2015-08-01

Neutral Parametric Canonical Form for 2D and 3D Wireframe CAD Geometry

Robert Steven Freeman
Brigham Young University - Provo

Neutral Parametric Canonical Form for 2D and 3D Wireframe CAD Geometry

Robert Steven Freeman

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
C. Greg Jensen
Steven E. Gorrell

Department of Mechanical Engineering

Brigham Young University

August 2015

ABSTRACT

Neutral Parametric Canonical Form for 2D and 3D Wireframe CAD Geometry

Robert Steven Freeman
Department of Mechanical Engineering, BYU
Master of Science

The challenge of interoperability is to retain model integrity when different software applications exchange and interpret model data. Transferring CAD data between heterogeneous CAD systems is a challenge because of differences in feature representation. A study by the National Institute for Standards and Technology (NIST) performed in 1999 made a conservative estimate that inadequate interoperability in the automotive industry costs them $1 billion per year. One critical part of eliminating the high costs due to poor interoperability is a neutral format between heterogeneous CAD systems. An effective neutral CAD format should include a current-state data store, be associative, include the union of CAD features across an arbitrary number of CAD systems, maintain design history, maintain referential integrity, and support multi-user collaboration. This research has focused on extending an existing synchronous collaborative CAD software tool to allow for a neutral, current-state data store. This has been accomplished by creating a Neutral Parametric Canonical Form (NPCF) which defines the neutral data structure for many basic CAD features to enable translation between heterogeneous CAD systems. The initial architecture developed begins to define a new standard for storing CAD features neutrally. The NPCF's for many features have been implemented in a multi-user interoperability program and work between NX and CATIA CAD systems. The 2D point, 2D line, 2D arc, 2D circle, 2D spline, 3D point, extrude, and revolve NPCF's will be specifically defined. Complex models have successfully been modeled and exchanged in real time and have validated the NPCF approach. Multiple users can be in the same part at the same time in different CAD systems and create and update models in real time.

Keywords: interoperability, heterogeneous CAD, CAD features, design history, neutral format, multi-user CAD, design cycle, collaboration, asynchronous messaging, MUObject, client-server architecture

ACKNOWLEDGMENTS

I express thanks to my wife, Michelle Freeman, for being my motivation and support through this long process. I also give thanks to my professors, Dr. Ed Red, Dr. C. Greg Jensen, and Dr. Steve Gorrell for guiding me in the right direction and ensuring that I completed my thesis successfully. The interoperability team in the BYU PACE lab, Eric Bowman, Josh Coburn, Dan Staves, Bryce DeFigueiredo, Scott Christensen, Alex La, and the interoperability software testers deserve a lot of credit for making this research possible. I give a special thanks to all my other family members and professors for their encouragement and support.

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, including methods and tools
$CSYS$	Coordinate System
GUI	Graphical User Interface
$GUID$	Global Unique Identifier
IAB	Industry Advisory Board
$IGES$	Initial Graphics Exchange Specification
JT	Lightweight Neutral Data Format by Siemens PLM Software
MU	Multi-User
$NIST$	National Institute for Standards and Technology
NMC	Neutral Modeling Command
$NPCF$	Neutral Parametric Canonical Form
NSF	National Science Foundation
ORM	Object-Relational Mapper
PLM	Product Lifecycle Management, including methods and tools
$SMCH$	Solid Model Construction History, latest development with STEP
$STEP$	Standard for the Exchange of Product model data
UFO	Universal Features Object
UML	Unified Modeling Language
UPR	Universal Product Representation
XML	Extensible Markup Language

CHAPTER 1. INTRODUCTION

The use of CAD software is central to product design and begins in the early stages of development. It allows a product to be visualized in 3D space and analyzed with engineering tools. A product is typically modeled by a single user working on their own client so if the user wants feedback on their model they must have co-workers come to their workstation and discuss the model on one screen. Many CAD vendors exist and the algorithms and model representations vary between them. Primary suppliers and sub-contractors on their supply chain often choose different CAD applications because of differences in company resources. When the model is exchanged down the supply chain it needs to be converted to a format that can be read by the receiving CAD software. Due to incompatibilities between systems many features or entire models need to be remodeled downstream. The lack of a neutral format which contains sufficient data to properly translate the model is central to the problem. Interoperability is the ability of different computer systems or software to exchange data between themselves. To better convert data between CAD systems a better interoperable neutral data architecture is needed. The architecture must be capable of neutralizing the data so that all participating systems can work on the model without needing to remodel or convert the data manually. Features in the model should have creation, edition, and deletion functionality to enable complex modeling. The addition of multi-user interaction between CAD clients where multiple users can edit and review models simultaneously from any location is also needed to improve collaboration.

1.1 Challenges of Interoperability

A 1999 study performed by the National Institute for Standards and Technology (NIST) made a conservative estimate that inadequate interoperability in the automotive industry costs them $1 billion per year [1]. Interaction with Industry Advisory Board (IAB) members of the National Science Foundation (NSF) Center for e-Design has revealed similar waste in the U.S. aerospace

industry. The cost of interoperability problems comes from the time and money spent to convert data manually that wasn't automatically converted, lengthening the design cycle time. Iterating on a design is an essential aspect of engineering to produce an optimal design so if the turnaround time on each iteration could be shortened, considerable time and money would be saved. Figures 1.1 and 1.2 illustrate the difference between the typical design process and the new interop design process.

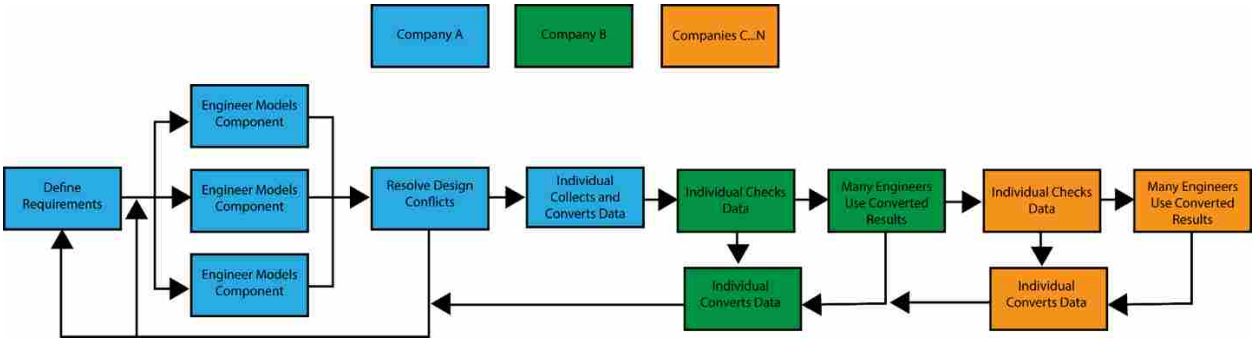


Figure 1.1: Typical Design Process

The interop design process places the work of all companies into parallel with each other which reduces feedback loops that are common in the typical design process. This allows improvements to be made to the design sooner because engineers inside and outside of a company can view and edit the work being done by everyone else, instead of waiting for a single user to finish their part. This speeds up the design process because the designers can immediately receive feedback on their design before wasting energy producing a design that is flawed. Productivity and accuracy is improved because different minds can notice different potential issues. Opening up the single user world of CAD modeling to allow multiple people to view and edit the model from the beginning is central to shortening the design process.

Developing a complex product requires the effort of many diverse engineering companies with different expertise and resources all working together. Companies choose a CAD system for a variety of reasons such as available functionality, price, and ease of use. It is highly likely that the CAD software used by a team of engineering companies will differ and this leads to the need to exchange data between different programs. Since CAD companies can decide to represent the

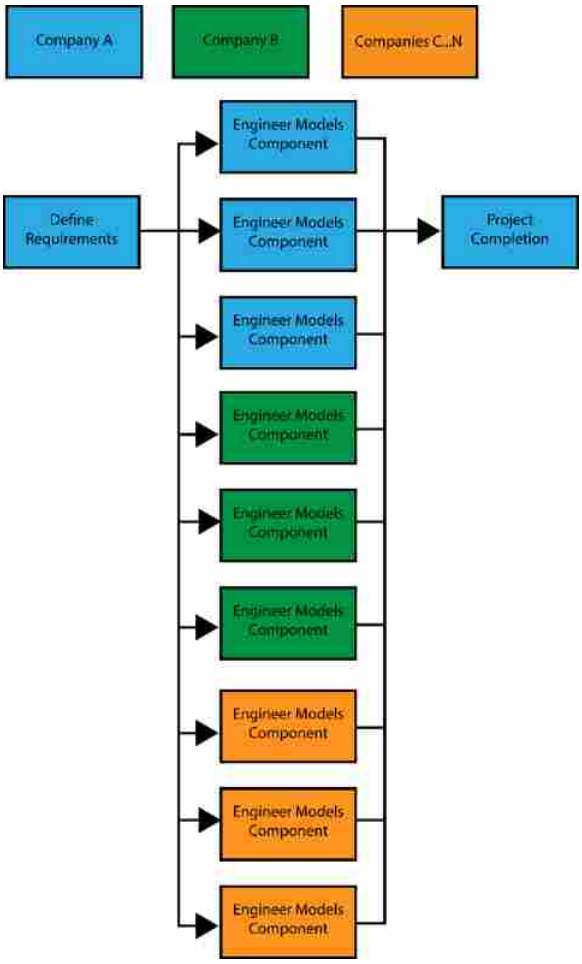


Figure 1.2: Interop Design Process

data for features in any number of ways differences will naturally result. While many features are common among all CAD systems many are not. An example will illustrate these challenges. A designer from a large company is modeling a complex part using NX, the CAD software available at his company. The manufacturer of the part uses CATIA as their CAD system so once the designer is done he must convert the model into an IGES file so that it can be read by CATIA. When the manufacturer opens the IGES file they find that the model has a few gaps between surfaces which must be fixed before it can be used to make the part on a machine. They decide to use repair software their company has purchased which fills in the gaps to make a water-tight solid. After doing this, however, it is clear that the resulting geometry of the part is not the same as it was before translation. In other words, the model integrity has been reduced and the part

will no longer meet its design requirements if manufactured as is. The manufacturer decides to remodel the part from scratch using the design drawings provided with the IGES file. This adds significant time to the design process and the resulting part is still found to need more revisions before it is mass produced. The information is passed back to the original designer and the process starts over. The feedback loops from Figure 1.1 illustrate this process. Significant time could have been saved if the manufacturer had received a feature-based part he could edit in CATIA as needed. An even better experience would have resulted if the manufacturer could have viewed the part being designed before it was ever passed to him because he likely would have noticed issues in the design and could have provided immediate feedback. Many such interactions occur in large supply chains each year so small improvements in CAD interoperability would save companies billions of dollars. Similar challenges exist when importing neutral CAD files into Computational Fluid Dynamics (CFD) software and other engineering tools.

This research found solutions to the interoperability problem by using a client-server architecture that supports multiple CAD systems interacting at the same time in a multi-user way. Advanced object-oriented programming methods such as polymorphism, encapsulation, and inheritance were all required to successfully create the program. A messaging system capable of receiving and managing many messages at once and carrying out the required tasks was central to the architecture's ability to enable multi-user CAD because individual users could model without worrying about what other users were doing. A relational database that supports the complex hierarchies of CAD feature trees was created to maintain the proper parent-child relationships within CAD models. A working program was developed which validates the interoperability concepts of this research.

1.2 Proposed Solution

The NSF Center for e-Design, BYU site, has been researching new tools, methods, and data structures to support the needs of its industrial members. The main body of this research has addressed the long lead times required to design complex mechanical components and assemblies by developing multi-user synchronous collaborative design tools. These tools reduce costs by reducing the time to design long lead time components and thus overall design cycle times. The objective of this research was to define the neutral data structure, called the Neutral Parametric Canonical

Form (NPCF), for many basic CAD features to enable translation between heterogeneous CAD systems. The NPCF stores the parameters which define a feature and defines a standard for neutral CAD features. Another objective was to design the neutral structure so that it would fit into the overall asynchronous client-server architecture of the interoperability program developed by BYU to support multi-user CAD interaction. The architecture of the NPCF is currently being developed to promote interoperability between NX, CATIA, and Creo CAD systems. These systems have been chosen initially because the industry sponsors of this research use them and they are three of the most commonly used CAD systems by industry leaders. Additional CAD systems could be added and configured to work with the developed neutral format. The NPCF for the 2D point, 2D line, 2D arc, 2D circle, 2D spline, 3D point, extrude, and revolve features will be defined in this thesis and used to demonstrate multi-user data interoperability between Siemens NX and CATIA CAD systems.

The solution to most of the interoperability problems is the definition of a neutral data format which can be translated to the proprietary formats used by existing CAD systems. Data neutralization is central to the issue because each CAD system is essentially trying to do the same thing, which is to represent a 3D model. At the heart of a 3D model are the parameters that define its boundaries and the relationships between its features. To translate between CAD systems the parameters of the model need to be neutralized so that any CAD system can use them to create a model with its own proprietary API methods. This research defined the NPCF for an initial set of CAD features which focuses on the parameters of CAD features and interfaces with the proprietary methods of individual CAD systems.

The important difference between data neutralization and data translation can be seen by looking at previous attempts to enable interoperability between CAD systems. IGES and STEP, in spite of their shortcomings, made it possible to exchange data with CAD systems that represent data in completely different ways. The reason they were successful was because large companies, aided by NIST, forced existing CAD vendors to support them [2]. For CAD vendors to stay competitive they had to comply with their customers' demands. Any future, long term solution needs to follow a similar course. The ultimate goal of translation is to move the data from one system to another but the goal of neutralization is to define a neutral format which contains the data and can be read by existing CAD systems. If greater interoperability is desired there needs to be a better

neutral representation that has all the required data to interface with the methods employed by a variety of CAD systems. If industry leaders adopt a neutral format that solves their interoperability needs then they will demand that CAD vendors support it and significant progress will be made.

With the parameters being the focus of the NPCF, the API's of each CAD system were used to extract the parameters from the proprietary methods so that they could be stored in the database. Multi-user (MU) classes are used for each feature which use API methods to translate from the CAD system feature to the neutral feature. CATIA's basic API is called the CATIA Automation API and provides somewhat limited access to the data. When greater access was needed the CAA Rade library was used which allows the programmer to do almost anything that might be needed. The NX API is called NXOpen and provides a level of access in between CATIA's Automation API and its CAA Rade library. The NXOpen API proved to be sufficient for supporting interoperability of the initial feature set in this research. Using CATIA and NX's API's allowed the data to be extracted and neutralized to support data conversion.

In addition to neutralizing the data, creating a multi-user environment further breaks down barriers to collaborative design. The client-server architecture of the program handles messages asynchronously and is able to support modeling synchronously. Whenever a feature is created, edited, or deleted a message is sent to the server which notifies all other clients of the change and updates the model. Not only is interoperability achieved, real time CAD modeling collaboration is made possible between users located anywhere in the world. Multi-user interaction between heterogeneous CAD systems is supported by the NPCF defined in this research.

The interoperability program has the capacity to neutralize CAD data in real time on a feature by feature basis between different CAD systems. This makes multi-user CAD interaction possible which greatly reduces the time to market for products as shown in Figure 1.3. Comparing this figure to Figure 1.1 shows that the "Resolve Design Conflicts" step has been removed because any conflicts were caught and fixed during multi-user collaboration. When this multi-user paradigm is applied across companies through interop the process appears more like Figure 1.2. The NPCF developed in this research allows the interoperability program to successfully exchange data. All basic features required to model 3D parts have been implemented and the NPCF for the 2D point, 2D line, 2D spline, 2D arc, 2D circle, 3D point, extrude, and revolve features will be defined in this thesis.

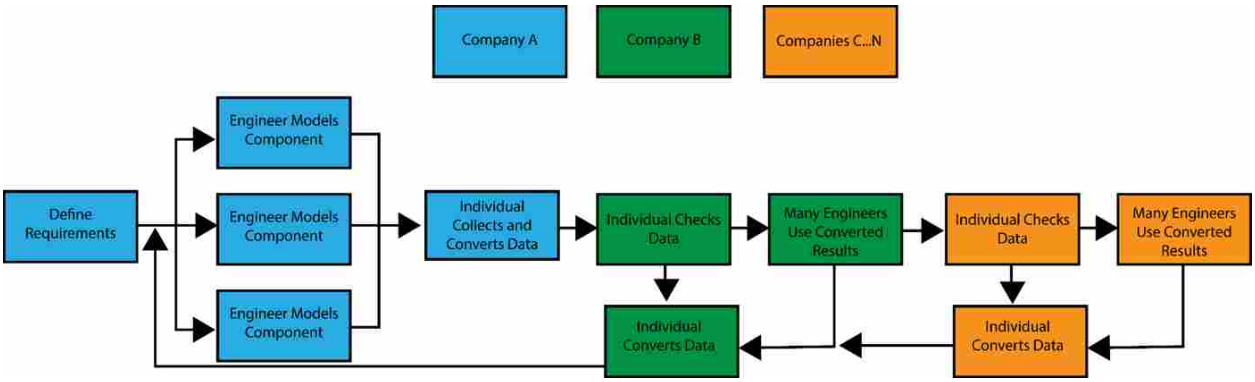


Figure 1.3: Multi-User CAD Modeling Process Eliminates Conflicts

CHAPTER 2. RELATED WORK

Neutral formats already exist which attempt to neutralize the data but data loss and corruption is a common issue. Very promising approaches focusing on modeling commands that are related to the API commands of each CAD system have made significant progress. Many companies today use direct translators which convert a model all at once between many existing CAD formats. The benefits of multi-user collaboration have been recognized by others as well and attempts have been made to better facilitate interaction. These many approaches have started to chip away at the interoperability problem but more needs to be done to improve collaboration and shorten the design cycle. The main contributors to the interoperability research area will be credited and their methods will be briefly explained in this section

2.1 Current Neutral Formats

The International Graphics Exchange Standards (IGES) format was introduced to facilitate data transfer between heterogeneous CAD systems. A technical committee met together in 1979 to address the issue of data exchange and IGES Version 1.0 was developed in 1980 [3]. IGES was the first attempt at resolving data exchange challenges between CAD systems and allowed the geometry of the CAD part to be viewed by those who were using different CAD systems. It is the most widely accepted neutral format and is therefore usually the best to use currently when working between different systems. It falls short in that only geometric information is transferable which isn't sufficient in many cases. Another major problem is that model geometry is often corrupted when translated into an IGES format. Holes or gaps are common in the geometry which then need to be repaired, costing time and money. Corruption results because numerical approximations are used when converting vertices to points and edges to curves to form a 3D solid [4].

To improve model data representation, the Standard for the Exchange of Product Model Data (STEP) was created in 1984 [5]. It has the advantages over IGES of storing product life-

cycle information, separating the application specific data from the general shape data, and using a formal language to define the data structure to avoid ambiguities when interpreting data [6]. Data corruption is still a common problem, however, when using a STEP file as the neutral format. The latest developments with STEP are Solid Model Construction History (SMCH) which is a hybrid approach [7]. B-rep and modeling construction history information are used to define features which provide design intent and geometry information together. A drawback of SMCH is that the file size is large due to B-rep and construction history being transmitted [8]. In a multi-user environment that is constantly sending data back and forth between clients a smaller data size is necessary. It remains to be seen if SMCH will improve the interoperability problem.

Rappoport introduced a new neutral architecture called the Universal Product Representation (UPR) which provided universal support for all data levels employed by the existing CAD systems at the time [9]. It represents the union of data types supported by CAD systems, not just the intersection. When the target CAD system doesn't support an incoming data type a rewrite is performed that attempts to convert the data type into something usable by the target. The results of this research have been implemented in the commercial translation software, Proficiency [10]. Feature-based CAD data translation is possible between five high-end CAD systems with high accuracy in translation. The drawback of this approach is that translation occurs all at once on the part instead of feature by feature so that multi-user interaction is not possible. There is no saved transaction history with this approach either so design intent can be lost.

Another neutral format used in commercial translation software is the Universal Features Object (UFO) used by Iyer and Ganapathi in their research [11] [12]. The program created by this research used the UFO to transfer binary files and translate them using the FeatureExchanger program. The FeatureExchanger is a collection of CAD API calls and methods that has CAD to UFO and UFO to CAD translation processes. The UFO is a superset of all the CAD features that existed in the CAD industry at the time. The CAD translation company Imagecom, now Aspire3D, uses the UFO to perform their translations [13]. The FeatureExchanger software successfully translated a feature-based model in SolidWorks to a feature-based model in Mechanical Desktop. Ganapathi expanded the research to include CATIA V5 R8 and Unigraphics V 18.0 [12]. The limitation of the research was that the entire model was translated at once so collaboration in real-time was limited. To successfully use this method would require consistent transferring of the file

back and forth when either side makes changes to the model. This neutral format is hindered in the same way the UPR format discussed earlier was hindered.

2.2 Modeling Command Approaches

Other promising research has tried a new approach which focuses on the modeling commands CAD systems use to create their proprietary models. These approaches analyze the API's of each CAD system and determine which methods are important to generate certain CAD features. The three main approaches that utilize this method are the macro-parametric approach, the neutral modeling command approach, and the ontology approach. They have been successful at translating feature-based CAD models between a variety of CAD systems.

2.2.1 Macro-Parametric Approach

The macro-parametric approach introduced by Guk-Heon Choi, et. al. has made strides in transferring features between heterogeneous CAD systems [8]. Choi came up with 167 standard modeling commands which were derived from the modeling commands of CAD systems. The research produced an external GUI where the translation of macro files between CAD systems could be handled. Duhwan Mun, et. al. continued the macro-parametric approach research by deciding upon a common set of modeling commands after surveying six different CAD systems [14]. A pre-processor was created which separates the source CAD system from the neutral modeling commands and a post-processor which separates the neutral modeling commands from the target CAD system. The pre- and post-processors map modeling commands between CAD system and neutral representation. This research was able to successfully translate simple parts from SolidWorks to CATIA with the feature trees in each being correct. The use of an external program, however, deters multi-user CAD. Standard modeling commands do not work with the actual feature data and therefore do not define a neutral format that all CAD systems can translate to.

2.2.2 Neutral Modeling Commands

Min Li also compiled a set of modeling commands, calling them neutral modeling commands (NMCs), and successfully translated data on a command by command basis [15] [16] [17].

Since the model isn't translated all at once, users are able to collaborate more effectively by seeing updates as another user makes changes to the model. Each client has an add-on that performs the translation of modeling commands from the source CAD system into neutral modeling commands. The server relays messages between clients to keep each system in sync. The command by command translation makes a multi-user experience possible but currently it forces each user to take turns making edits. The database of modeling commands is a superset containing all modeling commands found in the CAD systems surveyed. Wanfeng Dou, et. al., created a similar program but instead of compiling the union of all modeling commands created a unique set which is a minimum command set [18] [19]. A CAD Adaptor was attached to each CAD system which keeps track of all the operations being performed in the CAD system and prepares them for transfer. A CoCAD middleware module converts the local operations from the CAD Adaptor to a neutral command and vice versa. Other contributors to the NMC approach are Song [20], Chen [21], and Zhang [22]. While these approaches make progress by making multi-user CAD possible they still focus on the modeling commands and not on the feature data which prevents a feature-based neutral data representation from being defined.

2.2.3 Ontology Approach

Using the neutral modeling commands of the macro-parametric approach, Tae-Sul Seo, et. al., presented an ontology approach [23]. The meaning, or semantics, associated with each modeling command was added to improve the interpreting of modeling commands based on what they are trying to create. A source ontology was created for each CAD system used along with a shared ontology which acts as the neutral representation. Axioms were used between source and shared ontologies to interpret the data. Sun [24] and Tessier [25] also used these methods and developed similar programs. This approach is potentially more flexible than previous approaches because the architecture can handle a wider variety of inputs. So far the results of the approach have been the translation of very simple models. The disadvantage is still the same as with previous modeling command approaches because a neutral data representation for features is not defined and multi-user interaction is still limited.

2.3 Direct Translators

Many direct translators have also been created to transfer data between specific CAD systems and are in use by companies to reduce costs while translating data [10] [26] [27] [13]. This approach has been successful at translating models in a feature-based manner with high accuracy. Most of the popular, high-end CAD systems used in industry are supported which makes this software very useful. A common issue, however, is that the small percentage of features that are not supported are very often found in CAD models. This results in the model still needing to be altered or fixed after translation. Also, since a messaging system is not used between clients and a server the code is not conducive to a multi-user environment. A fundamental difference between a direct translator and the neutralization approach of this research is that the data is moved from one CAD system to another instead of converted to a neutral format and then translated to the proprietary CAD format. Instead, these solutions usually convert an entire macro file at once using an external GUI. If the solution allows multi-user collaboration it often requires users to take turns making edits which limits productivity. Proprietary neutral file formats are sometimes used within these systems such as the UPR and UFO formats mentioned earlier [9] [11] [12].

2.4 Multi-User CAD

Some of the past approaches already mentioned have created some form of multi-user interaction between CAD modelers. Since interoperability is very difficult, programs which support just one CAD system have had less roadblocks to supporting multi-user modeling. BYU developed NXConnect which provides real-time multi-user modeling between multiple NX clients and is now being turned into a commercial product [28] [29] [30]. Cai created a multi-user SolidWorks experience using similar ideas to NXConnect [31]. Maher did similar research using AutoCAD [32]. The interoperability software discussed in this research is a broader approach that encapsulates multi-user modeling between users of one CAD system and is thus more versatile.

While progress has been made to solve the CAD interoperability problem a neutral data structure still has not been defined for common CAD features. The architectures used also do not provide a synchronous multi-user CAD experience. More work is needed to define a neutral format which can be translated between common industry CAD systems in a multi-user way.

CHAPTER 3. INTEROPERABILITY METHODS

The methods used to create the Neutral Parametric Canonical Form (NPCF) and test if it better solves the interoperability problem will be explained in this section. This research defined the NPCF for a set of commonly used graphical commands and elements called features. The word "Parametric" alludes to the idea that the parameters which define a feature are stored as the neutral representation for the feature. The term "Canonical" portrays that this is a new standard form for representing CAD features. The NPCF is a new standard that supports multi-user CAD modeling. The initial feature set was chosen based on user experience and time constraints. Journaling, where a user's modeling commands are recorded and saved as a text file, was used to determine the important methods needed in creating the features in each CAD system. The required parameters to define a feature were found by viewing the inputs to the methods. By comparing the parameters needed in each CAD system a neutral representation became apparent and could be defined. Multi-User Object (MUObject) classes which contain all the needed logic to translate a feature between the CAD systems supported were used to store the neutral and CAD specific data for each feature. A client-server architecture connects all the participating clients to the server which relays messages between systems and uses the MUObject classes to translate the data. By using these methods the neutral format for a feature was defined and its validity was tested within a multi-user CAD setting.

The reasoning behind these methods becomes clear when the purpose of this research is considered. The purpose is to alleviate interoperability issues in industry by providing a program that works with existing CAD software packages. For the program to be helpful it needs to support the most common features used by real CAD designers. Implementing the software in the workplace also needs to be straightforward and simple to adopt. Creating a new CAD system that isn't compatible with existing CAD systems would not be well received by industry leaders because of the difficulty in training a large staff and the adjustment required.

Table 3.1: Common CAD Features in NXConnect Used to Determine Initial Feature Set

Feature	Count
Spline	3011
Scale	1842
Datum Plane	909
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
Line	36

3.1 Determining a Feature Set

The ultimate goal of the interoperability project is to support the union of all CAD features. Due to time constraints and feature complexity, the goal of this particular research was to support enough basic features that moderately complex parts could be modeled. An initial feature set was constructed by using the design experience of those familiar with CAD systems.

An aerospace senior design project where complex CAD models of aerospace parts were created in NXConnect was analyzed as a starting point in determining the most common features used during typical modeling. NXConnect is a multi-user version of NX developed by BYU. The database of features created in the NXConnect software during modeling was queried to find the features used. Table 3.1 summarizes the findings.

Based on these findings and the CAD experience of others working on the project the current feature set was chosen. The features that were minimally used during the project were not listed because they weren't required for basic modeling. Also, individual sketch features were not able to be queried in the database so the sketch feature set had to be determined by experience.

Table 3.2: Interoperability Program Initial Feature Set

Sketch Features	3D Modeling Features
Point2D	Datum CSYS
Line2D	Datum Plane
Arc2D	Point
Circle2D	Line
Spline2D	Spline
	Extrude
	Revolve
	Sketch

Many of the features listed are advanced features that can be replicated using other, simpler features and were not implemented to save time. The mirror feature, for instance, could be replicated by modeling both sides of a symmetrical model instead of just one. Sew, trimmed sheet, and thicken sheet are surface modeling features and will be implemented in later releases of the interoperability software. Any features that were auto-generated by NX during modeling were also not included. After considering the above points, Table 3.2 provides the initial feature set chosen for the CAD interoperability software. This basic set of features makes the modeling of complex parts possible.

3.2 Defining a Neutral Format

The value of using a neutral format that stores the parameters which define a feature instead of just transferring the data was discussed earlier. Previous approaches that access the API's of each CAD system are essentially extracting the data and moving it to another CAD system. Since the data is already extractable the method of this research is to store the data in a neutral format as parameters and in a database for later access and greater flexibility in software design. This allows translation to occur on a feature by feature basis and for the current state to be stored. If the interoperability system needs to be expanded to include more CAD systems it only involves adding the necessary API function calls which can translate to the neutral format. Therefore, to define the neutral format an understanding of the API methods used to generate features in each CAD system is required.

A great starting point to understanding the parameters which define CAD features is to record a journal or a script of the feature being used. Li and Mun used this idea to determine a set of neutral modeling commands that could be used as an intermediary between CAD specific commands [14] [33]. Most major CAD systems have journaling capability but a good understanding of the CAD system's API can allow a developer to program without it. NX and CATIA both provide journaling capabilities where the user starts recording, uses the feature within the program, and then stops recording. A text or script file is created which provides the actual functions called to create the feature. The inputs to the functions used are often the data needed to define the feature. This approach sheds light on the parameters which define a neutral format and was done before implementing the feature in the interoperability program. Figure 3.1 shows this idea being used to understand the data needed to define the 2D spline feature.

Areas boxed in the same color are accomplishing the same task in each CAD system. These journal files clearly show that a spline needs a sketch and control points to be created. Note that these are simplified journal files and so additional data is actually needed to completely define a 2D spline. The matching coordinates for the control points in each system are seen in the blue boxes. These coordinate values are the actual data that are important to extract and neutralize. The lines of code not boxed demonstrate how each CAD system goes about creating features in a different way. Although the methods used by each CAD system vary the end goal is the same and the parameters used are the same. Following this pattern the NPCF was constructed for each CAD feature.

3.3 MUObject Structure

The architecture of the program consists in multi-user object classes (MUObject) which store the neutral and CAD specific versions of each feature. This is necessary to support the creation, editing, and deletion of features. These classes are written in the C# language and extract the feature data using the CAD systems' API's. Within the multi-user class there are methods that translate back and forth between server features and CAD specific features. These methods are called when messages are received by the server that changes have been made by a user. Each feature has its own multi-user class which handles translations on a feature by feature basis. The architecture of an MUObject class is illustrated in Figure 3.2.

CATIA VB Script

```
//Open sketch
Set factory2D = sketch.OpenEdition()

//Determine axes of sketch
Set geometricalElements = sketch.GeometricElements
Set axis2D = geometricElements.Item("AbsoluteAxis")
Set line2D1 = axis2D.GetItem("HDirection")
Set line2D2 = axis2D.GetItem("VDirection")

//Create control points
Set controlPoint2D1 = factory2D.CreateControlPoint(-37.62, -10.15)
Set controlPoint2D2 = factory2D.CreateControlPoint(-54.26, -18.13)
Set controlPoint2D3 = factory2D.CreateControlPoint(9.35, -55.06)
Set controlPoint2D4 = factory2D.CreateControlPoint(125.85, -5.59)
Set controlPoint2D5 = factory2D.CreateControlPoint(80.25, 67.14)
Set controlPoint2D6 = factory2D.CreateControlPoint(41.27, 31.35)

//Create array of control points
Dim arrayOfObject(5)
Set arrayOfObject(0) = controlPoint2D6
Set arrayOfObject(1) = controlPoint2D5
Set arrayOfObject(2) = controlPoint2D4
Set arrayOfObject(3) = controlPoint2D3
Set arrayOfObject(4) = controlPoint2D2
Set arrayOfObject(5) = controlPoint2D1

//Create spline
Set spline2D = factory2D.CreateSpline(arrayOfObject)

//Close sketch
sketch.CloseEdition
```

NX C# Journal

```
//Activate sketch
sketch.Activate(NXOpen.Sketch.ViewReorient.False);

//Begin spline creation
Spline sp = null;

//Create spline builder
SketchSplineBuilder builder = workPart.Features.CreateSplineBuilder(sp);

//Create control points
Point3d coordinates1 = new Point3d(-37.62, -10.15, 0.0);
Point point1;
point1 = workPart.Points.CreatePoint(coordinates1);
Point3d coordinates2 = new Point3d(-54.26, -18.13, 0.0);
Point point2;
point2 = workPart.Points.CreatePoint(coordinates2);
Point3d coordinates3 = new Point3d(9.35, -55.06, 0.0);
Point point3;
point3 = workPart.Points.CreatePoint(coordinates3);
Point3d coordinates4 = new Point3d(125.85, -5.59, 0.0);
Point point4;
point4 = workPart.Points.CreatePoint(coordinates4);
Point3d coordinates5 = new Point3d((80.25, 67.14, 0.0);
Point point5;
point5 = workPart.Points.CreatePoint(coordinates5);
Point3d coordinates6 = new Point3d((41.27, 31.35, 0.0);
Point point6;
point6 = workPart.Points.CreatePoint(coordinates6);

//Create spline
NXObject spline;
spline = builder.Commit();

//Update the sketch with the new spline
theSession.ActiveSketch.Update();

//Delete the builder
builder.Destroy();

//Deactivate sketch
sketch.Deactivate(Sketch.ViewReorient.True, Sketch.UpdateLevel.SketchOnly);
```

Figure 3.1: Files Resulting from Journaling in NX and CATIA for 2D Spline Feature

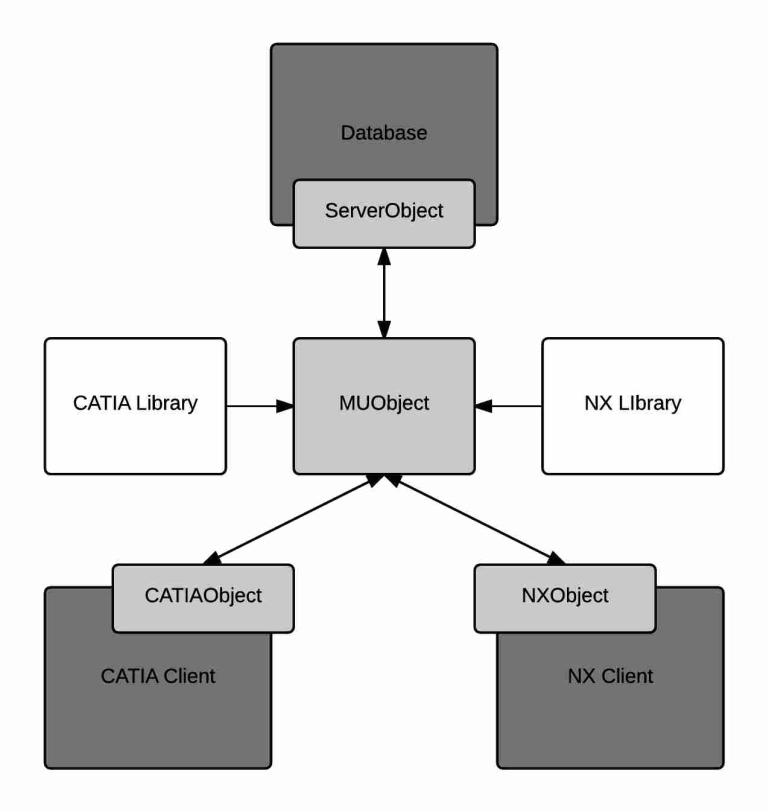


Figure 3.2: Architecture of MUObject Class

When a CATIA client creates a feature, shown in Figure 3.2 as the "CATIAObject", it is translated within the MUObject class into a server version and an NX version, designated "Server-Object" and "NXObject" respectively. The MUObject has access to the NX and CATIA libraries, meaning that it can access the API methods of each system to perform the translation. The data stored in the ServerObject is saved on the database for later retrieval. The same process occurs when a feature is created on an NX client, but in reverse. Using MUObject classes makes implementing new features relatively straightforward because all the feature specific information needed for translation is collected in one place. It also allows multi-user CAD modeling because the MUObject can be created or updated asynchronously during run-time by each CAD client to send data between the clients and the server in real-time.

3.4 Client-Server Architecture

Multi-user CAD requires a messaging system that can handle messages asynchronously while supporting modeling synchronously. A thick client thin server architecture achieves this by storing messages in a queue and then executing them in between the creation of other features. The interoperability program features a plug-in that is installed on each client which can communicate with a server running on the same client or elsewhere. A class for each CAD system is in charge of sending and receiving messages from the server via the MUObject classes and calls the appropriate feature class to create the needed feature on the client. The versions of each of the features are stored in MUObject classes on each client as discussed earlier. The server communicates with the database and stores the neutral data of the feature. Since translations are performed on a feature by feature basis, multi-user modeling is supported. The architecture is illustrated in Figure 3.3 between two NX client and two CATIA clients.

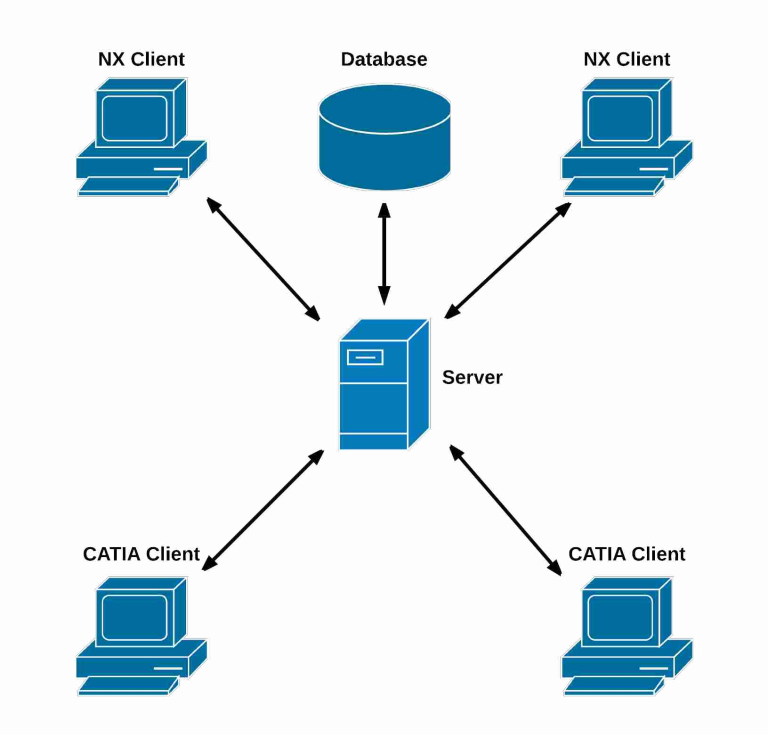


Figure 3.3: Client-Server Architecture

A likely issue when multiple users are sending updates to the server is that messages can get out of order. This can cause duplicate features to be created. To ensure that data is consistent across all clients and the server the pseudo-singleton pattern was used. The singleton pattern guarantees that only one copy of a class ever exists. The pseudo-singleton pattern modifies the singleton pattern to allow multiple instances of the same class but only one instance of an individual feature. For example, a user can create more than one line in a sketch but duplicates of the same line are not wanted. Using a pseudo-singleton pattern allows data to remain consistent and supports multi-user CAD.

3.5 Unit Testing

A common problem when writing a program of this size is regression. When changes are made to certain sections of the code it can cause previously working code to stop working. An effective way to prevent this is to use unit testing. In unit testing, a section of code that has a specific function is tested to make sure that it accurately performs its function. For instance, if a method is meant to create the server version of a feature when given the CATIA version, a unit test would create a test CATIA version, pass it into the method, and see if the server version that results has the correct data. Since regression can occur on either the server side or the client side unit testing was used to check both sides. Performing unit tests was very helpful in making the interoperable program robust and was done for every method of every feature.

3.6 NPCF Verification

Creating, editing, and deleting of features in both CAD systems were tested to verify the NPCF approach. Table 3.3 provides the tests that were carried out for each feature. For multi-user CAD modeling to be effective, features must be creatable in every supported CAD system. Furthermore, every feature must be editable and deletable in every CAD system regardless of which CAD system originally created it. Testing was done in three stages to validate the state of the program. Initial testing involved one user operating an NX client, a CATIA client, and the server all on the same machine. This setup allowed simple testing of the software to see if features pass back and forth. The next stage of testing involved three computers and two users. The NX

Table 3.3: Verification Tests Performed on Features

Test
Create feature in NX
Create feature in CATIA
Edit feature from NX in NX
Edit feature from NX in CATIA
Edit feature from CATIA in CATIA
Edit feature from CATIA in NX
Delete feature from NX in NX
Delete feature from NX in CATIA
Delete feature from CATIA in CATIA
Delete feature from CATIA in NX

and CATIA clients were on separate machines operated by different users and the server was on a third machine. This setup allowed geographically dispersed CAD modeling to be tested. The final stage of testing required five computers and four users. Two NX clients and two CATIA clients dispersed across four computers were operated by four different users with a server running on the fifth computer. This test determined the initial scalability of the software and simulated real world engineering where many engineers located across the country or the world are collaborating using CAD software. If a feature passed all these tests it was determined to be validated and ready for more rigorous testing.

Once all features passed the initial stages of testing the modeling of a moderately complex part was tested. When features are used in combination with each other and dependencies are generated the code is required to be more robust to work properly. This last test verified that the code is stable so focus could be turned to adding more features and functionality to the program. The features tested in this research were the 2D point, 2D line, 2D spline, 2D arc, 2D circle, 3D point, extrude, and revolve features.

CHAPTER 4. INTEROPERABILITY IMPLEMENTATION AND RESULTS

This research has worked on a Neutral Parametric Canonical Form (NPCF) which defines the neutral data structure for many basic CAD features to enable data conversion between heterogeneous CAD systems. The architecture of the program developed at BYU consists of database, communication, logic, and client tiers. In the database tier, parameters are stored in a database which has a table for each feature type supported in the program. When changes are made to the database, the communication tier handles messages to and from the server, client, and database. The logic tier contains classes with methods for creating and updating neutral features from CAD system features and vice versa. The client tier tracks a users actions and tells the logic tier what features need translation. The architecture of the NPCF supports the multi-user CAD initiatives of the lab and is being developed to promote interoperability between NX, CATIA, and Creo CAD systems. This thesis focuses on interoperability between NX and CATIA only. The 2D point, 2D line, 2D arc, 2D circle, 2D spline, 3D point, extrude, and revolve features have been implemented and will be discussed in this section. The capabilities and limitations of the features along with a complex modeling demonstration will be given.

4.1 Program Architecture

The interoperability program is divided into four main levels or tiers of architecture. This architecture was chosen because it enables the transfer of data on a feature by feature basis between all participating clients. Each tier will be explained in greater detail and the overall architecture is illustrated in Figure 4.1.

4.1.1 Database Tier

The database tier contains tables for each feature type. Each feature table contains parameters for all instances of that feature from all the parts in the database. Entity Framework 6 was

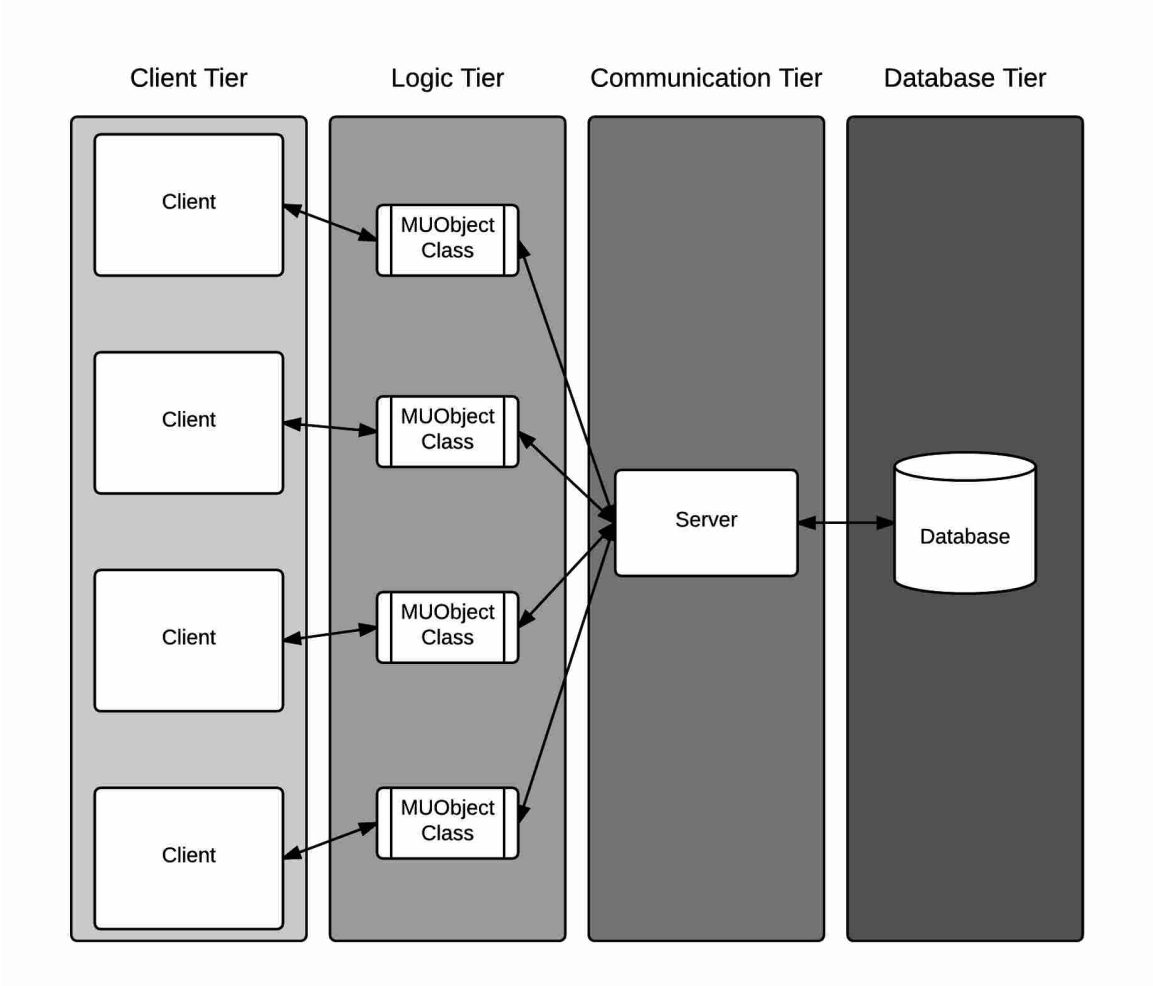


Figure 4.1: Interoperability Program Architecture

used as the object relational mapper (ORM) to handle relationships between tables. One-to-one, one-to-many, and many-to-many relationships are all used between tables depending on the need. For example, a 2D spline feature contains many 2D control point features which represents a one-to-many relationship. Primary and foreign keys are used to link tables together. A primary key uniquely identifies a row of data in a table. It is usually associated with a Global Unique IDentifier, or GUID. Every feature created in the interoperability program is assigned a GUID which allows the feature to be identified and retrieved from the database when needed. Foreign keys are used to reference primary keys in other tables in the database. Each row of data in the 2D control points table contains a 2D spline GUID which references the GUID for a row of data in the 2D spline table. The column of 2D spline GUIDs in the 2D control points table is the foreign key while the

column of GUIDs in the 2D spline table is the primary key. Using primary and foreign keys is essential to maintaining data consistency and is required for collaborative design. The database tier successfully maintains consistency to support multi-user CAD.

4.1.2 Communication Tier

A unique aspect of this interoperability architecture is the messaging system that handles changes made by the clients asynchronously while supporting synchronous modeling. The communication tier of the program architecture sends and receives messages between clients whenever a feature is created, edited, or deleted. When messages are received at the same time by the server a system is in place which applies changes to the client in between operations. In single-user CAD modeling operations are handled in series and applied to the model. Multi-user environments require a parallel scheme which can manage modeling operations coming from many sources at once. Figure 4.2 compares single-user communication to multi-user communication.

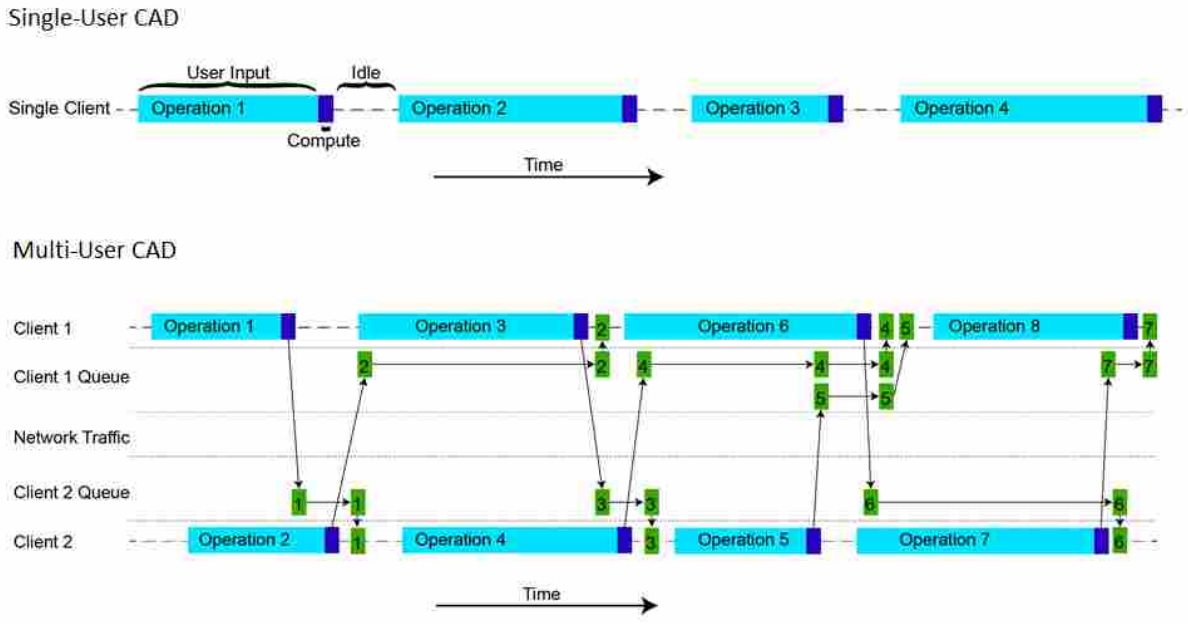


Figure 4.2: Single-User CAD vs. Multi-User CAD Communication

When client 1 completes an operation during client 2's operation it is stored in client 2's queue. Once client 2's operation is complete the operations in the queue are then applied. This

process proceeds throughout the entire modeling session. By serializing the operations the communication tier makes multi-user CAD modeling a reality.

4.1.3 Logic Tier

The core of translation between clients is handled by the logic tier. At the heart of the logic tier are multi-user object classes, or MUObjects, which were discussed earlier. The neutral format architectures for each feature are contained within their respective MUObject class. Each class stores the server, NX, and CATIA versions of the feature. When translating between NX and CATIA these properties are assigned and can be retrieved when needed. A GUID is created and assigned when the feature is first created on the client and persists with the feature throughout its life. This GUID is stored in the database along with the other neutral parameters required to define the NPCF for the feature. The API's of each CAD system provide methods for creating a CAD specific version of a feature. Each CAD system's API methods work differently and different approaches are needed to get to the feature data. The basic parameters needed to define a feature, however, are the same and are extracted and stored in the database. For a better understanding of the logic tier architecture, refer to Figure 3.2.

4.1.4 Client Tier

Each client has a class which orchestrates activities between the server and the multi-user feature classes. It checks for changes on the client and triggers messages that are handled by the communication tier. The client class also calls the multi-user feature class methods to initiate data neutralization. CATIA has an update loop that continuously runs and checks to see if changes have occurred and then notifies all other clients in real time. NX has events that are triggered when changes are made. When the program is started it connects with the clients using the client tier and initiates communication with the server. If an existing part is opened by the clients it pulls the data from the server and loads it onto the clients. As the part is edited or updated all changes occur through the client classes. The client tier is the boss that tells all the other classes what to do.

4.2 Sketching Implementation

Most CAD systems have sketching functionality upon which 3D modeling is based. The idea is to create two dimensional drawings and then extrude or sweep them through a distance or angle to form three dimensional shapes. Sketching is often an area of CAD software that CAD companies approach differently. This difference can greatly complicate the translation process. For example, CATIA immediately adds sketch elements within a sketch to the feature tree upon creation. NX does not update the feature tree with sketch elements until after the sketch is finished, at which time they are added to the tree within the sketch all at once. This small difference causes major problems when attempting to translate CAD data on a feature by feature basis because NX will receive the sketch elements from CATIA before it receives the sketch. To handle this difference, the sketch features created within CATIA are added to a queue. Once the sketch is finished, a packet of sketch features is sent to the server and are translated in the logic tier. This allows NX to pull down the sketch from the server before pulling down the features within the sketch. The neutral parametric canonical forms (NPCF's) for the 2D point, 2D line, 2D arc, 2D circle, and 2D spline sketch features were implemented in this research to fit into the upper level sketch feature architecture of the program. Figure 4.3 shows what each of these sketch features looks like.

Another major difference in how NX and CATIA create features is the use of a builder object by NX. Each feature has its own builder class in the NX API which creates and modifies features. The feature that needs creation or modification is passed into the builder creation method and the data is accessed through the builder object. The feature is created when the builder is committed using the Commit() command. CATIA uses a factory object which creates the feature but additional manipulation is done using the feature object directly. The feature is added to the part when the Update() method is called. Once the differences are understood the data can be accessed successfully.

Figure 4.4 gives the multi-user class diagram for the 2D point feature to highlight the main parts of a multi-user class. Plus signs indicate public methods, minus signs indicate private methods, and pound symbols indicate protected methods. The private methods are called by the public methods of the MUObject class which are called by the client classes when changes are made on a client. The private methods in the class assign values to the server, CATIA, and NX

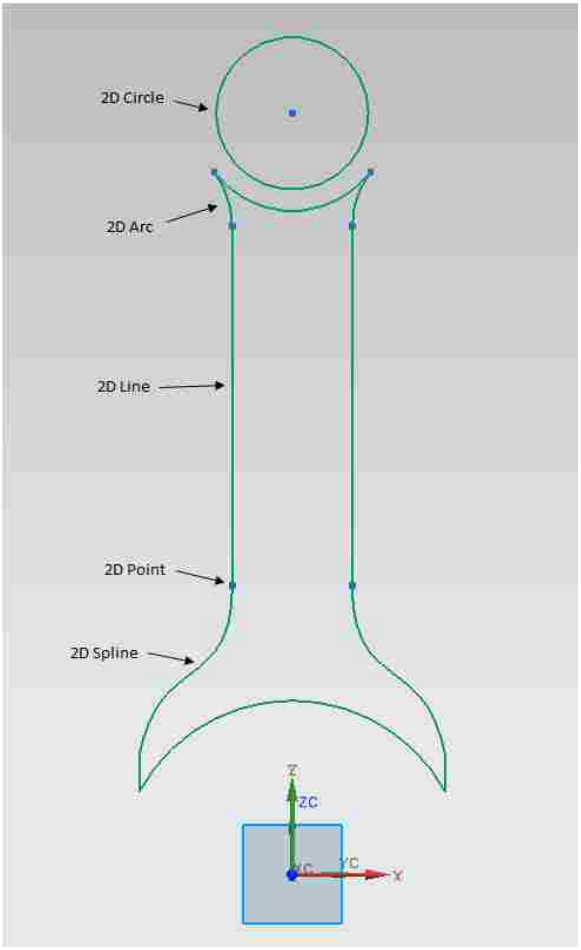


Figure 4.3: Demonstration of Sketch Features in NX

feature properties of the class. The CATIA and NX API's are used in the private methods to extract and neutralize data. The MUObject constructors create the server version of the point when it is first created on the client and the other private methods are called to generate CAD specific features from server features. GetInstance() methods retrieve a specific feature based on the GUID of the feature wanted and the CAD system. Update methods handle edits made to existing features and Delete methods manage deletion of features by the user. This architecture allows features to be translated between CATIA and NX in a multi-user environment.

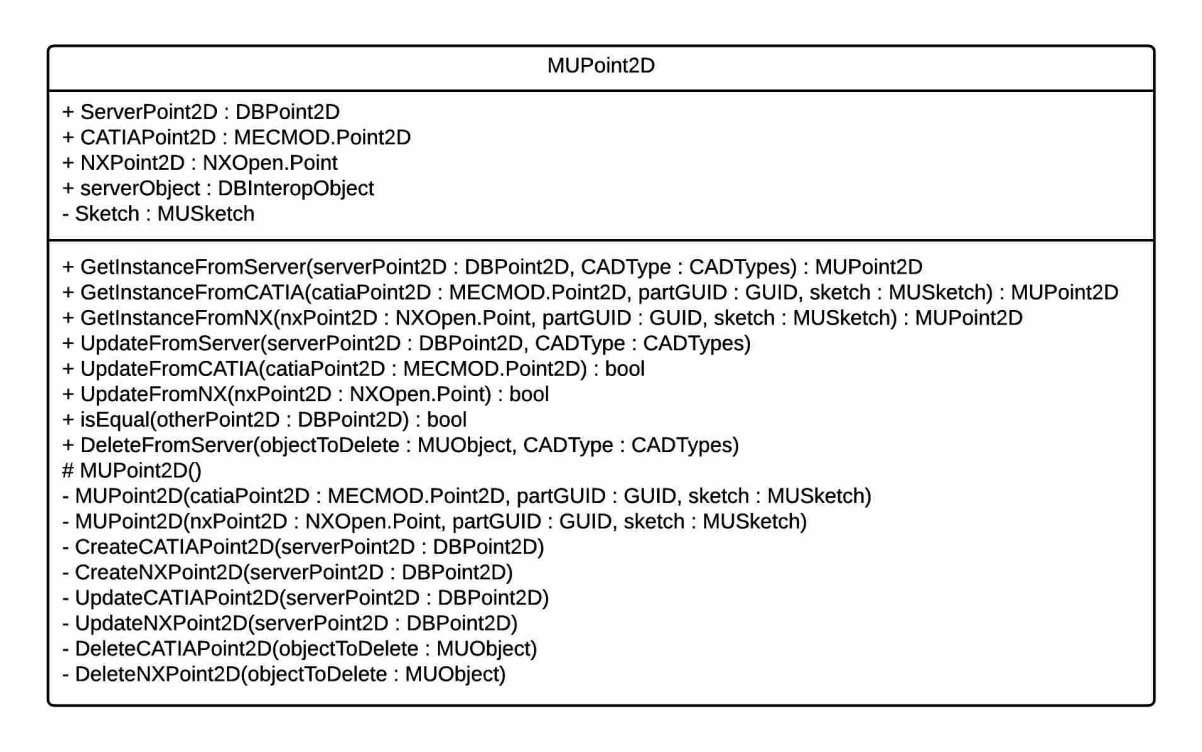


Figure 4.4: Properties and Methods of MUPoint2D Class

4.2.1 2D Point Implementation

A 2D point is the most basic sketch element that can be created. It is an important feature because many other sketch features are based on it, such as 2D lines. A 2D point in CATIA is shown in Figure 4.5 as the orange plus sign. Even though a 2D point is the same as a 3D point constrained to a specific plane, the addition of a sketch greatly complicates the translation process. Most of these complications, however, are managed by the MUSketch class created by others in this research group and only the MUPoint2D class will be explained. The unique parameters in the NPCF for a 2D point are an X and Y coordinate. The GUID which uniquely identifies the 2D point from other 2D points is inherited from the Interop Object through the Feature and Sketch Feature objects. The X and Y coordinates position the point on the sketch plane and the SketchGUID inherited from the Sketch Feature object links it to a specific sketch. The Sketch Feature object and the Sketch object both inherit from the Feature object and the Feature and Part objects inherit from the Interop Object. The Feature object is linked to the Part object through a one-to-many

relationship which means that a part can have many features. These links between objects allow the model to be edited and updated more easily than other solutions. Figure 4.6 illustrates the 2D point NPCF. Outlined arrows indicate inheritance relationships and a short line crossing a main line on one end with three lines spreading out on the other end represents a one-to-many relationship.

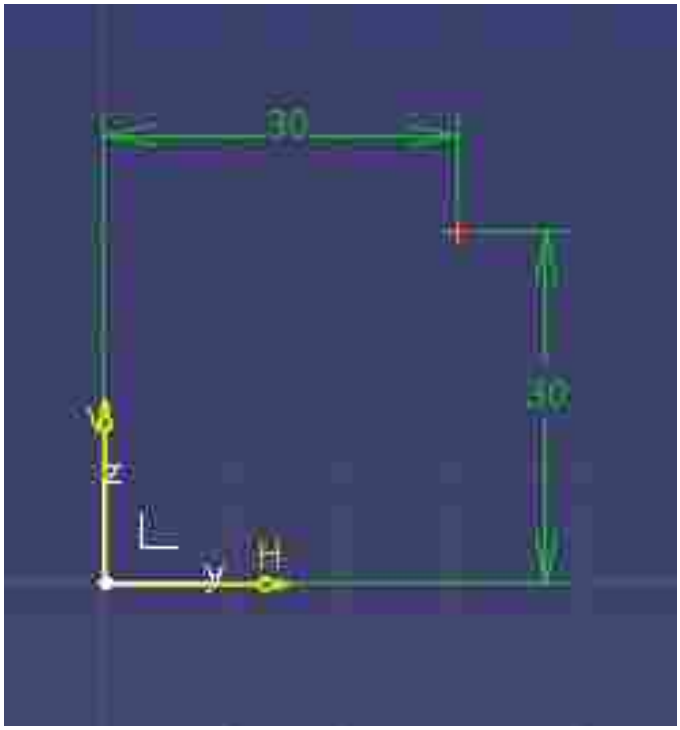


Figure 4.5: 2D Point Feature in CATIA

The figure expresses the idea that the NPCF for a feature encompasses more than just the basic data required to define it. It also includes the feature's relationships with other base objects in the architecture. All features implemented inherit from the Feature object so the top three boxes in the figure are the foundation of the NPCF. To extract the parameters that go into a 2D point, the MUPoint2D class is used. This class contains the methods which assign the properties.

4.2.2 2D Line Implementation

The 2D line feature adds some complexity to translation because a 2D line depends on two 2D points as shown in Figure 4.7. Therefore, the parameters that define a 2D line feature are a start

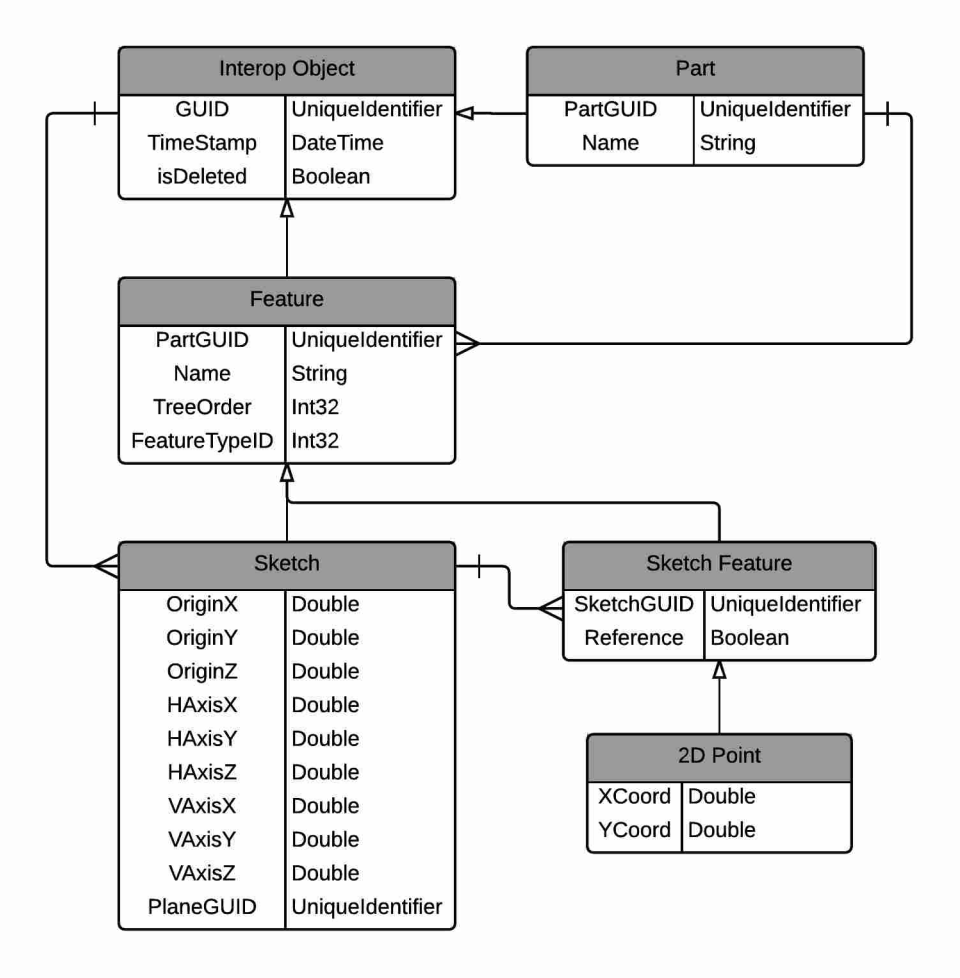


Figure 4.6: NPCF for 2D Point Feature

point GUID and an end point GUID. The 2D line feature inherits a sketch GUID from the Sketch Feature object just like a 2D point. Just as a Sketch Feature object's SketchGUID property links the feature to the proper sketch, the start and end point GUID properties link the line to the proper 2D points in the database. Figure 4.8 shows the added complexity of the NPCF for the 2D line feature. Notice that the foundation of the NPCF is the same but there are some added relationships and data for the 2D line.

Two one-to-many relationships attach two 2D point objects to the 2D line object. The MULine2D class extracts the data for the 2D line NPCF and differs from the MUPoint2D class in the inputs required for the methods. Instead of just needing the feature and the sketch to be passed in, a 2D line also needs the start and end points passed in. This is because when a line is

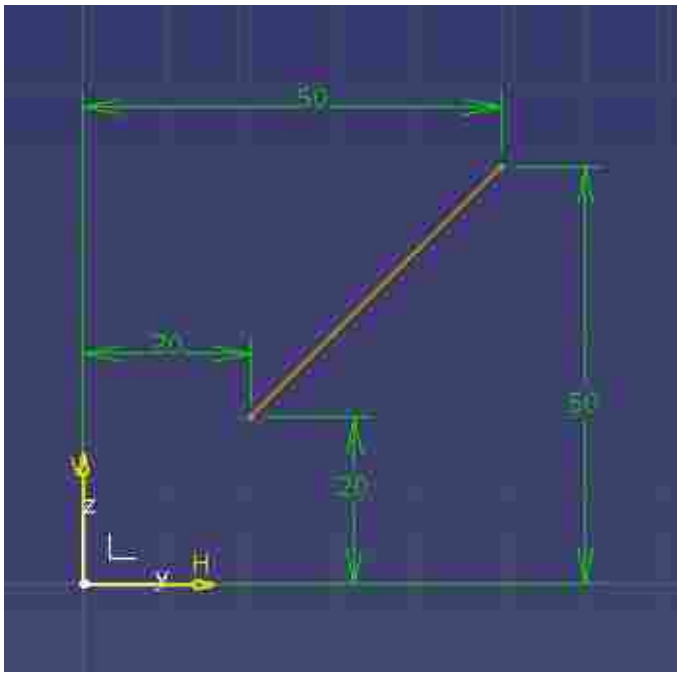


Figure 4.7: 2D Line Feature in CATIA

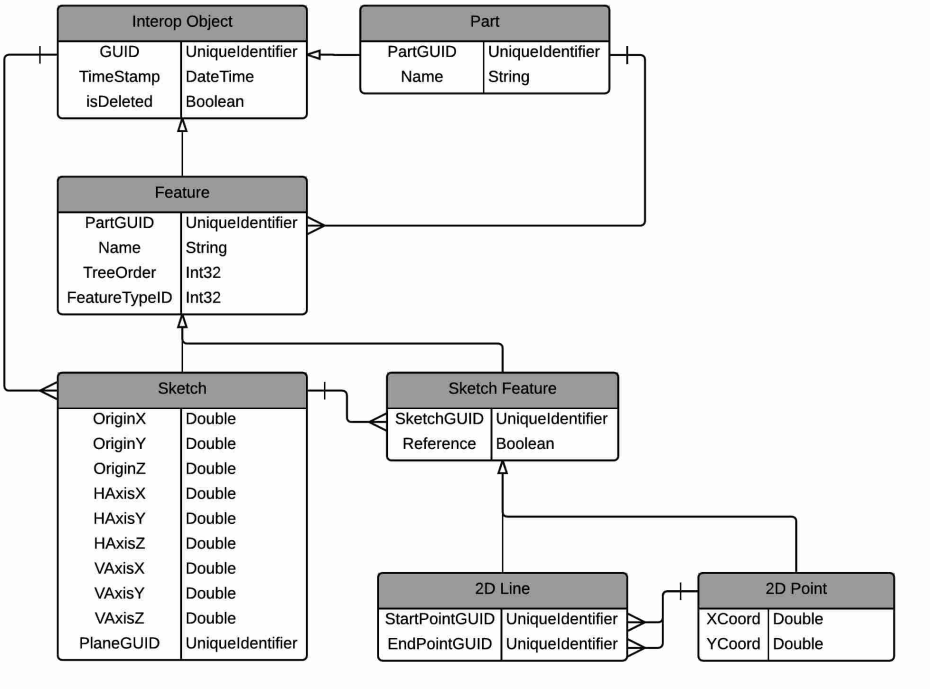


Figure 4.8: NPCF for 2D Line Feature

created or edited the start and end points need to be created or edited first. The MULine2D class is able to call the MUPoint2D class and use it to create the points. The same thing is done with the sketch. Separating the code into feature based classes is central to the software's ability to perform translation on a feature by feature basis.

4.2.3 2D Arc Implementation

The implementation of a 2D arc feature follows a similar pattern to that of the 2D line feature. The important data is visualized in Figure 4.9. Looking at the NPCF architecture in Figure 4.10 reveals that the 2D arc object has simply replaced the 2D line object which was in Figure 4.8. The only extra parameters needed to define a 2D arc are the center point GUID and the radius. The diagram shows an extra one-to-many relationship between the 2D point and the 2D arc objects for the center point. The rest of the diagram is the same.

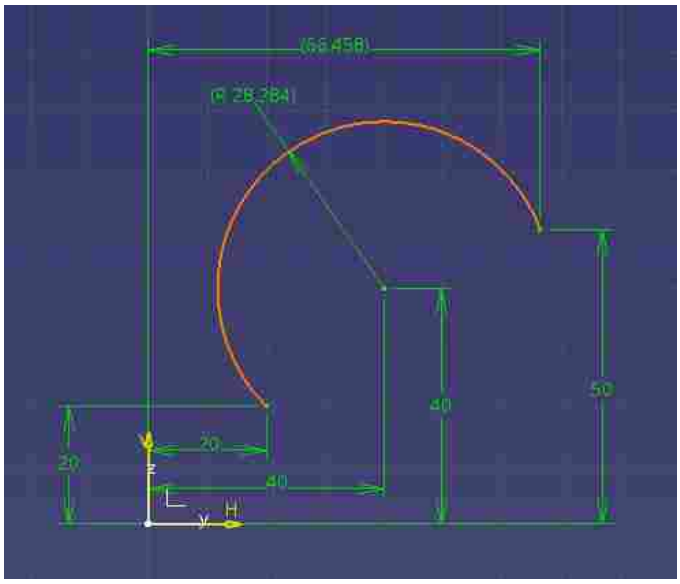


Figure 4.9: 2D Arc Feature in CATIA

CATIA's approach to creating a 2D arc is to create a circle and then set the start and end points. NX needs a start and end angle to create an arc so using the start, end, and center points the angles are calculated using trigonometry. The CreateCircle() method is used by CATIA and

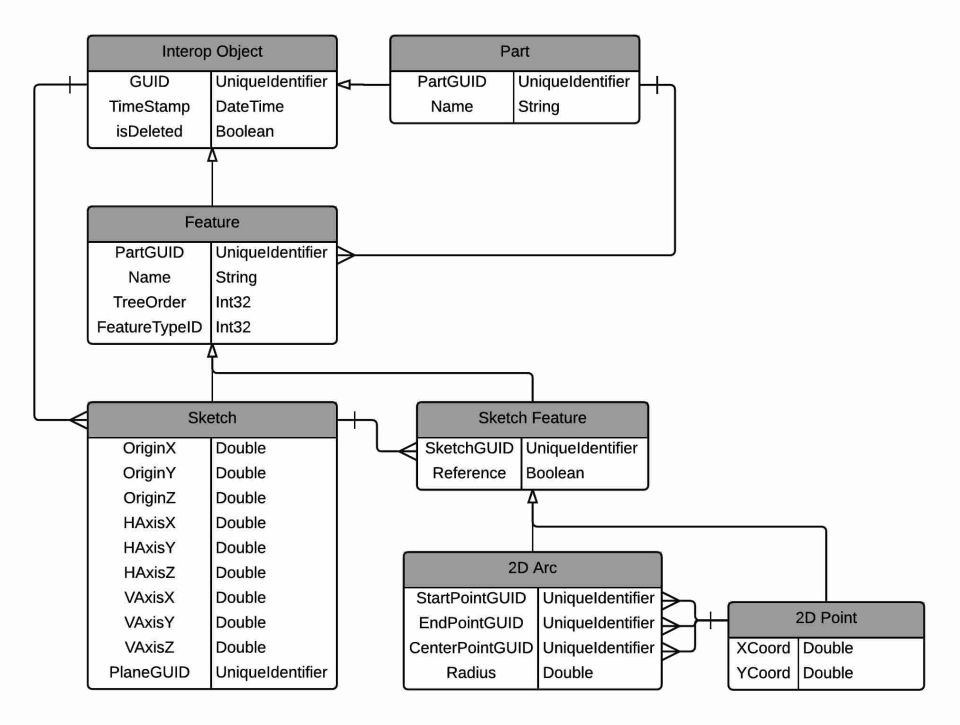


Figure 4.10: NPCF for 2D Arc Feature

the CreateArc() method is used by NX. In spite of the differences, the data is the same and can be extracted.

4.2.4 2D Circle Implementation

Both NX and CATIA consider a 2D circle the same as an arc with different boundary conditions. CATIA calls the CreateClosedCircle() method instead of the CreateCircle() method it used to create an arc and takes as inputs the X and Y coordinates of the center point and a radius. NX uses the same CreateArc() method used for creating arcs but uses a start and end angle of 0°and 360°, respectively, for inputs. Because of this, only a center point and a radius are needed to define a 2D circle, which is clear from Figure 4.11. The NPCF given in Figure 4.12 shows this with a single one-to-many relationship between the CenterPointGUID property and the 2D Point object. Everything else about a 2D circle is the same as a 2D arc.

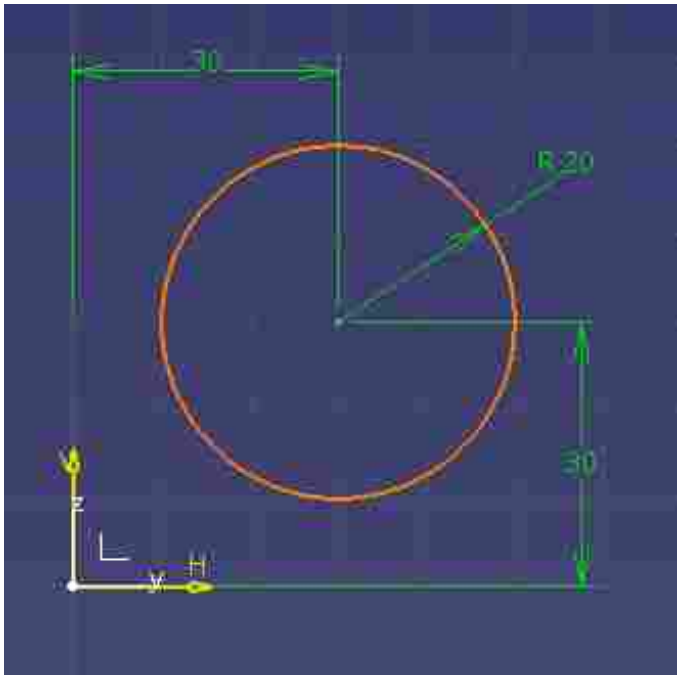


Figure 4.11: 2D Circle Feature in CATIA

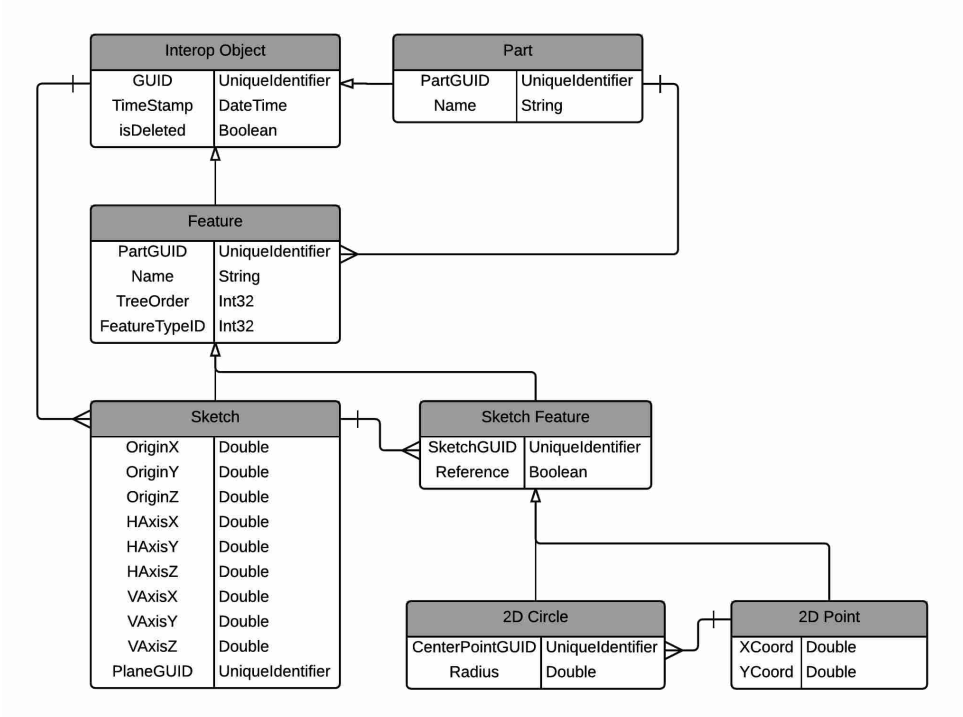


Figure 4.12: NPCF for 2D Circle Feature

4.2.5 2D Spline Implementation

A 2D spline allows complex surfaces and shapes to be generated through extrusion or revolution. A spline is shown in orange in Figure 4.13. The NPCF consists of an array of control points and is shown in Figure 4.14. Other important properties such as the GUID and sketch are inherited as was the case for the other sketch features.

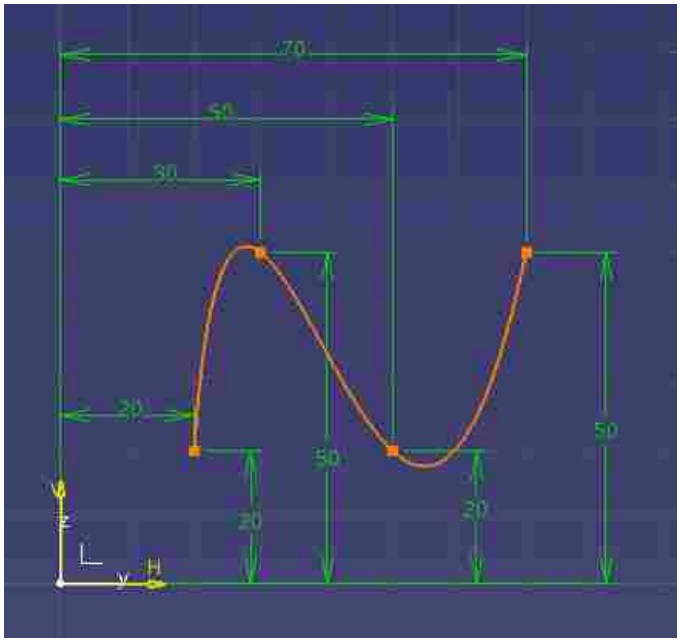


Figure 4.13: 2D Spline Feature in CATIA

Implementing the 2D spline feature was similar to what was done for the 2D line feature but the addition of control points complicated the translation as the next sub-section discusses. Instead of passing in the start and end points to the class methods the control points array is passed in.

2D Control Point Implementation

A 2D control point is its own sketch element and a separate multi-user class was written for it. The extra parameters needed to define a control point is a 2D spline GUID and an order number. Figure 4.14 shows how the NPCF for a 2D control point feature relates to the 2D spline feature

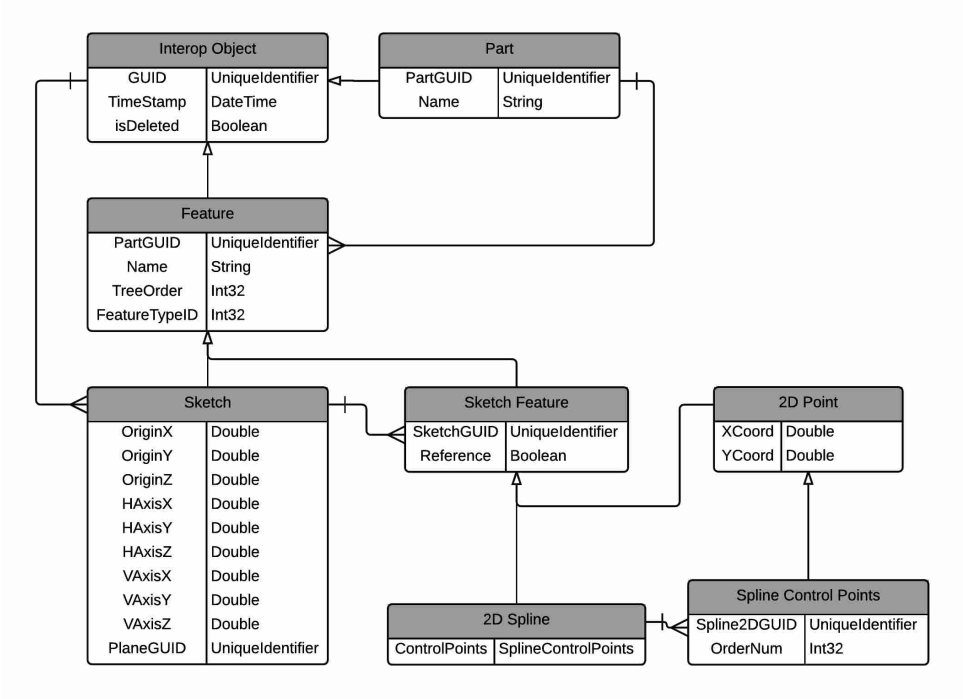


Figure 4.14: NPCF for 2D Spline Feature

NPCF. The control points can be seen in Figure 4.13 as orange boxes. The relationship between a 2D spline and a 2D control point is a one-to-many relationship because one 2D spline has multiple 2D control points. The 2D control point inherits its X and Y coordinates from a 2D point object as shown in the figure. The use of an unspecified number of control points makes implementing the 2D spline more complicated. When translating, the number of control points for the specific 2D spline needs to be extracted using the CAD API's to properly fill the array of control points. The order in which the control points are added to the 2D spline is also very important which is why an order number is assigned to each control point.

4.3 3D Modeling Implementation

Once the basic sketch features have been implemented the 3D modeling features can be added. To become familiar with 3D translation the 3D point feature was implemented first. 3D points can be used as a reference to create other 3D features and are simple to implement. Two of the most common and useful 3D features are the extrude and revolve features and they were

also implemented. CATIA and NX have a fundamental difference in how these two features are handled, as they did for sketch features. NX uses boolean operations when solids are created which determines whether its to be used as an addition, subtraction, or another type of extrusion or revolution. All these operations are included within the same extrude or revolve feature. CATIA separates the addition and subtraction operations into two different features. This makes translation more difficult but fortunately the problem can be solved because the data used to define the features is the same. The use of a builder object by NX is another difference from CATIA as was seen with sketch features. The NPCF was defined for the 3D point, extrude, and revolve features which enabled many moderately complex parts to be created. The upper level architecture for 3D modeling features which includes the "Interop Object", "Part", and "Feature" objects was used as a starting point for developing the architecture.

Multi-user object classes were created for the 3D modeling features just as they were for sketch features. A fundamental difference between the two is that sketch features need to be assigned to a sketch to be valid but a 3D modeling feature typically needs a sketch as an input. Other than this, the overall architecture and methods used in the MUObject classes were the same. The CATIA and NX API's are utilized to extract the needed parameters for the NPCF and private methods in the class convert from CAD system version to neutral format and back again. Update and delete methods allow 3D features to be edited or deleted in either CAD system. 3D features are translated on a feature by feature basis to support multi-user CAD modeling.

4.3.1 3D Point Implementation

The first 3D modeling feature implemented was the 3D point feature because it is relatively easy to understand. The specific parameters which define a 3D Point feature are an X, Y, and Z coordinate as seen in Figure 4.15. While the definition is simple, extracting the data from each CAD system and translating between the two requires more effort. NX and CATIA have their own unique methods and approaches to creating 3D points in their respective CAD systems. For example, to get the point coordinates from a CATIA point the GetCoordinates() method needs an empty array passed in which can be filled with the coordinate values. An NX point uses a builder which links the point created by the CreatePoint() method to a PointFeature object. Even though

the methods between CAD systems differ the data they are manipulating is the same so a neutral representation can be defined.

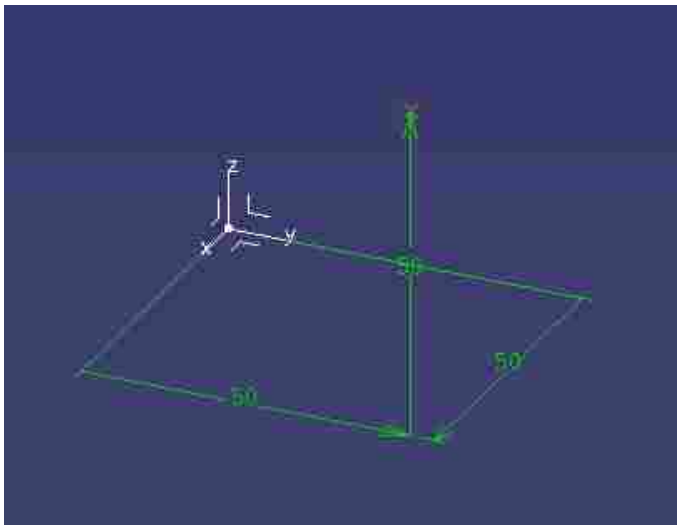


Figure 4.15: 3D Point Feature in CATIA

The NPCF of the 3D point feature is simpler than any sketch feature previously discussed because no sketch is involved. The 3D point object is connected directly to the Feature object as seen in Figure 4.16. The GUID for the point is again inherited from the Interop Object and other important properties are inherited from the Feature object. The 3D point feature is the simplest feature that can be implemented.

A multi-user point, or MUPoint, class was created which contains the logic and methods to extract the X, Y, and Z coordinates from the CAD specific features and store them in the database. Upon creation, a new GUID is assigned and persists with the point throughout its life. Methods for translating back and forth between each CAD point through a neutral, or server point are within the MUPoint class and get called by the client tier when changes are made as discussed earlier.

4.3.2 Extrude Implementation

An extrude, or extrusion, is the expansion of a 2D sketch into a 3D shape. It is the most common feature used to generate 3D shapes. Figure 4.17 shows an extrusion in CATIA. NX has an extrude feature that contains boolean options for uniting or subtracting extrudes from other

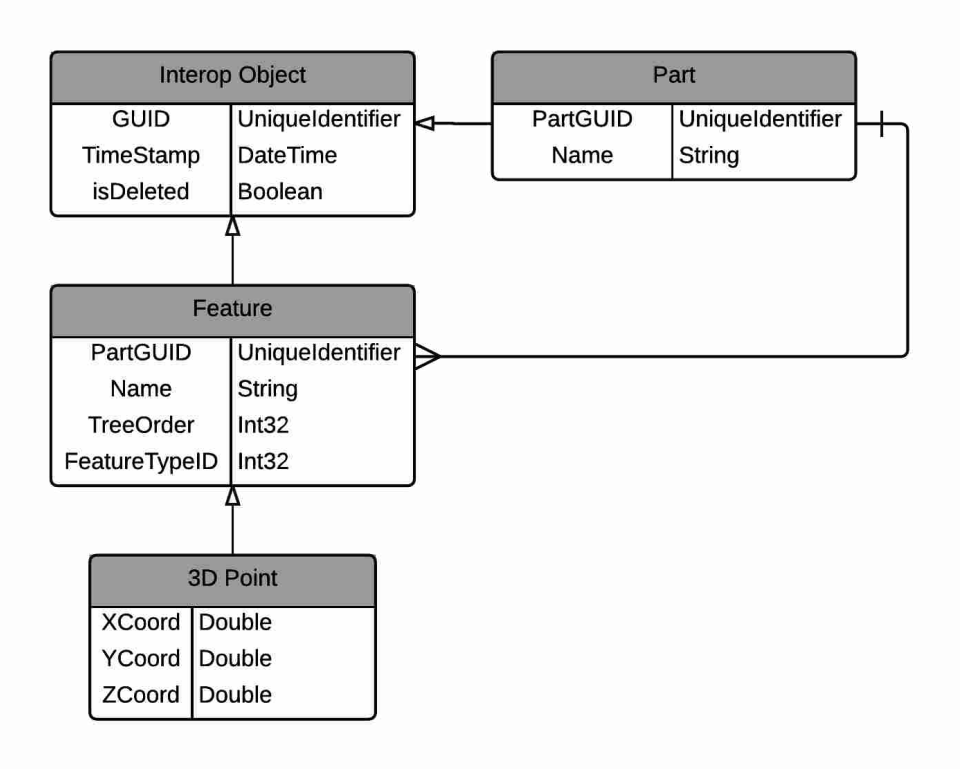


Figure 4.16: NPCF for 3D Point Feature

extrudes. CATIA has a pad feature for extrusion creation and a pocket feature for subtracting a shape from an existing shape. To support interoperability, the pad and pocket features of CATIA were combined to interface with the extrude feature of NX. The parameters which define an extrude feature are a direction, starting limit, ending limit, an IsPocket bool, and a sketch GUID. The NPCF architecture is given in Figure 4.18.

The direction, starting limit, and ending limit together determine the size and orientation of the extrude. The sketch GUID links the sketch being extruded to the extrude feature. The IsPocket property handles the difference between NX and CATIA's extrude representation and checks the created extrude to see if it is a subtraction or pocket instead of a normal extrude. Within the CATIA API is a pocket property that can be compared to the created extrude to see if it is a pocket. The NX API has a subtract property that can be compared to the extrude which performs the same function. Using these properties the type of extrude is determined before passing it to other clients.

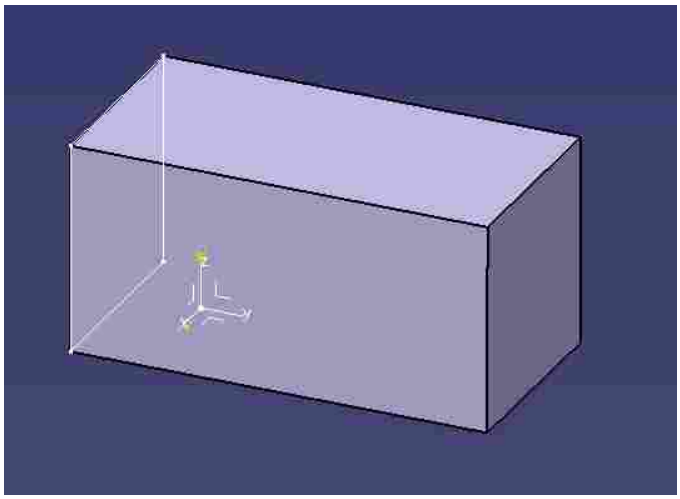


Figure 4.17: Extrude Feature in CATIA

An isExtrudeValid method was created which measures the surface area of the existing geometry without the extrude and then compares it the surface area of the geometry with the extrude. If the volumes do not agree then the extrude is rejected. This extra check ensures data consistency between clients. Just as was the case with sketches, CATIA immediately creates and pushes an extrude to the server when the pad dialog box is opened. This requires that edit functionality is implemented for CATIA for creation methods to work.

4.3.3 Revolve Implementation

A revolve is the shape created by rotating a 2D sketch about an axis. While an extrude can often be used instead of a revolve to create the same geometry, if the sketch being revolved is complex the revolve feature is very useful. A revolve feature created in NX is shown in Figure 4.19. Similar to the extrude feature, CATIA has two features that are represented by one in NX. In CATIA a revolution and a groove are used for addition and subtraction operations, respectively. NX has one revolve command with addition and subtraction boolean operations that perform the same function. The CATIA commands were combined so that a neutral format could be defined to translate between NX and CATIA. The specific parameters which define a revolve feature are a start angle, end angle, an IsGroove bool, and a sketch GUID. This is summarized in Figure 4.20.

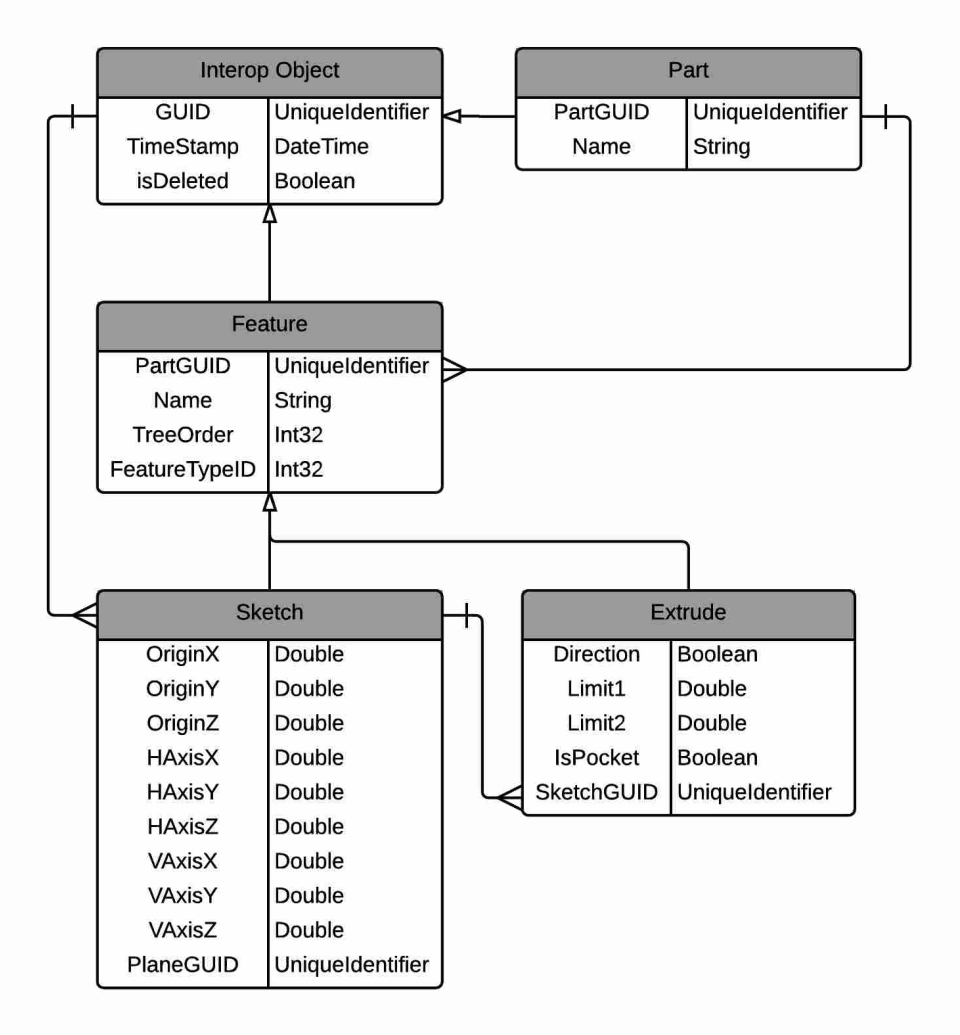


Figure 4.18: NPCF for Extrude Feature

The start and end angles define the boundaries of the revolve and the sketch GUID links the revolve to the sketch being revolved. The IsGroove property allows the revolution and groove commands to be compatible with the boolean operations in NX. Upon opening the revolve command in CATIA, a revolve is automatically sent to the server and pulled by other clients. This means that edit functionality must already be implemented on the CATIA side in order for creation to work, as was the case with the extrude feature. A revolve feature requires an axis to revolve the sketch around and a Csys Axis object was created to provide this function. A one-to-many relationship says that an axis can be associated with many different revolve features.

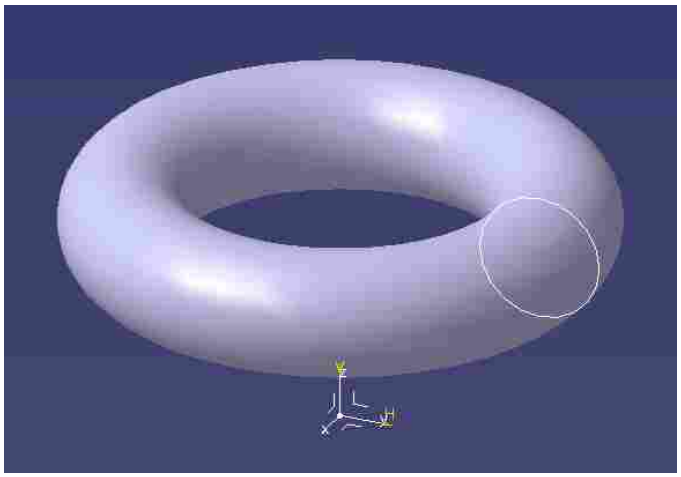


Figure 4.19: Revolve Feature in CATIA

An isRevolveValid method performs the same function as the isExtrudeValid method discussed earlier and makes sure that any revolve created has volume before it is sent to the server. If this method fails a message is presented to the user giving them the chance to change the revolve parameters and try again.

4.4 Feature Capabilities and Limitations

All the features discussed can be created, edited, and deleted in either CAD system during modeling. The test methods summarized in Table 3.3 have all passed through this implementation. Multiple users using either NX or CATIA can collaborate on models without restrictions. While this research has focused on many of the features which have been implemented, a few more have been implemented by others and include datum CSYS, datum plane, 3D line, and 3D spline.

Some of the existing limitations of the software include unexpected bugs and a lack of conflict resolution logic. The software bugs include occasional program crashes and features that are created or edited sometimes not being pulled to the other clients. It is possible for two people to try and edit the same feature at the same time, resulting in conflicts. Ammon Hepworth and Robert Moncur implemented conflict resolution and feature locking in NXConnect and similar ideas could be applied to the interoperability software [30] [34]. More work will be done to make the software more robust but the existing solution is effective.

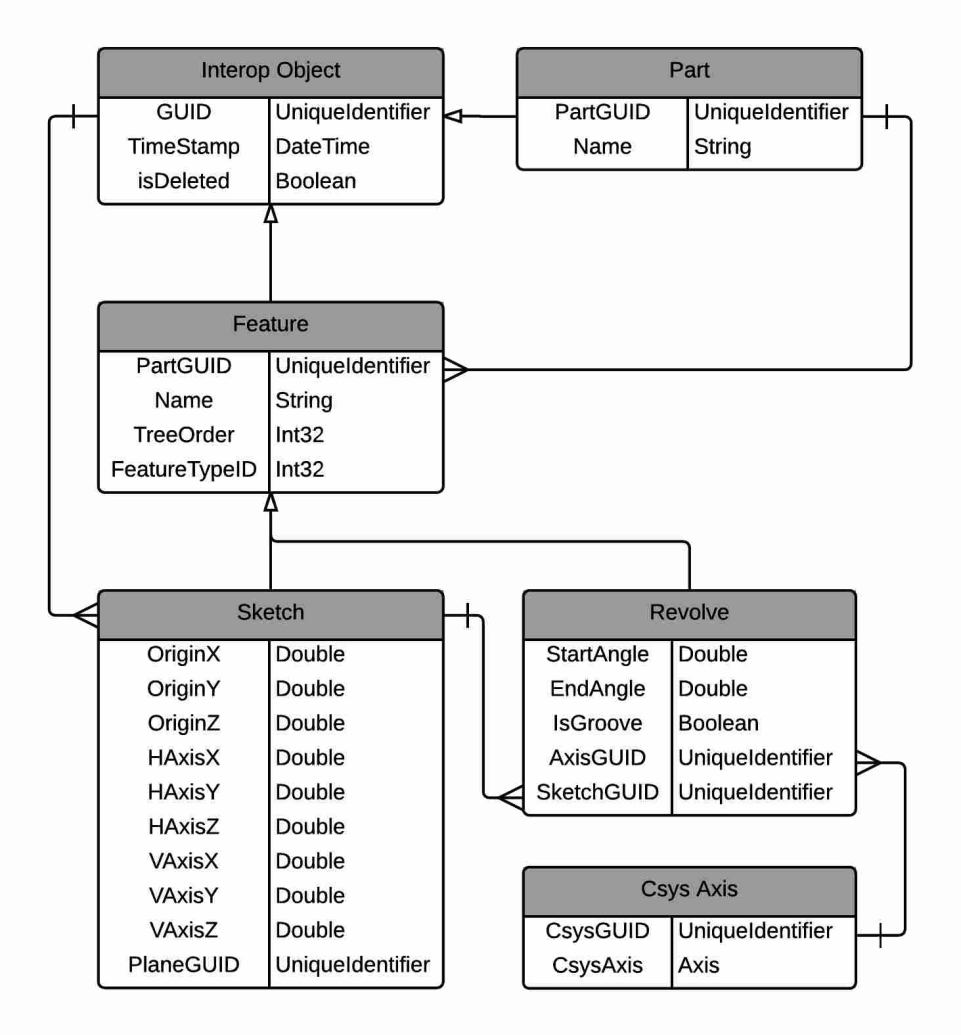


Figure 4.20: NPCF for Revolve Feature

4.5 Complex Modeling Demonstration

The NPCF for the 2D point, 2D line, 2D arc, 2D circle, 2D spline, 3D point, extrude, and revolve features have been implemented in the multi-user interoperability program and work between NX and CATIA CAD systems. Fig. 4.21 summarizes the NPCF of the aforementioned features. This research has focused on defining the neutral representation of the most common and foundational features needed to generate CAD models. The capabilities of the software will be demonstrated by the modeling of a complex part.

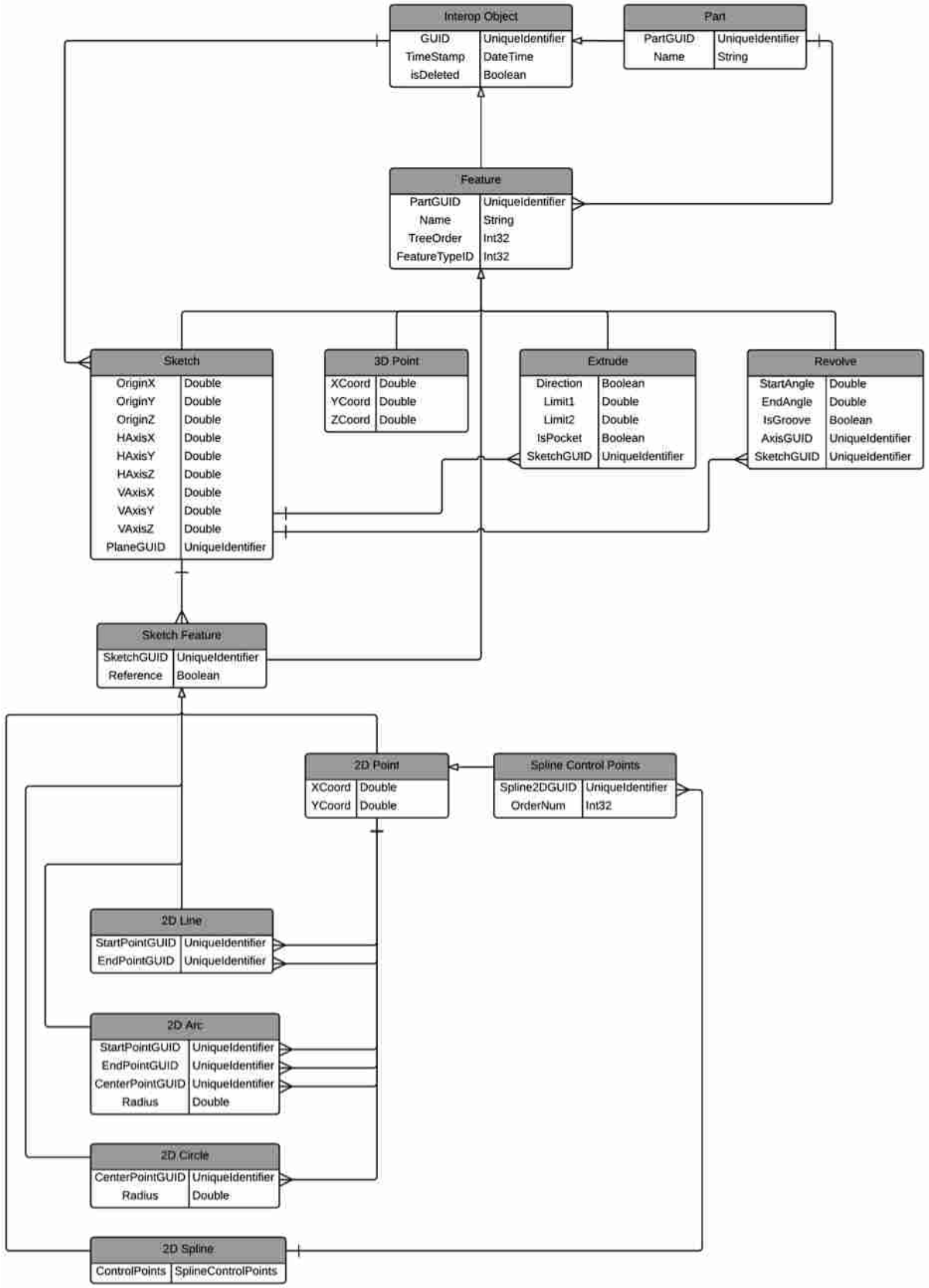


Figure 4.21: Overall NPCF of Interoperable CAD Features

Table 4.1: Components of the Piston and Connecting Rod Assembly

Component	Quantity
Connecting Rod Lower	1
Connecting Rod Lower Bearing	1
Connecting Rod Upper Bearing	1
Connecting Rod Upper	1
Connecting Rod Bushing	1
Piston Pin	1
Snap Ring	2
Piston Ring	3
Piston	1

A piston and connecting rod were modeled by four users on four different computers. Two users used NX and the other two used CATIA. All users were given instructions for modeling their individual parts of the model and allowed to collaborate prior to beginning modeling. The piston and connecting rod part is actually an assembly made up of nine unique components which are listed in Table 4.1. All nine part files were added to the assembly before modeling started and different users modeled different components in the part files. View Appendix A to see each component's part file and the instructions used during the demonstration. While these users were co-located, similar sessions could occur between geographically dispersed users by using a chat or video conferencing service. All the features implemented in this research were used in the modeling of the piston and connecting rod.

Figure 4.22 is a screenshot near the beginning of the modeling session. All users are working at the same time creating features on their own CAD clients. As a user exits their sketch the changes made by other users are pulled to their client in real time so that the model on all screens stays in sync. For example, Figure 4.23 shows that a piston was created on a CATIA client in the lower right quadrant of the image and Figure 4.24 shows the piston appearing on the upper left NX client after they have exited a sketch. The piston also appeared on the other NX and CATIA client when they exited the sketches they were working on.

The revolve feature was used to create the piston and the extrude feature was used to create the connecting rod. All sketch features were needed to generate the complex shapes of the part. Some users are creating sketches at the same time other users are creating 3D geometry. Updates

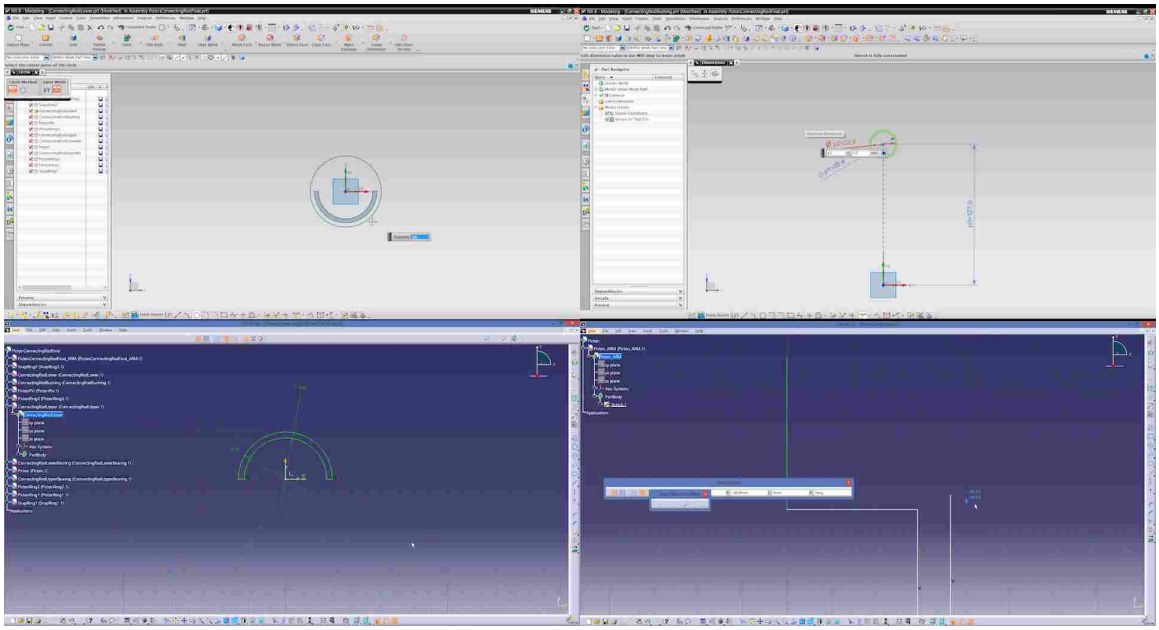


Figure 4.22: Interoperable Modeling Session with 2 NX Clients and 2 CATIA Clients

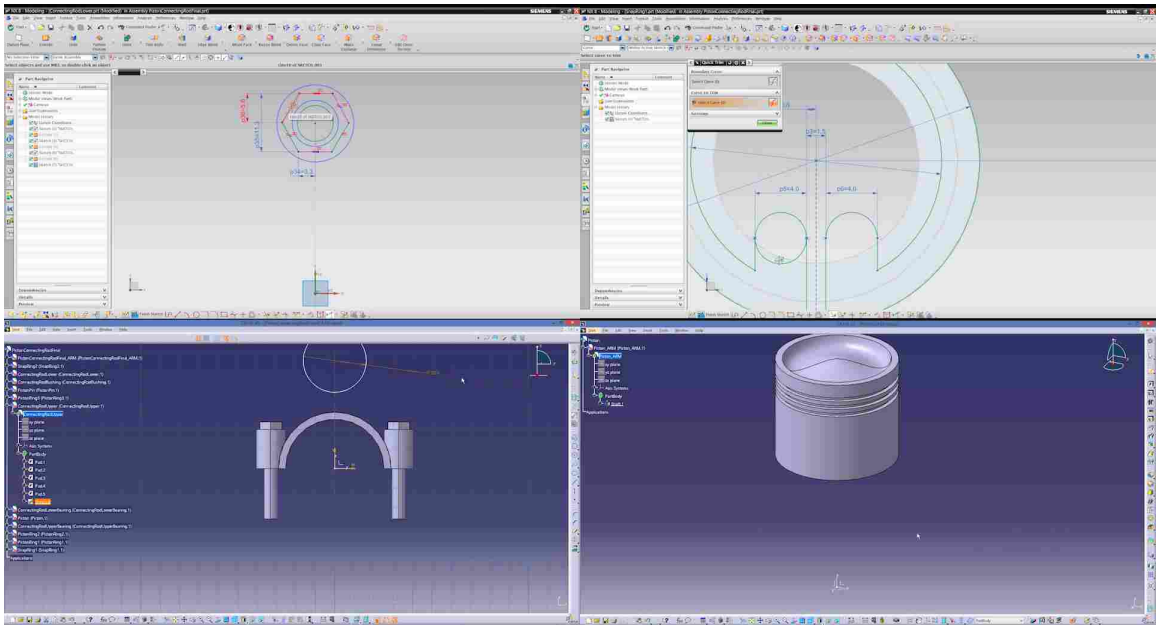


Figure 4.23: Interoperable Modeling Session before Piston Transfer

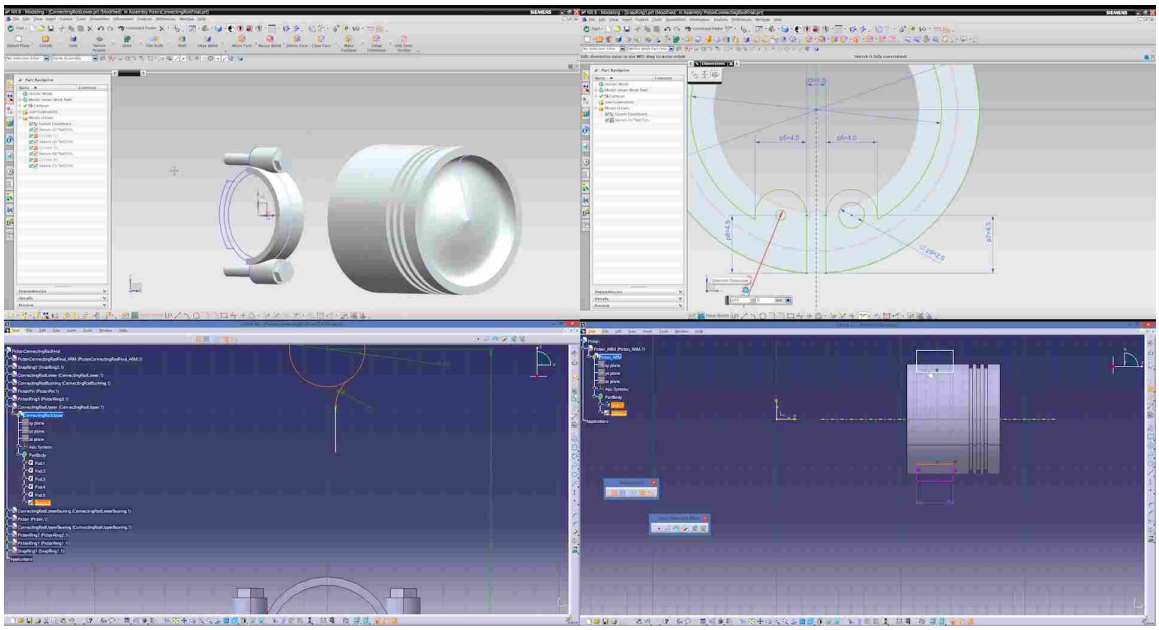


Figure 4.24: Interoperable Modeling Session after Piston Transfer

are placed in a queue on each client while creating a feature and the updates are applied after the feature is finished. A view of the completed piston and connecting rod model is given in Figure 4.25 and shows the same model on all four users' screens. A closeup of the finished part in NX and CATIA is shown in Figure 4.26.

To illustrate how this approach is much more effective at exchanging data between NX and CATIA than the typical approach today, the piston and connecting rod was modeled in single user NX and then exported as an IGES file. The IGES file was then imported into CATIA to see how successful the translation was. Figure 4.27 compares the feature trees in NX to an imported IGES file in CATIA. The feature trees are boxed in red. Although the models look about the same in each system, the feature tree in NX shows an assembly with the components of the piston and connecting rod model. Within each of these components is a more detailed feature tree containing editable sketches, extrudes, and revolves. The feature tree in CATIA only contains points, lines, arcs, and surfaces that cannot be edited. In order to work with the IGES model requires that the surfaces be joined and a solid created, which can take hours for a model of this complexity. Neutralizing the data using the NPCF developed in this research allows data to be exchanged in a much more useful way.

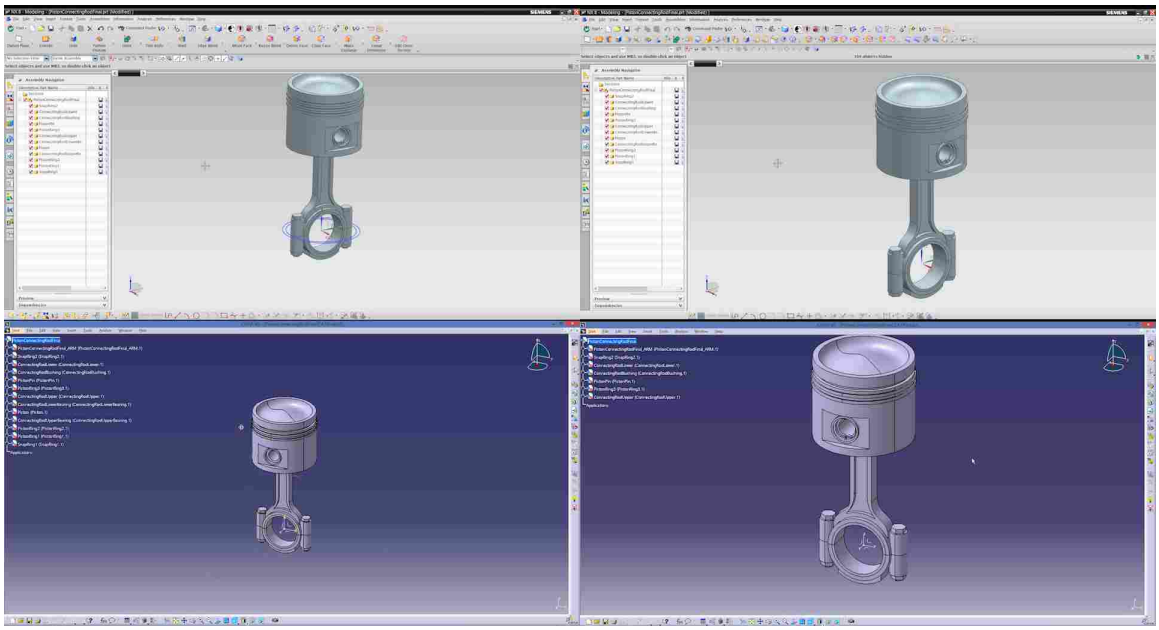


Figure 4.25: Interoperable Modeling Session after Completion

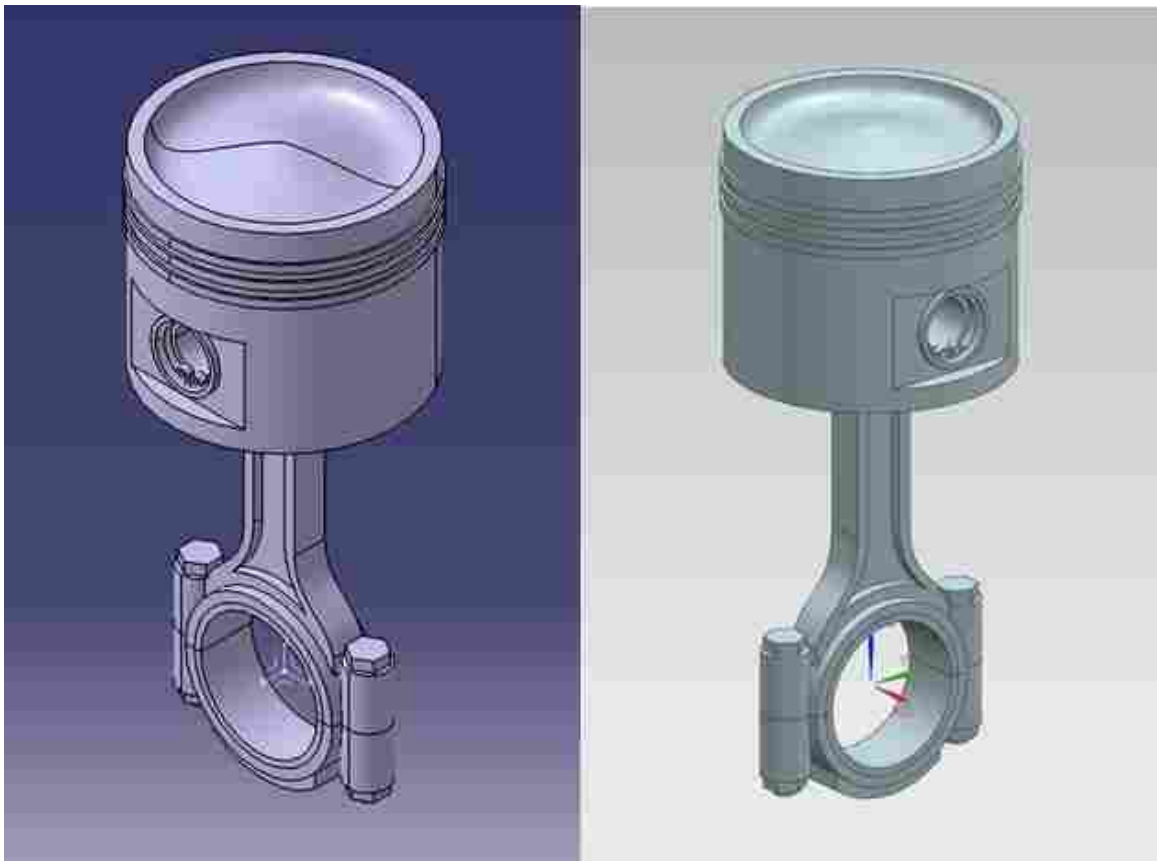


Figure 4.26: Piston and Connecting Rod Model: CATIA (left) and NX (right)

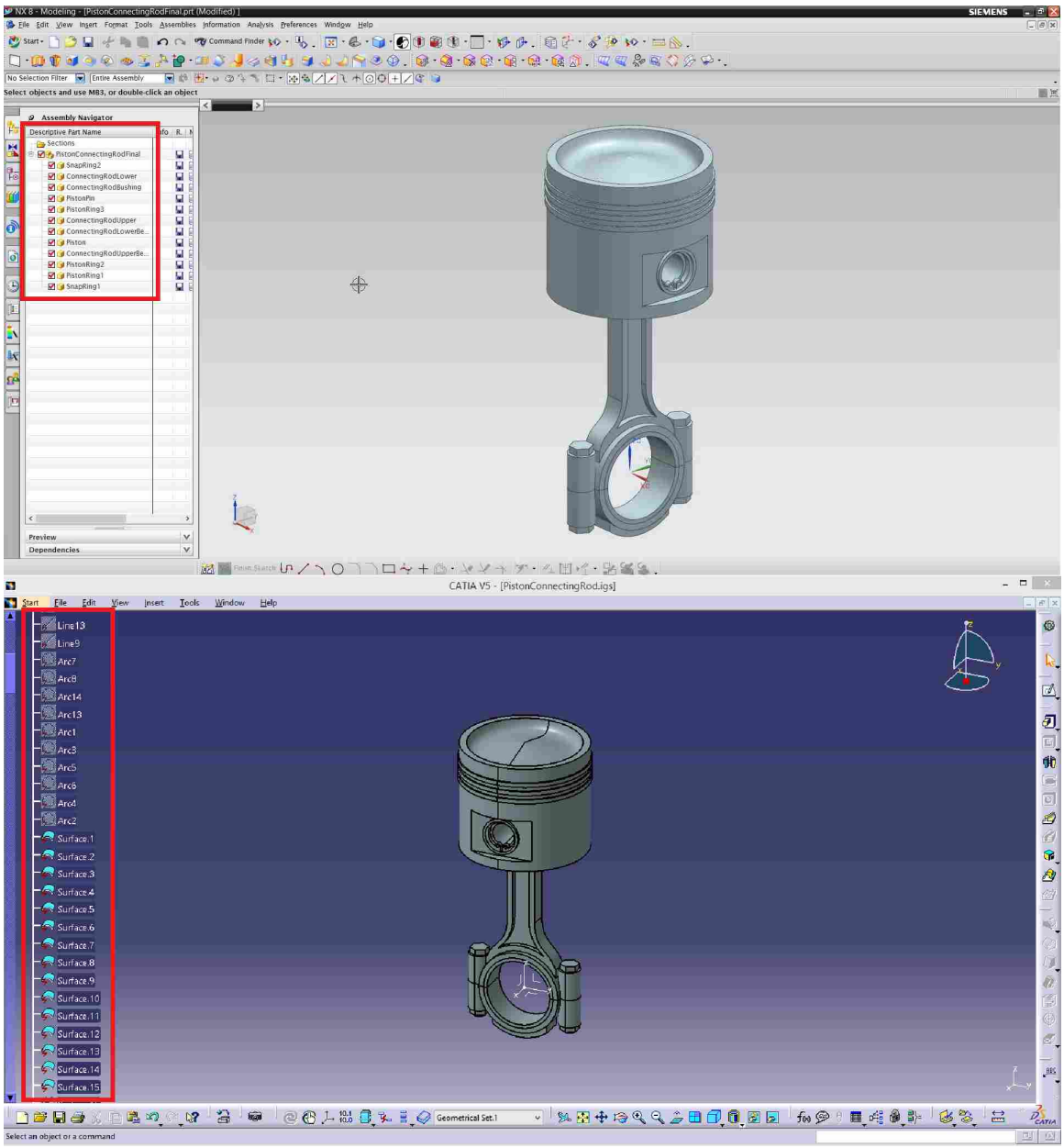


Figure 4.27: Feature Tree Comparison between NX (top) and CATIA IGES Import (bottom)

CHAPTER 5. SUMMARY

The objective of this research was to define a neutral parametric canonical form (NPCF) for basic CAD features that supports multi-user modeling and includes creation, editing, and deletion functionality. The current implementation supports modeling between NX and CATIA CAD systems. The interoperability problem consists in feedback loops in the design process due to the use of insufficient neutral data formats for exchanging CAD data between heterogeneous CAD systems. Solutions exist but none of them define a neutral data standard that supports multi-user CAD modeling. The definition of a standard is needed for data conversion between CAD systems to be consistent. Replacing the existing single user environment of CAD systems with a multi-user modeling environment improves collaboration in the design process by removing feedback loops and parallelizing the process. The client-server architecture of the program makes multi-user collaboration possible and the NPCF defined in this research enables the data to be transferred successfully. The neutral standard was defined by using journaling in the CAD system to understand the parameters used to create specific CAD features. A database architecture was created to store these parameters so they could be accessed in real time during modeling. This represents a neutralization approach instead of a translation approach because the data persists in the program. The NPCF was defined for the 2D point, 2D line, 2D arc, 2D circle, 2D spline, 3D point, extrude, and revolve features. The effectiveness of the NPCF at performing its function was verified through a modeling demonstration where a piston and connecting rod assembly was modeled by four users, with two using NX and two using CATIA. The modeling demonstration was successful and the model was correctly stored and created in both CAD systems. This represents the first synchronous, multi-user interoperable CAD solution.

CHAPTER 6. CONCLUSION AND FUTURE WORK

The Neutral Parametric Canonical Form (NPCF) effectively supports multi-user CAD modeling between NX and CATIA. The feature set chosen can be created, edited, and deleted in both CAD systems, validating its usefulness. This approach has achieved more unrestricted multi-user interaction than previous solutions because each user can model in their own CAD system without needing to take turns working on the model. The modeling demonstration performed was successful, showing that the NPCF architecture is capable of supporting multi-user modeling as claimed. Model awareness between users has been achieved because updates made to the model appear on all users' screens in real time. This improves the design process by reducing feedback loops. 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. Boeing, which primarily uses CATIA, could also better interact with their suppliers who use NX, such as Pratt & Whitney.

The NPCF approach defines the initial architecture for a new data standard. Focusing on defining a standard instead of simply translating the data proved to be more useful because the data was accessible at any time while the program was running which allowed multi-user support. Past neutral formats that were successful had support from industry leaders and also focused on neutralizing the data. The NPCF approach presented in this thesis is funded by several industry leaders and successfully neutralizes the data and stores it in a database. Updates to a model can be made more easily because the architecture of the NPCF links features together so that edits are propagated downstream. The interoperability software also encapsulates multi-user homogeneous CAD modeling, meaning it includes the basic capabilities of NXConnect while adding the ability to interact with different CAD clients. This shows the versatility that defining a neutral format can provide.

Great effort is required to completely solve the interoperability problem. The current solution developed is functional but limitations exist. For instance, the union of CAD features needs to

be supported for greater interoperability but only an intersection of features is currently supported. Also, no logic is in place which verifies that the feature trees of each CAD client match and sometimes differences result. Unexpected bugs also occur during modeling and work is needed to make the program more stable and robust. While there is edit functionality the undo/redo options are currently not implemented. Finally, if two people attempt to edit the same feature at the same time there is no logic in place to lock a feature being edited by another user. This can cause inconsistencies in the model. In spite of these limitations the existing solution is effective and should continue to be improved and expanded.

REFERENCES

[1] Brunnermeier, S. B., and Martin, S. A., 1999. "Interoperability cost analysis of the us automotive supply chain.". 1

[2] Kemmerer, S. J., 1999. *STEP: the grand experience*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology. 5

[3] Basu, D., and Kumar, S. S., 1995. "Importing mesh entities through iges/pdes." *Advances in Engineering Software,* **23**(3), pp. 151–161. 8

[4] Gu, H., Chase, T. R., Cheney, D. C., Johnson, D., et al., 2001. "Identifying, correcting, and avoiding errors in computer-aided design models which affect interoperability." *Journal of Computing and Information Science in Engineering,* **1**(2), pp. 156–166. 8

[5] Chang, T., Wysk, R. A., and Wang, H., 2006. "Computer-aided manufacturing." *New Jersey: Pearson Education*. 8

[6] Marjudi, S., Fahmi, M., Amran, M., Abdullah, K. A., Widyarto, S., Amlya, N., Majid, A., and Sulaiman, R., 2010. "A Review and Comparison of IGES and STEP." In *World Academy of Science, Engineering and Technology 62*, pp. 1013–1017. 9

[7] Ranyak, P., 1994. "Application interface specification (ais), version 2.1." *Consortium for Advanced Manufacturing International (CAM-I). Integrity Systems, USA*. 9

[8] 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. 9, 10

[9] Rappoport, A., 2003. "An architecture for universal CAD data exchange." *Proceedings of the eighth ACM symposium on Solid modeling and applications - SM '03*, p. 266. 9, 12

[10] Proficiency http://www.transcendata.com. 9, 12

[11] Iyer, G. R., 2001. "Development of API-Based Interfaces to Enable Interoperability Between CAD Systems During Design Collaboration." PhD thesis, The University of Texas at Arlington. 9, 12

[12] Ganapathi, S., 2002. "A Software Model for Interoperability between Heterogeneous CAD Systems." PhD thesis, University of Texas at Arlington. 9, 12

[13] Aspire3d http://www.aspire3d.com/3dto3d. 9, 12

[14] 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), Nov., pp. 1171–1179. 10, 16

[15] Li, M., Yang, Y., Li, J., and Gao, S., 2003. "A preliminary study on synchronized collaborative design based on heterogeneous CAD systems." PhD thesis, Zhejiang University. 10

[16] Li, M., Gao, S., Li, J., and Yang, Y., 2004. "An Approach to Supporting Synchronized Collaborative Design Within Heterogeneous CAD Systems." *Volume 4: 24th Computers and Information in Engineering Conference*, pp. 511–519. 10

[17] Li, M., Gao, S., and Wang, C. C. L., 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. 10

[18] Dou, W., Song, X., and Zhang, X., 2009. "A Language of Neutral Modeling Command for Synchronized Collaborative Design among Heterogeneous CAD Systems." In *The 1st International Conference on Information Science and Engineering*, IEEE, pp. 12–15. 11

[19] Dou, W., and Song, X., 2013. "Operation Command Transformation of Synchronized Collaborative Design Upon Heterogeneous CAD Systems.". 11

[20] Song, X., Dou, W., and Zhu, J., 2010. "Implementation of collaborative design system upon heterogeneous CAD systems using a feature-based mapping set." *Proceedings of the 2010 14th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2010*, pp. 510–515. 11

[21] Chen, J. Y., Ma, Y. S., Wang, C. L., and Au, C. K., 2005. "Collaborative Design Environment with Multiple CAD Systems." *Computer-Aided Design,* **2**(March 2015), pp. 367–376. 11

[22] Zhang, X., and Dou, W., 2009. "An Approach of Constructing Neutral Modeling Command Set of Synchronized Collaborative Design upon Heterogeneous CAD Systems." PhD thesis, Nanjing Normal University, Sept. 11

[23] Seo, T.-s., 2005. "Sharing CAD models based on feature ontology of commands history." PhD thesis, Korea Institute of Science and Technology Information. 11

[24] Sun, L.-j., and Ding, B., 2009. "Heterogeneous CAD Data Exchange Based on Cellular Ontology Model." In *World Congress on Software Engineering*, IEEE, pp. 46–50. 11

[25] Tessier, S., 2011. "Ontology-Based Approach to Enable Feature Interoperability between CAD Systems." PhD thesis, Georgia Institute of Technology. 11

[26] Coretechnologie http://www.coretechnologie.com. 12

[27] Theorem solutions http://www.theorem.com. 12

[28] Hepworth, A. I., Nysetvold, T., Bennett, J., Phelps, G., and Jensen, C. G., 2014. "Scalable Integration of Commercial File Types in Multi-User CAD." *Computer-Aided Design and Applications,* **11**(4), July, pp. 459–467. 12

[29] 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), Mar., pp. 021008–1 to 021008–9. 12

[30] 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), Mar., pp. 141–152. 12, 42

[31] Cai, X., Li, X., He, F., Han, S., and Chen, X., 2012. "Flexible Concurrency Control for Legacy CAD to Construct Collaborative CAD Environment." *Journal of Advanced Mechanical Design, Systems, and Manufacturing,* **6**(3), pp. 324–339. 12

[32] Maher, M. L., and Rutherford, J. H., 1997. "A model for synchronous collaborative design using CAD and database management." *Research in Engineering Design,* **9**(2), pp. 85–98. 12

[33] Li, W., Ong, S., Fuh, J., Wong, Y., Lu, Y., and Nee, A., 2004. "Feature-based design in a distributed and collaborative environment." *Computer-Aided Design,* **36**(9), Aug., pp. 775–797. 16

[34] Moncur, R. A., Jensen, C. G., Teng, C.-C., and Red, E., 2013. "Data Consistency and Conflict Avoidance in a Multi-User CAx Environment." *Computer-Aided Design and Applications,* **10**(5), Jan., pp. 727–744. 42

APPENDIX A. MODELING DEMONSTRATION INSTRUCTIONS AND PART FILES

A.1 Piston and Connecting Rod Demo Instructions

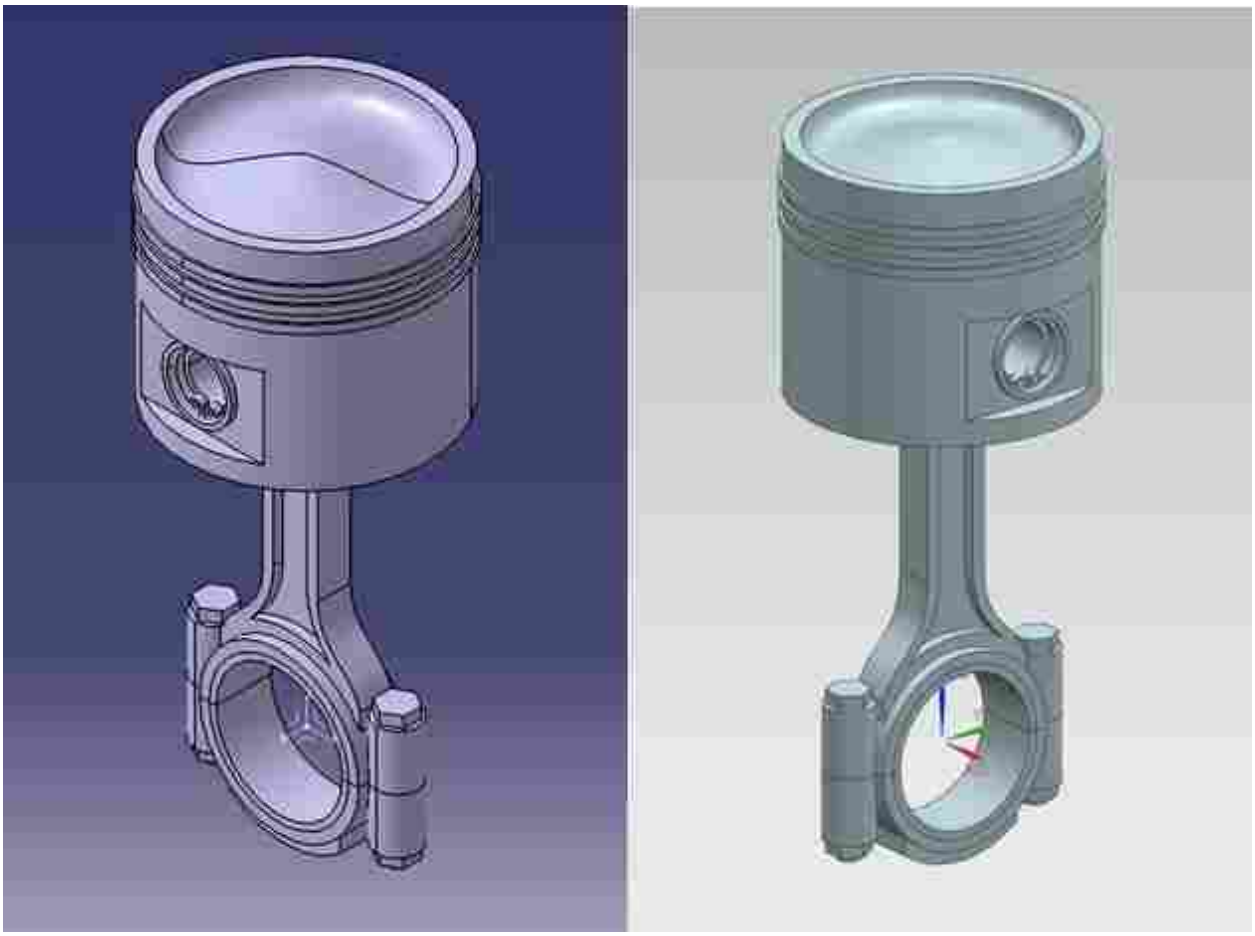
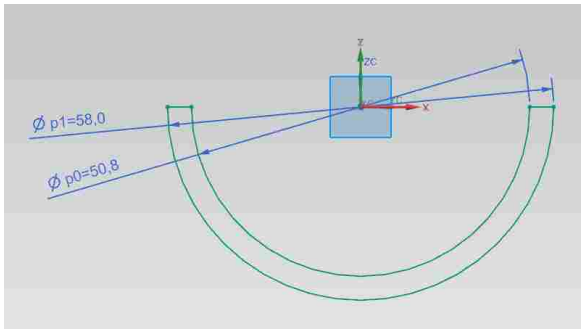


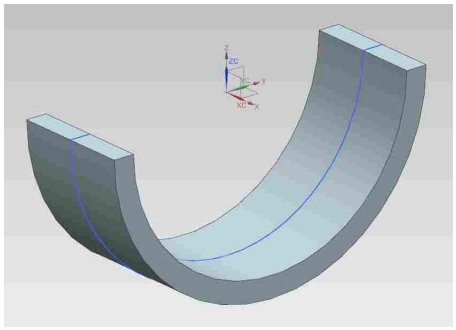
Figure A.1: Final Outcome of Instructions in CATIA (left) and NX (right)

User 1:
Connecting Rod Lower

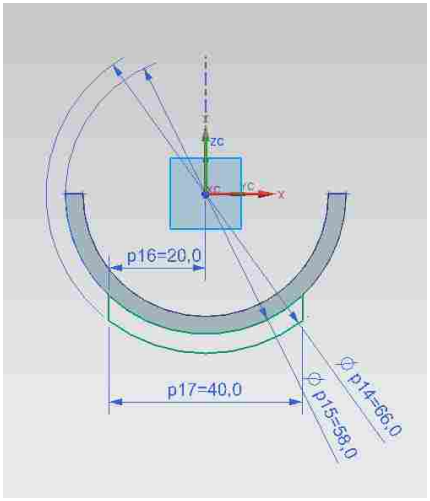
1. Create **Sketch** on yz plane



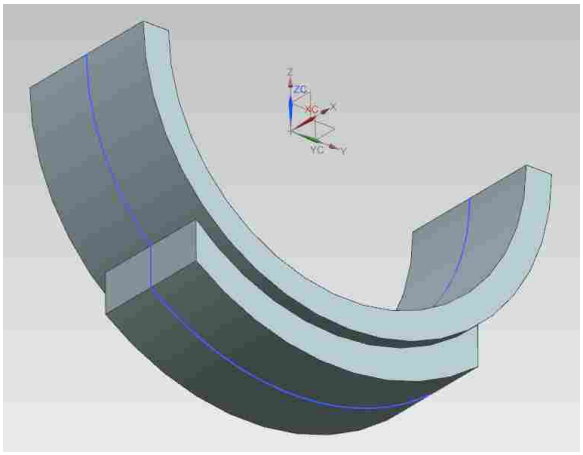
2. **Extrude** Sketch
    a. From -10.0 to 10.0



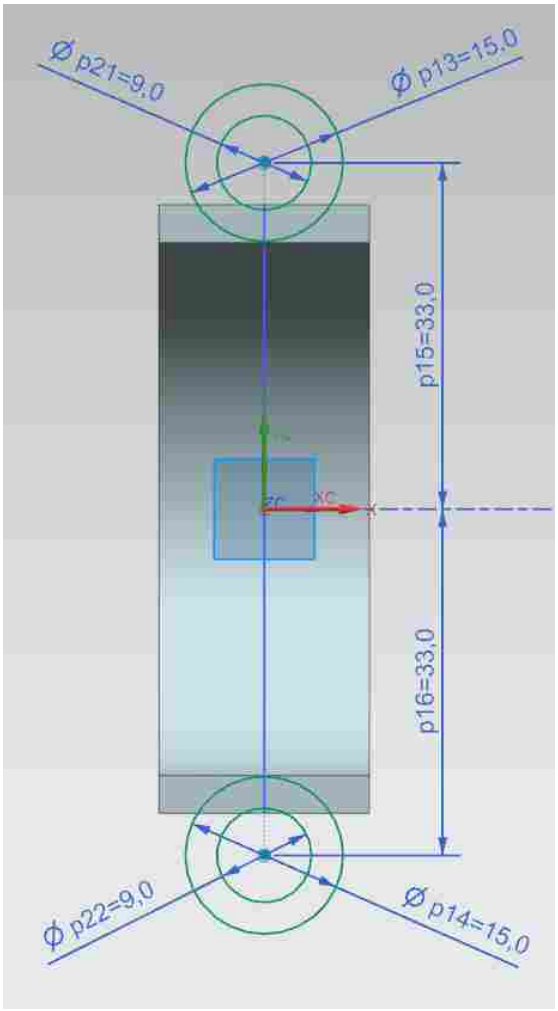
3. Create **Sketch** on yz plane



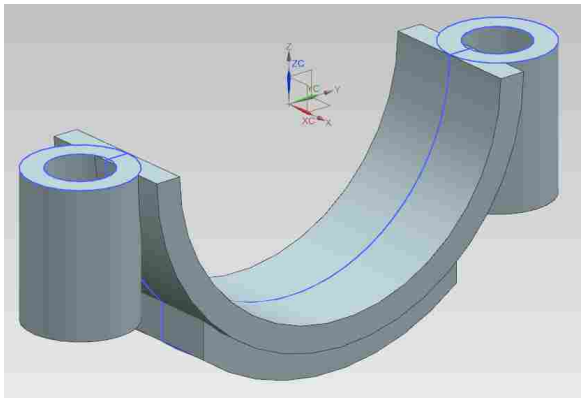
4. **Extrude** Sketch
    a. From -8.0 to 8.0



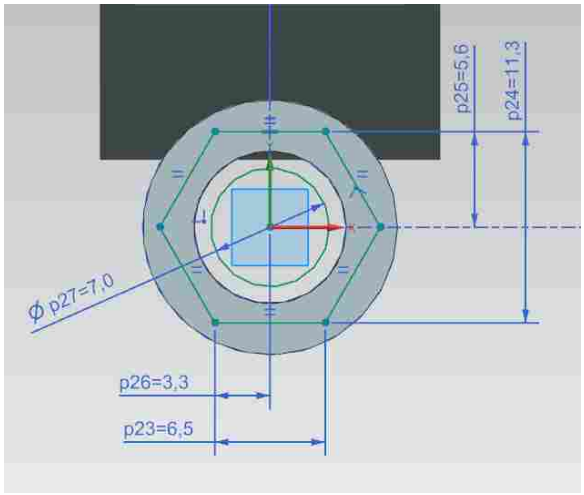
5. Create **Sketch** on xy plane



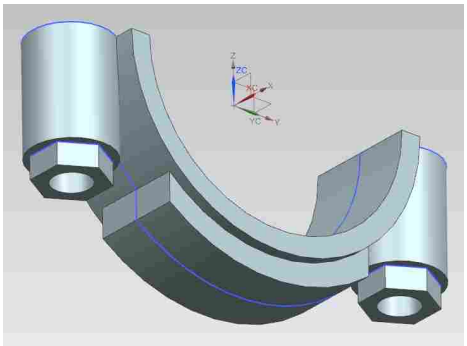
6. **Extrude** Sketch
    a. From 0.0 to -20.0



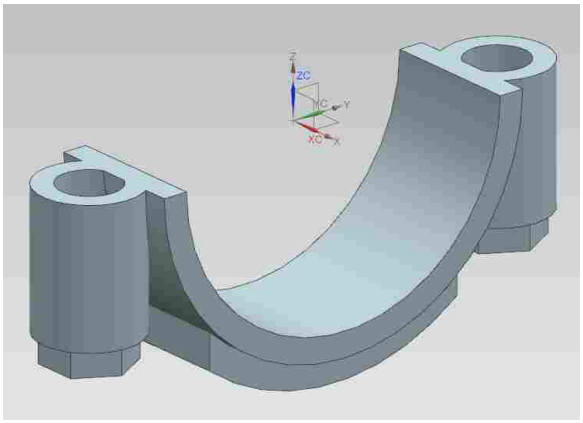
7. Create **Sketch** on xy plane (repeat for other side)



8. **Extrude** Sketch
    a. From -20.0 to -24.0

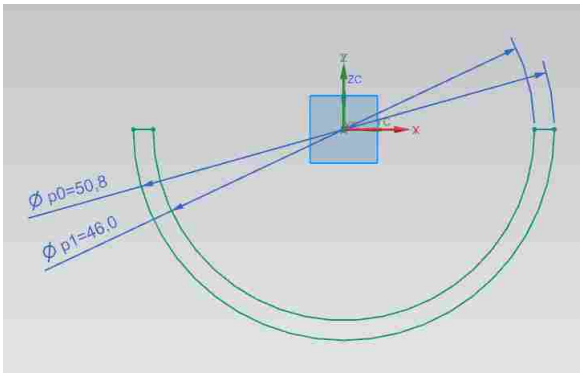


End Result

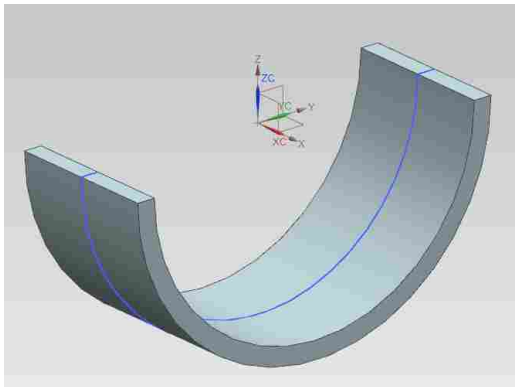


Connecting Rod Lower Bearing

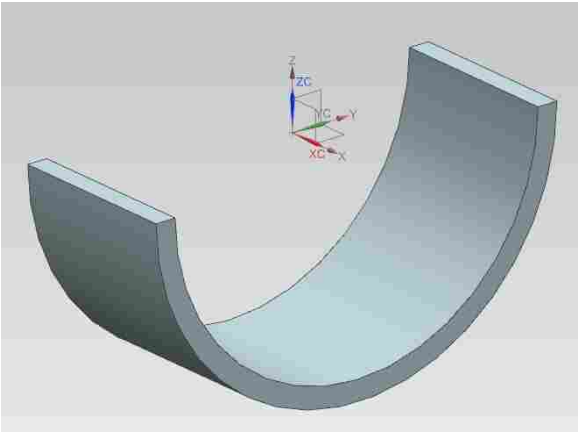
1. Create **Sketch** on yz plane



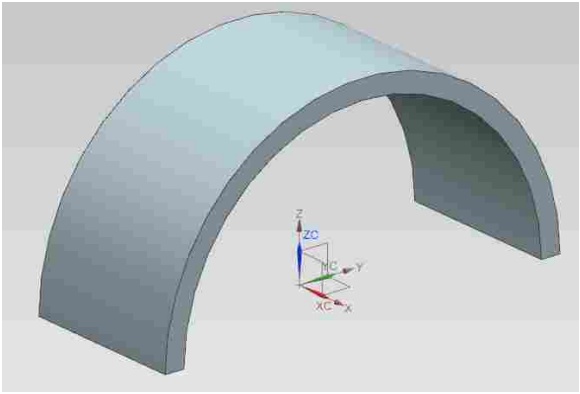
2. **Extrude** Sketch
    a. From -10.0 to 10.0



End Result

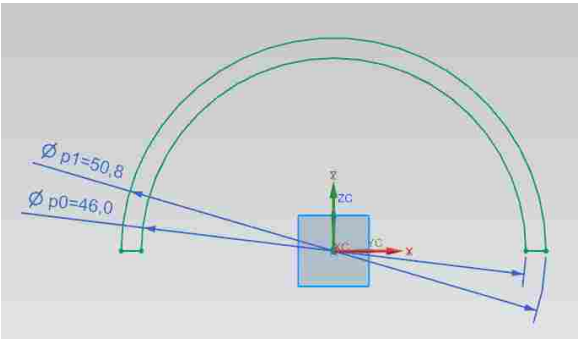


End Result

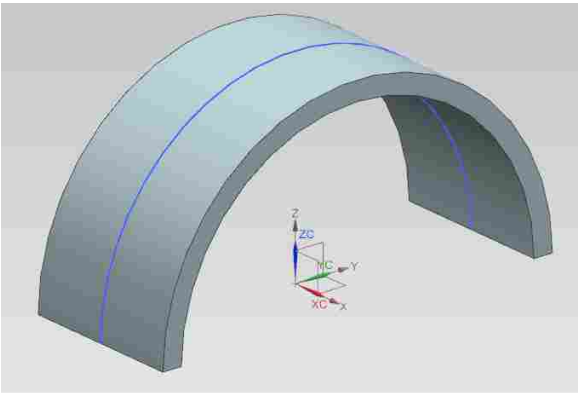


Connecting Rod Upper Bearing

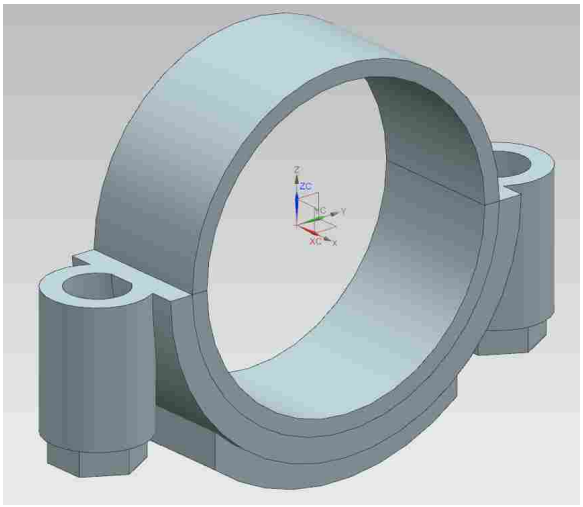
1. Create **Sketch** on yz plane



2. **Extrude** Sketch
    a. From -10.0 to 10.0

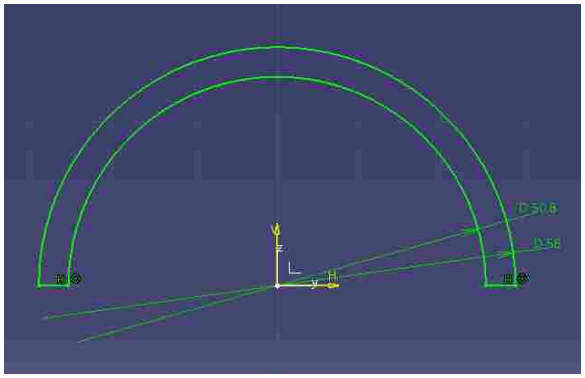


Summary for User 1

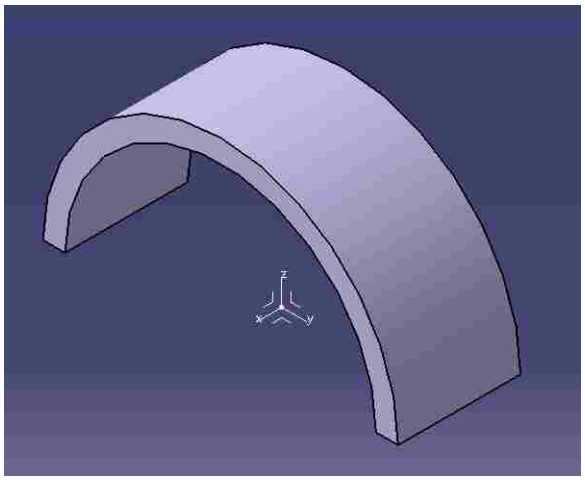


User 2:
Connecting Rod Upper

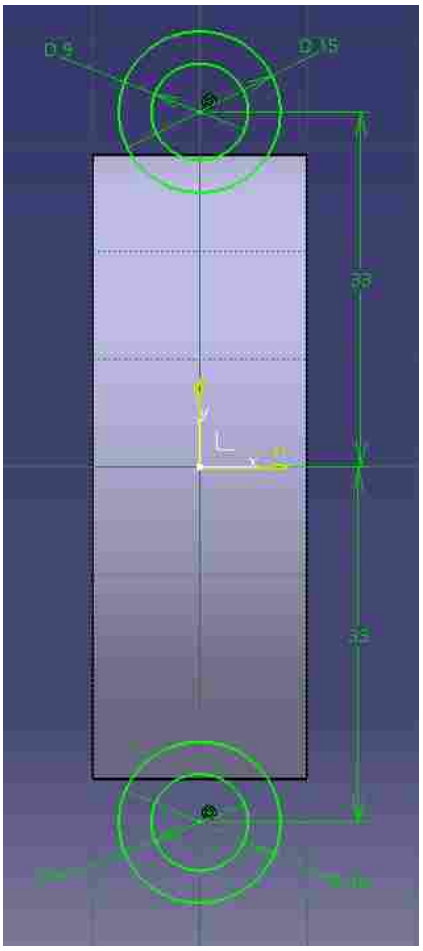
1. Create **Sketch** on yz plane



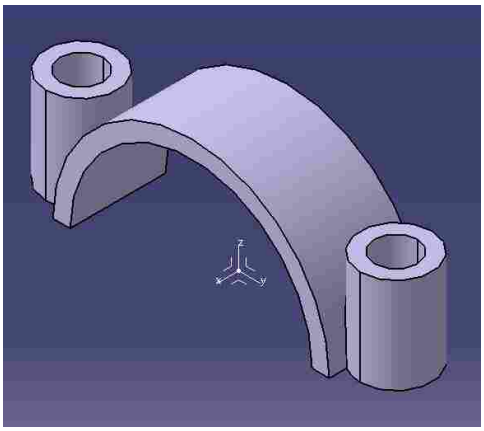
2. **Extrude** Sketch
    a. From -10.0 to 10.0



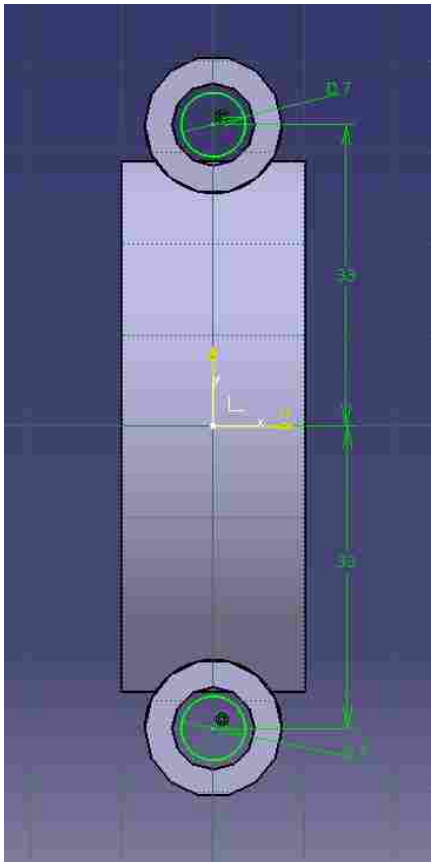
3. Create **Sketch** on xy plane



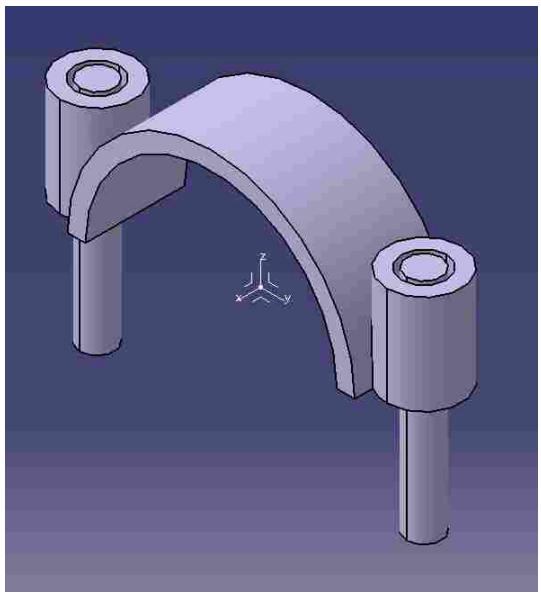
4. **Extrude** Sketch
    a. From 0.0 to 20.0



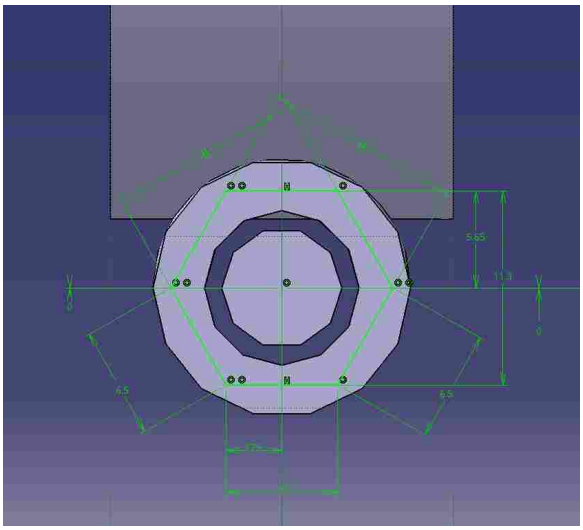
5. Create **Sketch** on xy plane



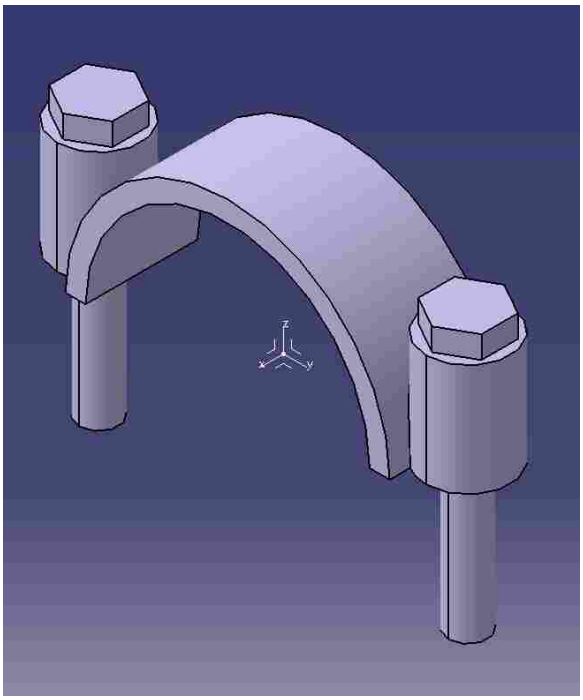
6. **Extrude** Sketch
    a. From -26.0 to 20.0



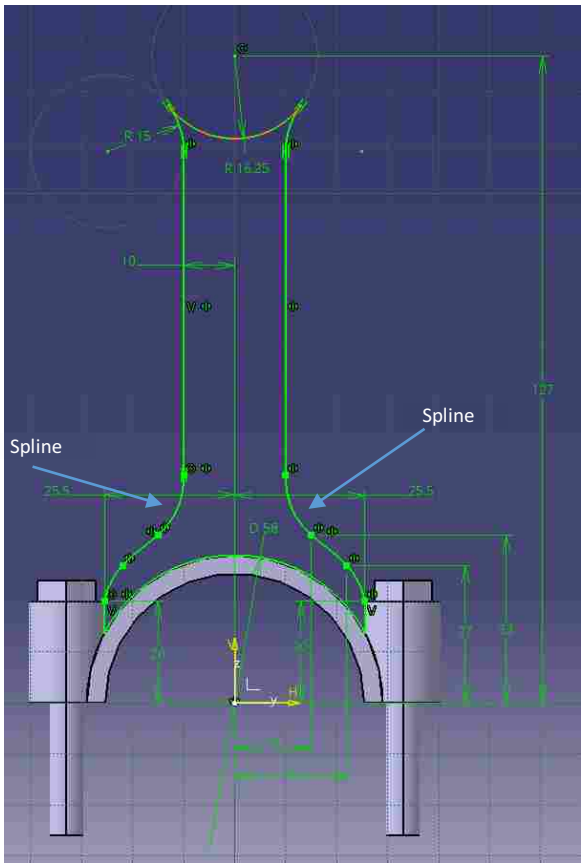
7. Create **Sketch** on xy plane (repeat for other side)



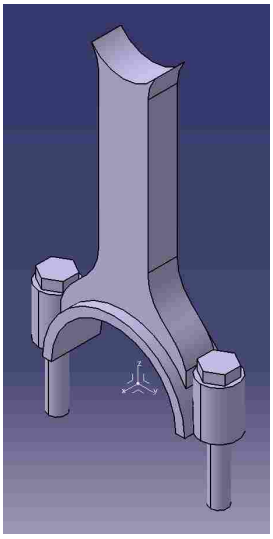
8. **Extrude** Sketch
    a. From 20.0 to 24.0



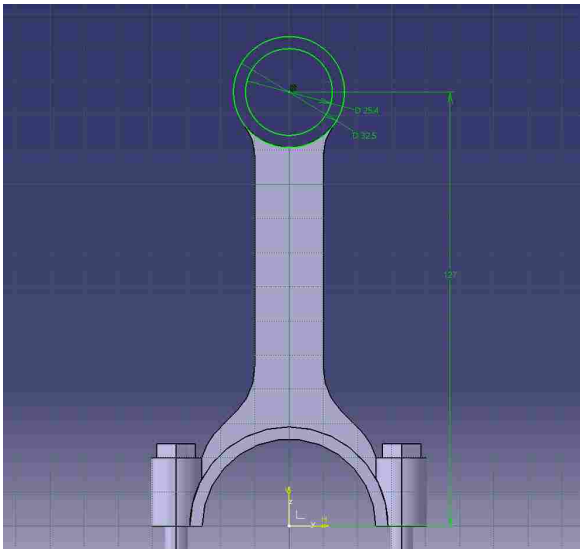
9. Create **Sketch** on yz plane (use spline as shown)



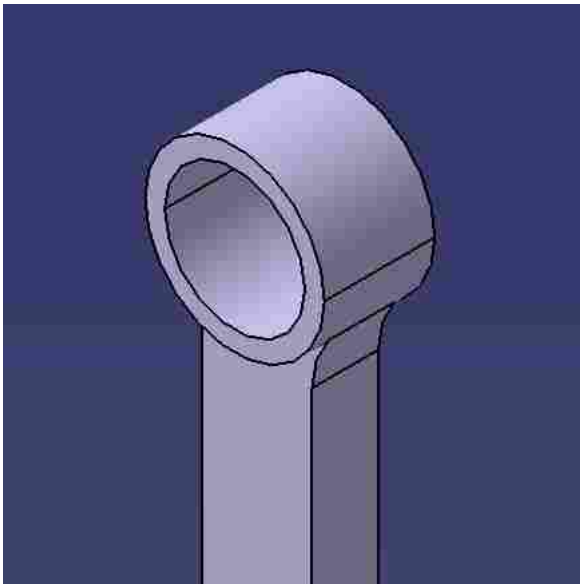
10. **Extrude** Sketch
    a. From -6.0 to 6.0



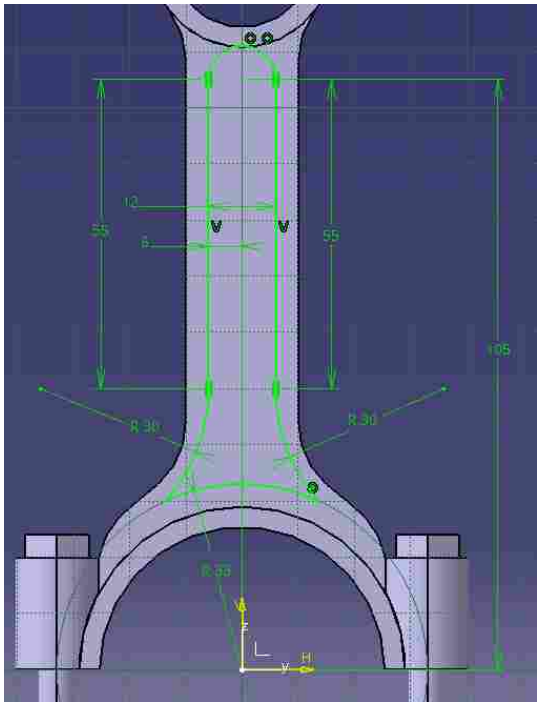
11. Create **Sketch** on yz plane



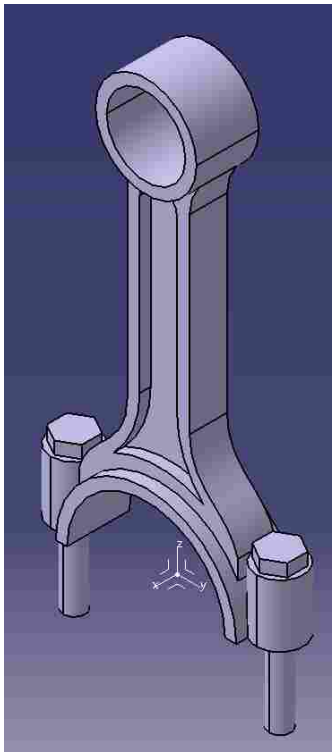
12. **Extrude** Sketch
    a. From -10.0 to 10.0



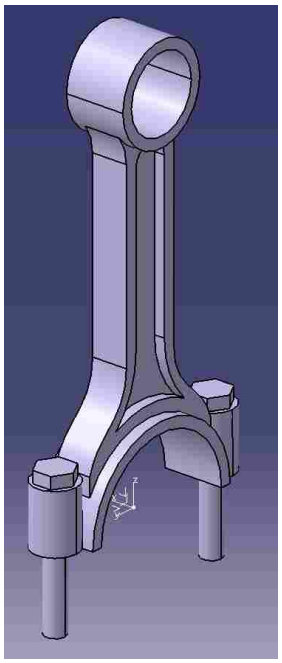
13. Create **Sketch** on yz plane



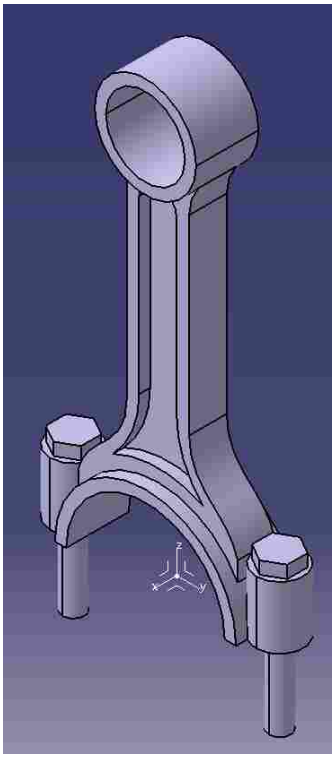
14. **Extrude** Sketch (subtract)
    a. From 2.5 to 6.0



15. **Extrude** Sketch (subtract)
    a. From -2.5 to -6.0



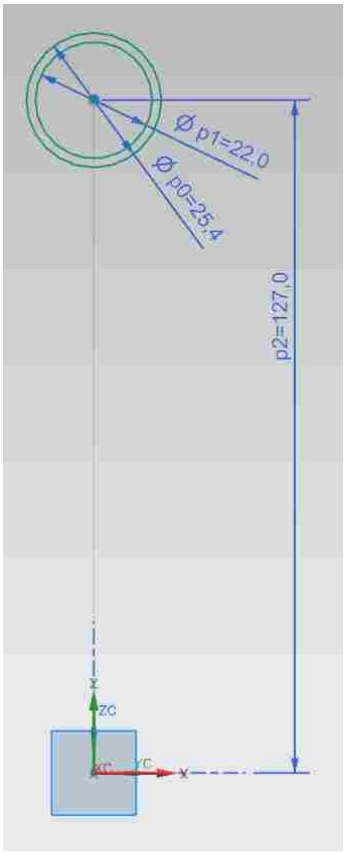
Summary for User 2



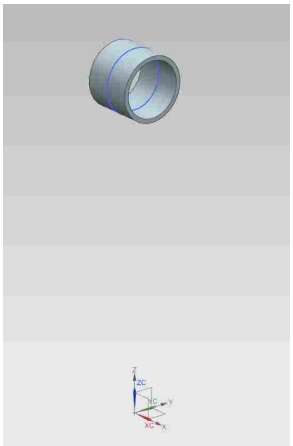
User 3:

Connecting Rod Bushing

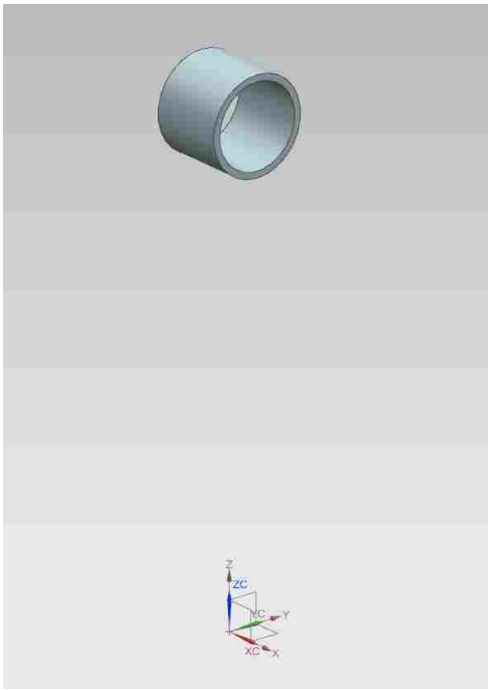
1. Create **Sketch** on yz plane



2. **Extrude** Sketch
    a. From -10.0 to 10.0

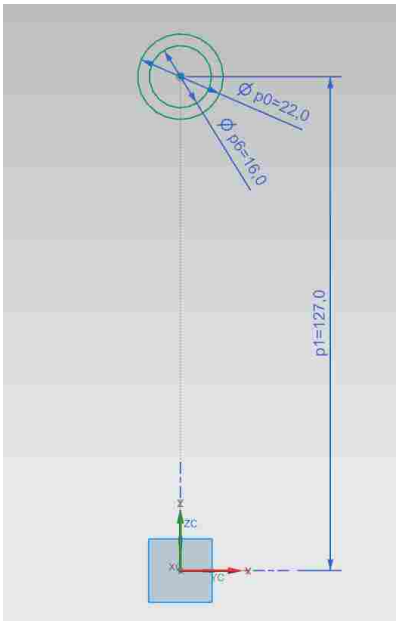


End Result

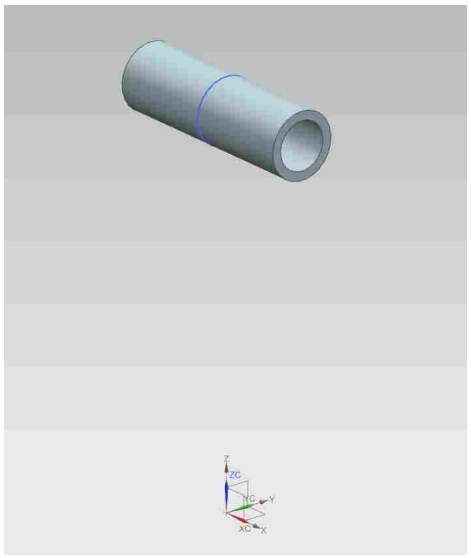


Piston Pin

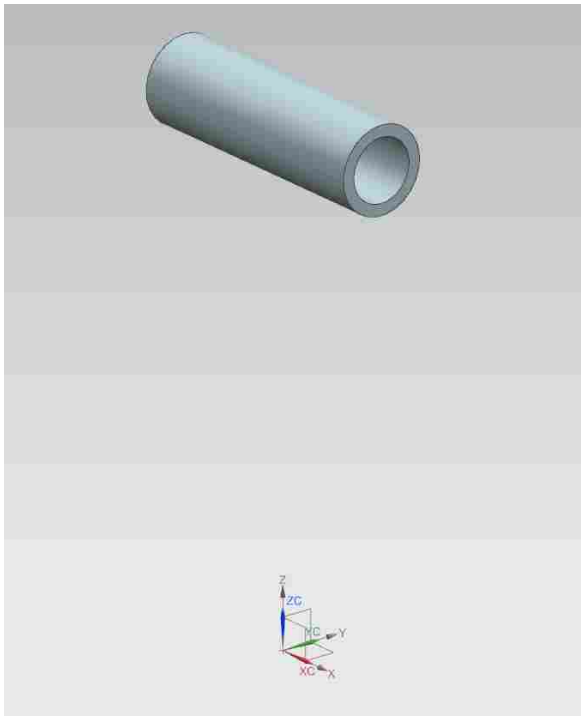
1. Create **Sketch** on yz plane



2. **Extrude** Sketch
    a. From -35.0 to 35.0

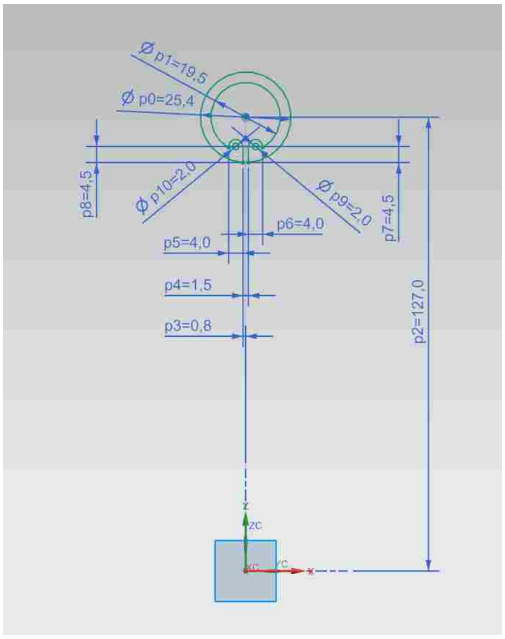


End Result

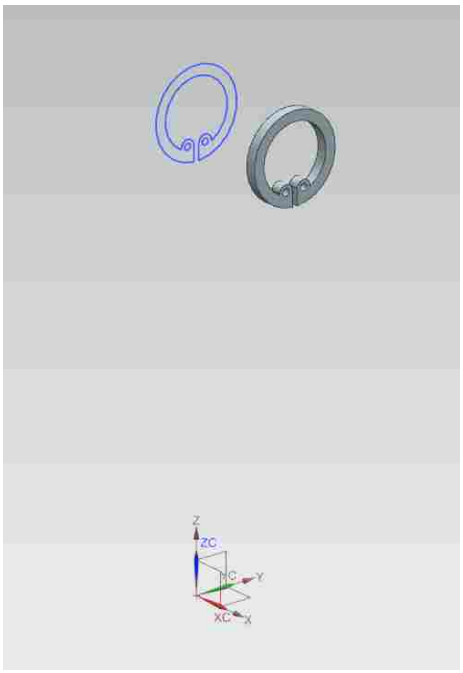


Snap Ring (2)

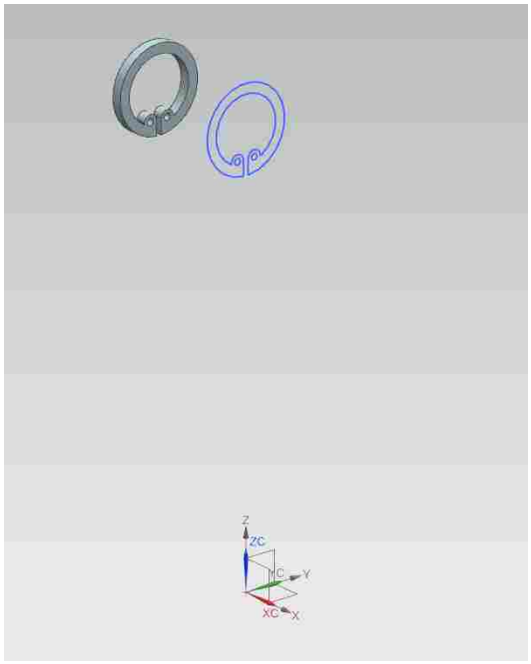
1. Create **Sketch** on yz plane



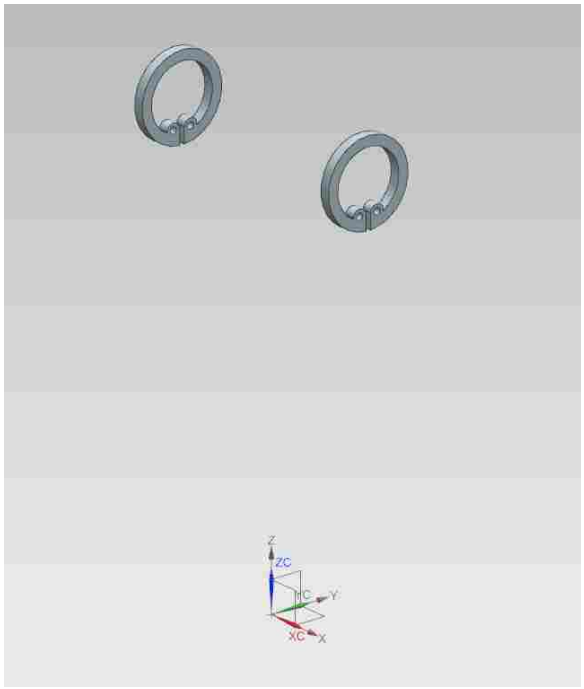
2. **Extrude** Sketch (1st time)
    a. From 35.0 to 38.0



3. **Extrude** Sketch (2nd time)
    a. From -35.0 to -38.0

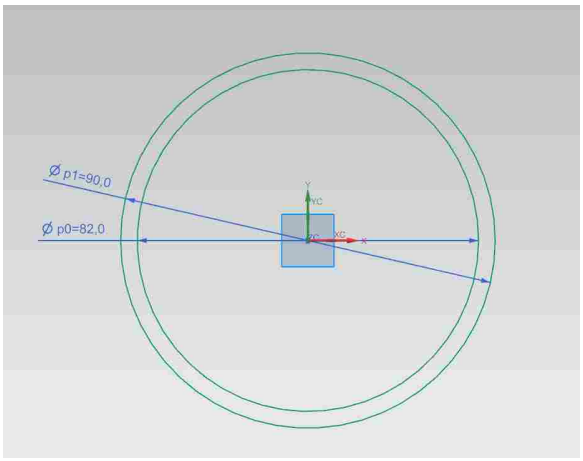


End Result

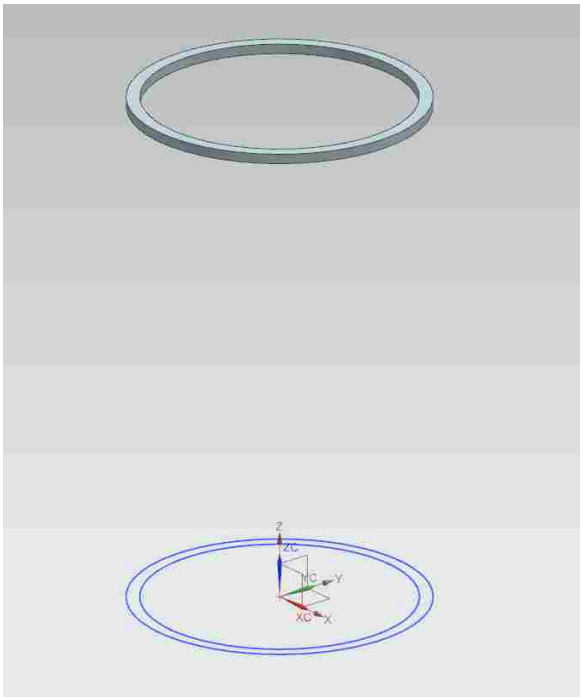


Piston Ring (3)

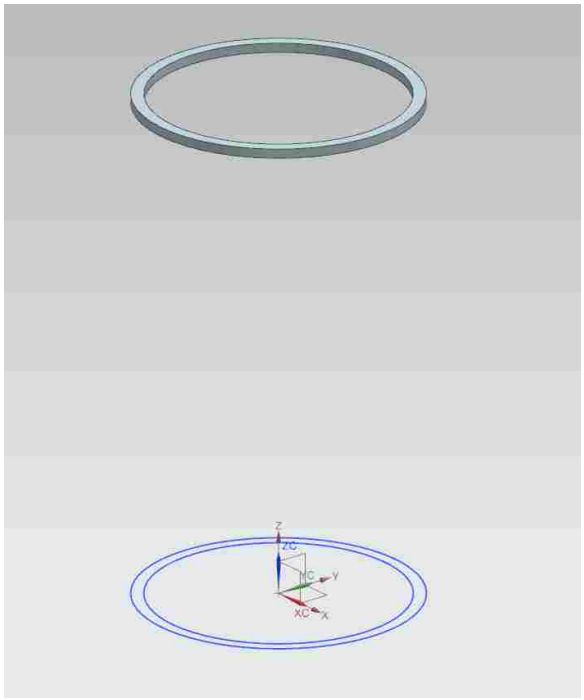
1. Create **Sketch** on xy plane



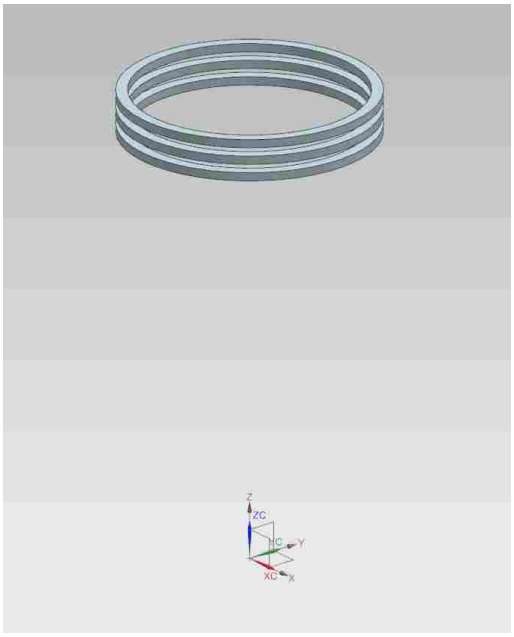
2. **Extrude** Sketch (1st time)
    a. From 155.0 to 158.0



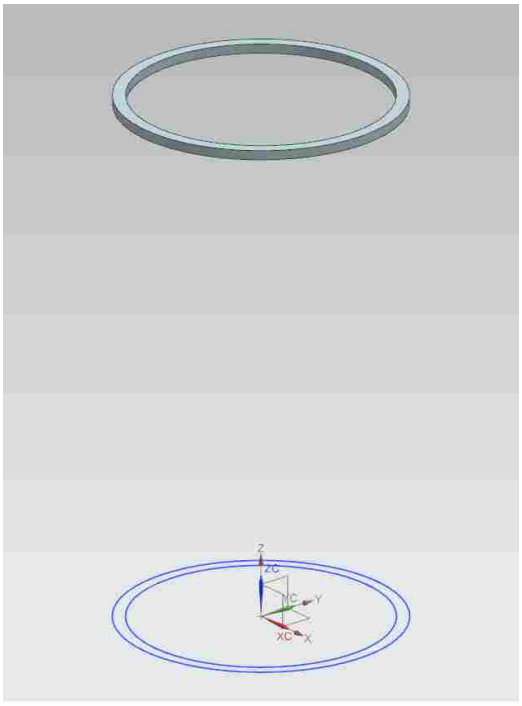
3. **Extrude** Sketch (2nd time)
    a. From 161.0 to 164.0



End Result



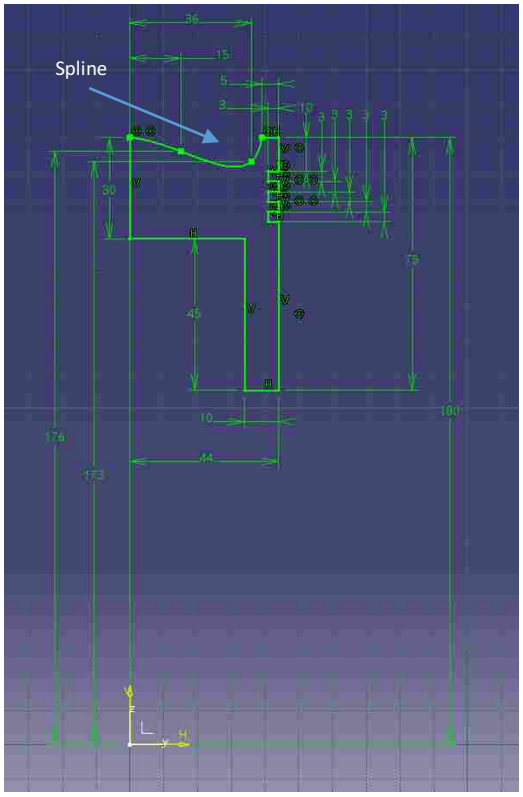
4. **Extrude** Sketch (3rd time)
    a. From 167.0 to 170.0



Summary for User 3

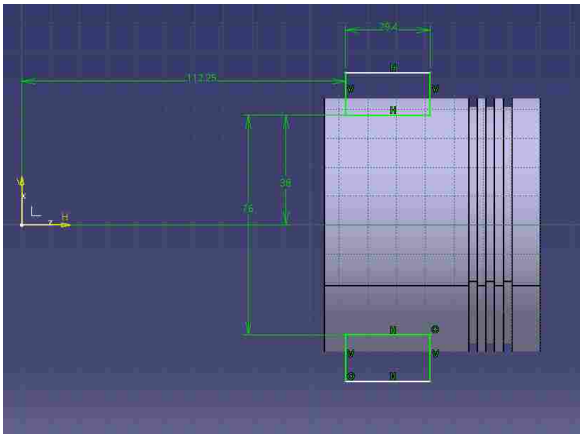
User 4:
Piston

1. Create **Sketch** on yz plane (use spline as shown)

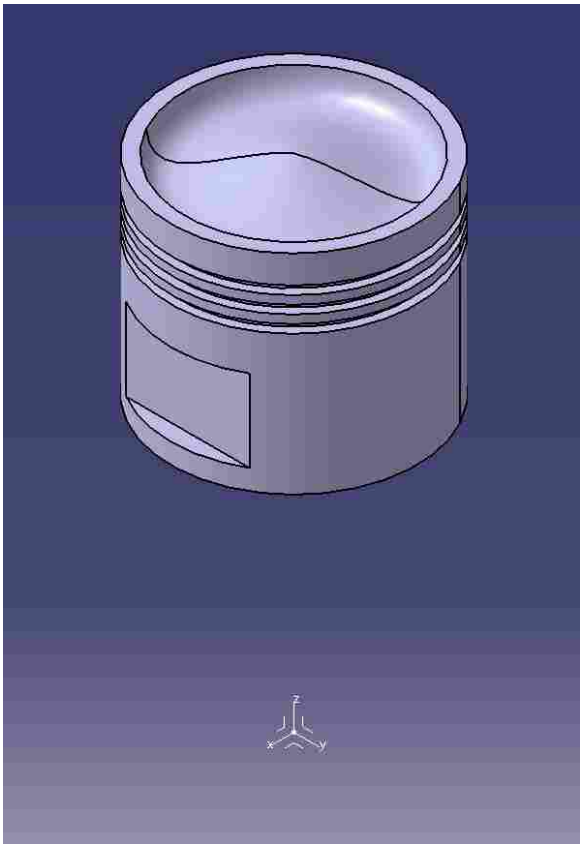


2. **Revolve** Sketch
    a. From 0.0 to 360.0

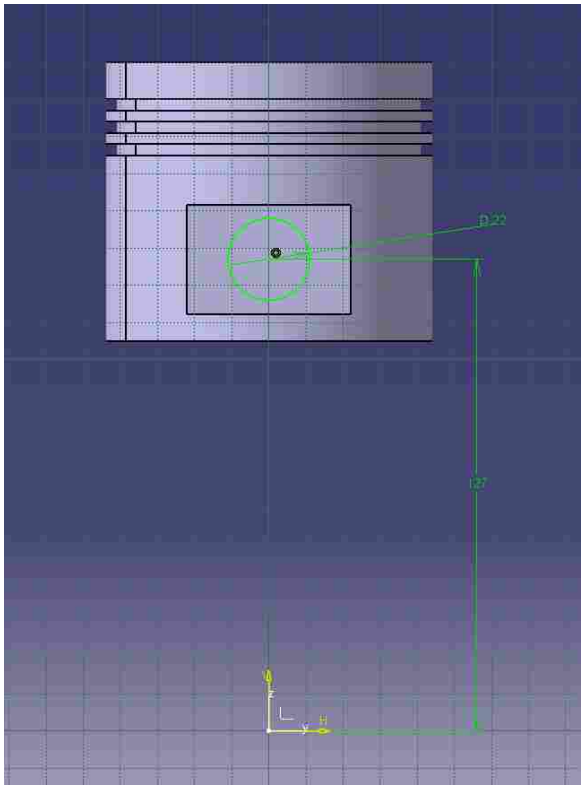
3. Create **Sketch** on xz plane



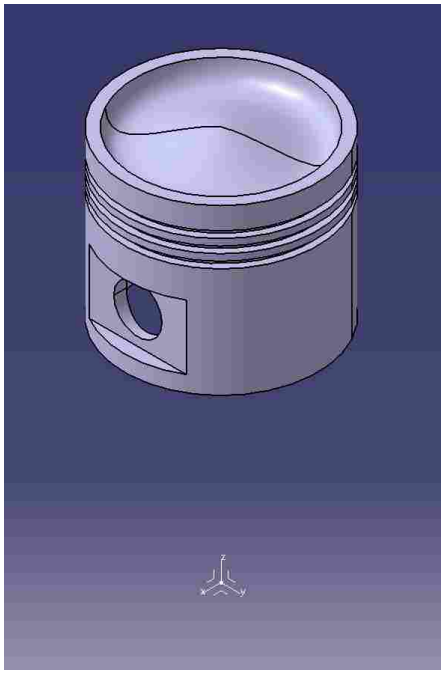
4. **Extrude** Sketch (subtract)
    a. From -40.0 to 40.0



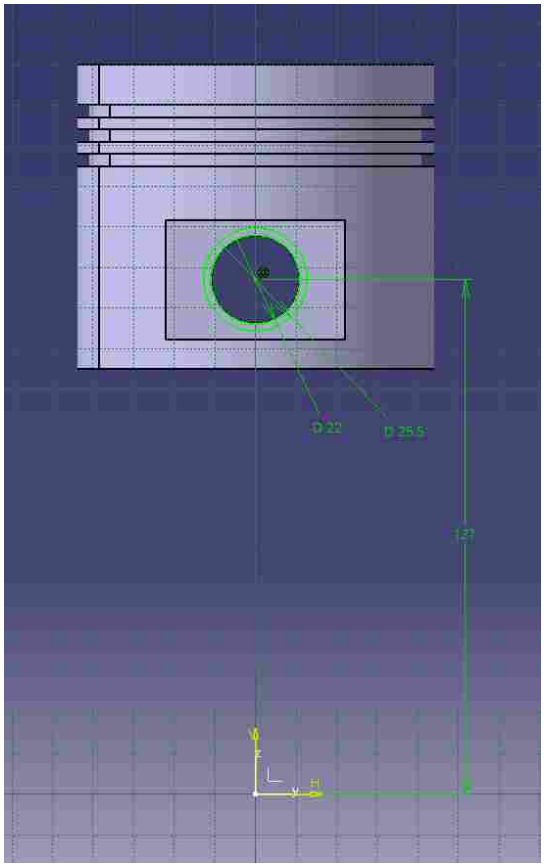
5. Create **Sketch** on yz plane



6. **Extrude** Sketch (subtract)
    a. From -80.0 to 80.0



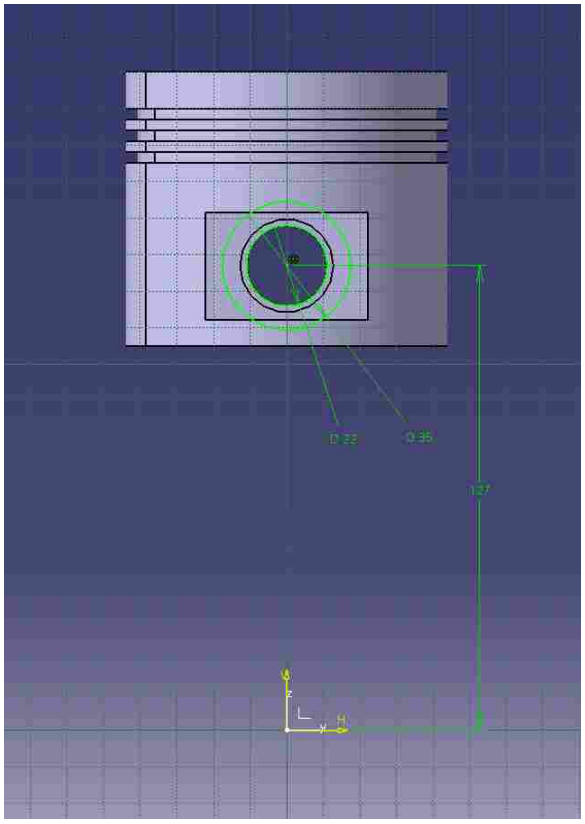
7. Create **Sketch** on yz plane



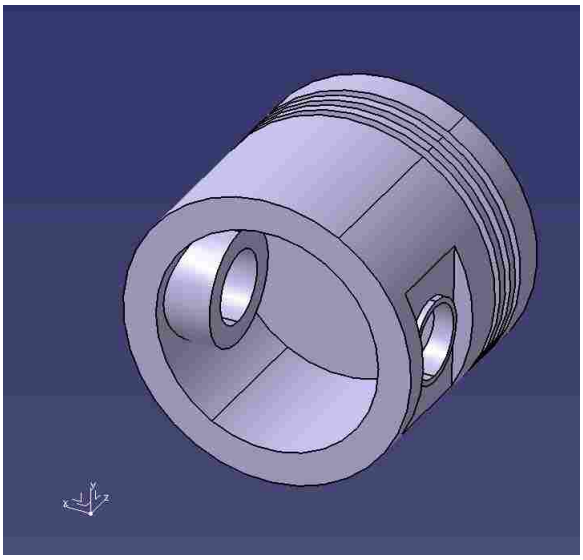
8. **Extrude** Sketch
    a. From 38.0 to 40.0

9. **Extrude** Sketch
    a. From -38.0 to -40.0

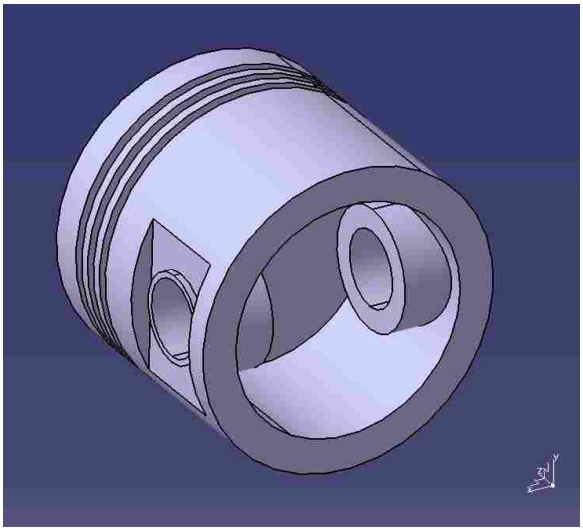
10. Create **Sketch** on yz plane



11. **Extrude** Sketch
    a. From 20.0 to 37.0



12. **Extrude** Sketch
    a. From -20.0 to -37.0



Summary for User 4

A.2 Part Files

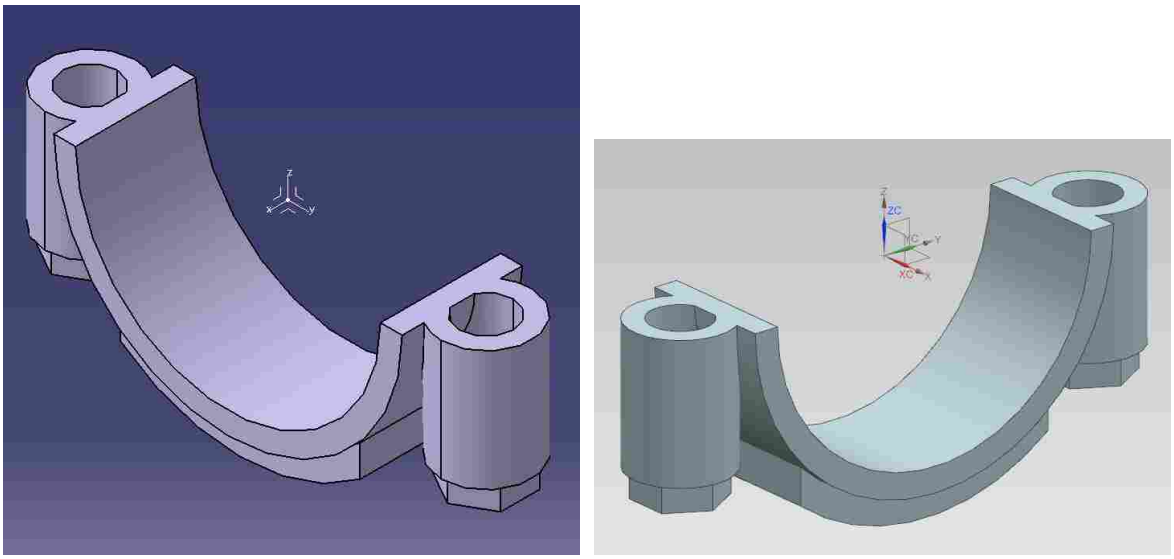


Figure A.2: Connecting Rod Lower Part File in CATIA (left) and NX (right)

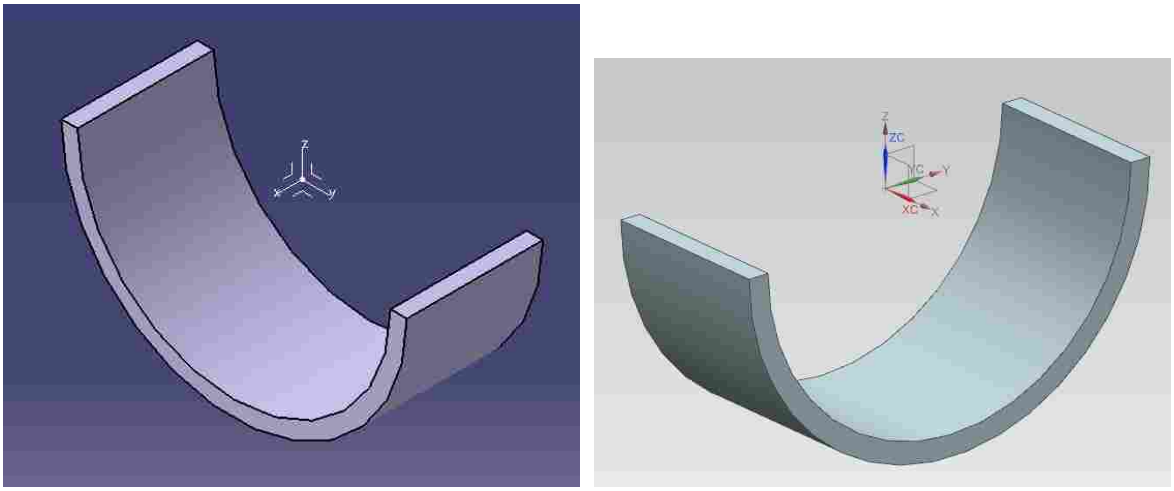


Figure A.3: Connecting Rod Lower Bearing Part File in CATIA (left) and NX (right)

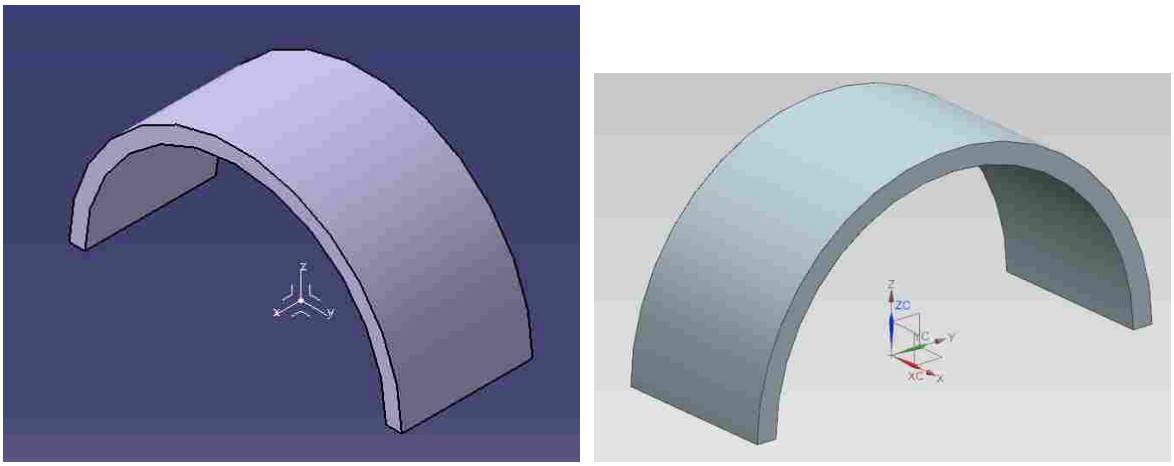


Figure A.4: Connecting Rod Upper Bearing Part File in CATIA (left) and NX (right)

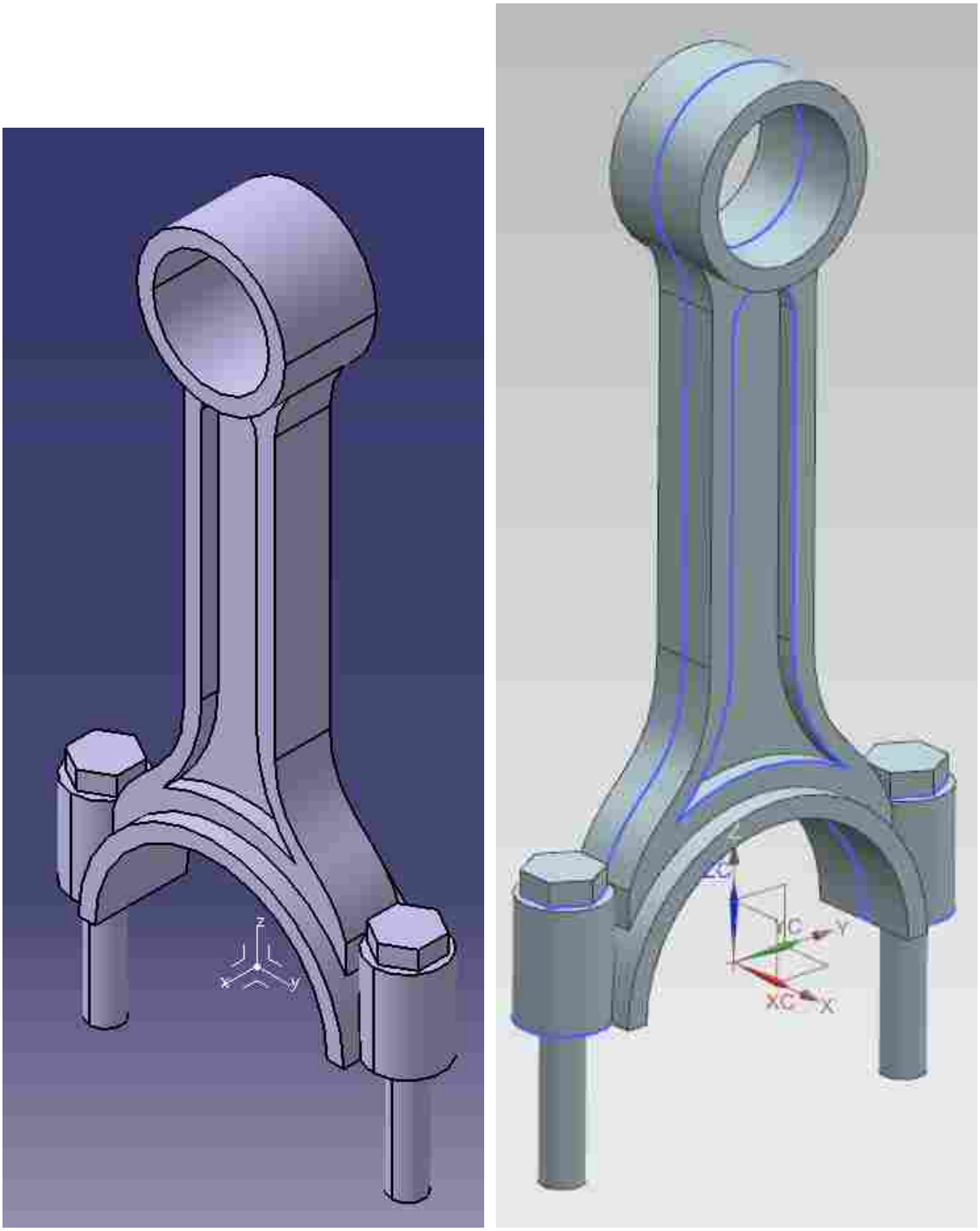


Figure A.5: Connecting Rod Upper Part File in CATIA (left) and NX (right)

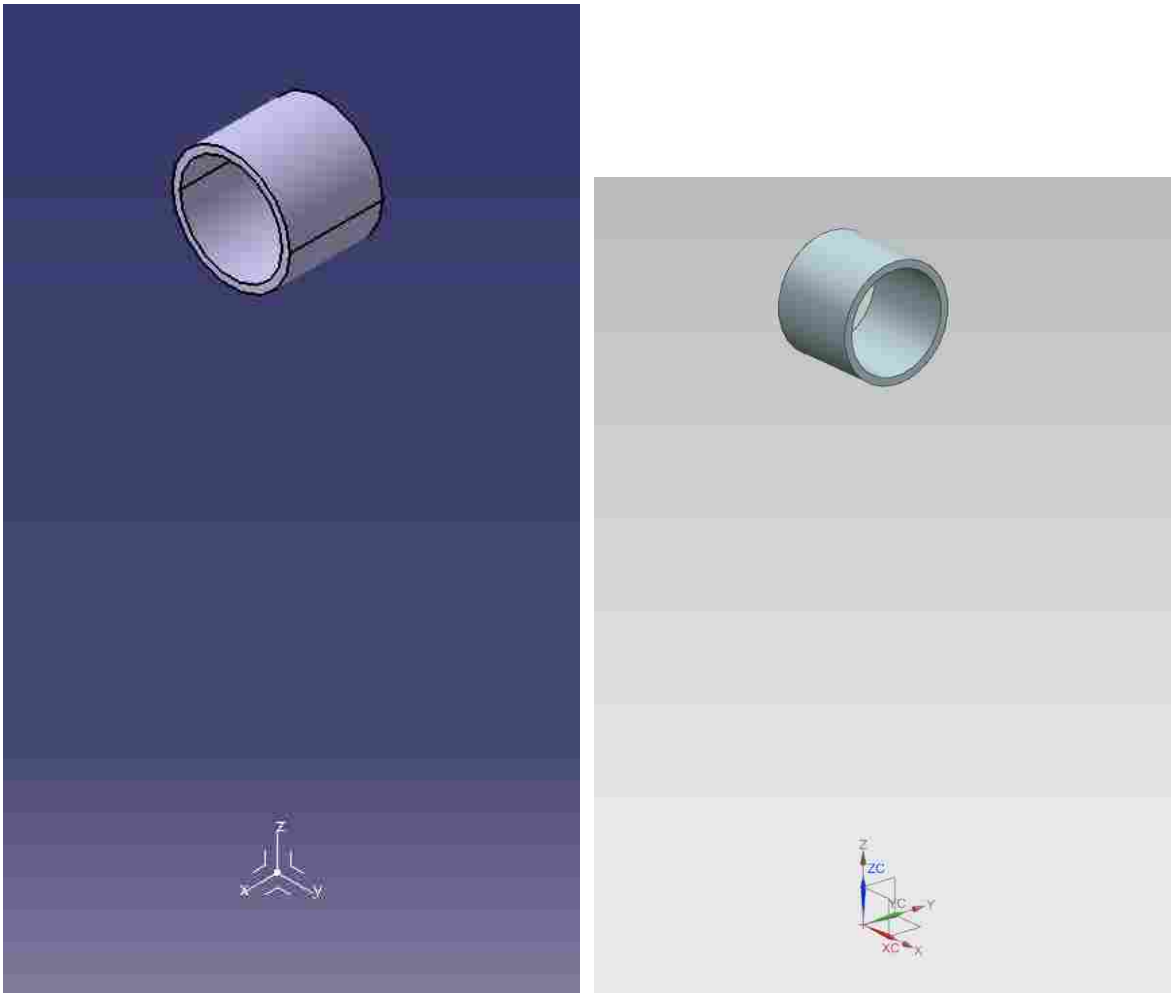


Figure A.6: Connecting Rod Bushing Part File in CATIA (left) and NX (right)

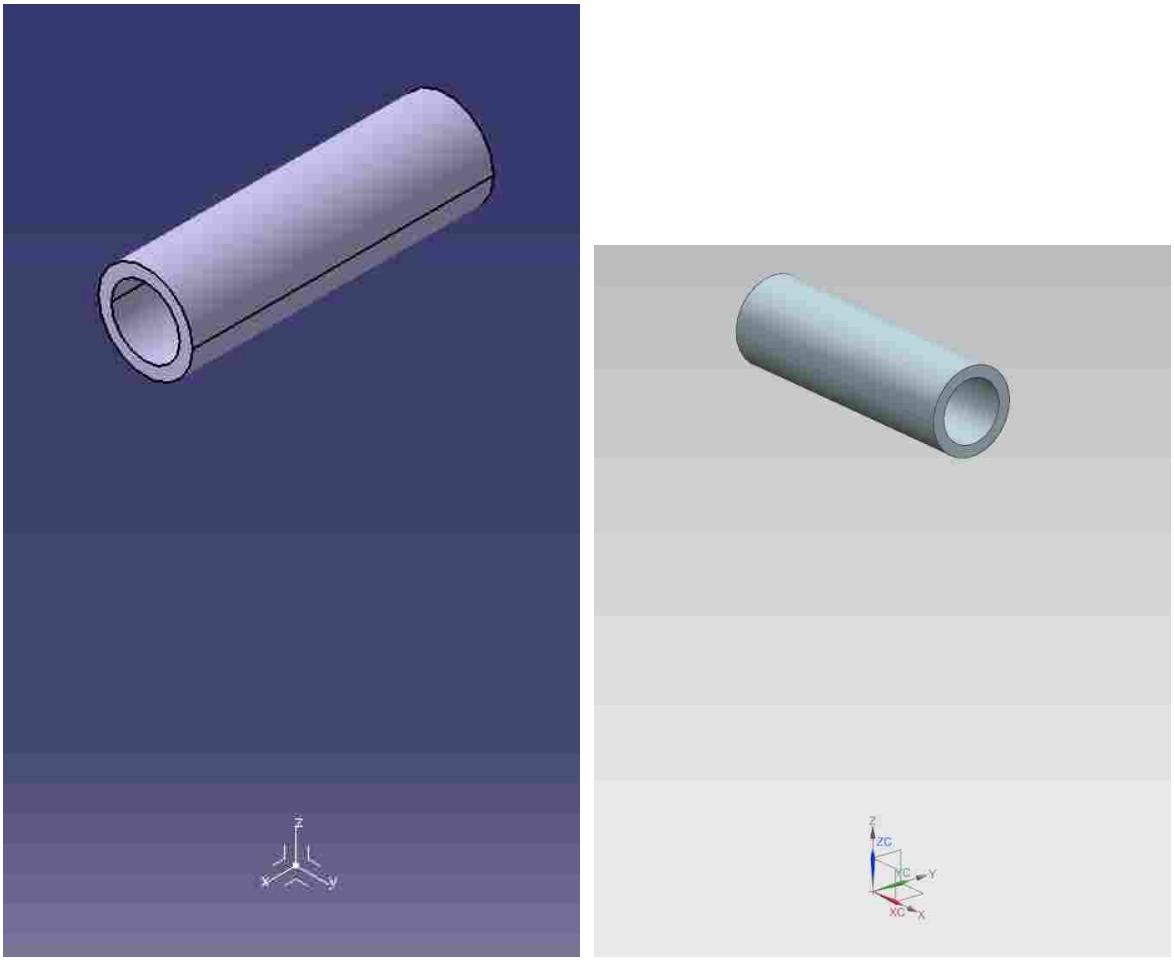


Figure A.7: Piston Pin Part File in CATIA (left) and NX (right)

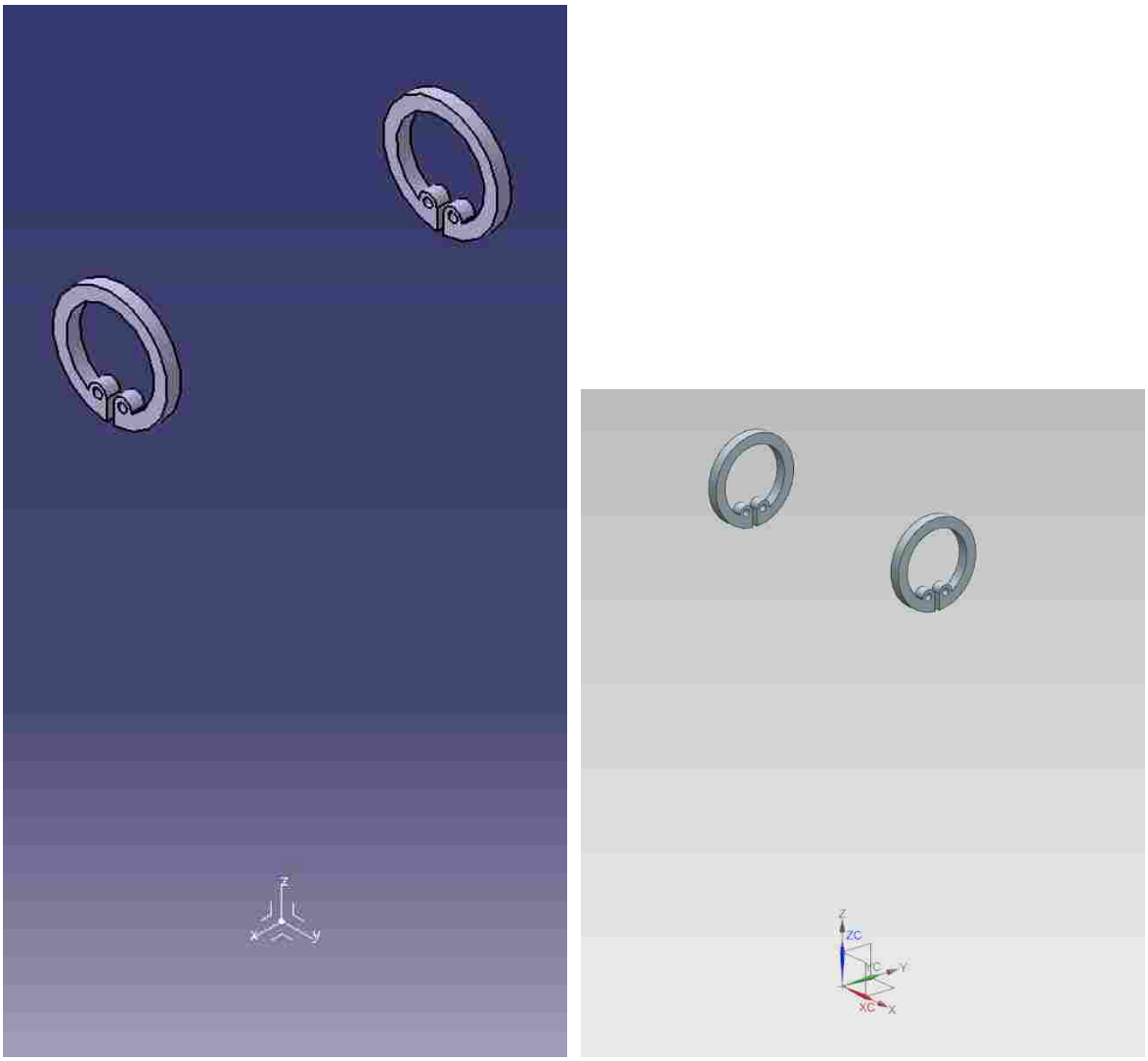


Figure A.8: Snap Ring Part File in CATIA (left) and NX (right)

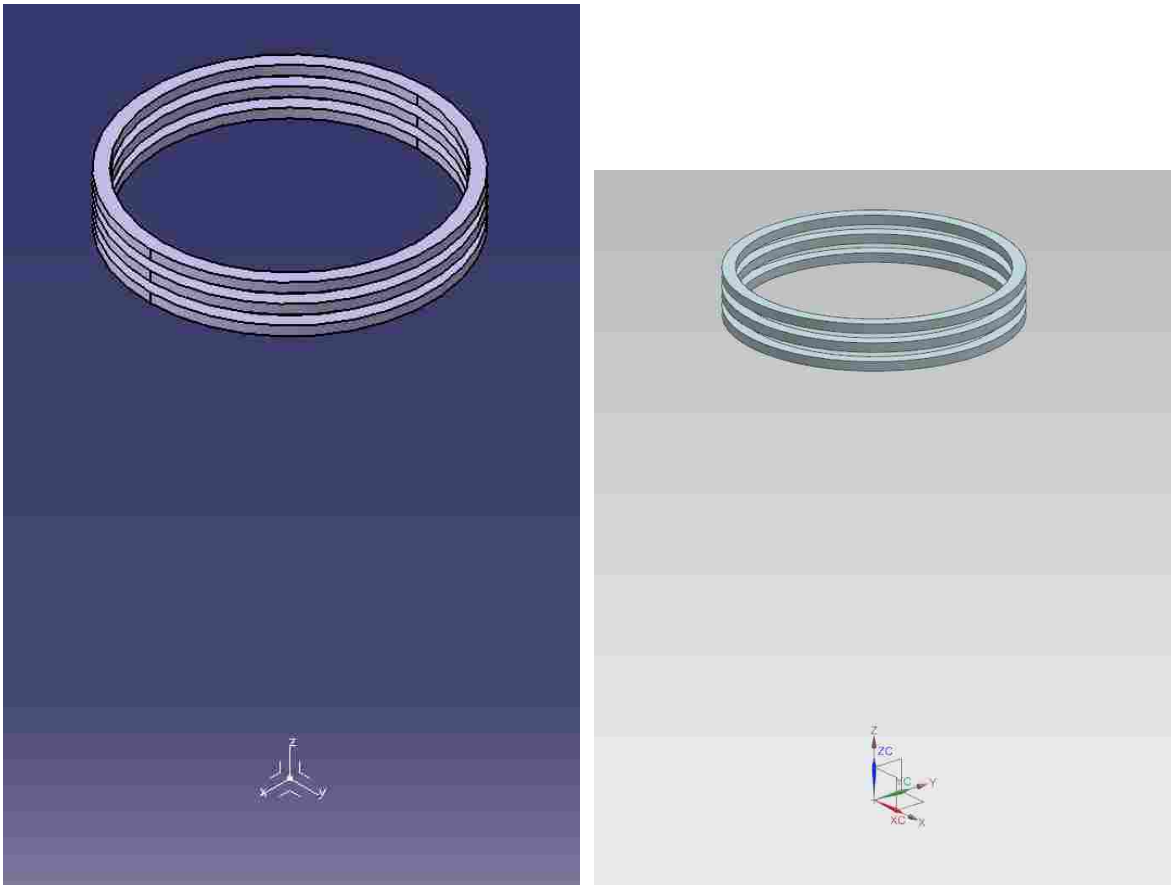


Figure A.9: Piston Ring Part File in CATIA (left) and NX (right)

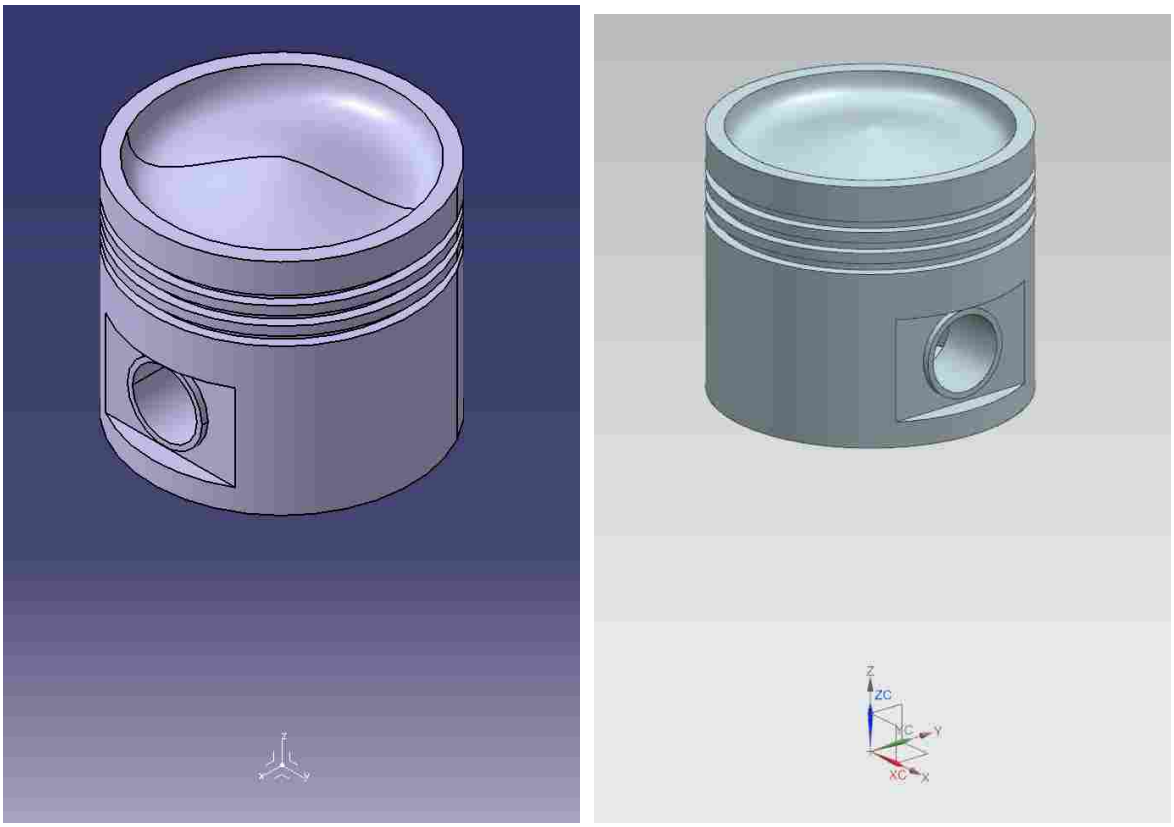


Figure A.10: Piston Part File in CATIA (left) and NX (right)