COMPUTER-AIDED DESIGN & APPLICATIONS, 2017
VOL. 14, NO. 4, 408–421
http://dx.doi.org/10.1080/16864360.2016.1257184

Associative CAD references in the neutral parametric canonical form

Daniel R. Staves, John L. Salmon and Walter E. Red

Brigham Young University, USA

ABSTRACT
Due to the multiplicity of computer-aided engineering applications used in industry, interoperability between programs has become increasingly important. A 1999 study by the National Institute for Standards and Technology (NIST) estimated that inadequate interoperability between the original engineering manufacturers (OEM) and their suppliers cost the US automotive industry over $1 billion per year, with the majority spent fixing data after translations. The Neutral Parametric Canonical Form (NPCF) prototype standard developed by the BYU Site of the NSF Center for e-Design offers a solution to this problem by enabling real-time collaboration between heterogeneous systems while preserving design intent. The NPCF is implemented within a SQL database and defines the schema both for neutral features and for the parameters defining the inter-feature relationships and associations.

KEYWORDS
Interoperability; heterogeneous CAD; design intent; collaboration

1. Introduction

Iterative engineering processes have long been integral to engineering design. Before computer technology assisted the design process, designers and engineers gathered around the drafting table to coordinate their work on products. Detailed drafting standards (ISO 128, DIN 6, ASME Y14.5) were employed to ensure accurate interpretation of drawings when transferred to other designers and manufacturers. Even with these standards, however, close personal contact between the designer and manufacturer was usually necessary [11]. With advances in CAD and communications via the Internet, computers are now a recognized necessity in the design process. While email has allowed designers and manufacturers to be geographically separated, mutual and interactive communication of design information is no less vital now than in the past [24]. Part models and drawings must be frequently exchanged, translated, and checked between designers and manufacturers at each iteration of the design. These designers and manufacturing personnel often use different applications that require translation or rework to make the models consistent between systems.

While many methods of the design process have remained the same during the transition from the drafting tables of the past to the modern computer aided engineering processes of today, the tools of engineering have changed dramatically. A wide variety of computer-aided tools are available that specialize in the many different aspects of design, like finite element analysis (FEA), computational fluid dynamics (CFD), and solid or surface modeling (CAD). Companies, engineers, and designers choose these tools based on their strengths, ease of use, and familiarity.

Translation processes and standards have been developed for designers and engineers to work with the multiplicity of file types and mathematical definitions of features in order to carry out their work. The translation process, however, often strips the original model of its design intent, or intelligence embedded into the model. In addition, the process can introduce tolerance variations, which impedes proper manufacturing. Translating also slows the design process and hinders meaningful collaboration between designers working in separate systems.

1.1. The cost of poor interoperability

Due to a shift in production practices in the 1980s, US automakers increased their market share and became more competitive in the US automotive market [2]. A large part of this increase in production is due to a move toward concurrent engineering and design outsourcing. This shift resulted in the sharing of design data between a greater number of people and organizations both within the company and between the company and its suppliers. While overall productivity increased, the move highlighted many difficulties related to the use

CONTACT Daniel R. Staves danstaves@byu.edu

of heterogeneous CAD packages. Estimates found that imperfect interoperability, or model transfer between CAD packages, during this time cost between $1.02 billion and $1.05 billion within the US automotive supply chain alone [2]. The vast majority of this cost is due to the time spent fixing data from poor CAD model translations between both the OEMs and their suppliers. Interactions with Industry Advisory Board (IAB) members of the National Science Foundation (NSF) Center for e-Design have revealed a similar trend in the US aerospace industry.

Difficulties in the translation process arise because of differences in the way each CAD system represents the part model. Because there is no standard, modern CAD packages store feature and design data in proprietary formats, even though each system represents the same type of data: three-dimensional geometric models. These systems often only support interoperability by translating geometric Boundary REPresentation (BREP) data through formats such as IGES and STEP. While these translation methods have been extensively and successfully used in industry, the variances in the translation process between applications can cause geometric errors among transferred models [5]. In addition, by only translating BREP data, crucial design intent stored within feature data is lost, which greatly hinders meaningful collaboration. Any modifications or adjustments to the model must either be made on the originating system and re-translated or completely redesigned in the new system. In addition to these limitations, this method inherently follows a serial, single-user work-flow, only allowing one user at a time to design or update the model. Before a design can be shared, it must be exported in a neutral format and sent to the end-user. This significantly impedes the advantages of concurrent engineering.

The interoperability challenge is illustrated by the simplified product development process depicted in Figure 1. Black arrows in the figure represent the flow of the design and related data as it is passed between designers and groups. Before the design data is passed to another group, it is often converted to a neutral format, such as IGES or STEP, as illustrated by the convert data stage. After importing into the new system, the model must undergo a fidelity check to ensure an accurate import. If the model is to be used for CAE analysis, it may need to be updated to fix geometric continuity problems associated with the neutral format. The red arrows represent the feedback loops through which the design must pass if any of these tests are failed. As the number of designers using CAD systems increases, the time spent resolving these conflicts in the feedback loops increases, which can significantly delay the product launch and increase costs.

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

1.2. Background

Due to the multiplicity of commercial CAD applications, companies often interface with a wide variety of CAD file formats through interactions with their supply chain. To open and use these formats, CAD applications must translate part models and assembly files between heterogeneous CAD systems. Translation inconsistencies arise

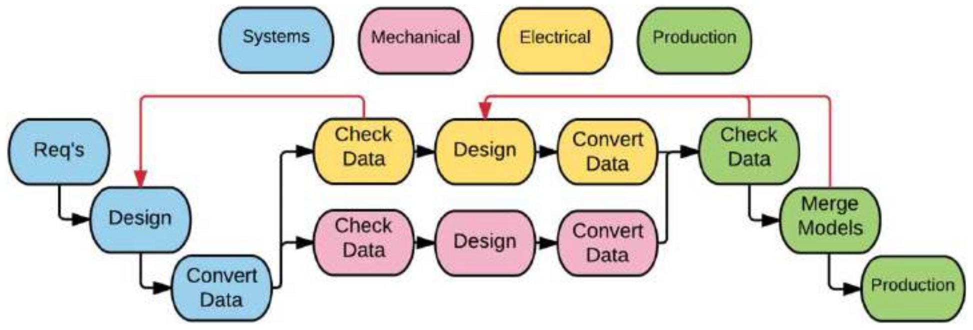


Figure 1. Simplified Serial Product Development Process.

because different CAD systems generate different features or may represent similar features differently. With no standard definition representing CAD features, each system may define a feature differently. Due to these different definitions, translating between programs is nontrivial.

The International Graphics Exchange Standard (IGES) format was first developed in 1980 to facilitate the translation between heterogeneous CAD systems [1]. IGES was the first attempt at resolving the data exchange challenge between CAD systems. It works by translating the CAD model of each system to its basic geometric data and is the most widely used neutral format today. While IGES has been very successful in allowing models developed in different systems to be exchanged, it falls short in that only BREP data is translated. All associative links between features are broken, and design intent is lost.

Created in 1984, the Standard for the Exchange of Product Model Data (STEP) is a redesign of IGES, aiming at a more advanced, database-oriented, and integrated solution based on product lifestyle data [19]. It fixes some shortcomings of IGES, including standardizing the processors, and uses a formal language to define the data structure to avoid ambiguities during interpretation, which could result in as much as 50% failure rates [6]. Work has continued on the STEP standard, most recently by the Solid Model Construction History (SMCH) group, which is seeking to preserve design intent through the translation process by recording the history of the model as it is created. SMCH adds an implicit, or history-based, representation of the model to the explicit BREP data. This effectively creates solids that maintain their original relationships with other solids after the translation process, preserving design intent and allowing them to be edited and manipulated accordingly [23]. A drawback of SMCH is that the file size is large due to both the B-rep and construction history data being stored, making it difficult for collaborative, multi-user applications in which data is transmitted frequently between clients.

A number of successful solutions have investigated preserving design intent through model translation by utilizing the command history used to create the CAD model. The Macro-Parametric approach proposed by Choi et al. [4] utilizes each CAD system's built-in macro functionality, which automatically records each action performed by the user. The macro file is translated into a format readable by the new CAD system and executed, mirroring the original actions performed in the new format. This approach was continued by Mun, et al. by determining a common set of modeling commands from six different CAD systems [21]. Continued research into the Macro Parametric Approach [7,14,22] has further expanded the capabilities of the format and improved the state of CAD interoperable methods. The Neutral Modeling Command method addresses the need for a collaborative heterogeneous CAD solution through the use of each CAD system's application programming interface (API) [12–15]. Each client runs an add-on program that translates system modeling operations (SMO) into neutral modeling commands (NMC) in real time and is synchronized between clients through a server. The modeling commands are then translated to the SMO of the designation CAD system and applied to the model, effectively recreating the model's operation history. The development of other collaborative CAD solutions [3,10,17–18,25] have also improved upon the processes of transferring geometric models between multiple clients using heterogeneous CAD systems. These approaches have made strides in the effort to preserve design intent through the translation process. However, the design intent is transferred by mirroring the design history of the part between all clients. While this mirrored history enables feature edits on remote clients, it is not stored in a way that preserves referential integrity to ensure validity of stored data.

1.3. The neutral parametric canonical form (NPCF) approach

The BYU Site of the NSF Center for e-Design has sought to reduce the design cycle time by developing synchronous, multi-user collaborative design tools. Its most recent work to develop the NPCF seeks to solve the interoperability problem by formulating neutral standards for representing CAD features while maintaining associativity between features. It utilizes a client-server architecture for communication, and a SQL database for persistent storage to enable multiple users to access the same CAD model. This format enables simultaneous design among heterogeneous CAD systems. Currently, this interoperable design workflow supports real-time collaboration between Dassault Systemes CATIA, Siemens NX, and PTC Creo CAD systems.

Like previous approaches to solving the CAD interoperability problem, the NPCF method seeks to preserve design intent through the translation process. Design intent is preserved during the translation process by translating data required to re-create features to a neutral format as opposed to translating the geometric results of the features themselves. This type of data enables designers to not only edit the geometry that is imported from different systems, but understand how the features relate to the rest of the model. This, for example, enables the entire part to update after a parameter change. Rather than using the modeling history to preserve this design intent, as in the case of the Macro-Parametric or Neutral Modeling Command approach, the features themselves are neutralized while retaining their original parameterization and associativity. The neutral features are stored in the database using design rules to ensure referential integrity of all associative links and data integrity of the feature parameters. The neutral features are automatically forwarded to all clients subscribed to the part or assembly model to be incorporated into their local models, enabling simultaneous collaboration between geographically separated users running heterogeneous CAD systems.

This paper describes the process utilized to determine a normalized, neutral format for CAD features and incorporate it into the NPCF SQL database.

2. NPCF methods

The goal of the Neutral Parametric Canonical Form (NPCF) is to enable real-time, simultaneous translation of both model and design intent between heterogeneous CAD clients. This is made possible by a client-server architecture in which CAD operations performed by clients running a CAD system are neutralized and sent to the server for distribution among other clients. In addition to this architecture, other methods facilitate the transfer of design intent and feature associativity during the modeling process. They enable modern, commercially available CAD packages to work simultaneously in a heterogeneous environment.

The current state of the NPCF provides support for a limited subset of CAD features as the focus is centered on transferring design intent while maintaining the data integrity of the model. Continued development is supporting new features in the NPCF database.

2.1. Maintain referential data integrity

Traditional neutral formats like IGES and STEP store part information in a file, either as text-based or binary files. While convenient for sharing files via secure file transfer or email, they are unable to maintain the integrity of the data. Often, corrupt data is written by a CAD system rendering the neutral file unreadable. To prevent this type of corruption within the NPCF, a SQL database is used in conjunction with unique identifiers to link feature data together. Neutral features are represented by tables arranged in a hierarchal format with common parameters shared by many features being placed in tables at the top of the hierarchy, and parameters specific to features are placed in dedicated feature tables. This arrangement enables feature references and inter-feature references to be stored. In this instance, referential integrity is maintained because the references are only allowed between certain features. For example, an extrude feature can reference a sketch as the profile to extrude where is it not able to reference a coordinate system. When defining the neutral extrude feature table on the NPCF database, associations are created linking the extrude table with the sketch table, enabling the correct feature associations.

In addition to the hierarchical arrangement of distinct features represented on the database, sub-hierarchies representing feature variations are also incorporated into the NPCF. These feature variations include the multiple definitions CAD systems use to define a feature. For example, a plane feature can be defined using a 3D point in space and a normal vector, or by an offset from another surface. Both of these parameter sets make up a different sub-feature table located on the database hierarchy directly below the plane feature table. General parameters used to describe all planes are included in the generic plane table while specific parameters used to define a plane variation are stored in the sub-feature tables.

An important step in preserving referential integrity is to utilize a sub-hierarchy within the NPCF database to handle feature variations rather than including all feature parameters in a single table. Each parameter required to define a particular feature variation is marked in the database as non-nullable, signifying that all fields must be filled with the proper data type if it is to be saved in the database. By imposing this requirement, only data structures that are completely filled out with all required information are stored in the database and transmitted to other clients.

Table 1 describes the process used to incorporate new features into the neutral database. When a new feature is to be added, feature creation methods are identified and compared between CAD systems to identify commonalities. For example, both a blind extrude in Creo and an extrude-by-values in NX expect a sketch feature on which the operation is to be performed and numerical limits to identify the start and end points of the extrude. Though these feature variations are given different names and treat the limits differently, both can be neutralized into the same format without loss of data or semantic meaning. Contrasting this feature variation with an extrude-to-plane operation, both methods utilize the same sketch parameter, but accept different objects to define limits.

Table 1. Feature parameter distribution process utilized when adding new features to database.

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

When splitting parameters between common tables, as described in Table 1, a generic extrude table will be created that stores the referenced sketch parameter, and two feature variation tables will be created to hold the parameters required by the blind extrude and planar extrude, respectively.

2.2. Defining a neutral format

The principle of neutralization versus translation is an important distinction between the way the NPCF enables interoperability between heterogeneous CAD systems and methods employed by previous CAD interoperability formats. The neutralization method converts CAD data from an originating system to a neutral format after a CAD process is performed. The neutral format completely defines the feature by its parameters and is stored within a database. It is then converted to the CAD format of the destination system and incorporated into the local part model for interaction by the other user. The collection of neutral features associated with a single part remain on the database as the master copy of the model from which all systems load data. This single neutral copy ensures model consistency between clients and sessions.

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

To understand the data requirements for representing a feature in a neutral format, creation and edit actions were recorded using each CAD system's built-in scripting or journaling functionality. This is the process used by Li et al. to determine the neutral modeling commands format [16]. Most major CAD systems support this functionality, but a good understanding of the CAD system's API can allow a developer to program without the recorded script.

The inputs to the scripted feature are compared between systems to identify commonalities and shed light on the data required to define a neutral format for the feature. The format was chosen by identifying the minimum ideal set of parameters that fully defines the feature. During the translation process, these parameters may be converted to equivalent representations based on the requirements of the individual CAD system's API. For example, a 2D point used in sketching in CATIA and Creo is defined by a sketching plane and an X and Y coordinate. On NX, however, the point used in sketching is defined using X, Y, and Z parameters to specify location. Though both formats are mathematically equivalent, for ease in conversion, the neutral format was defined using the X and Y coordinates and a sketching plane. This same process was used to determine the neutral parameters for all supported CAD features within the NPCF.

2.3. Object relational manager

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

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

The ORM, as shown in Figure 2, converts the features and feature variations distributed through the various database tables into single CAD objects that contain all the parameters required to define each feature

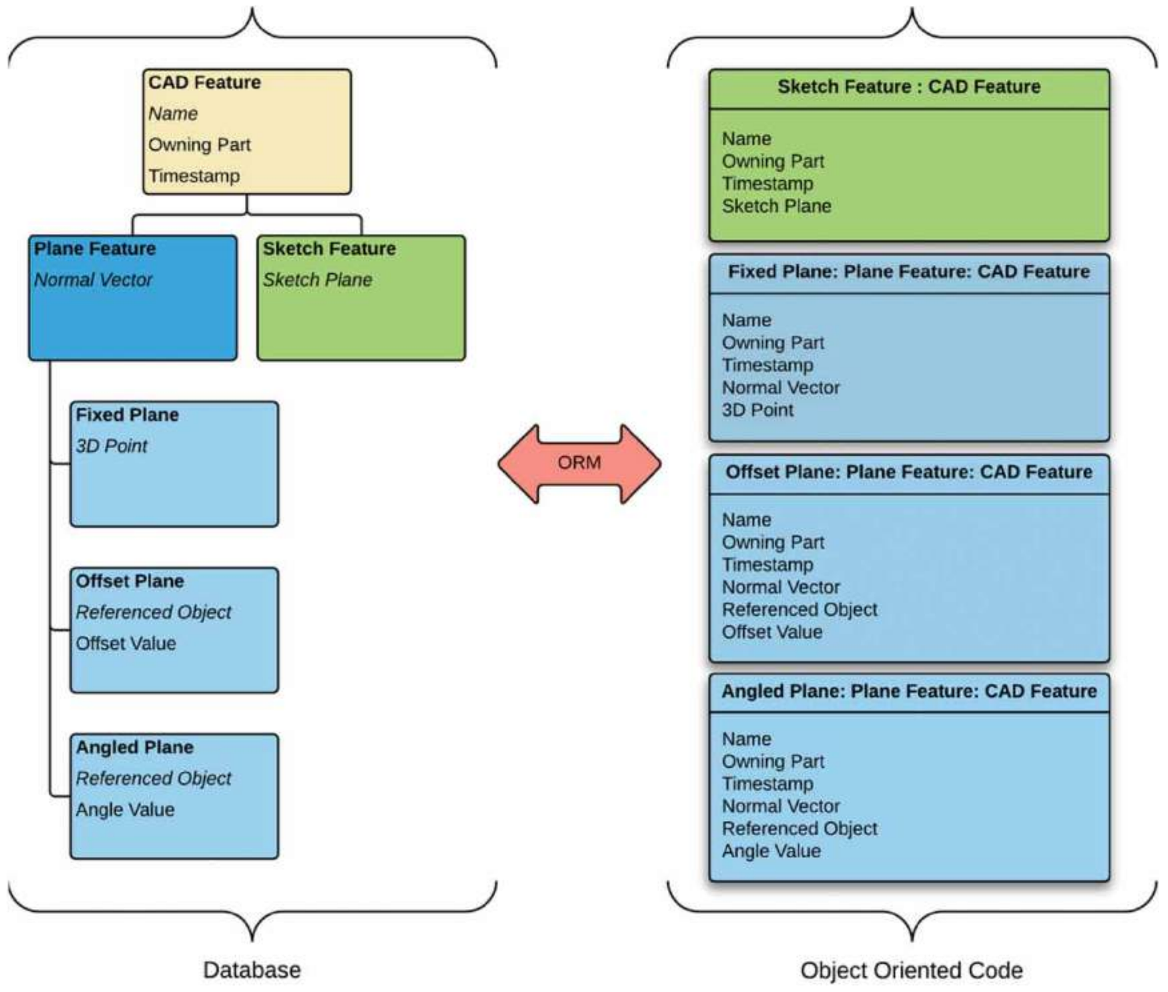


Figure 2. The ORM converts database tables and schema into programming data structures.

or feature variation in a neutral format. The database tables for sketch feature and plane feature variations are combined into their own data structures. These structures retain their relationships with the corresponding database parameters from which they are derived and are used for accessing and writing data within the database. The hierarchy established within the database between tables is also retained in the data structures created by the ORM as object inheritance.

Apart from combining feature parameters into a single feature object, inheritance is used to define data types. Since each feature variation is represented in the database as a separate table and modified by the ORM to be represented as separate objects inheriting properties from the base feature class, each feature variation object is of the same type as the base feature object. This is an important principle that enables feature references. A 2D sketch feature in CAD, for example, references a plane feature to define location. A plane feature has multiple feature variations from which it can be defined. Using the method describe above, the ORM will generate feature objects for each feature variation. However, since each plane feature variation inherits the base plane feature, the parameter used to reference a plane feature in the sketch feature object needs to be only of type plane object, and any variation of the plane object can be stored. This method both simplifies development and helps to preserve data integrity because only one reference parameter is needed to reference multiple feature definitions.

2.4. *Interface inheritance*

While the ORM alone enables multiple feature definitions to be referenced, many CAD features can reference completely different types of objects in a single parameter. This is best illustrated by the example in Figure 3. A revolve feature revolves a 2D sketch around an axis to generate 3D geometry. The axis of the revolve could be an axis feature defined by a 3D point and vector, or it could be a line object in a 2D sketch. These two objects in no way share an inheritance structure but must be able to be referenced in a single parameter to maintain

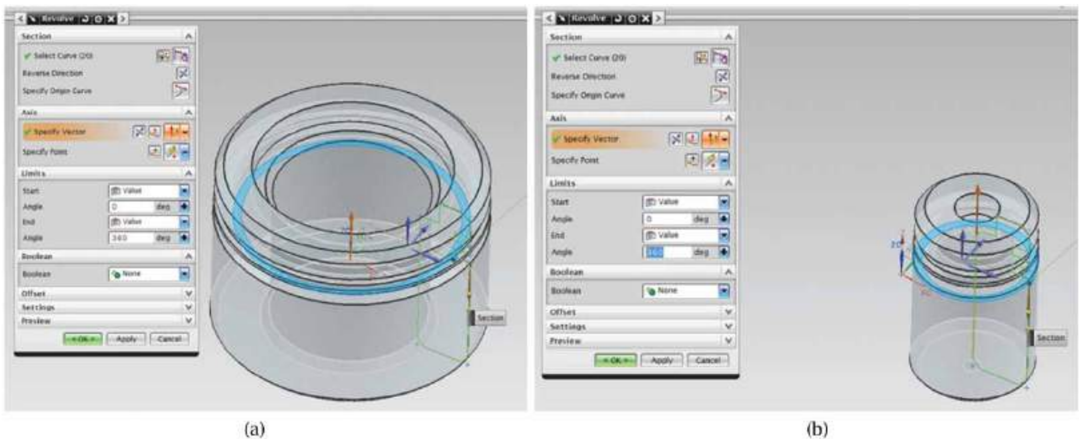


Figure 3. A revolve feature can revolve a 2D sketch around (a) an axis of a coordinate system or (b) a line of a sketch.

data integrity. Further modifications to the ORM are necessary to support this functionality.

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

Database tables and data structures that reference an interface rather than a feature need to be modified as well. Because each association in the database requires a single table to represent the association, any feature table using a parameter to reference an interface must create the association with the most basic database table from which all feature tables associate. The parameter used for the reference is given a reserved name that represents the type of interface used. For the case of an axis reference, the reserved name is ReferencedAxis while other reserved names can be seen in Table 2. The ORM is modified to identify these reserved names and replace all associated references to the basic CAD object data structure with the interface type corresponding to the reserved name.

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

Table 2. Reserved names for database columns used to define parameters that reference interfaces.

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

2.5. *Client plug-in software*

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

To facilitate the wide variety of APIs associated with the CAD programs, the client-side architecture of the plug-in software is divided into two parts, as seen in Figure 4. The multi-user object (MUObject) classes are responsible for handling communications with the server and for neutralizing CAD data. These MUObject classes are all written in the C# programming language and contain the code that is generated by the ORM to read from

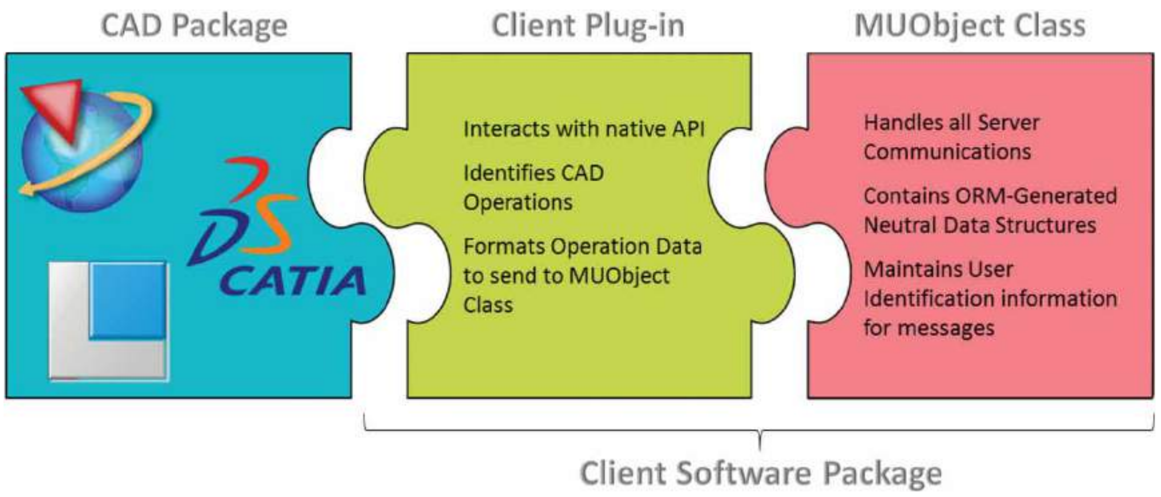

CAD Package
Client Plug-in
MUObject Class
CATIA
Interacts with native API
Identifies CAD Operations
Formats Operation Data to send to MUObject Class
Handles all Server Communications
Contains ORM-Generated Neutral Data Structures
Maintains User Identification information for messages
Client Software Package

Figure 4. Client plug-in between a CAD application and the MUObject class.

and write to the database. When a CAD operation is performed, the MUObject classes convert the operation data to a neutral format and fill the ORM-generated data structures. A message is then sent to the server containing the neutral data along with information pertaining to the originating user, the part model that the message is associated with, and a time stamp to be used with ordering operations. The MUObject classes also receive and deserialize messages from the server. Since all clients perform these same basic functions, the MUObject classes can be identical for all CAD packages.

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

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

By separating the client software package into two parts, developers working on integrating new CAD packages into the interoperable software are, in essence, further abstracted away from the communication aspects of the multi-user program. Instead, developers are only concerned about translating between the CAD specific data and the neutral format defined in the MUObject class. This work flow has significantly accelerated development time as new features are supported in an incremental process between CAD systems.

3. Implementation and verification

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

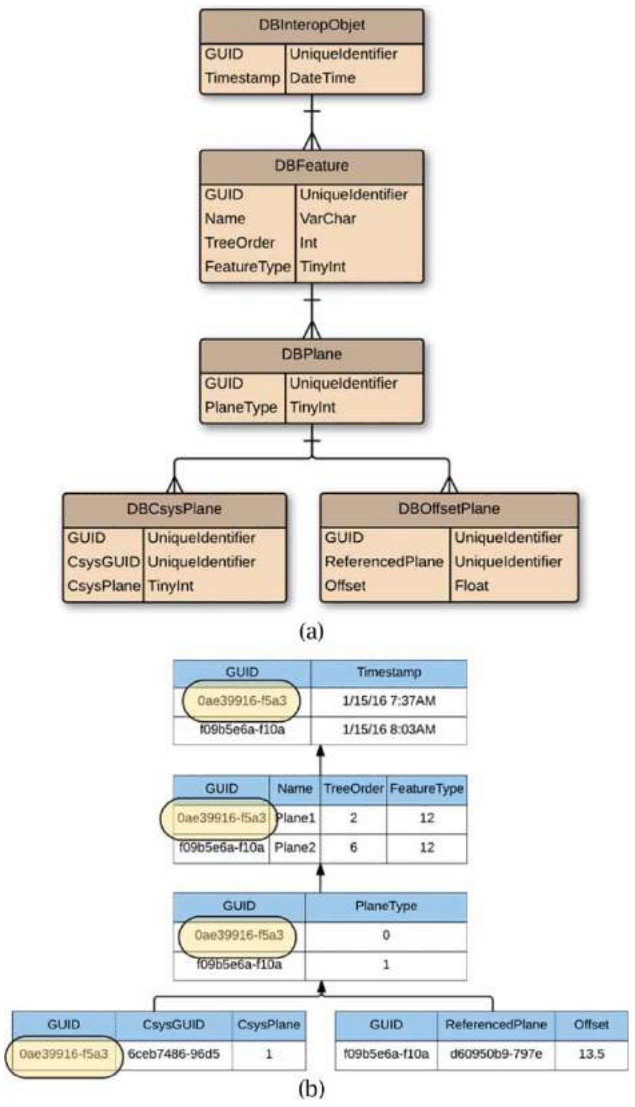
To support this research, 22 database tables were created, with associations between tables representing object inheritance. These tables supported the initial 13 features which were chosen by surveying a list of most used CAD operations utilized by a BYU senior design project. These features include coordinate system, datum plane, datum axis, 3D point, 3D spline, 2D sketch, extrude, and revolve.

Supported 2D sketch features include 2D point, 2D line, 2D arc, 2D circle, and 2D spline. Additional tables used to enable assembly operations were also created.

3.1. Unique identifiers

The NPCF defines a set of parameters and relationships that express CAD features in a neutral format. To test their viability, the NPCF was implemented in Microsoft SQL Server with a database table for each feature variation, and a database column within the tables for each feature parameter. In this implementation, each table contains a Globally Unique Identifier (GUID) parameter used to uniquely identify neutral CAD features stored in the database. It is generated using the Microsoft Windows API to guarantee uniqueness between all clients with a high degree of certainty. Figure 5 describes the process of creating and propagating GUIDs through the NPCF process. After a new feature is created in the CAD system, a GUID is generated by the plug-in software and stored with both the neutral data in the MUObject class and with the CAD feature in the CAD system. The GUID is used to identify features on all clients, update feature data in the database when edits are performed in the CAD system, and link feature data dispersed between tables in the SQL database.

Because the GUID is unique between all clients, it can be used to allow a single feature to be partially represented by multiple database tables in a hierarchical format, similar to the way CAD features would be represented in a CAD system's API. For example, data for a coordinate system plane feature is stored in four separate tables as seen in Figure 6. Figure 6(a) shows both the database schema that describes how the database tables relate to each other as well as a representation of the data stored within each table. As seen in Figure 6(b), a GUID and time-stamp for the feature are stored in the InteropObject table. The same GUID, feature name, tree order, and feature type are stored in a feature table. The GUID and plane type are stored in the DatumPlane table and the GUID, associated coordinate system, and plane type are stored in the DBCsysPlane table. Each table is linked together using a primary-key/foreign-key reference that automatically matches the correct data when querying the database for a feature.



GUID	Timestamp
0ae39916-f5a3	1/15/16 7:37AM
f09b5e6a-f10a	1/15/16 8:03AM

GUID	Name	TreeOrder	FeatureType
0ae39916-f5a3	Plane1	2	12
f09b5e6a-f10a	Plane2	6	12

GUID	PlaneType
0ae39916-f5a3	0
f09b5e6a-f10a	1

GUID	CsysGUID	CsysPlane
0ae39916-f5a3	6ceb7486-96d5	1

GUID	ReferencedPlane	Offset
f09b5e6a-f10a	d60950b9-797e	13.5

Figure 6. (a) Neutral data for a single feature is partially represented by multiple tables. (b) GUIDs are used to link the partial data in each table together to completely define a neutral feature.

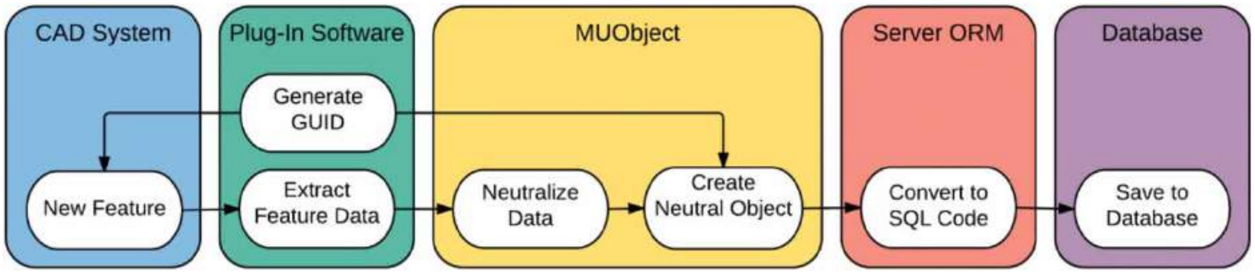


Figure 5. The order of operations after a new feature is created in the CAD Interop Multi-user prototype.

3.2. Software implementation

To support interoperable CAD on the local client, a software plug-in package was written for each supported CAD system. As the primary purposes of the plug-in client software are to detect CAD operations performed by the user and to perform remote CAD operations locally, each software is written to utilize the native API of the CAD system – NXOpen for NX, Pro\Toolkit for Creo, and Automation for CATIA.

The API's of both NX and Creo expose events performed by the user. Callback functions can be registered so that when a feature is created, for example, parameters relating to the new feature can be stored and converted to the neutral format. CATIA's automation API, however, exposes no events. To identify when new features are created, the plug-in iterates through each feature within the part's feature tree. In all cases, when a new feature is identified, either via an event or through iteration, a GUID is created using the Windows API and stored as a parameter in the feature. The plug-in software then converts the CAD-specific feature parameters to the CAD-neutral parameters defined by the NPCF.

The 2nd part of the plug-in software is the same for each client, and handles the transmission of neutral data to and from the server. To accomplish this, Microsoft's entity framework is utilized as an ORM to read and write the neutral data to the database. When an operation is performed remotely, the neutral data is sent from the server in the ORM object and converted by the CAD-specific part of the plug-in to a CAD feature. The CAD feature is then incorporated into the CAD model using methods outlines in each system's API.

3.3. Feature capabilities and limitations

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

(1) Some bugs are caused by programmer error. When implementing a new feature into three different CAD systems, crucial information is sometimes omitted causing a feature to be created incorrectly or not at all. The majority of these bugs are located in the plug-in portion of the client code when extracting or setting CAD feature information. These types of bugs are given the highest priority during the modeling sessions and are fixed immediately.

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

(3) Sometimes a single feature can pass all tests, but when used during a team-modeling session, it can cause the program to crash. Often this is because of the lack of an overall consistency manager to ensure multiple operations are not performed on the same feature. To limit the effect of this architecture limitation, methods were implemented in each client plug-in to queue remote operations when a local operation is performed. This fix has greatly reduced the amount of reported bugs during team-modeling sessions. This solution, however, does not prevent potential conflicts that may arise when multiple users are editing the same feature or feature tree. In its current implementation, CAD Interop avoids these conflicts because the only features supported do not remove geometry. An edge blend feature, for example, removes an edge during its operation. If another user were to, at the same time, create a feature that uses that edge, a conflict would arise. A server-level consistency manager should be implemented to make the software more robust and handle this type of conflict, though the existing solution is effective. In addition, a feature-locking conflict avoidance system as implemented in homogeneous multi-user CAD applications [8,9,20] would solve many of these problems.

This process for identifying, prioritizing, and fixing errors that arise has been effective in rapidly improving the state of the software. As development continues on the NPCF, this categorization process will continue to classify these types of errors to maintain this level of progress.

3.4. NPCF verification

As a result of this research, associations between features and methods for preserving design intent have been incorporated into the multi-user interoperability program utilizing the NPCF prototype standard. Neutral definitions for coordinate systems and datum planes, as well as extended definitions for extrude and revolve features, have been defined and included in the heterogeneous system. Additionally, support for PTC Creo has been added to enable multi-user synchronous modeling between NX, CATIA, and Creo CAD systems. The capabilities of the software and NPCF prototype standard are demonstrated by modeling two separate assemblies. The assembly models were selected to utilize the features and test the methods developed and implemented as a result of this research. Both assemblies were modeled by multiple users simultaneously. All users were co-located and allowed to collaborate prior to beginning modeling. While users in these sessions were co-located, the same process could occur between geographically separated users by using a video-conferencing service.

The multi-user assembly modeled during this verification process was designed to showcase the associativity methods developed within the Neutral Parametric Canonical Form. Three clients using NX, CATIA, and Creo were given instructions prior to modeling, which included the basic dimensions of each component of the assembly and the order of operations employed by a single user to model the part. During modeling, clients simultaneously modeled within the same parts, collectively working to complete the design before moving to the next component. The absence of any conflict resolution implementations in this architecture, however, necessitated that clients communicate to avoid interfering with other's operations. This method was illustrated early in the modeling session as displayed in Figure 7. After the initial cylinder was extruded to define location and size, each client simultaneously modeled geometry based off the initial feature, although in relative isolation.

Figure 8 shows the finished pump bearing housing part model on all three CAD clients. Collaboration between clients enabled each user to work on portions of the model to collectively complete the model. A process similar to the one employed in this session not only enables users to work with the CAD system in which they are most comfortable, but also enable users with different specialties to access and contribute to the part model during the design stage.

To test the software's ability to exactly replicate models between CAD systems, the finished part model from

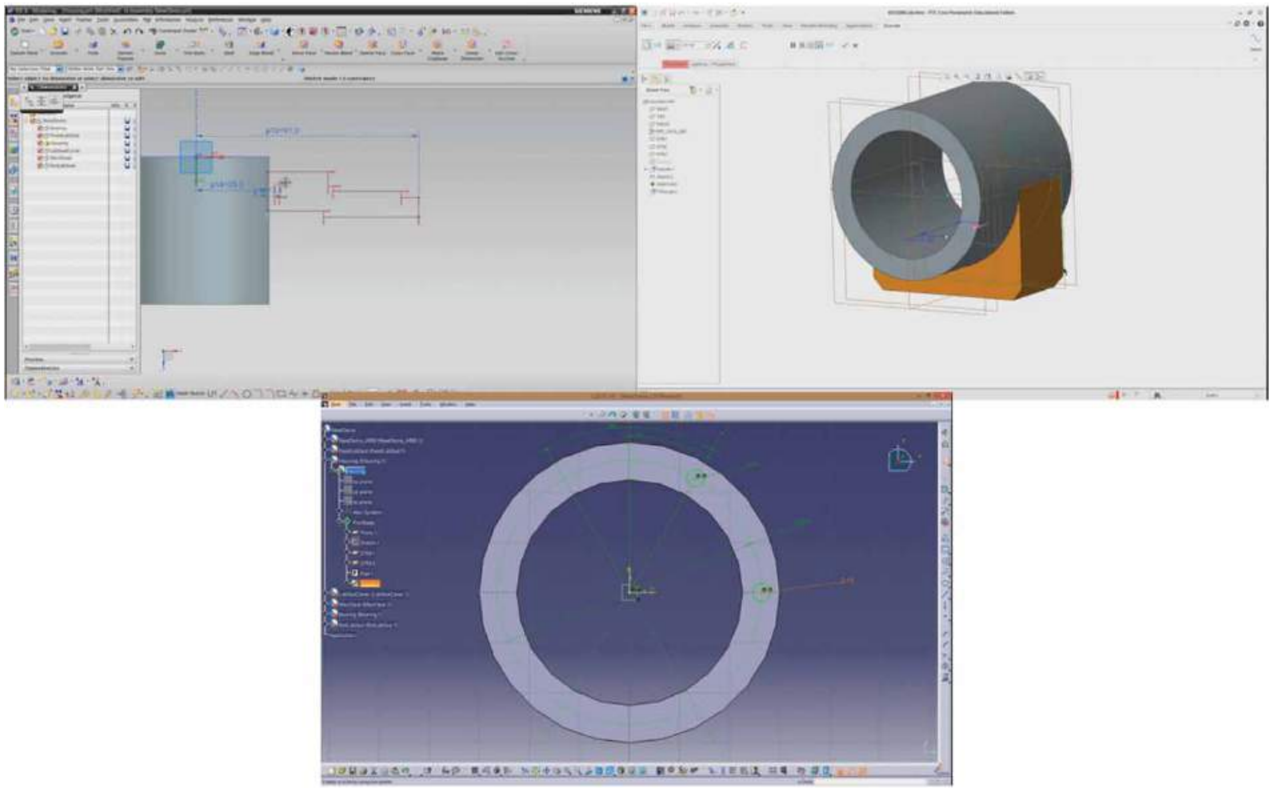


Figure 7. Early in the modeling session, the NX client (top left), Creo client (top right), and CATIA client (bottom middle) created new features referencing other's modeling operations.

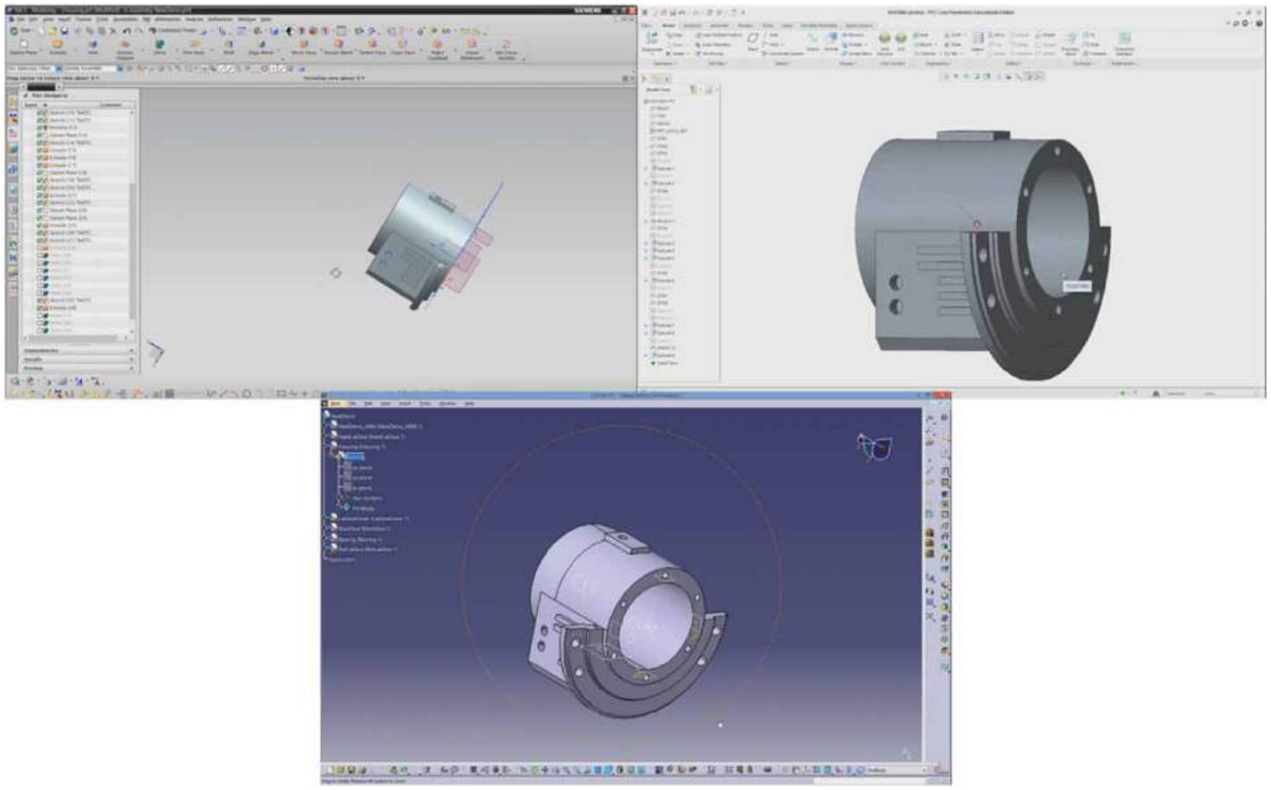


Figure 8. Completed assembly modeled simultaneously by an NX client (top left), a Creo client (top right), and a CATIA client (bottom middle).

Table 3. Model parameter comparison of exported stereolithography files.

Parameter	NX	CATIA	Creo
X-Dimension	100.000 mm	100.000 mm	99.990 mm
Y-Dimension	82.500 mm	82.478 mm	82.495 mm
Z-Dimension	60.000 mm	60.000 mm	60.000 mm
Volume	114612.7 mm^3	114721.5 mm^3	114666.0 mm^3
Surface Area	35687.9 mm^2	35663.1 mm^2	23665.4 mm^2

Table 4. Guided Model Rocket Components List.

Component	Quantity
Motor Subframe	1
Servo Motor	4
Guidance System	1
Nose Cone	1
Lower Fin	3
Upper Fin	4
Motor	1
Batter	1
Nozzle	1
Skin	1
IMU	1

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

In addition to testing the accuracy of recreating models in multiple CAD systems, load tests were performed to determine the scalability of the NPCF architecture. A guided model-rocket assembly, consisting of the 11 unique components in Table 4, was designed and modeled by six clients simultaneously within the heterogeneous environment. Modeling responsibilities were divided and strategy discussed between participants prior to beginning modeling. Though some components were multiple instances of the same part (i.e., Servo Motors and Fins), they were all explicitly modeled in their proper position and orientation due to the lack of support for assembly constraints.

While each client modeled their respective components, geometry created by remote users were automatically integrated into the local assembly. As users alternate between creating sketches and constructing geometry,

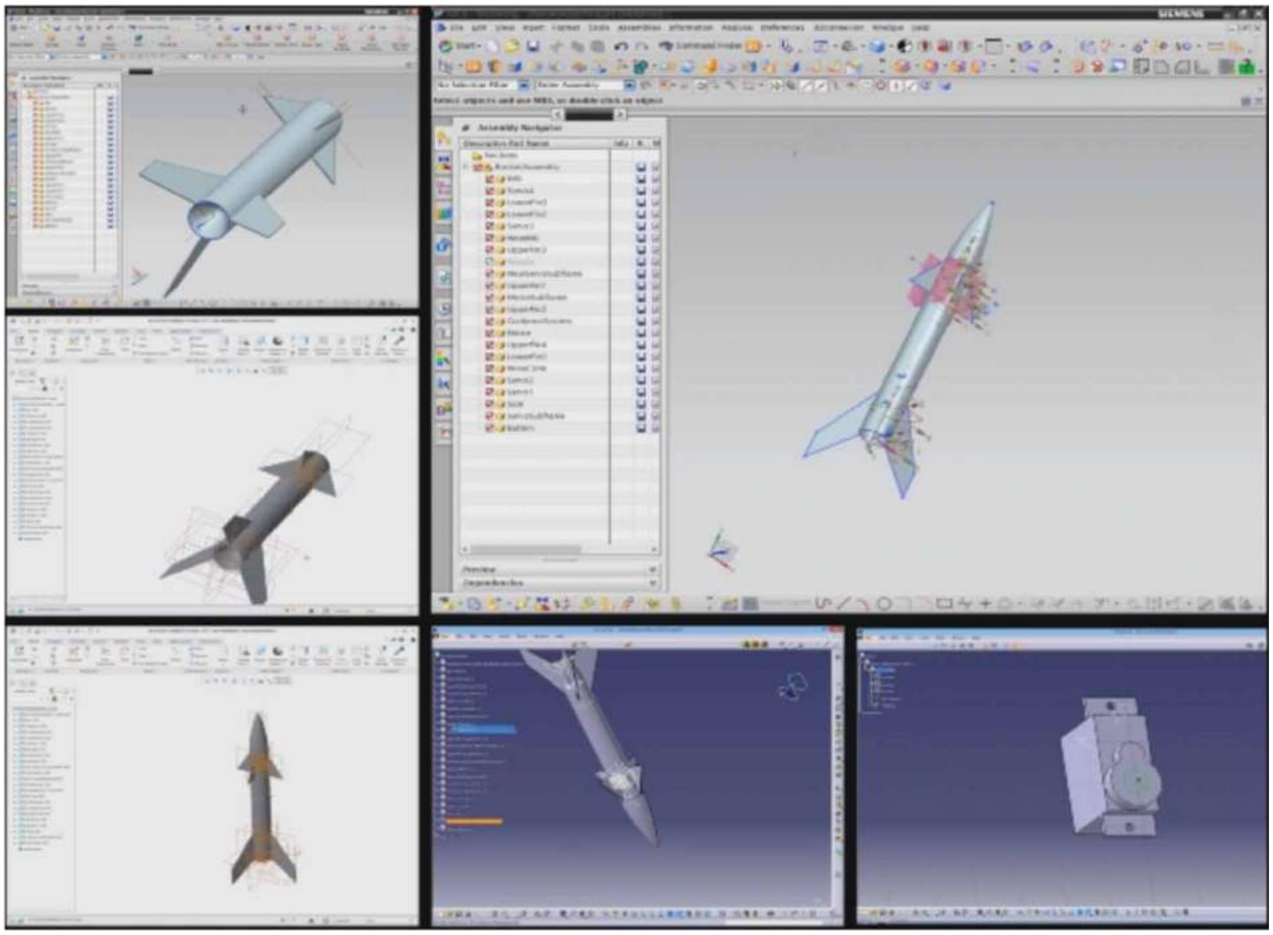


Figure 9. Screenshot of rocket assembly as modeled by six clients simultaneously.

remote users' work can be referenced and checked. This kind of awareness, similar to the collaboration around the drafting table, fosters interaction between designers to help and check each other where needed. A graphic of the six clients collaborating is included as Figure 9.

4. Conclusion

Inadequate CAD interoperability solutions continue to affect the design process employed by engineers and designers. As concurrent engineering and design sharing practices continue to be implemented into the typical design process, the shortcomings of the current heterogeneous CAD environment will become increasingly evident. With billions of dollars spent on fixing data after a CAD part or assembly model is transferred, there is a great need to update the translation process to preserve the design intent stored within the original model. Design intent enables CAD parameters and geometry to update according to the constraints set by the original design team. It is especially important in a multi-user setting where designers could be dispersed geographically thereby limiting face-to-face communication. The Neutral Parametric Canonical Form seeks to address these concerns by creating a heterogeneous CAD environment supporting simultaneous modeling. By utilizing a SQL database to store the neutral representations of features, part and assembly models can be created and simultaneously edited by multiple clients at once. In addition, rules implemented on the database enforce referential data integrity by declaring the parameters required to represent a feature and by enabling only valid inter-feature references. These architectural decisions improve the translation of design intent between CAD systems and preclude invalid data from corrupting neutral models stored within the database.

While only a limited number of features are currently supported, they are sufficient to create interesting and complex part and assembly models. An initial limited feature set was specifically chosen to focus on the robustness of the solution. Current work is focused on increasing the number of supported features while maintaining stability.

ORCID

Daniel R. Staves http://orcid.org/0000-0003-3365-1160
John L. Salmon http://orcid.org/0000-0002-8073-3655
Walter E. Red http://orcid.org/0000-0002-9321-6913

References

[1] Basu, D.; Kumar, S. S.: Importing mesh entities through IGES/PDES, Advances in Engineering Software, 23(3), 1995, 151–161. http://dx.doi.org/10.1016/0965-9978(95)00075-5.

[2] Brunnermeier, S. B.; Martin, S. A.: Interoperability costs in the US automotive supply chain, Supply Chain Management: An International Journal, 7(2), 2002, 71–82. http://dx.doi.org/10.1108/13598540210425821

[3] Cheng, Y.; He F.; Wu, Y.; Zhang, D.: Meta-operation Conflict Resolution for Human-Human Interaction in Collaborative Feature-Based CAD Systems. Cluster Computing. 19(1), 2016, 237–253

[4] Choi, G.-H.; Mun, D.; Han, S.: Exchange of CAD Part Models base on the Macro-Parametric Approach, International Journal of CAD/CAM, 2(1), 2002, 13–21

[5] Gu, H.; Chase, T. R.; Cheney, D. C.; Johnson, D.: Identifying, correcting, and avoiding errors in computer-aided design models which affect interoperability, Journal of Computing and Information Science in Engineering, 1(2), 2001, 156–166. http://dx.doi.org/10.1115/1.1384887.

[6] Haenish, J.: CAD-Exchange Towards a First Step Implementation, Industrial Electronics Society, 16th Annual Conference of IEEE, 1990, 734–739. http://dx.doi.org/10.1109/IECON.1990.149231.

[7] Han, S.: Macro-parametric: an approach for the history-based parametrics, in Soonhung Han (guest editor), Special issue: The future of CAD interoperability: History-based parametrics, Int. J. Product Lifecycle Management (IJPLM), 4(4): 321–325 Dec. 2010.

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

[9] Hepworth, A.; Tew, K.; Trent, M.; Ricks, D.; Jensen, C.; Red, W. E.: Model consistency and conflict resolution with data preservation in multi-user computer aided design, Journal of Computing and Information Science in Engineering, 14(2), 2014. http://dx.doi.org/10.1115/1.4026553.

[10] Jing S, He F, Han S, et al. A method for topological entity correspondence in a replicated collaborative CAD system. Computers in Industry, 60(7), 2009, 467–475.

[11] Leach, L. M.: Language interface for data exchange between heterogeneous CAD/CAM databases, Dissertation Abstracts International Part B: Science and Engineering, 44(5), 1983

[12] Li, M.; Yang Y.; Li, J.; Gao, S.: A preliminary study on synchronized collaborative design based on heterogeneous CAD systems, PhD thesis, Zhejiang University, 2003. http://dx.doi.org/10.1109/CACWD.2004.1349025.

[13] Li, M.; Gau, S.; Li, J.; Yang, Y.: An approach to supporting Synchronized Collaborative Design within Heterogeneous CAD Systems, ASME 2004 International Design Engineering Technical Conferences, 2004, 511–519. http://dx.doi.org/10.1115/DETC2004-57703.

[14] Li, J.; Han, S.; Shin, S.; Lee, S.; Kang, Y.; Cho, H.; Kim, H.; Song, I.; Kim, I.; Rathore, P. S.: CAD Data Exchange Using the Macro-Parametrics Approach: An Error Report, International Journal of CAD/CAM, 10(2), 2010.

[15] Li, M.; Gao, S.; Wang, C. C. L.: Real-Time Collaborative Design With Heterogeneous CAD Systems Based on Neutral Modeling Commands, Journal of Computing and Information Science in Engineering, 7(2), 2007, 12–15. http://dx.doi.org/10.1115/1.2720880.

[16] Li, W.; Ong, S.; Fuh, J.; Wong, Y.; Lu, Y.; Nee, A.: Feature-based Design in a distributed and colaborative environment, Computer-Aided Design, 36(9), 2004, 775–797. http://dx.doi.org/10.1016/j.cad.2003.09.005.

[17] Li, X.; He, F.; Cai, X.; et al: A method for topological entity matching in the integration of heterogeneous CAD systems. Integrated Computer-Aided Engineering, 20(1), 2013, 15–30.

[18] Li, X.; He, F.; Cai, X.; et al: CAD data exchange based on the recovery of feature modelling procedure. International Journal of Computer Integrated Manufacturing, 25(10), 2012, 874–887.

[19] Marjudi, S.; Amran M.; Abdullah, K. A.; Widyarto, S.; Majid, N.; Sulaiman, R.: A Review and Comparison of IGES and STEP, Proceedings of World Academy of Science, Engineering And Technology. 62, 2010, 1013–1017.

[20] Moncur, R.; Jensen, C.; Teng, C.; Red, E.: Data consistency and conflict avoidance in a multi-user CAx environment, Computer-Aided Design and Applications, 10(5), 2013, 727–744. http://dx.doi.org/10.3722/cadaps.2013.727-744.

[21] Mun, D.; Han, S.; Kim, J.; Oh, Y.: A set of standard modeling commands for the history-based parametric approach, Computer-Aided Design, 35(13), 2003, 1171–1179. http://dx.doi.org/10.1016/S0010-4485(03)00022-8.

[22] Mun, D.; Han, S.: Identification of Topological Entities and Naming Mapping for Parametric CAD Model Exchanges, International Journal of CAD/CAM, 5(1), 69–82, Dec. 2005.

[23] Pratt, M.J.: Extension of the Standard ISO10303 (STEP) for the exchange of parametric and variational CAD Models, Proceedings of the Tenth International IFIP WG, 5(3), 1998

[24] Wisnosky, D. E.: ICAM Program Prospectus, DTIC Document, 1977

[25] Zhang, D. J.; He, F. Z.; Han, S. H.; et al: Quantitative optimization of interoperability during feature-based data exchange. Integrated Computer-Aided Engineering, 23(1), 2016, 31–50.