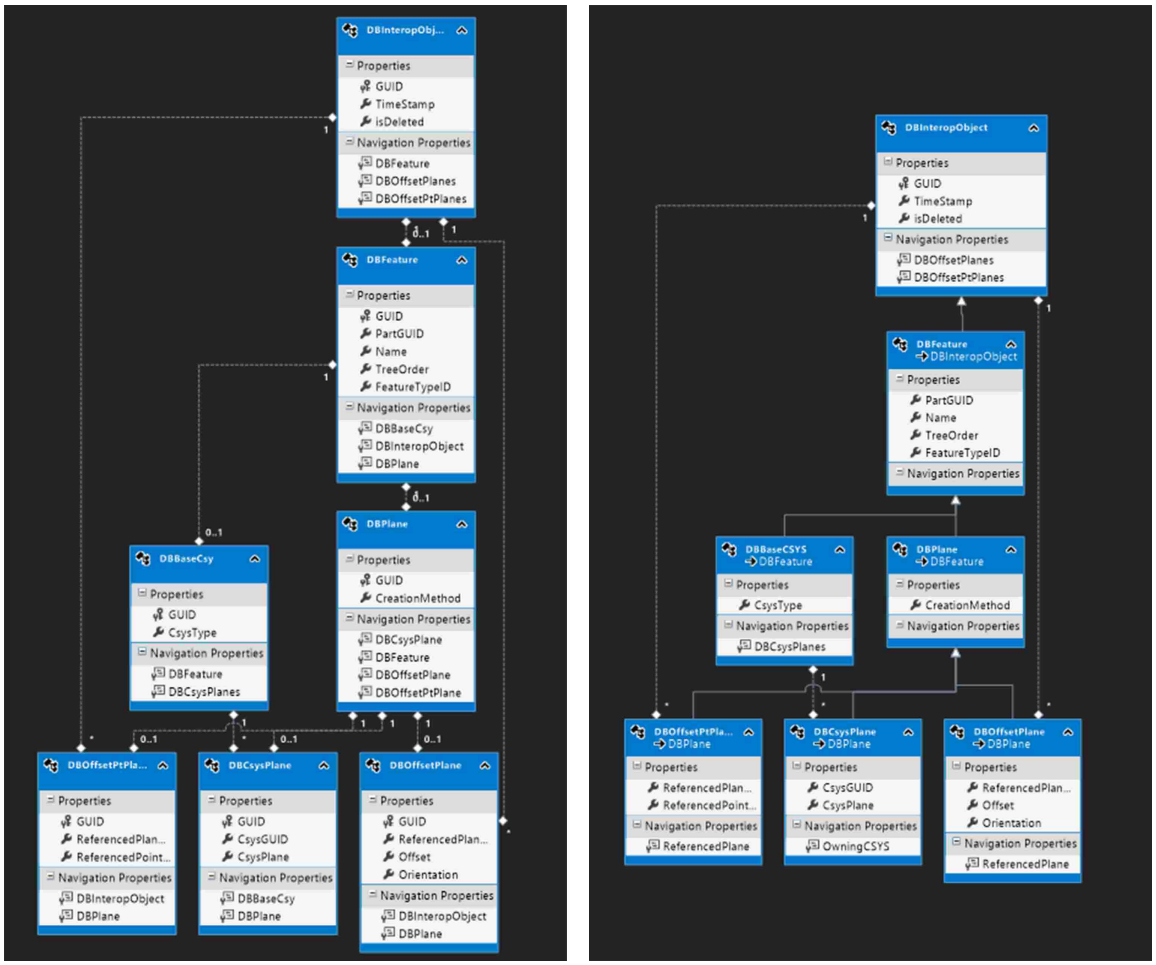
This SQL database implementation also allows feature references to be stored as parameters within the database tables. Since features are represented as database tables, these CAD references are stored in the SQL database as primary-key/foreign-key associations between tables, with the primary-key being the feature table being referenced, and the foreign-key being the feature table containing the reference parameter. Any feature variation which requires a feature reference will store the GUID of the referenced feature in the associated parameter database column. In Figure 4.2 this occurs in the DBCsysPlane table in the CsysGUID column. The data stored within the CsysGUID column is the unique identifier of the coordinate system feature that is referenced by the coordinate system plane. When querying the database for a feature that utilizes a feature reference done in this manner, information about the referenced feature is also automatically returned with the query.

4.2 Object Relational Manager Implementation

The main purpose of the object relational manager in the CAD Interop prototype is to provide an effective interface with the database. The ORM generates programming classes based on the database schema. To write to the database, the developer writes to the properties of the ORM class. When querying the database, the ORM fills the ORM class properties based on the values stored in the database. In the Interop implementation, the majority of the ORM processes occur server-side. When geometry is created in the CAD system, the client software populates the neutral feature parameters in the associated ORM class and serializes the ORM class to be sent to the server. The server uses the ORM to save the feature parameters stored in the ORM class to write to the database.

A secondary purpose of the ORM specific to this implementation of the CAD Interop software is to enable an interface-like structure when interacting with the database. A typical SQL database only supports associations between two tables when using a single database column to hold the foreign key. While multiple columns could be used to support associations between multiple tables, this would violate the principle of referential integrity for a CAD system because only one parameter should be filled out at a time while the database could potentially fill out multiple

parameters. To support associations between multiple tables while only allowing one association at a time to be utilized, thus supporting referential integrity of the data, the classes generated by the ORM are modified implement an interface.



(a) (b)

Figure 4.3: Table hierarchy data automatically generated by ORM (a) is modified to form an inheritance structhre (b)

Microsoft's Entity Framework 6.1.3 was chosen to be the ORM for the CAD Interop prototype because it is customizable and is closely integrated with the C# programming language. Entity Framework has a visual interface which is used to set most properties of the objects that represent database tables. After each feature table is added to the SQL database, the Entity Framework model representing the tables of the database is updated to incorporate the new feature and

all associated references. Basic table schema and associations are automatically downloaded and incorporated into the entity framework model. Associations made to other tables are called navigation properties which store the object that they reference.

For example, the OffsetPlane object defined by the ORM is based off of the InteropObject, Feature, DatumPlane, and OffsetPlane tables described previously. The associations between these tables are changed in the ORM to inheritance relationships on the GUID property. This instructs the ORM to combine all parameters defined in the four tables together when creating the OffsetPlane data structure. The FixedPlane data strucuture is created in a similar manner - combining the parameters defined in the InteropObject, Feature, and DatumPlane tables with the FixedPlane table.

Figure 4.3(a) shows the datum plane objects generated by the ORM immediately after import from the database. Microsoft's Entity Framework GUI represents database table associations with dotted lines between objects. While some of these associations will be used to represent feature reference parameters, most need to be converted to form inheritance relationships. After post-processing, the ORM GUI looks like Figure 4.3(b). During post-processing, associations are replaced with inheritance relationships, represented by solid arrows pointing to the base class. GUID parameters are deleted from derived members to signify that the single GUID parameter of the base class is used for all derived members. When uploading data to the database, the proper GUID database columns are filled with the correct data.

The final modifications to the ORM code enable the objects to implement and reference interfaces. Classes are automatically generated from the database schema through the use of a t4 text template which is included with entity framework. This template is modified to identify navigation properties with the reserved names in Table 3.3. The names of the navigation properties are set by the database columns they are associated with. Any matches to the names are replaced with the associated interface type using the following code:

```
if(navProp.Name == "ReferencedPlane")
{
        navPropType = "IPlanarReference";
}
else if(navProp.Name == "ReferencedAxis")
{
```

```
            navPropType = "IAxisReference";
        }
```

Partial classes are then created for each object which is to implement an interface. For example, the heading of the partial class for a DBLine2D object is:

```
namespace ConnectData
{
    public partial class DBLine2D : IAxisReference
    {
        . . .
```

Utilizing partial classes ensures that any code implemented within them will not be overwritten when new ORM code is auto-generated.

4.3 Datum Features

A major focus of this research was to improve the way the NPCF captures and preserves design intent through associativity within and between features. As a result, this research produced an expanded definition of the NPCF schema which allows the associative references to be stored. While the definitions of the defined neutral features are captured within the database schema as the table structure and associations, interactions with the NPCF occur in code through the data structures generated by the ORM.

To further test the ability of the NPCF definitions for preserving design intent through associativity, new features were supported. Because datum features are used by designers and engineers to define important locations and parameters of a CAD part model or assembly, they are ideal for this test. The datum features of coordinate system, axis, and plane were implemented as associative features within the NPCF database schema and incorporated into each supported client CAD system. Figure 4.4 shows how these features are used to define position and orientation in PTC Creo 3.0. The layouts of the data structure used to neutralize these features are contained in this section.

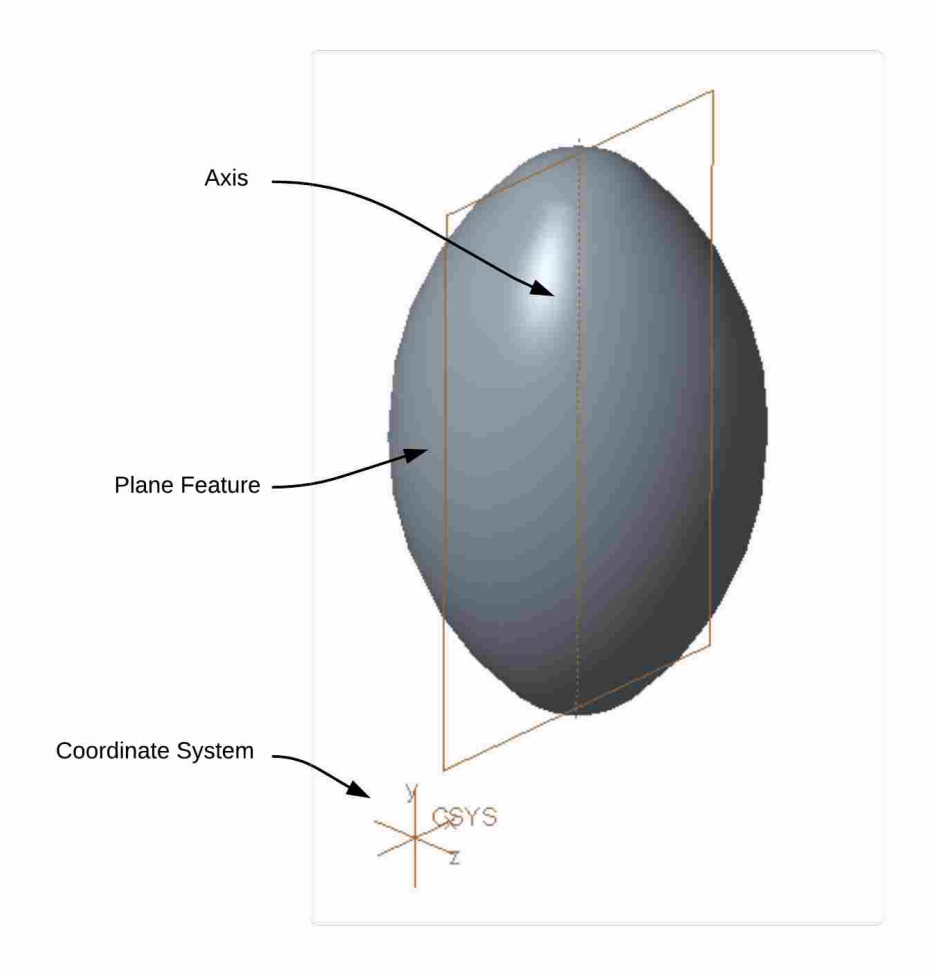
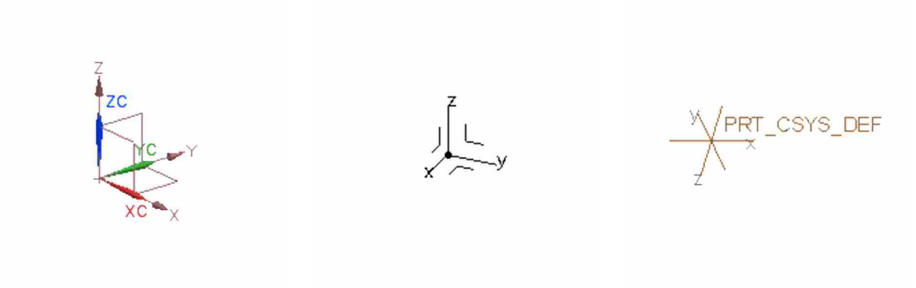


Figure 4.4: Datum Features in PTC Creo 3.0 are used to define the position and orientation of solid models.

4.3.1 Coordinate System Features

A coordinate system is often the first CAD feature in a part or assembly, and is used to define the directions and positions of the axes in the model. It is so commonly used that many CAD systems have a default setting to automatically include this feature when a new model is created. Even with its common use in all major CAD systems, however, they are defined in very different ways. Figure 4.5 shows coordinate system features from all three supported CAD systems. Coordinate systems in both Siemens NX and Dassault Systemes CATIA contain a 3D point for defining location, 3 axis systems for defining orientation, and 3 planes for construction of geometry. In PTC Creo, however, coordinate systems have only a 3D point and 3 axes.



(a) (b) (c)

Figure 4.5: Coordinate systems in default orientations as represented in Siemens NX (a), Dassault Systemes CATIA (b), and PTC Creo (c)

Because the planes are automatically included with the coordinate systems for NX and CATIA, it is necessary to support these planes on Creo as well. To mirror functionality on all systems, they must be created in Creo when a coordinate system is created. This is important because during the neutralization process, any feature which has a reference to one of these coordinate system planes, must be able to reference the same plane on a remote CAD system to be created correctly. When a coordinate system feature is created in Creo while using the Interop prototype software, three planes are automatically created to mirror the planes which are created by default on NX and CATIA when the neutral feature is Incorporated into their respective part models. When opening neutral models containing a coordinate system in Creo, these planes are also created.

Furthermore, part models and assemblies in Creo are automatically created with a coordinate system and three separate planes as default features. They are used to define the global coordinate system of the model, and don't have any parameters definable by the user. These features are necessary because they are used to define locations and orientations for new features. Though this default coordinate system is identical to coordinate systems created later by the user, it is defined using different parameters and is entered into the NPCF as a feature variation of the neutral coordinate system feature. As discussed in Chapter 3, supporting these two feature types as variations maintains the referential integrity of the database by only requiring each parameter in feature table to be filled completely.

Figure 4.6 shows the complete schema for a neutral coordinate system feature in the NPCF. The schema accounts for both feature variations of the coordinate system: a coordinate system with parameters set by the user and a coordinate system found at the origin of the part. Both of

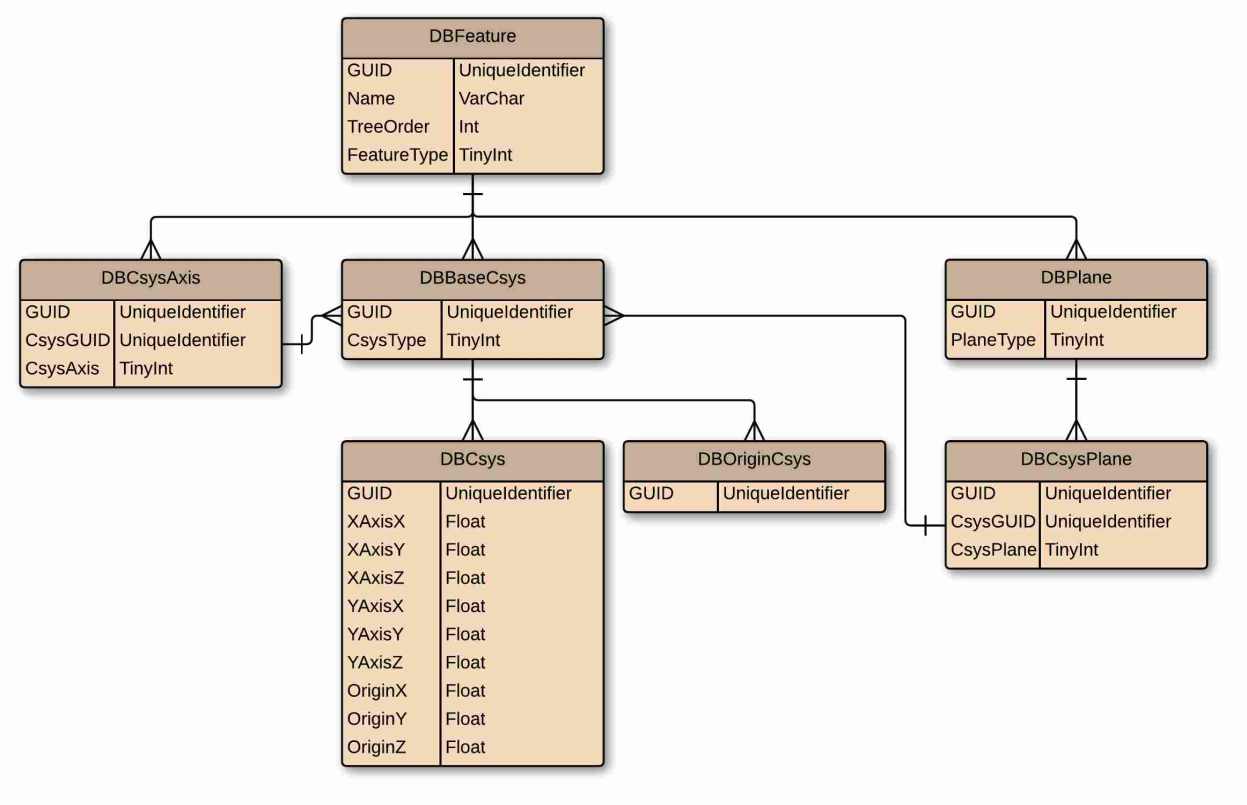


Figure 4.6: Database schema used to define the neutral parametric canonical form for a coordinate system feature contains associations with the axis table and plane table for a complete definition

these variations share an association with the DBBaseCsys table which maps to an inheritance relationship in the classes generated by the ORM. This DBBaseCsys table is referenced by the DBCsysAxis table and the DBCsysPlane table enables the axes and planes of the coordinate system feature to be referenced independently of the coordinate system but still be associated with it for identification on all systems. The ability to reference these sub features independently of the coordinate system is important because each sub feature has very different properties. The planes, for example, are used in very different ways from the axes. By keeping them separated in the database allows each to be referenced by other features. This would not be possible if, for example, all parameters were contained in a single DBCsys table.

Though not always expressed as a separate feature, all part models in all CAD systems reference a global coordinate system. Because some CAD systems, like Creo, create a feature to define this global coordinate system, it is necessary to include it in the neutral format so that when it is referenced, it can be referenced in any CAD system. To support this, all neutral part files created

using the NPCF must include, as its first feature, an origin coordinate system with its associated planes and axes. When a new part is created using the Interop prototype, a DBOriginCsys object is created and associated with three DBCsysAxis objects and three DBCsysPlane objects. Since the creation of neutral parts happen independently of a CAD system, this process is located within the MUObject portion of the client software. This ensures that the same part definition occurs on any system without any involvement from future developers, thus reducing the chance of data consistency errors.

4.3.2 Plane Features

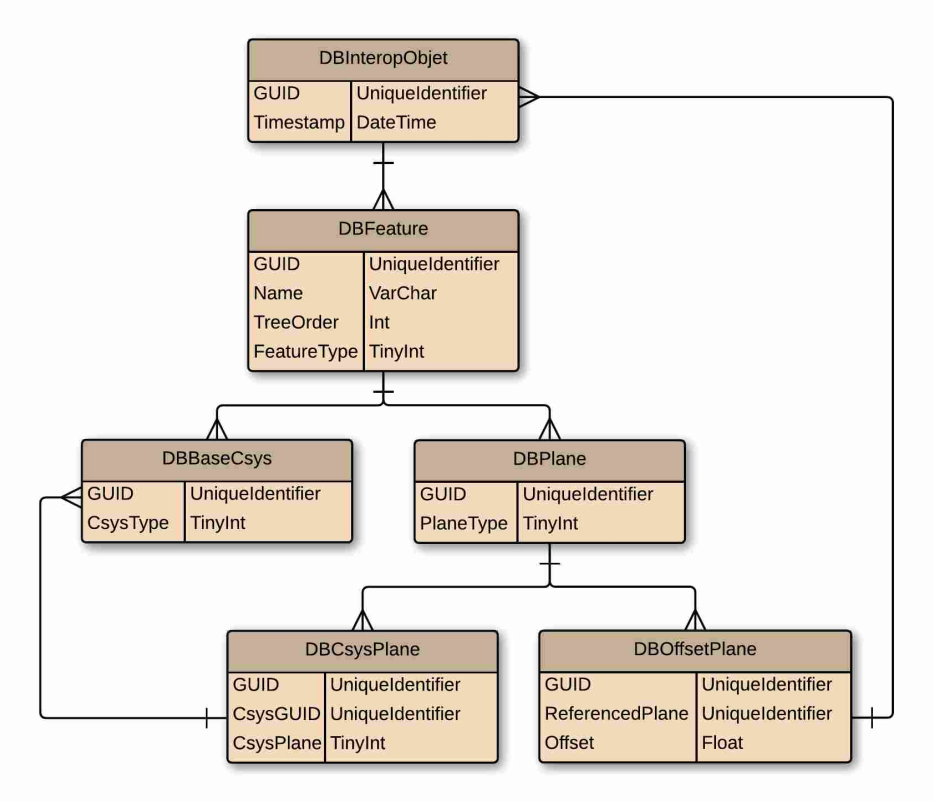


Figure 4.7: Database schema illustrating the two variations of the plane feature that were implemented into the Interop prototype software

Because datum planes were among the top used features in the BYU Senior Capstone project, a considerable amount of attention was given to them to support their required functional-

ity. Datum planes are often used to define the location of sketch features or the limits of extrude and revolve features. The implementation of this feature was divided amongst the two supported feature variations, coordinate system plane and offset plane. Both of these variations are associated with a base plane database table which, after translated by the ORM, becomes a base plane class implementing the IPlanarReference interface. This interface signifies that the base plane, and all feature variations of the plane, can be referenced by other feature objects as a planar reference.

The implementation of the coordinate system plane, illustrated in Figure 4.7 by the DBCsysPlane database table, includes a reference to the associated coordinate system and a property to represent the type of plane. This reference is made by a one-to-many association between the DBBaseCsys table and the DBCsysPlane table with the CsysGUID column of the DBCsysPlane table being used to store the foreign key. This type of association allows one coordinate system to associate with multiple coordinate system planes. To identify each of these planes, the DBCsysPlane table contains a CsysPlane column which stores an enumeration detailing the type of plane the data represents. The values of the enumeration stored in this property are described in Table 4.1.

Table 4.1: Coordinate System Plane Type Enumeration

Value	Enumeration	Description
0	xyPlane	Plane defined by the X and Y axes of the coordinate system
1	yzPlane	Plane defined by the Y and Z axes of the coordinate system
2	zxPlane	Plane defined by the Z and X axes of the coordinate system

An offset plane feature describes a plane which is parallel to another planar object separated by a finite offset. This feature, implemented in the NPCF definitions in the DBOffsetPlane database table, contains an offset property for storing a floating point value for the offset and a ReferencedPlane property to enable the reference with the parallel planar object. Because a planar object could be a variety of features, rather than associate with a plane feature, it must associate with the PlanarReference interface described in chapter 3. To support this interface method, the ReferencedPlane property stores the foreign key for its one-to-many association with the generic

DBInteropObject table. ReferencedPlane is a reserved name which triggers the custom code in the ORM to associate the property with the PlanarReference interface.

The MUObject portion of the client was written specifically to handle the hierarchical format of the plane feature variations. In an effort to reduce future development time and to minimize potential errors, only the MUPlane class is publicly accessible from other code within the MUObject structure. Both the MUOffsetPlane and MUCsysPlane are child classes of the MUPlane class and are marked private. When a remote plane feature operation is sent from the server, it is passed to the MUPlane class to determine the type of plane. The MUPlane class then passes the data to the respective MUObject class for the feature variation which converts it a format readable by the plug-in portion of the client software.

4.4 Feature Extensions

Continued research and development on the Neutral Parametric Canonical Form has expanded the definitions and capabilities of features previously overlooked [5]. In this case, the expansions served two purposes. First, the features are more complete and more closely resemble definitions employed by typical CAD systems. Secondly, and most importantly, the extended features better capture the design intent of the modelers as they support associativity with other features. Both the extrude and revolve features received these expansions.

4.4.1 Sketch Features

In modern 3D CAD, sketches are used for defining 2D geometry that, when associated with other CAD features like extrudes and revolves, defines 3D geometry. Figure 4.8 illustrates the sketch referenced during a revolve operation in Creo. Is this example, the sketch location and orientation is defined by the part coordinate system. In the previous NPCF implementation, the sketch positioning data was dis-associated from the coordinate system. As a result of this research, the sketch plane of the neutral sketch feature definition references the plane feature. Updates to the location and orientation of the plane automatically propagate to the sketch.

The neutral database schema for a sketch feature is shown in Figure 4.9. To make a reference to a planar object, the ReferencedPlane parameter is associated with the generic InteropObject

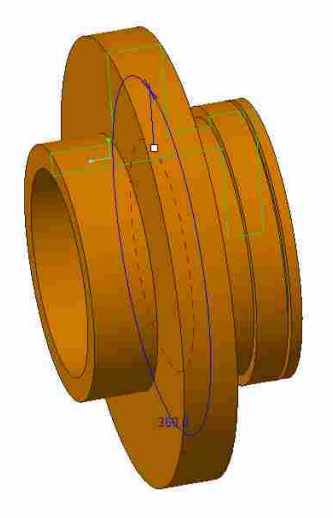


Figure 4.8: A revolve feature in PTC Creo 3.0 references a sketch feature and an axis to generate 3D geometry

table. This association, as well as the reserved name, signifies to the ORM to create a PlanarReference interface parameter. The HAxis and VAxis parameters are used to define the horizontal and vertical axes of the sketch respectively. These axes define how the sketch geometry lies on the plane.

4.4.2 Extrude Features

An extrude feature creates three dimensional geometry by expanding a two dimensional sketch and was one of the top used features in the BYU Senior Capstone project. Like most CAD features, there are often a variety of extrude definitions that can be used to capture design intent. These variations, however, are not implemented in the same way as the features previously discussed. An example illustrates this point. A blind extrude is created when the two dimensional sketch is expanded a specified, finite distance. If the sketch plane is moved, the extrude is moved as well, but the extrude distance remains the same. A planar extrude, however, expands the sketch to meet a planar object. If the sketch plane is moved, the extrude is moved but is expanded to the same plane. The two types of extrudes can, however, be used together, with one side of the sketch being expanded to a planar object while the other is expanded a set distance.

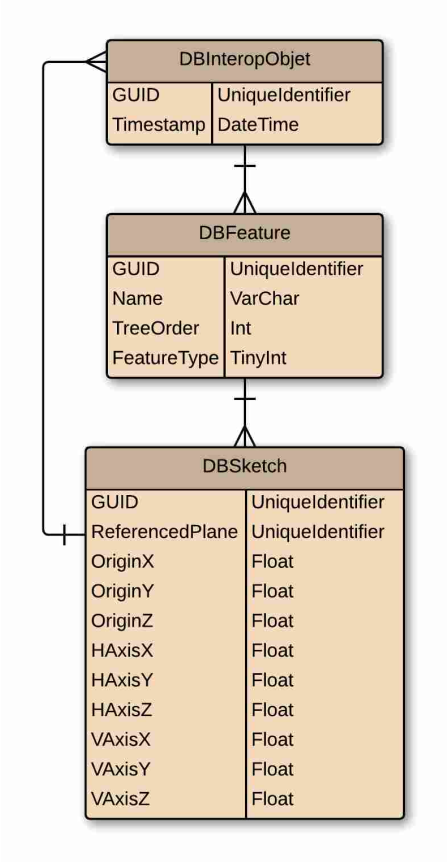


Figure 4.9: Neutral sketch features in the NPCF reference plane features to define sketchlocation

To capture both types of extrude limits while maintaining the referential integrity of the database, two associations with the DBLimit table are employed which store data for each side of the extrude feature, as seen in Figure 4.10. The DBLimit table, in turn is associated with both the DBBlindLimit table and the DBPlanarLimit table to store values and planar references respectively. This database schema allows for mixed limit types to be used when creating an extrude feature.

4.4.3 Revolve Features

A revolve feature, like an extrude feature, expands a two dimensional sketch into three dimensional geometry. It does so by revolving the sketch about a user-defined axis by certain values. Because multiple CAD objects can be used as an axis for the revolve, the DBRevolve table, seen in Figure 4.11, is associated with the DBInteropObject table with the foreign key being stored in the AxisGUID property. After being processed by the ORM, a property is created which

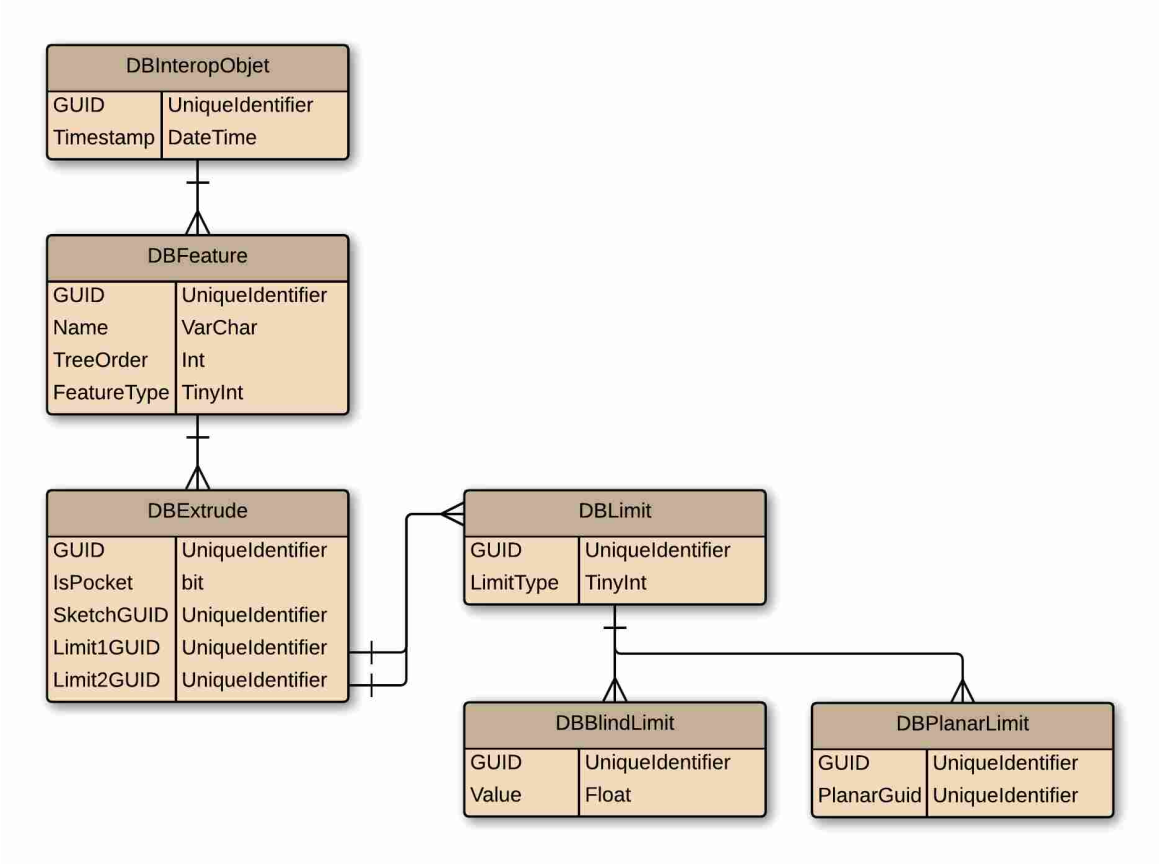


Figure 4.10: Multiple limit objects are associated with the DBExtrude database table

implements the AxisReference interface. This interface is also used by each of the three axes in coordinate systems and by 2D line features in sketches.

4.5 Feature Capabilities and Limitations

Each feature supported by the NPCF can be created, edited, and deleted in NX, CATIA, and Creo during the modeling process. During the implementation process, each feature is tested individually and all features collectively during regular team-modeling sessions. During these sessions, errors that arise are recorded and investigated to determine the cause. Most are due to bugs and architecture limitations, and can be categorized into three underlying causes.

1. Some bugs are caused by programmer error. When implementing a new feature into three different CAD systems, crucial information is sometimes omitted causing a feature to be created incorrectly or not at all. The majority of these bugs are located in the plug-in portion

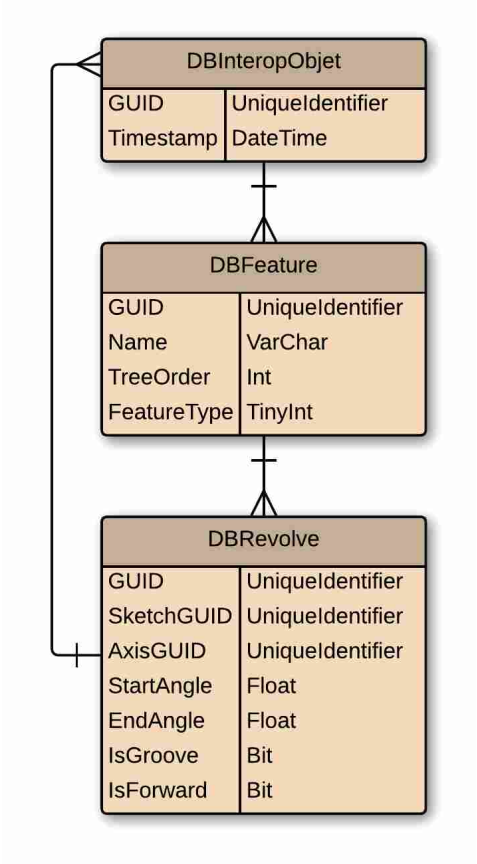


Figure 4.11: The DBRevolve table employs an association with the DBInteropObject table to employ the AxisReference interface

of the client code when extracting or setting CAD feature information. These types of bugs are given the highest priority during the modeling sessions and fixed immediately.

2. Some bugs arise because of limitations or errors within the CAD system's API. An example of this bug type occurred when implementing the revolve feature into PTC Creo. In the plug-in implementation for Creo, the integrity of local feature operations are preserved by ensuring remote operations don't conflict. This is done by subscribing to button events. During a team-modeling session, unexpected behavior was observed around the revolve feature and was traced to an API bug dealing with the revolve button event. The issue was reported to PTC 17 June 2015 and is currently under investigation. Workarounds are developed for these bugs to address the lost functionality. In the case of the example above, a new button

was placed in the user interface of Creo that must be pressed by the user after performing a revolve operation - simulating the functionality that was lost from the API bug.

3. Sometimes a single feature can pass all tests but, when used during a team-modeling session, can cause the program to crash. Often this is because of the lack of an overall consistency manager to ensure multiple operations aren't performed on the same feature. To limit the effect of this architecture limitation, methods were implemented in each client plug-in to queue remote operations when a local operation is performed. This fix has greatly reduced the amount of reported bugs during team-modeling sessions. This solution, however, does not prevent potential conflicts that may arise when multiple users are editing the same feature or feature tree. In it's current implementation, Interop avoids these conflicts because the only features supported don't remove geometry. An edge blend feature, for example, removes an edge during its operation. If another user were to, at the same time, create a feature which uses that edge, a conflict would arise. A server-level consistency manager should be implemented to make the software more robust and handle this type of conflict, though the existing solution is effective.

The current process for identifying, prioritizing, and fixing errors that arise has been effective in rapidly improving the state of the software. Future work into expanding the capabilities of the NPCF software should continue categorizing these types of errors to maintain the level of progress.

4.6 Modeling Demonstrations

As a result of this research, associations between features and methods for preserving design intent have been incorporated into the multi-user interoperability program utilizing the NPCF prototype standard. Neutral definitions for coordinate systems and datum planes, as well as extended definitions for extrude and revolve features have been defined and included in the heterogeneous system. Additionally, support for PTC Creo has been added to enable multi-user synchronous modeling between NX, CATIA, and Creo CAD systems. The capabilities of the software and NPCF prototype standard are demonstrated by modeling two separate assemblies. The assembly models were selected to utilize the features and test the methods developed and implemented

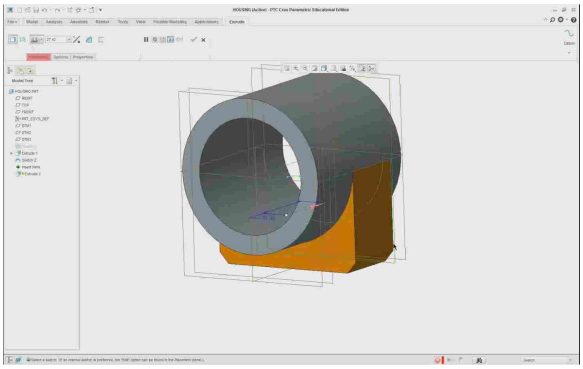
as a result of this research. Both assemblies were modeled by multiple users simultaneously. All users were co-located and allowed to collaborate prior to beginning modeling. While users in these sessions were co-located, the same process could occur between geographically separated clients by using a video conferencing service.

4.6.1 Water Pump Thrust Bearing

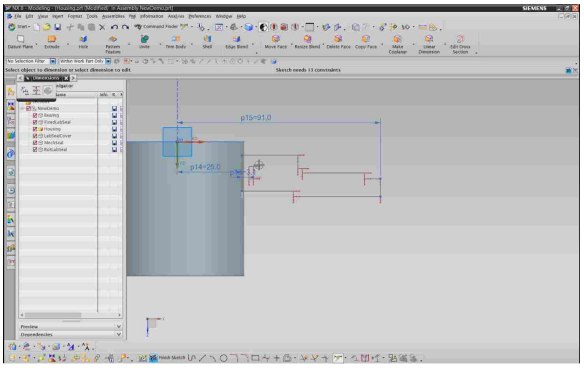
The first multi-user assembly presented was designed to showcase the associativity methods developed during this research. Three clients, using NX, CATIA, and Creo respectively, were given instructions prior to modeling which included the basic dimensions of each component of the assembly and the order of operations employed by a single user to model the part. Instructions for each model can be found in Appendix A. During modeling, clients simultaneously modeled within the same parts, collectively working to complete the design before moving to the next component. The absence of any conflict resolution implementations in this architecture, however, necessitated that clients communicate to avoid interfering with others' operations. This method was illustrated early in the modeling session as is displayed in Figure 4.12. After the initial cylinder was extruded to define location and size, each client simultaneously modeled geometry based off of the initial feature, albeit in relative isolation.

The plane and coordinate system features were used extensively to define the location of features in relation to other geometry. Planes, as seen in Figure 4.13, were used to define the locations of sketches and the limits of extrudes. Coordinate systems defined the axis of revolution of revolve features. Because of the associative nature of these features, when mistakes were made during the modeling session as to the limits or locations of new features by any client, the model could be edited to correct the mistake. Edits could be performed by any user, regardless of which client initially created the feature. Updates to the model were automatically propagated to all remote clients and stored within the database.

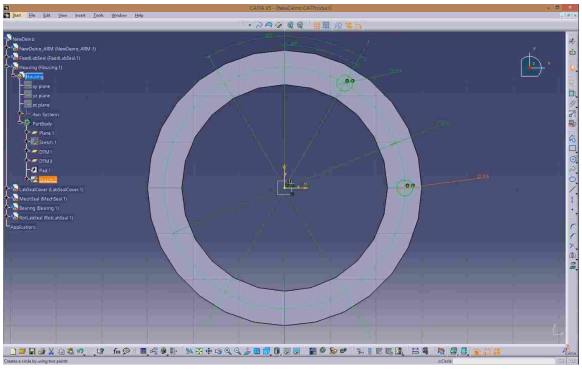
Figure 4.14 shows the finished pump bearing housing part model on all three CAD clients. Collaboration between clients enabled each user to work on portions of the model to collectively complete the model. A process, similar to the one employed in this session, not only enables users to work with the CAD system in which they are most comfortable, but also enable users with different specialties to access and contribute to the part model during the design stage.



(a)



(b)

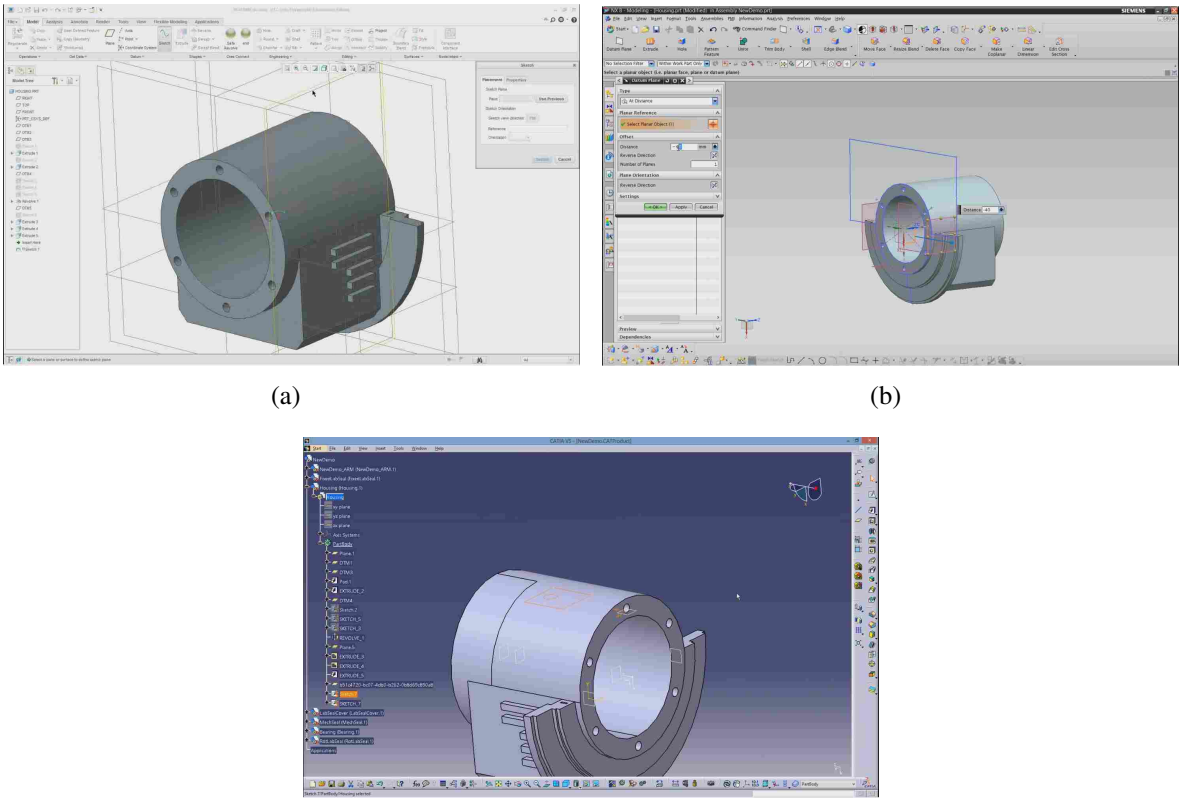


(c)

Figure 4.12: Interoperable Modeling Session between clients using the Creo (a), NX (b), and CATIA (c) CAD systems

Table 4.2: Comparison of model parameters of exported stereolithography files from each CAD system

Stereolithography File Comparison			
	NX	**CATIA**	**Creo**
X-Dimension	100.000 mm	100.000 mm	99.990 mm
Y-Dimension	82.500 mm	82.478 mm	82.495 mm
Z-Dimension	60.000 mm	60.000 mm	60.000 mm
Volume	114612.7 mm^3	114721.5 mm^3	114666.0 mm^3
Surface Area	35687.9 mm^2	35663.1 mm^2	35665.4 mm^2

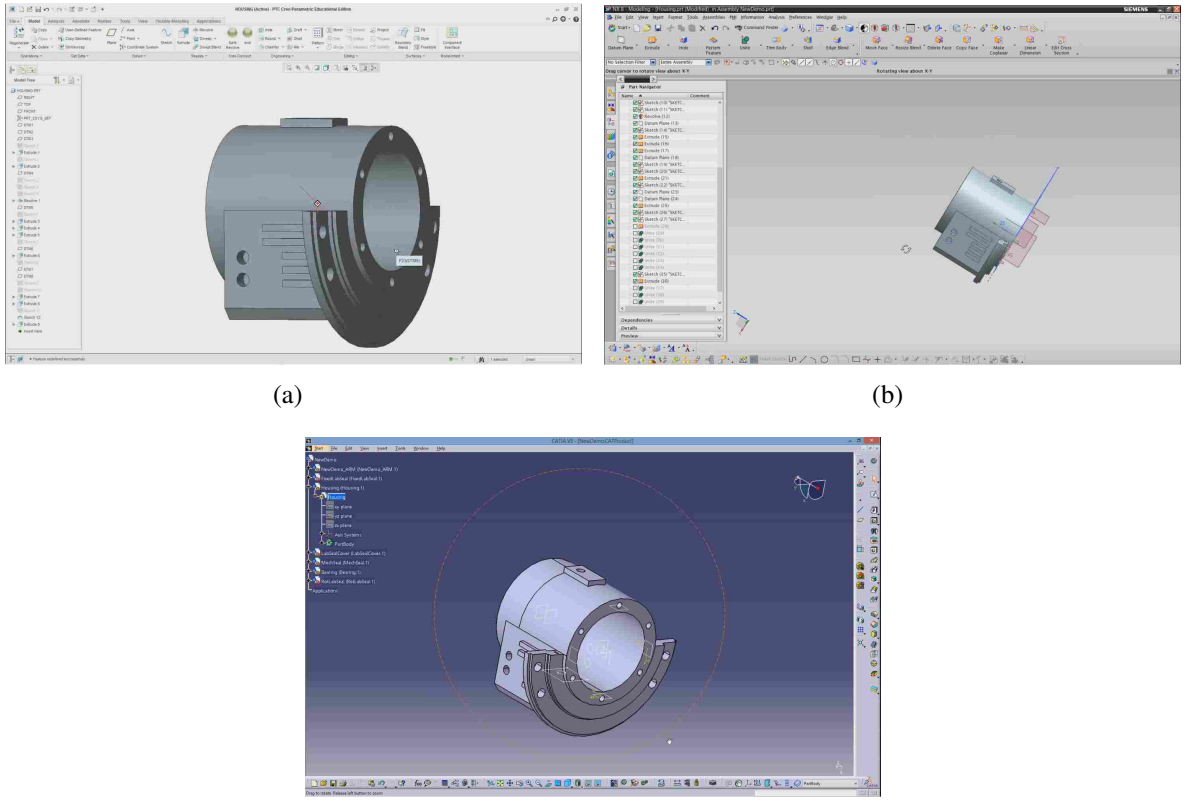


(a) (b)

(c)

Figure 4.13: Associative planes were used to define the location of features while modeling in Creo (a), NX (b) and CATIA (c)

To test the software's ability to exactly replicate models between CAD systems, the finished part model from each system was converted to a stereolithography (.stl) file often used for rapid prototyping. When exporting to this file type, each CAD system represents the part model by a triangular tessellation covering all surfaces of the model. The .stl files from each respective CAD system was then imported into a 3rd party analysis tool to extract the important model parameters found in Table 4.2. Though there are slight variations between the parameters, these are extremely minimal, with the largest percent difference coming from the volume calculation which is less than .095%. Even so, the differences in these parameters is most likely due to variations in how each CAD system creates the tessellations.



(a) (b)

(c)

Figure 4.14: Finished Water Pump Bearing Housing modeled using multi-user heterogeneous CAD as seen in Creo (a), NX (b) and CATIA (c)

4.6.2 Guided Model Rocket Assembly

The second multi-user assembly presented with this research tests the system's handling of multiple users ad multiple parts within an assembly. The challenge of this situation is to handle the constant stream of remote CAD operations from other users in a way as to not interfere with the local user's modeling process. In this example, a total of six users, two in CATIA, two in NX, and two in Creo, model a model rocket retrofitted with a simple guidance system. Components were divided amongst users in each system prior to modeling. Additionally, instructions for modeling were given to each user found in Appendix A. The model rocket assembly is actually an assembly made up of 10 unique parts seen in Table 4.3. Due to current architecture limitations in assembly component instance functionality, each component was modeled separately, ie. the servo motors were modeled four times in four configurations.

Table 4.3: Guided Model Rocket Components list

Guided Model Rocket Assembly	
Component	Quantity
Motor Sub-frame	1
Servo Motor	4
Guidance System	1
Nose Cone	1
Lower Fin	3
Upper Fin	4
Motor	1
Battery	1
Nozzle	1
Skin	1
IMU	1

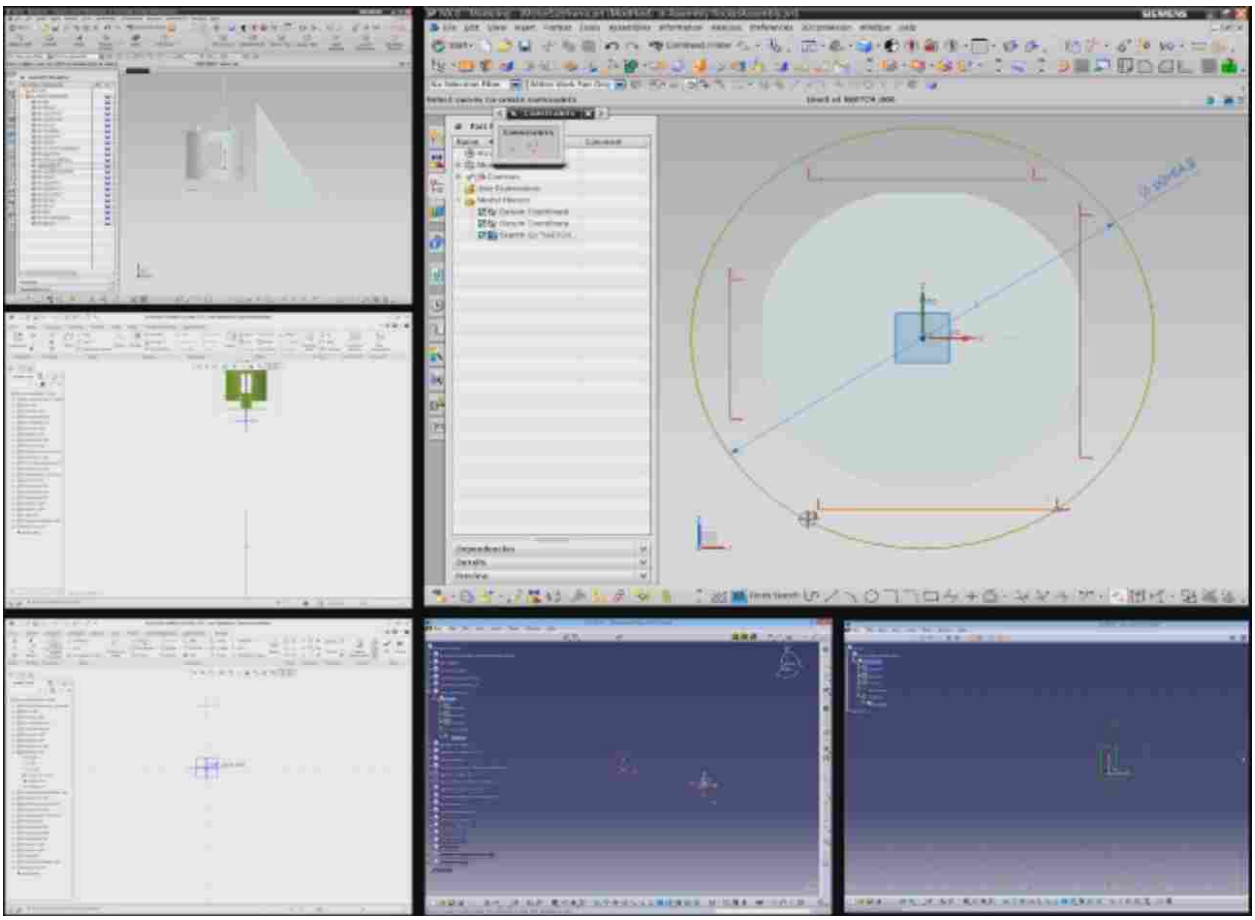


Figure 4.15: Interoperable Modeling Session between Six Heterogeneous CAD Clients

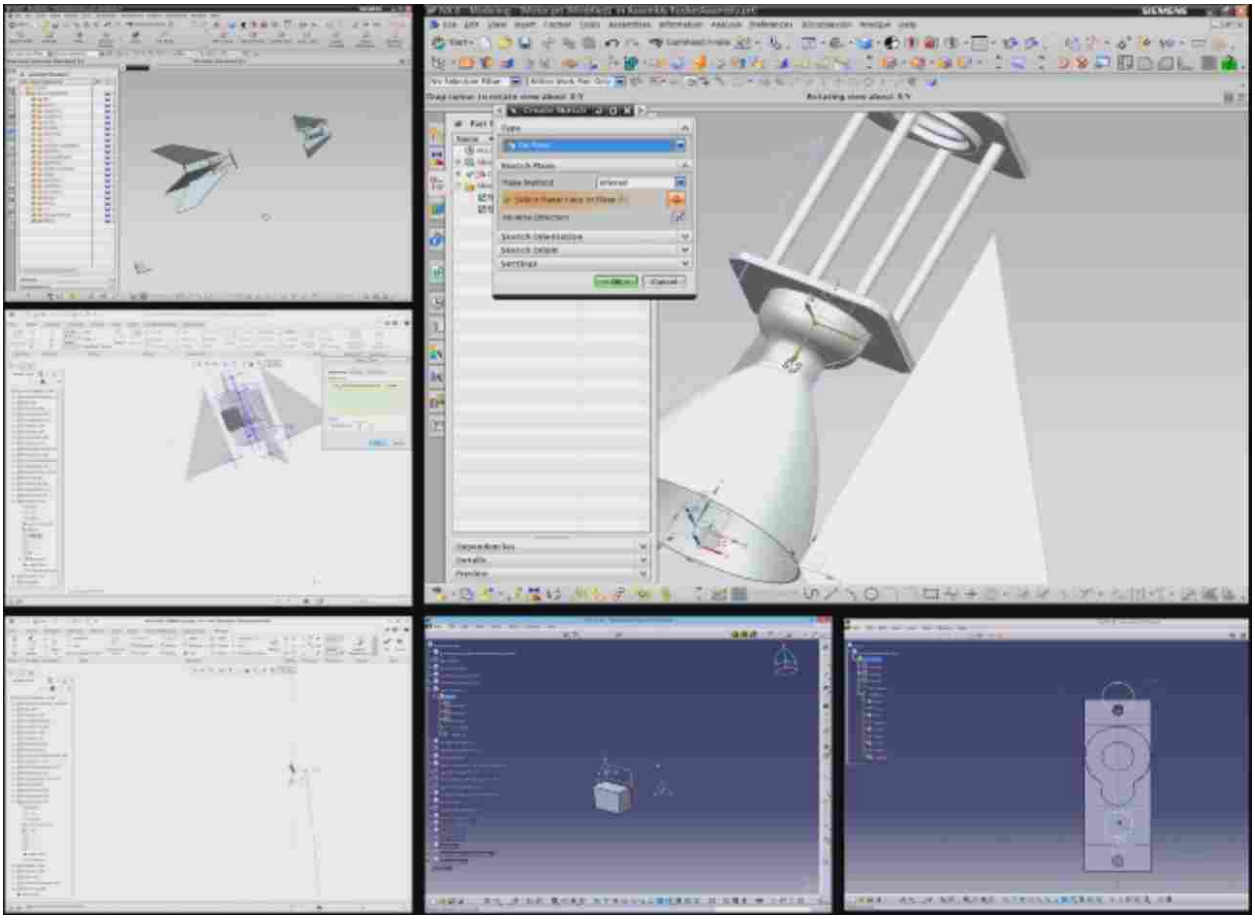


Figure 4.16: Components were modeled in place according to the rocket global coordinatesystem

While each client modeled their respective components, geometry created by remote users were automatically integrated into the local assembly. Figure 4.15 shows each client early in the modeling session. As users alternate between creating sketches and constructing geometry, remote user's work can be referenced and checked. This kind of awareness, akin to the collaboration around the drafting table, fosters interaction between designers to help and check each other where needed. While clients could see the operations performed by remote clients, each client worked alone on any particular component. This was important because it resulted in no operational conflicts - necessitated because of the lack of a conflict management implementation.

During this modeling session, coordinate systems were used extensively to define both location of objects and axes of rotation. Each part was modeled according to the rocket global coordinate system, so coordinate system features were often the first feature created and offset to the correct locations to define geometry. This can be seen in Figure 4.16 and is a common practice

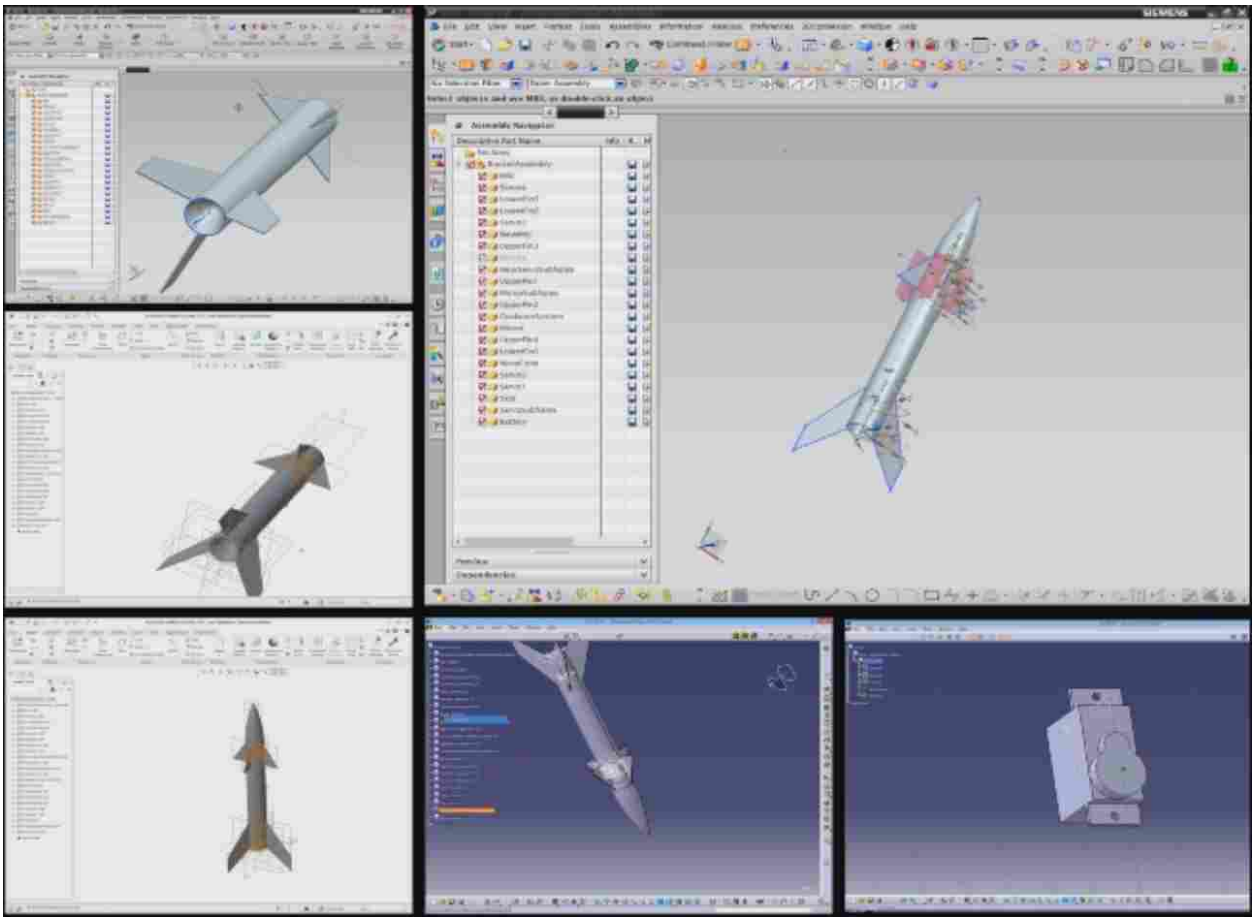


Figure 4.17: Image of finished rocket assembly modeled in a heterogeneous CAD system

with large assemblies in industry. A screen shot of the finished model rocket assembly is seen in Figure 4.17.

To handle the steady stream of remote CAD operations sent to the local client from the server, as was the case in this demonstration, remote operations are stored within a modeling queue. When the software determines it is 'safe' to apply the commands without interfering with the local user, the operations are performed in time-stamp order and the geometry is incorporated into the local CAD model. This happens when the user is no longer actively working on a feature. The modeling process described by this assembly demonstrates the software's ability to handle these messages without corrupting the local assembly model.

CHAPTER 5. SUMMARY AND CONCLUSION

The main objective of this research was to expand the neutral parametric canonical form (NPCF) developed by the NSF Center for e-Design, BYU Site to maintain the associativity of features while enforcing the data integrity of the format. The current implementation supports multi-user modeling and includes creation, editing, and deletion functionality between NX, CATIA, and Creo CAD Systems. Associative methods within the NPCF architecture now enable the design intent, implicitly defined by the designer, to be transferred between CAD systems. This is an important improvement to the previous implementation because many design parameters are incorporated directly into a CAD model during creation. With multi-user, collaborative CAD technologies enabling design sharing between people or groups geographically separated, these important parameters need to be shared for the proper interpretation of models.

Design intent was preserved in the NPCF standards by defining a schema allowing multiple definitions, or variations, of feature types. This allows modelers to select the feature creation type which will best define the part. Furthermore, an interface structure was designed and implemented to allow the NPCF standard to handle feature references like CAD software. These enhancements were implemented into Interop to support all previously defined features, as well as new datum features which were specifically chosen to test and showcase the added functionality. These methods enabled new CAD features to be incorporated into the NPCF neutral database. Coordinate system features, plane features, axis features, and extrude and revolve features were added to the system.

Part model edits performed in any client maintain the feature associations set during feature creation. This approach has achieved more unrestricted multi-user interaction than previous implementations because the design intent implicitly defined by the modeler is stored within the neutral format and translated to remote systems. This enhances model awareness between active clients within the collaborative session because CAD operations are performed in real-time on all users' screens. This improves the design process by reducing or eliminating the costly feedback

loops that occur when models are transferred between modelers or engineering groups. The solution developed in this research would be useful for a company such as Pratt & Whitney, whose US division uses NX and Canada division uses CATIA. Raytheon, which primarily uses Creo, could better interact with their suppliers using other systems.

The focus of this research has been to continue the original NPCF approach of defining a neutral standard instead of simply translating the data between CAD systems. This approach proved useful because data is stored in a database, easily accessible by other clients and facilitating multi-user collaborative processes. The NPCF approach, and the enhancements at the focus of this research, were funded by several industry leaders and its effectiveness been validated in many tests and demonstrations.

While the software resulting from this research is much improved over its predecessor in both the number of supported CAD systems, features, and functionality, further research is needed to neutralize other areas where design intent is implicitly stored. Sketch and assembly constraints are used to define the size locations of geometry. In the current implementation, these constraints are ignored and only the geometric data resulting from the constraints are neutralized and sent to other clients. Research into defining a neutral constraint will continue to improve transfer of design intent. Finally, referencing Boundary REPresentations (BREPs) such as bodies, faces, and edges has not yet been implemented and is a requirement for new features to be supported into the NPCF definitions. Each CAD system names and identifies these objects differently and a neutralized form is non-trivial. In spite of these limitations, the existing solution is effective and should continue to be improved and expanded.

REFERENCES

[1] Leach, L. M., 1983. "Language interface for data exchange between heterogeneous cad/cam databases.." *Dissertation Abstracts International Part B: Science and Engineering[DISS. ABST. INT. PT. B- SCI. & ENG.],,* **44**(5). 1

[2] Wisnosky, D. E., 1977. Icam program prospectus Tech. rep., DTIC Document. 1

[3] Boucher, M., 2010. "Working with multi-cad.". 1

[4] Brunnermeier, S. B., and Martin, S. A., 2002. "Interoperability costs in the us automotive supply chain." *Supply Chain Management: An International Journal,* **7**(2), pp. 71–82. 2

[5] Freeman, R. S., 2015. "Neutral parametric canonical form for 2d and 3d wireframe cad geometry.". 5, 19, 46

[6] 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. 10

[7] 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. 10

[8] Ramani, K., Agrawal, A., Babu, M., and Hoffmann, C., 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. 10

[9] 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. 10

[10] 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. 10

[11] 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. 10

[12] 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. 10, 11

[13] 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. 10, 11

[14] Stork, A., and Jasnoch, U., 1997. "A collaborative engineering environment." In *Proceedings of the TeamCAD97 Workshop on Collaborative Design*, pp. 25–33. 11

[15] 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*. 11

[16] Nam, T.-J., and Wright, D. K., 1998. "Collide: A shared 3d workspace for cad." In *1998 Conference on Network Entities, Leeds, UK*. 11

[17] 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. 11

[18] 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*, Vol. 6, pp. 131–137. 11

[19] 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. 11

[20] 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. 11

[21] 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. 11

[22] Basu, D., and Kumar, S. S., 1995. "Importing mesh entities through iges/pdes." *Advances in Engineering Software,* **23**(3), pp. 151–161. 12

[23] Marjudi, S., Amran, M. M., Abdullah, K. A., Widyarto, S., Majid, N. A., and Sulaiman, R., 2010. "A review and comparison of iges and step." In *Proceedings Of World Academy Of Science, Engineering And Technology*, Vol. 62, pp. 1013–1017. 12

[24] Haenisch, J., 1990. "Cad-exchange-towards a first step implementation." In *Industrial Electronics Society, 1990. IECON'90., 16th Annual Conference of IEEE*, IEEE, pp. 734–739. 12

[25] Pratt, M., 1998. "Extension of the standard iso10303 (step) for the exchange of parametric and variational cad models." *CDROM Proceedings of the Tenth International IFIP WG,* **5**(5.3). 12

[26] Ferris, S., 2005. "Cad export options.." *Cadalyst,* **22**(12), pp. 22 – 28. 12

[27] Eigner, M., Handschuh, S., and Gerhardt, F., 2010. "Concept to enrichen lightweight, neutral data formats with cad-based feature technology." *Computer-Aided Design and Applications,* **7**(1), pp. 89–99. 12, 13

[28] Choi, G.-H., Mun, D., and Han, S., 2002. "Exchange of cad part models based on the macro-parametric approach." *International Journal of CAD/CAM,* **2**(1), pp. 13–21. 13

[29] Mun, D., Han, S., Kim, J., and Oh, Y., 2003. "A set of standard modeling commands for the history-based parametric approach." *Computer-aided design,* **35**(13), pp. 1171–1179. 13, 19

[30] Li, M., Gao, S., Li, J., and Yang, Y., 2004. "An approach to supporting synchronized collaborative design within heterogeneous cad systems." In *ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. 511–519. 13

[31] Li, M., Gao, S., and Wang, C. C., 2007. "Real-time collaborative design with heterogeneous cad systems based on neutral modeling commands." *Journal of Computing and Information Science in Engineering,* **7**(2), pp. 113–125. 13

[32] Hepworth, A., Tew, K., Trent, M., Ricks, D., Jensen, C. G., and Red, W. E., 2014. "Model consistency and conflict resolution with data preservation in multi-user computer aided design." *Journal of Computing and Information Science in Engineering,* **14**(2), p. 021008. 20

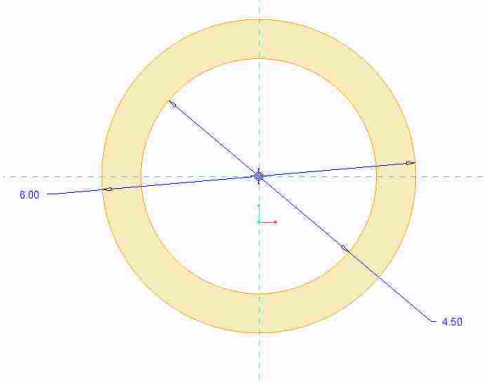
[33] Hepworth, A. I., Tew, K., Nysetvold, T., Bennett, M., and Greg Jensen, C., 2014. "Automated conflict avoidance in multi-user cad." *Computer-Aided Design and Applications,* **11**(2), pp. 141–152. 20

APPENDIX A. MODELING INSTRUCTIONS

A.1 Water Pump Thrust Bearing Instructions

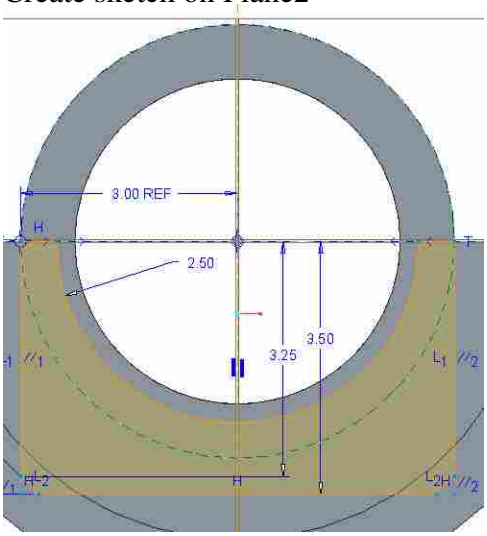
A.1.1 Thrust Bearing Housing

1. Create a plane offset 6” along z-axis
2. Create a sketch on Base XY Plane



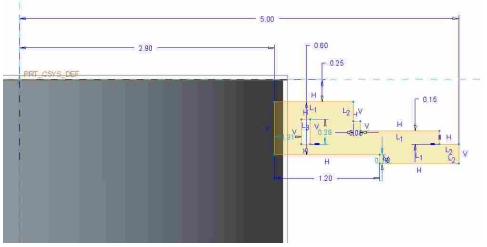
3. Create plane offset .75” from base XY plane along z-axis (Plane2)
4. Create plane offset .75” from plane 1 along negative z-axis (Plane3)

5. Create sketch on Plane2



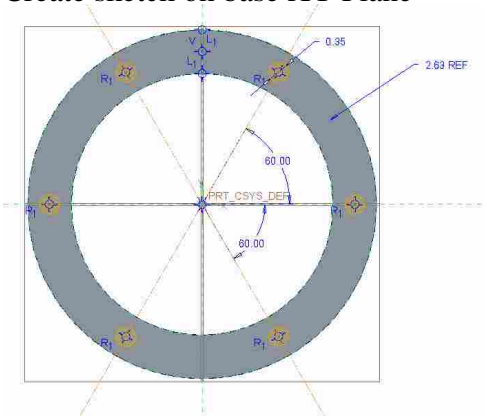
6. Extrude sketch to Plane3

7. Create sketch on Base XZ plane



8. Revolve sketch around part z-axis on bottom half only

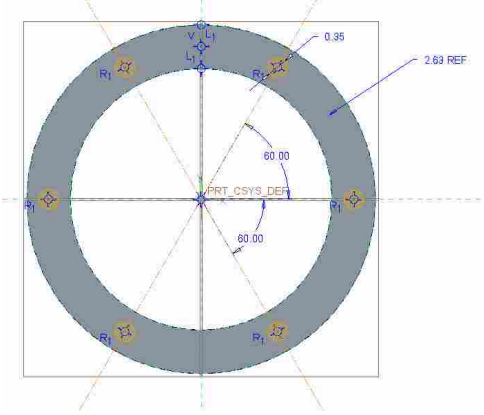
9. Create sketch on base XY Plane



10. Extrude holes through part (subtract)

11. Create plane offset 2.5” from base YZ-Plane along positive x-axis (Plane4)

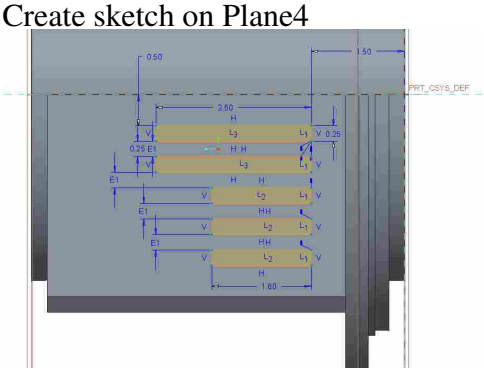
12. Create sketch on Plane4



13. Extrude sketch .75”

14. Create same sketch and extrude on opposite side

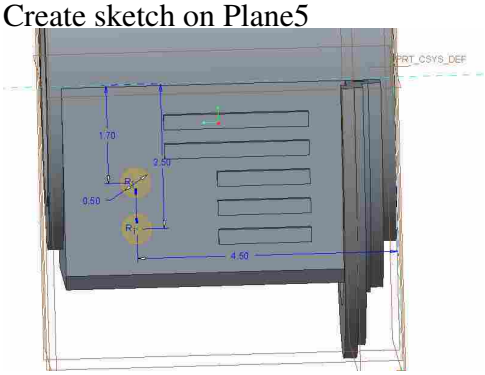
15. Create sketch on Plane4



16. Extrude sketch out (subtract)

17. Create plane offset 2.5” from base XZ-Plane along positive y-axis (Plane5)

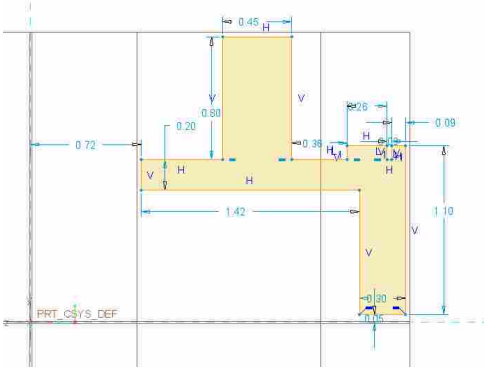
18. Create sketch on Plane5



19. Extrude sketch .75”

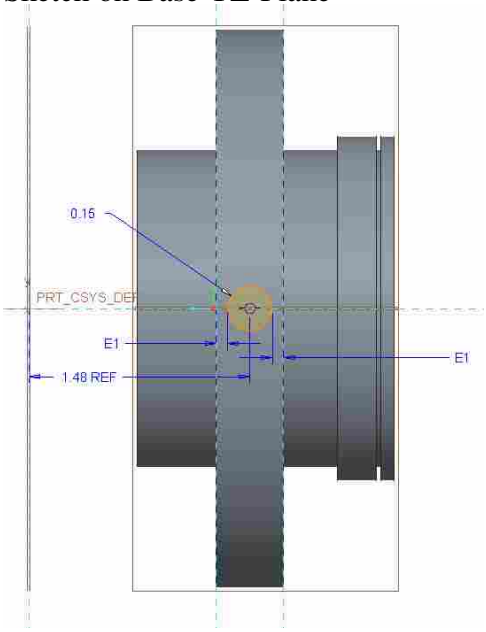
A.1.2 Mechanical Seal

1. Create a sketch on Base YZ-Plane



2. Revolve sketch around Part Z-Axis

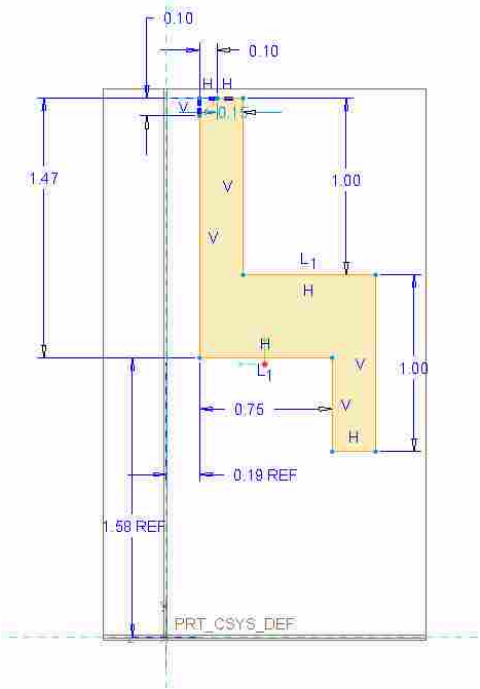
3. Sketch on Base YZ-Plane



4. Extrude along positive X-Axis through part (Subtract)

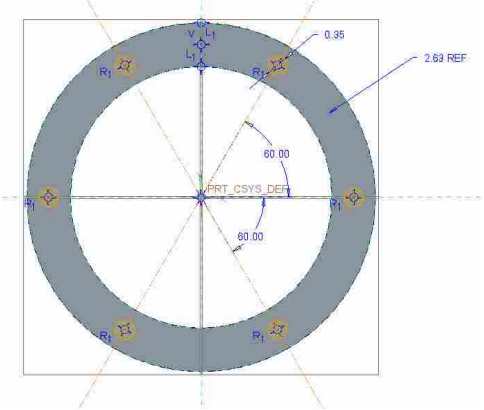
A.1.3 Labyrinth Seal Cover

1. Create a sketch on base YZ-Plane



2. Revolve sketch around part Z-Axis

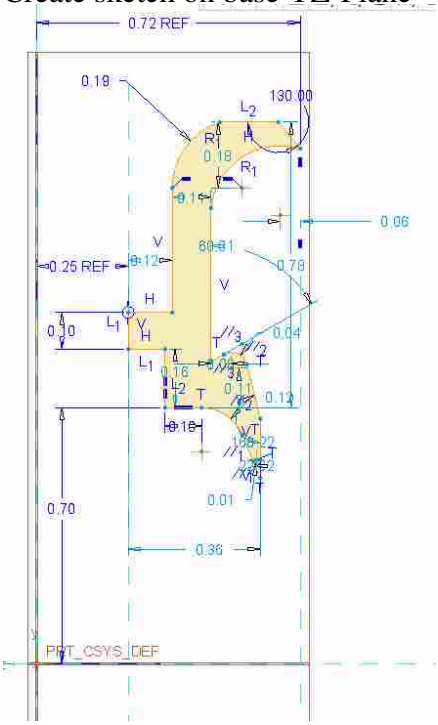
3. Create sketch on base XY-Plane



4. Extrude sketch through part (subtract)

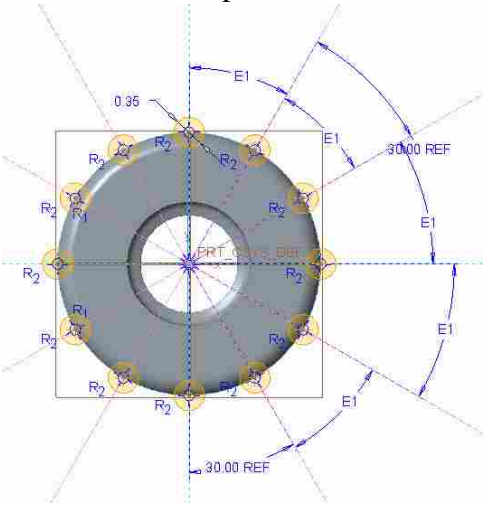
A.1.4 Rotating Labyrinth Seal

1. Create sketch on base YZ-Plane



2. Revolve sketch about part Z-Axis

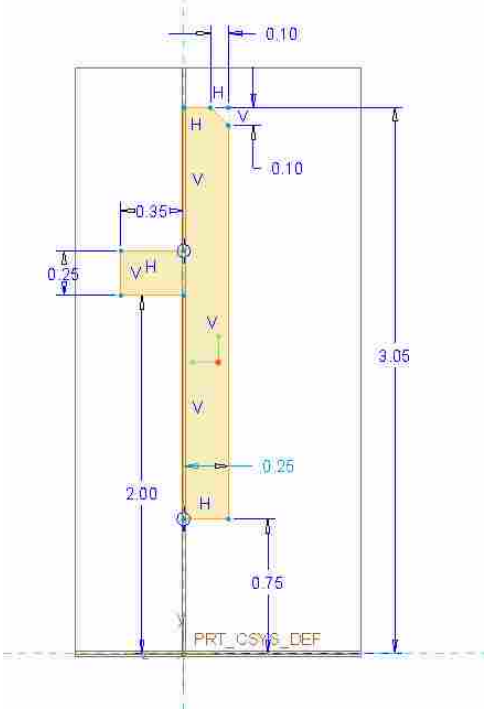
3. Create sketch on part XY-Plane



4. Extrude along part Z-Axis through part (subtract)

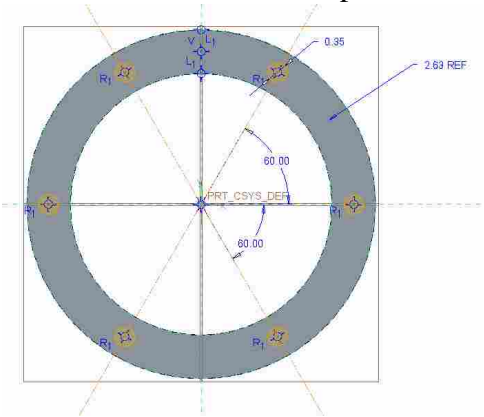
A.1.5 Fixed Labyrinth Seal

1. Create sketch on base YZ-plane



2. Revolve sketch about part Z-Axis

3. Create sketch on base XY-plane

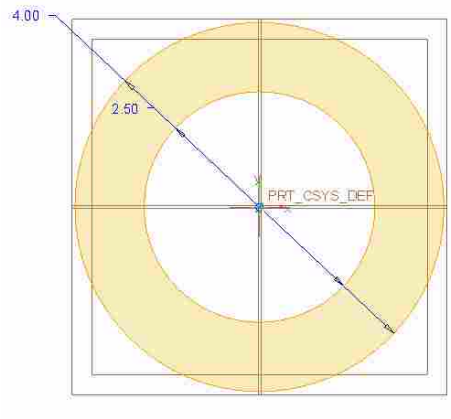


4. Extrude sketch through part (subtract)

A.1.6 Ball Bearing

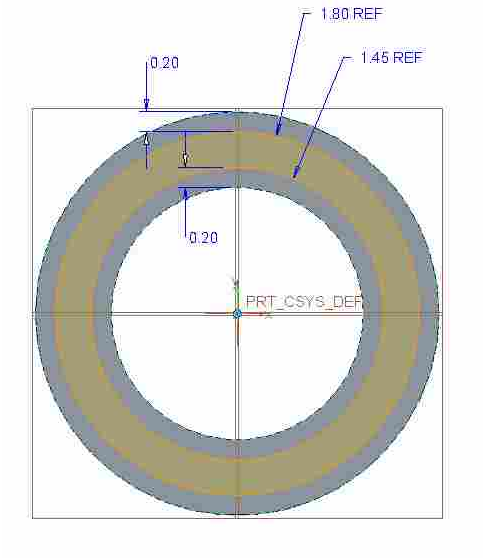
1. Offset plane 2.5 from base XY-plane along positive z-axis (PLANE 1)

2. Create sketch on base XY-plane



3. Extrude sketch to PLANE1

4. Create sketch on base XY-plane



5. Extrude .5 along positive z-axis

6. Create same sketch and extrude from PLANE 1

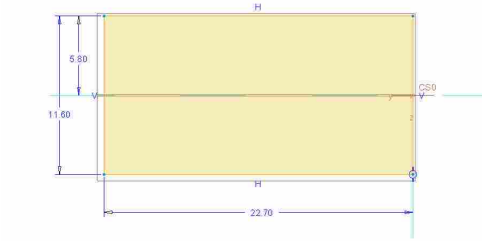
A.2 Guided Model Rocket Assembly Instructions

A.2.1 Servo Motors

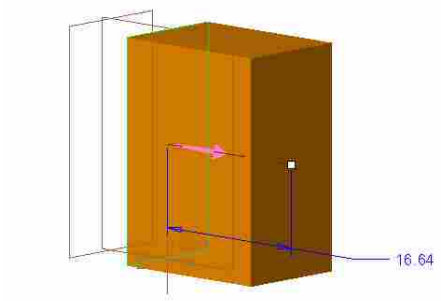
1. Create Datum CSYS

Datum CSYS Coordinates			
	X	**Y**	**Z**
Servo 1	7.6 mm	392.6 mm	0 mm
Servo 2	0 mm	392.6 mm	-7.6 mm
Servo 3	-7.6 mm	392.6 mm	0 mm
Servo 4	0 mm	392.6 mm	7.6 mm

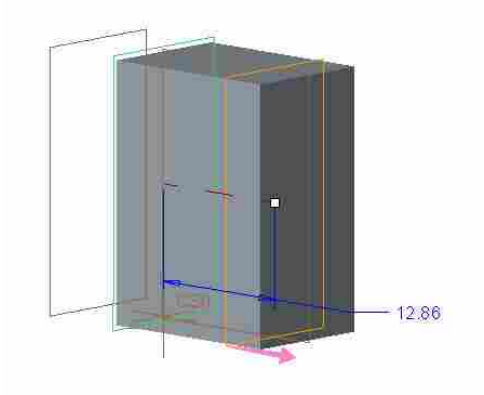
2. Create Sketch



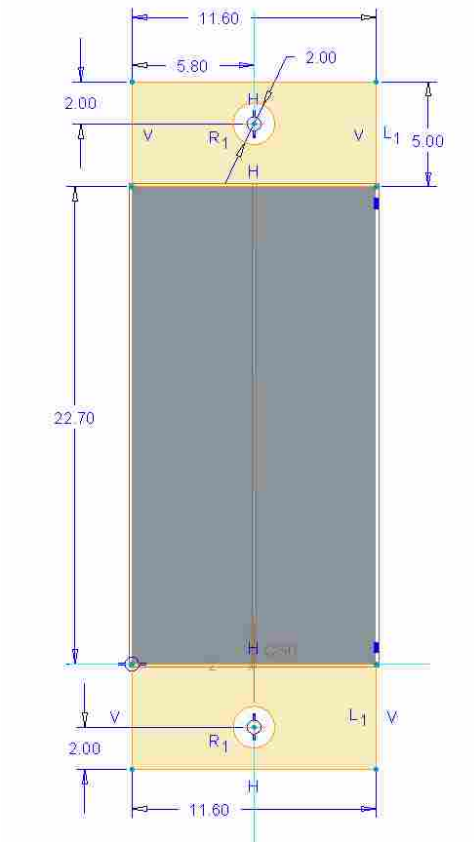
3. Extrude Sketch 16.64 mm



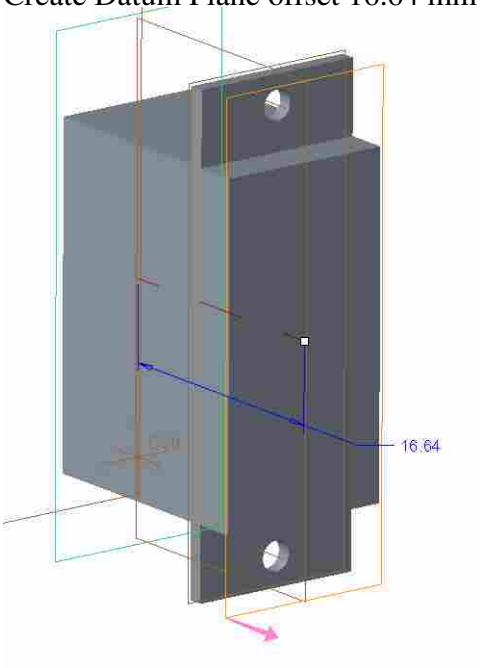
4. Create Datum Plane offset 12.86 mm



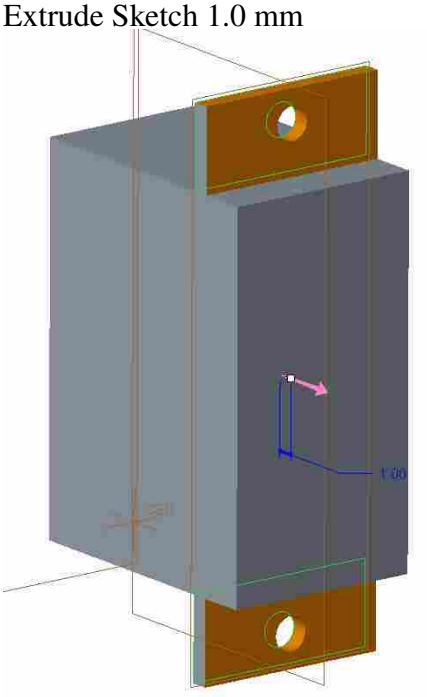
5. Create Sketch



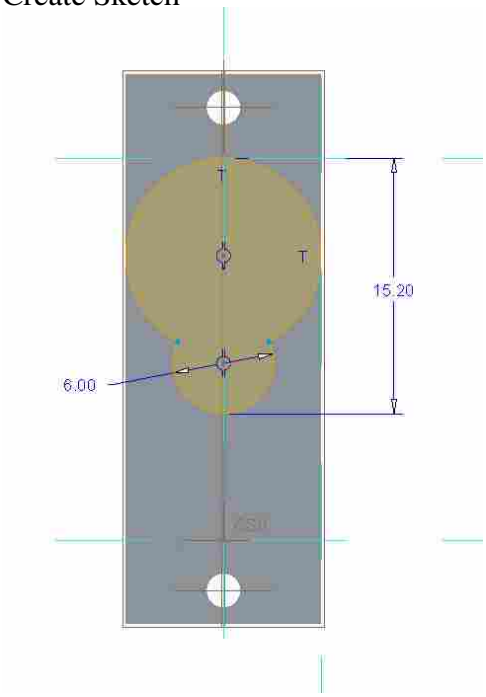
7. Create Datum Plane offset 16.64 mm



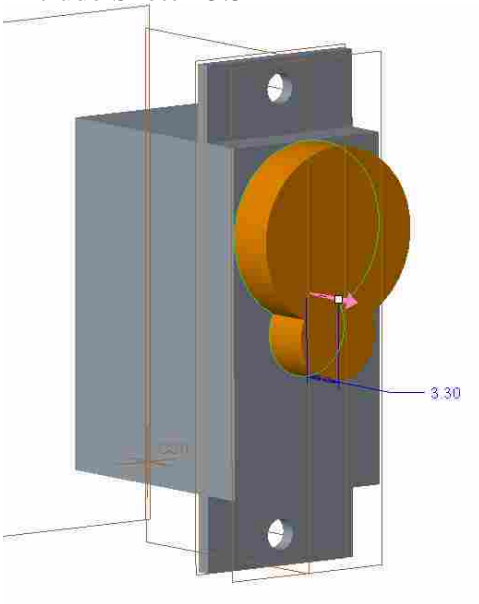
6. Extrude Sketch 1.0 mm



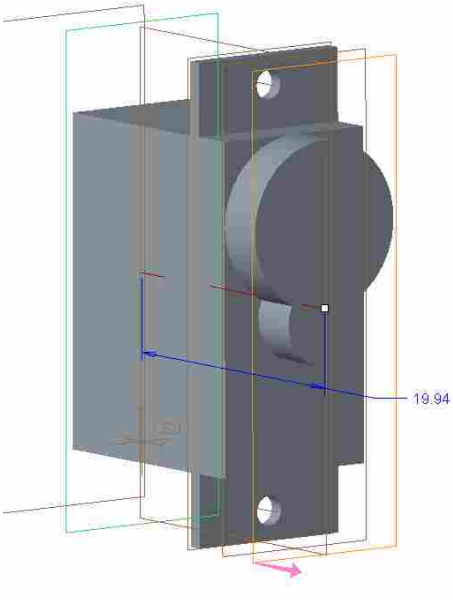
8. Create Sketch



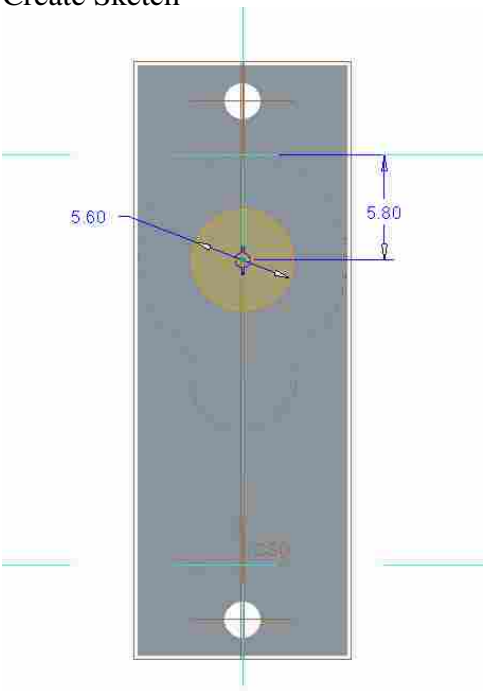
9. Extrude Sketch 3.3 mm



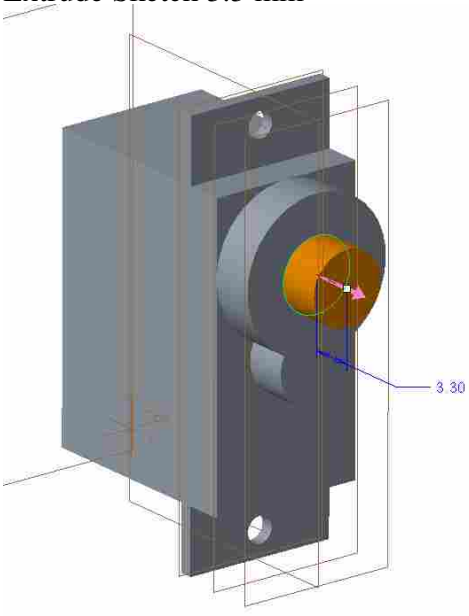
10. Create Datum Plane offset 19.94 mm



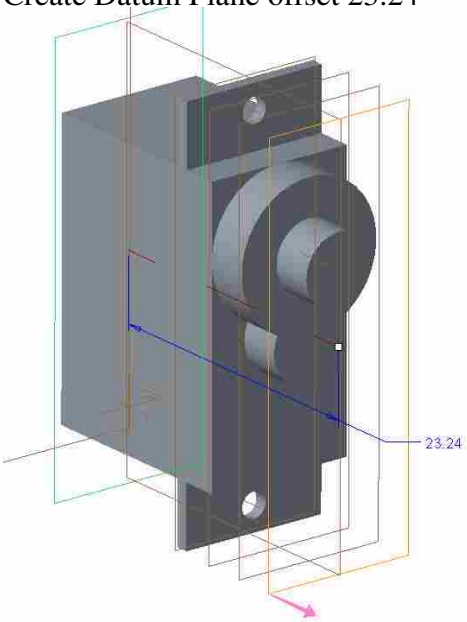
11. Create Sketch



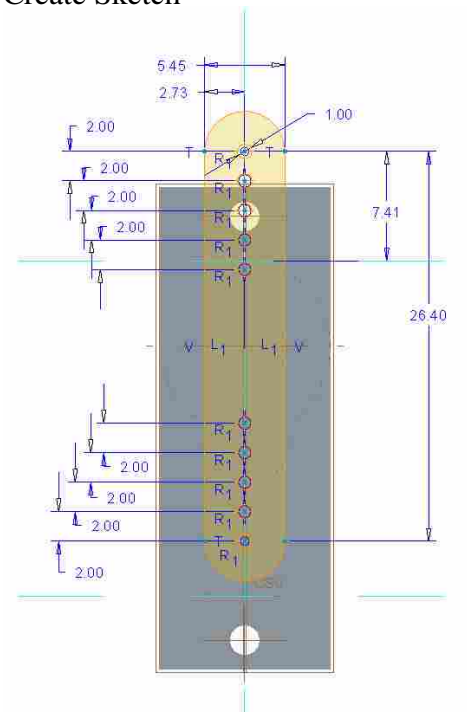
12. Extrude Sketch 3.3 mm



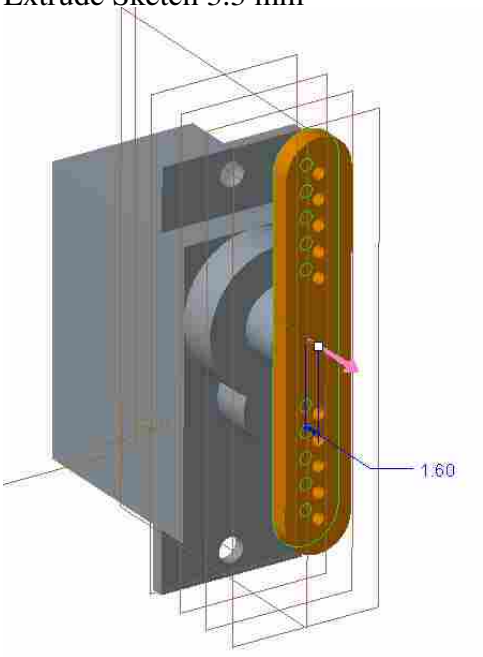
13. Create Datum Plane offset 23.24



14. Create Sketch



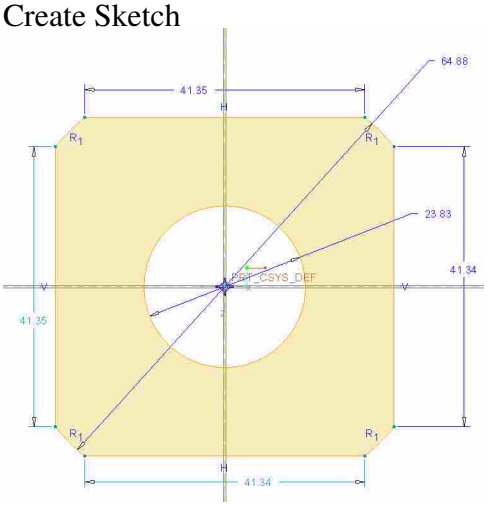
15. Extrude Sketch 3.3 mm



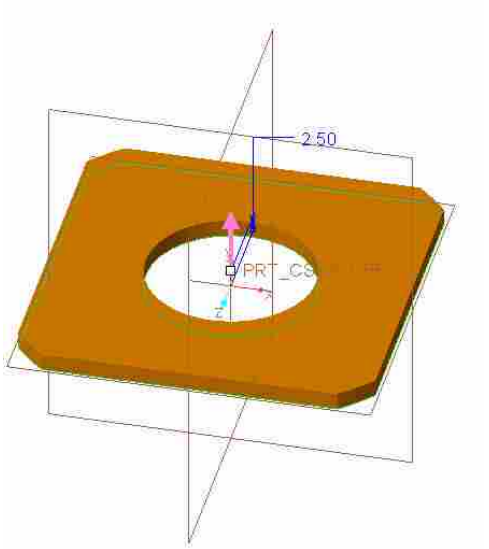
A.2.2 Motor Subframe

1. Create Datum CSYS offset 75.0 mm along Y-Axis

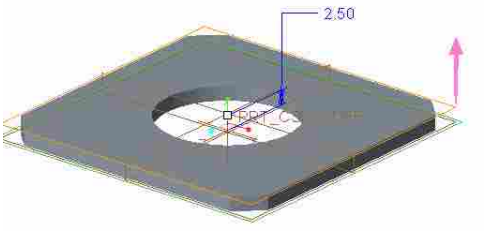
2. Create Sketch



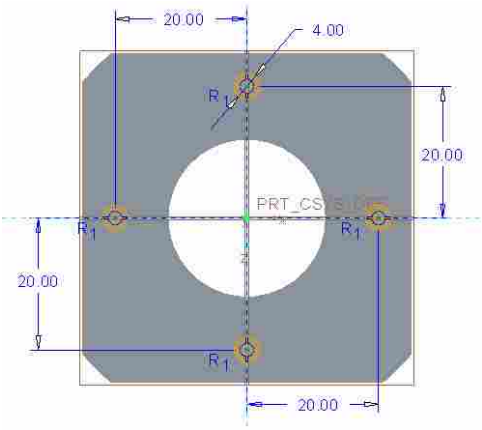
3. Extrude Sketch 2.5 mm



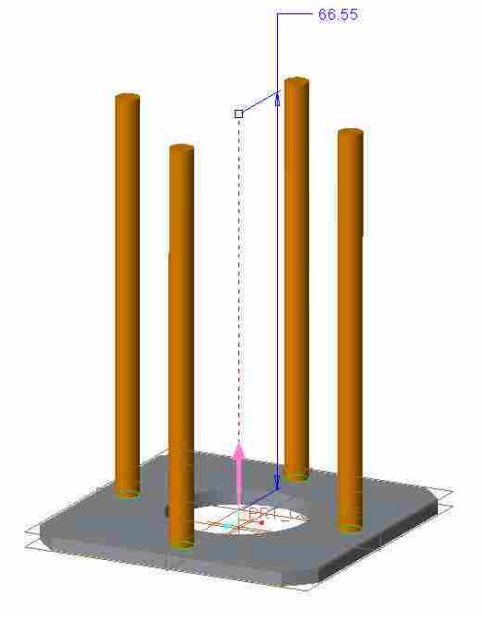
4. Create Datum Plane offset 2.5 mm



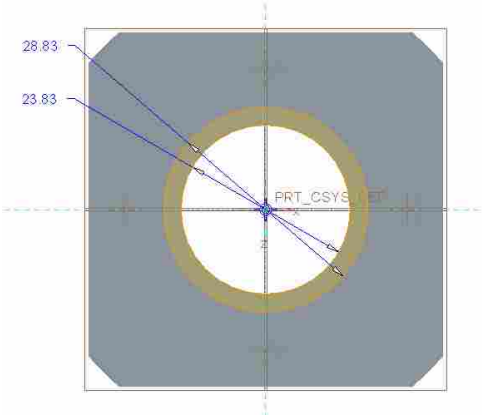
5. Create Sketch



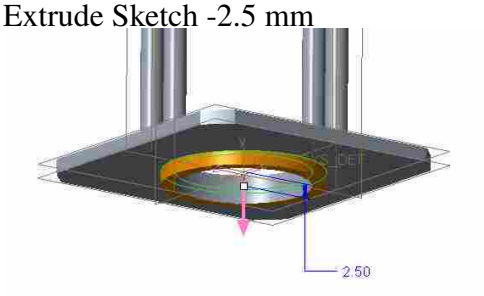
6. Extrude Sketch 66.55 mm



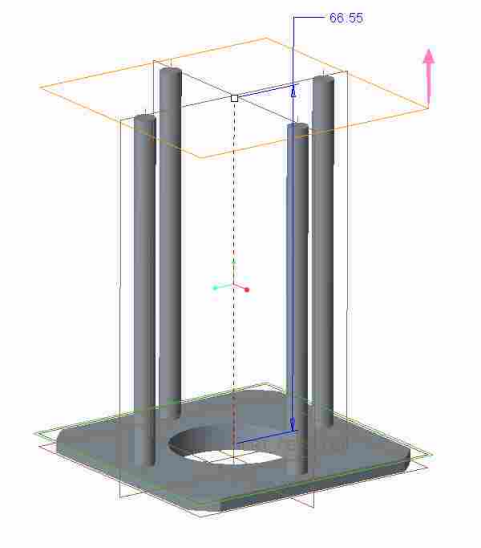
7. Create Sketch



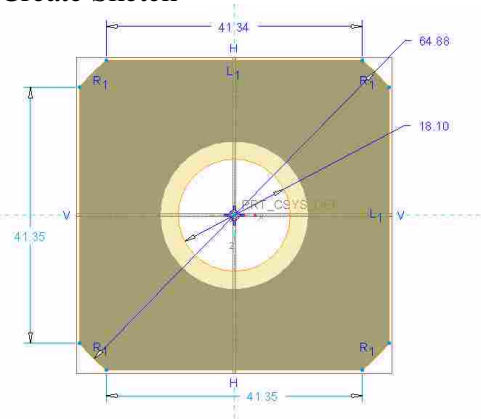
8. Extrude Sketch -2.5 mm



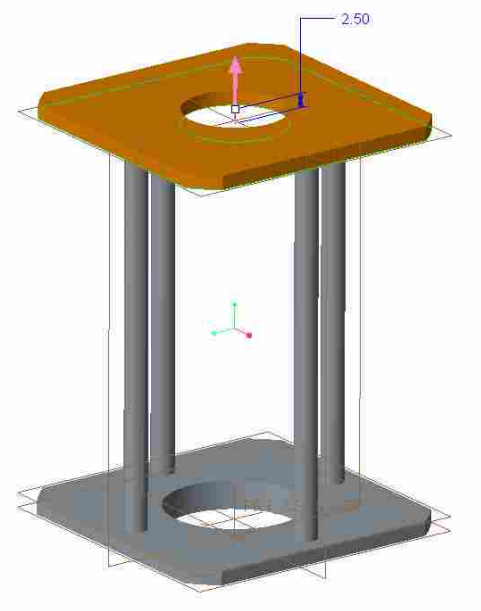
9. Create Datum Plane offset 66.55 mm



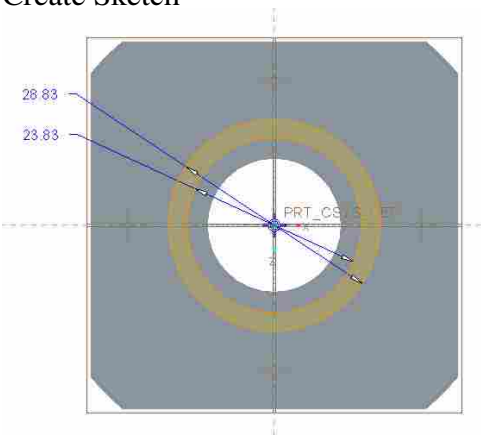
10. Create Sketch



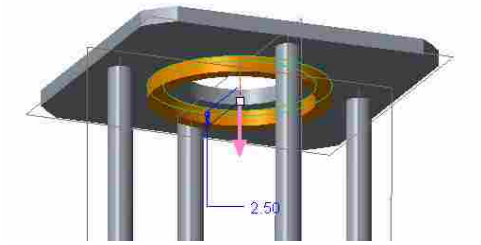
11. Extrude Sketch 2.5 mm



12. Create Sketch



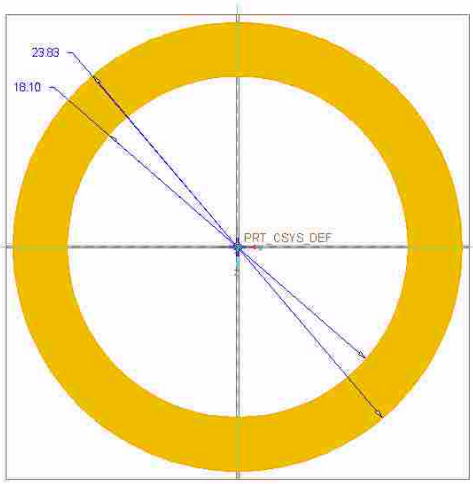
13. Extrude Sketch -2.5 mm



A.2.3 Motor

1. Create Datum CSYS offset 72.5 mm along Y-Axis

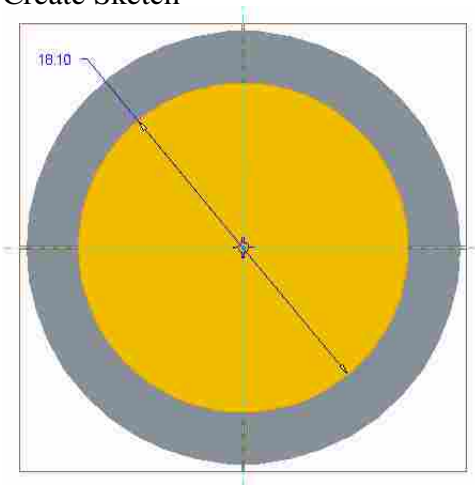
2. Create Sketch



3. Extrude sketch 69.05 mm

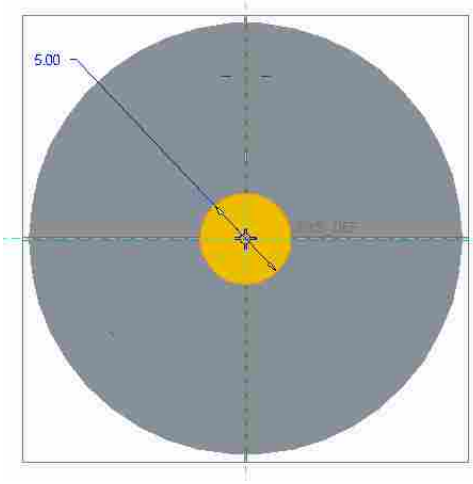
4. Create Datum Plane offset 4.0 mm

5. Create Sketch



6. Extrude Sketch 61.72 mm

7. Create Sketch



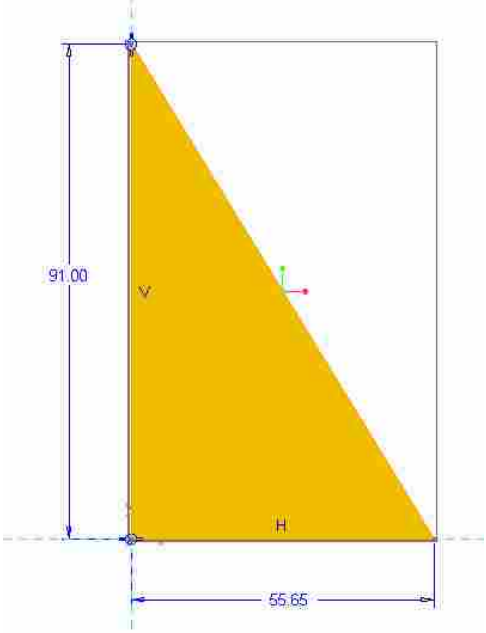
8. Extrude Sketch (subtract) 30 mm

A.2.4 Upper Fins

1. Create Datum CSYS

Datum CSYS Coordinates			
	X	**Y**	**Z**
Fin 1	32.44 mm	359.0 mm	0 mm
Fin 2	0 mm	359.0 mm	-32.44 mm
Fin 3	-32.44 mm	359.0 mm	0 mm
Fin 4	0 mm	359.0 mm	32.44 mm

2. Create Sketch



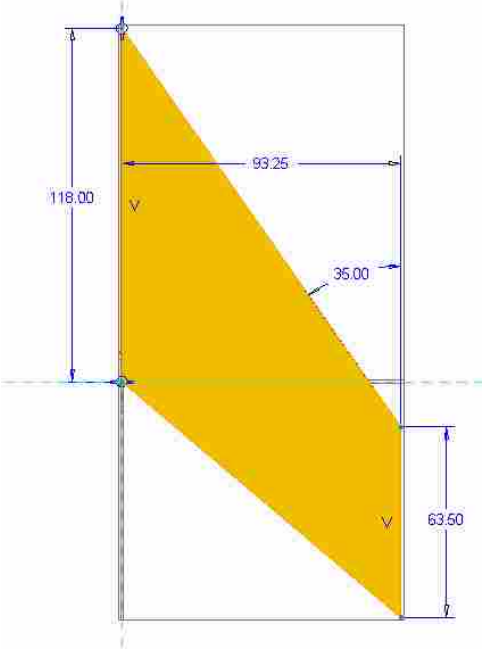
3. Extrude Sketch 1.28 mm in both directions

A.2.5 Lower Fins

1. Create Datum CSYS

Datum CSYS Coordinates			
	X	**Z**	**About Y**
Fin 1	32.44 mm	0 mm	0 degrees
Fin 2	-16.22 mm	28.09 mm	30 degrees
Fin 3	-16.22 mm	28.09 mm	-30 degrees

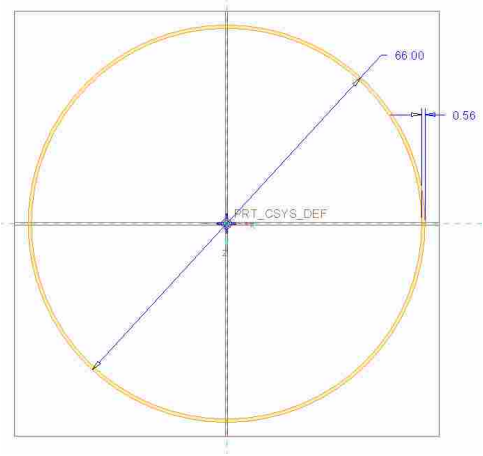
2. Create Sketch



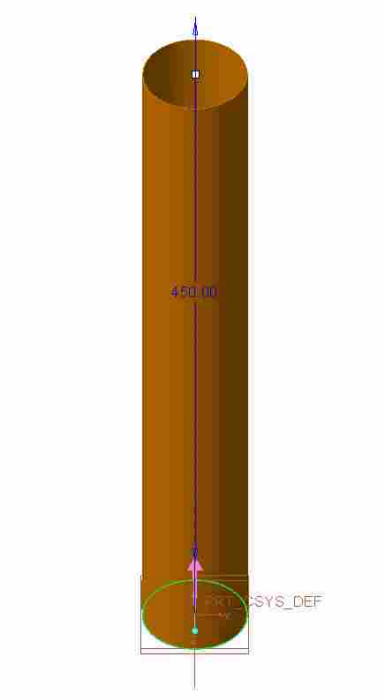
3. Extrude Sketch 0.94 mm in both directions

A.2.6 Rocket Skin

1. Create Sketch on Top plane



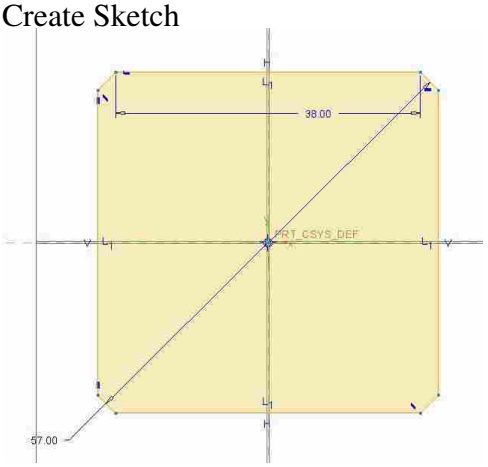
2. Extrude Sketch 450 mm



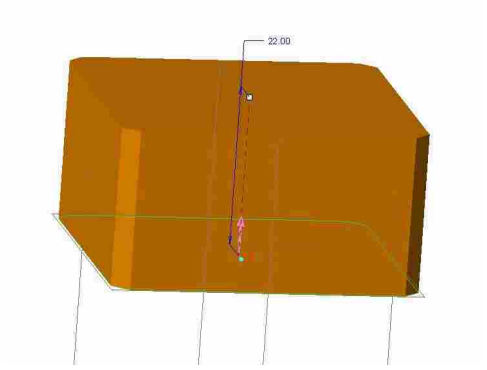
A.2.7 IMU

1. Create Datum CSYS offset 429.8 mm along y-axis

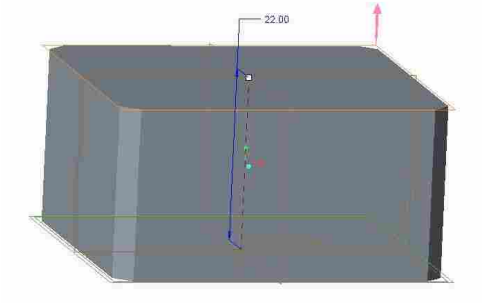
2. Create Sketch



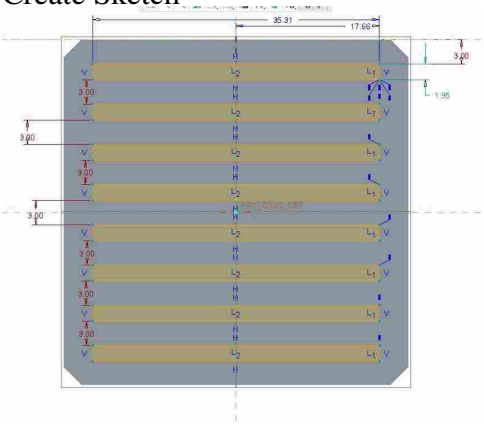
3. Extrude Sketch 22.0 mm



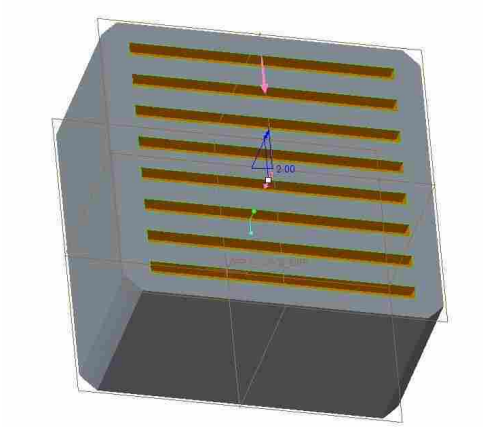
4. Create Datum Plane offset 22.0 mm



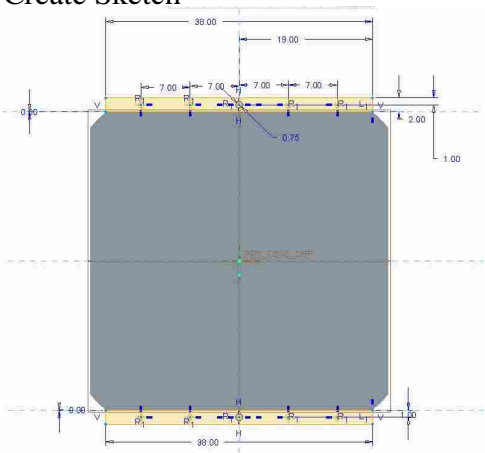
5. Create Sketch



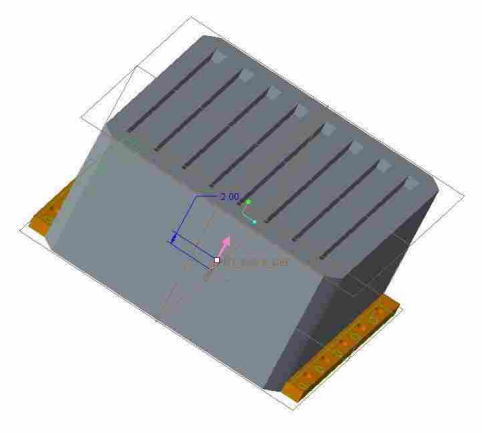
6. Extrude Sketch 2.0 mm (subtract)



7. Create Sketch

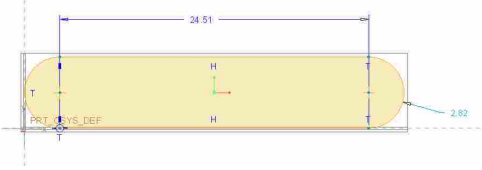


8. Extrude Sketch 2.0 mm



A.2.8 Battery

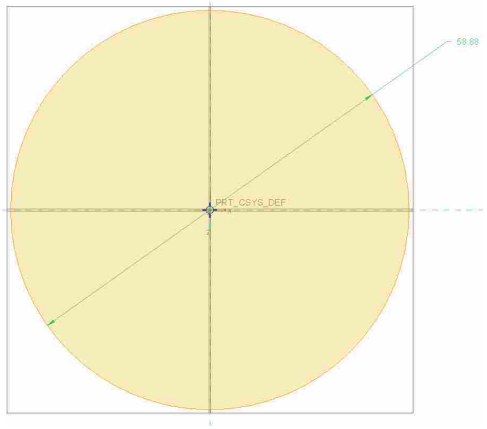
1. Create Datum CSYS offset 374.6 mm along y-axis
2. Create Sketch



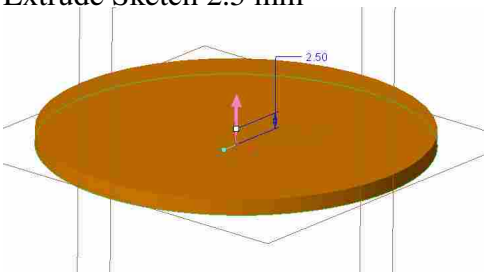
3. Extrude Sketch 23.5 mm in both directions

A.2.9 Guidance System

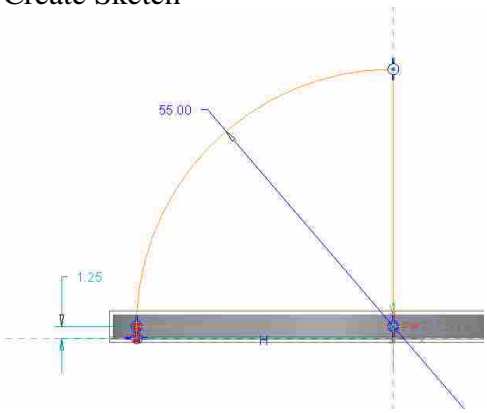
1. Create Datum CSYS offset 499.48 mm along y-axis
2. Create Sketch



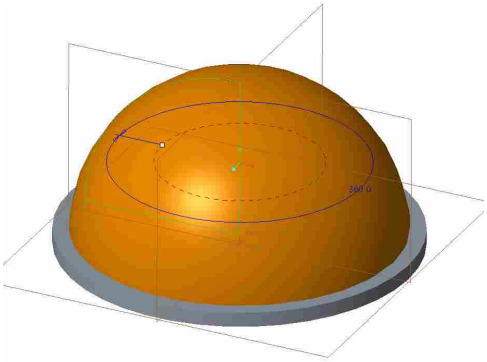
3. Extrude Sketch 2.5 mm



4. Create Sketch

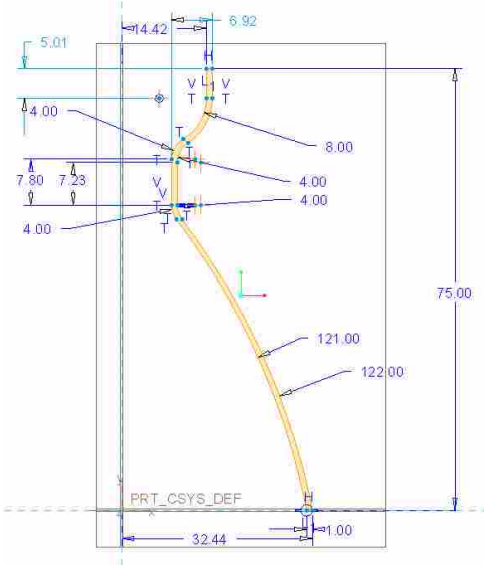


5. Revolve Sketch about y-axis 360 degrees

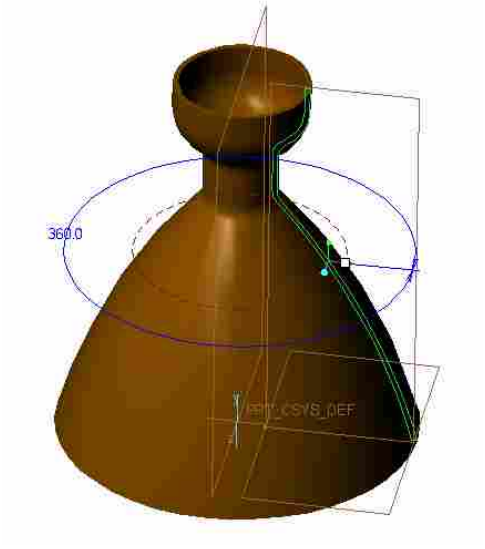


A.2.10 Nozzle

1. Create Sketch

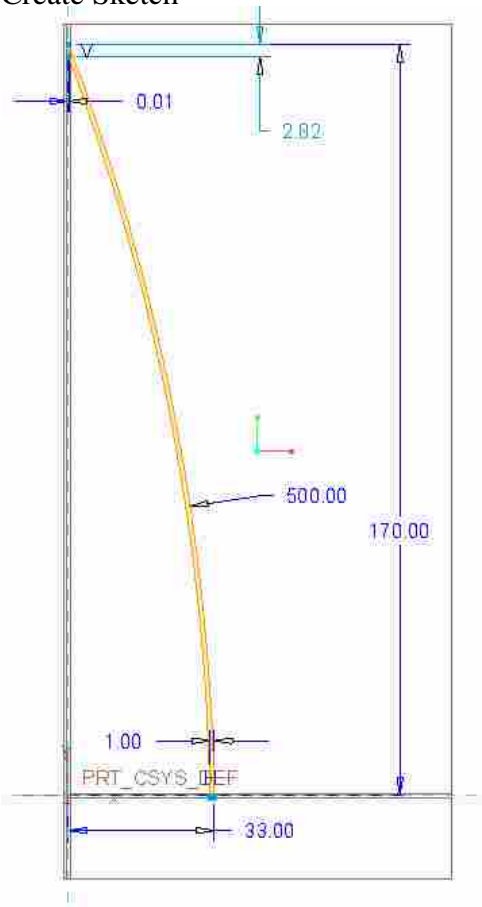


2. Revolve Sketch about y-axis 360 degrees



A.2.11 Nose Cone

1. Create Datum CSYS offset 450 mm along y-axis
2. Create Sketch



3. Revolve Sketch about y-axis 360 degrees