QUALITY ANALYSIS OF COPY PASTE FEATURE IN CAD SOFTWARE NX

By

Jawal J. Panchal

10MMCC07

Under the guidance of

Prof. J. M. Dave

DEPARTMENT OF MECHANICAL ENGINEERING

AHMEDABAD-382481

May 2012

QUALITY ANALYSIS OF COPY PASTE FEATURE IN CAD SOFTWARE NX

Major Project Part-I

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in CAD/CAM

By

Jawal J. Panchal

10MMCC07

DEPARTMENT OF MECHANICAL ENGINEERING

AHMEDABAD-382481

May 2012

Declaration

This is to certify that

i) The thesis comprises my original work towards the degree of Master of Technology in Computer Aided Design and Computer Aided Manufacturing at Nirma University and has not been submitted elsewhere for a degree.

ii) Due acknowledgement has been made in the text to all other material used.

Jawal J. Panchal

Undertaking for Originality of the Work

I, **Jawal J. Panchal**, Roll. No.**10MMCC07** , give undertaking that the Major Project entitled "**Quality Analysis Of Copy And Paste Feature In CAD Software NX**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **CAD/CAM** of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____

Signature of Student

Date:_____

Place: NU, Ahmedabad

Endorsed by

(Signature of Guide)

Certificate

This is to certify that the Major Project entitled " QUALITY ANALYSIS OF COPY AND PASTE FEATURE IN CAD SOFTWARE NX " submitted by Jawal J. Panchal (10MMCC07), towards the partial fulfilment of the requirements for the degree of Master of Technology in Mechanical Engineering(CAD/CAM) of Nirma University of Science and Technology, Ahmadabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. J. M. Dave
Project Guide,
Department of Mechanical Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Mr. Vishal Deshpande
Industry Guide,
Siemens Industry Software (India) Pvt. Ltd.,
Hinjewadi, Pune

Prof. R. N. Patel
Head of Department,
Department of Mechanical Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. K. Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad

Acknowledgements

It is indeed a pleasure for me to express my sincere gratitude to those who have always helped me throughout my project work. This dissertation is by far the most significant technical accomplishment in my life and it would be impossible without people who supported me and believed in me. I am highly indebted to Prof. R. N. Patel.

I owe my most sincere gratitude to my Senior Manager Mr. Avick Chakraborty and Team Manager Mr. Vishal Soni at Siemens PLM Software India Pvt. Ltd, Pune, who gave me the opportunity to work with them. Special thanks for accepting me to carry out substantial work at the project's facilities and enabling me access the project's data base. Most of all I would like to thank Mr. Vishal Deshpande. He is not only my industry guide with deep vision but most importantly a kind person. His trust and excitement inspired me in the most important moments of making right decisions and I am very glad to work with him.

Many thanks go to my project guide Prof. J. M. Dave, Head of the Department of Mechanical Engineering, Institute of Technology, Nirma University Ahmedabad, whose assistance was vital for this dissertation, his wide knowledge and his logical way of thinking have been of great value for me.

I am deeply grateful to acknowledge with much appreciation the crucial role of the NX-CAD team who gave me time of their tight schedules and help me out in difficult situations by detailed and constructive comments, and for their important support throughout this work.

- Jawal J. Panchal

10MMCC07

Abstract

In today's world most of the products developed are dependent upon CAD softwares for its design manufacturing and analysis. Thus a CAD software comprises of a very important position in product life cycle. Moreover there is a lot of competition in the market so if a CAD system wants to survive in the market it needs to developed and improved continuously without losing the robustness and consistency of the features.

For this reason CAD software NX needs to improve the robustness of its copy and paste feature as well as the framework improvement of the copy/paste is also needed. Thus this thesis deals with improving the robustness of the copy/pasting different features and various other issues related to the Copy/Paste. Some of the problems the copy/Paste feature of the NX software is facing for the time being are Direction sensitivity, No robust input, dependency of the copy/paste feature on many other features, improper failure messages and incorrect feature references names in the paste dialogue.

Related to the copy/paste framework, there are three large code pieces in copy/paste the need to be reorganized and cleaned up so that further improvement to the framework functionality can be easily implemented with robustness. It will also make the future maintenance and any further enhancement less costly. This capability would improve productivity and usability of part modeling a product using widely known, renowned and global leader of CAD software Siemens NX. Also it will maintain the robustness and consistency across its various features. It also ensures that the pre-existing functionality does not regress.

Key words: Topological entities, geometric entities, NX CAD, Feature Modeling, UI- components, UI Framework, Journamation, Copy, Paste, Mapping, Labeling, Parent Children Relationship, Direct Parents, Indirect Parents.

Contents

List of Figures

Nomenclature

NX	Next Generation
CSG	Constructive Solid Geometry
$BREP$	Boundry Representation
$NURB$	Non-Uniform Rational B-Spline
STE	Sketcher Task Environment
UI	User Interface
$UICOMP$	User Interface Component
$UIFW$	User Interface Framework
$GWIF$	Graphical Window Interface
$OpenGL$	Open Graphics Library
KID	Kernel Interface Driver
DLL	Dynamically Linked Library
WCS	Work Co-ordinate System
JA	Journamation - Journaling + Automation
API	Application Programming Interface
SC	Smart collector
$FREC$	Feature Record
PK	Parasolid Kernel System
FMI	Feature Method Interface

Chapter 1

Introduction

1.1 Introduction to NX CAD

NX is the commercial CAD/CAM/CAE software suite developed by UGS. It is commonly referred to as a 3D PLM software application. All stages of product development are supported, from conceptualization, to design (CAD), to analysis (CAE), to manufacturing (CAM).

NX was previously known as Unigraphics. NX is a parametric solid / surface feature based package based on Parasolid. NX is widely used throughout the engineering industry, especially in the automotive and aerospace sectors. NX can provide users with industry specific tools known as wizards to help streamline and automate complex processes, as well as improve product quality and user productivity. For example, the Mold Wizard module addresses the specific requirements of plastic injection applications while the Progressive Die Wizard addresses the specific requirements of die stamping applications.

1.2 Introduction to UI-Components

A UI Component is a class that defines the entire user interface for any particular piece of data, such as a vector, section, or application specific data. For example, the Sheet Metal Tab feature's UI appears below (Figure 1.1).

This UI accepts a value for the thickness of the sheet, which can be modified by the user in several different ways:

1. Dragging the linear handle
2. Keying in value from either the dynamic gwif or dialog
3. Using Design Logic from either the dynamic gwif or dialog
4. Selecting a location to snap to
5. Flipping the direction using either the pull-down menu or double clicking the handle.

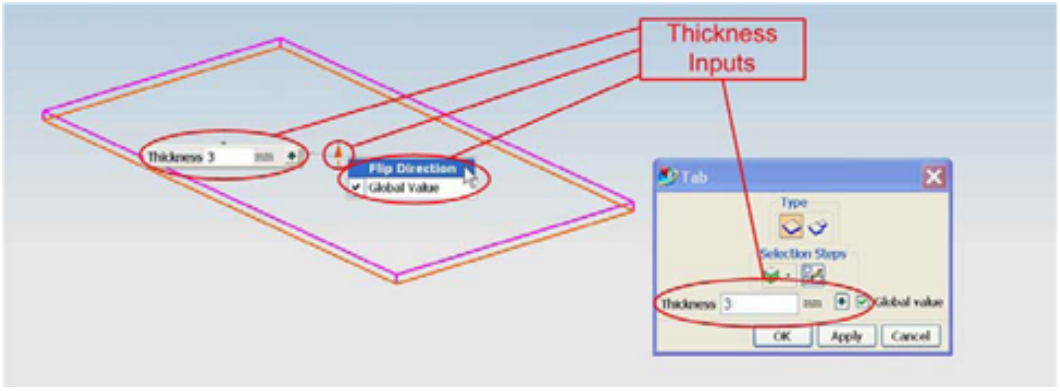

Figure 1.1: Various forms of input for a Tab Thickness

Rather than writing all the code necessary to handle these different forms of input, the Tab UI can use a previously written UI Component for values represented by a

linear handle - the UICOMP_linear_dim component.

All of the code necessary to support the interactions listed above is embedded within this Component, and the Tab UI developer can enable it with very little effort - he simply adds the component to the Tab UI. Any other application that needs a similar interaction (such as Extrude, Offset, Blend, and so on) can use it as well. Tab is notified when any change occurs to the value associated with this component, and can update its preview accordingly.

Each UI component can be considered a 'miniature application' since they can contain any aspect of UI that would typically appear in an application, including

1. Dialog Items
2. Handles
3. Dynamic Gwifs
4. Design Logic
5. Menu Bar (Static Gwif) Items
6. Popup menu entries
7. Selection
8. Selection Steps and Types

Components can also leverage other components internally. This use is completely shielded from the client application, just like the rest of the UI code contained within the component.

In addition, UI components ensure that the states of all these forms of interaction are managed, so that selections become active when expected, static gwifs and dialog items are synchronized when modified, and dynamic gwifs and dialog items are synchronized.

1.3 Introduction to UI Framework

There are two roles that the UI Framework (the UIFW module) serves. First, with all these UI Components floating around it is necessary to have a framework present to execute them and bind them together into a cohesive application.
Some examples of these responsibilities are:

1. Operating the application UI, passing the application UI as an argument.

2. Grafting all the UI Components together into a single UI. This includes producing the dialog, dialog bar, displaying all of the handles, and so on.

3. Managing selection steps, selection focus, dynamic gwif visibility, synchronization of various representations of the same data, and other state management.

4. Undo and failure recovery during ok, apply, and cancel

Secondly, UIFW contains many helper classes that make building lower level components easier. Without getting into specifics at the moment, these classes make working with handles, gwifs, mouse interpretation, and other aspects of UI much easier.

1.4 Introduction to Journamation

The word "Journamation" is a combination of the words "Journaling" and "Automation". This is the term used for the initiative to create an automation API and a record-and-replay capability in NX. This API is to be used by both NX's interactive UI to access application functionality and by our customers in order to create their own automation programs. It is designed such that when it is called by the UI, it is possible to generate a journal (in Visual Basic .NET) that can be replayed in order to re-execute the work that was performed interactively. The API is also designed

to support several different modern object-oriented programming languages for customer automation.

Fundamental to proper JA API design understands that the JA API is both a customer-facing interface and the interface to be used by NX UI code to access application functionality. This is quite different from previous Unigraphics/NX design practices where the customer interface was User Function, UG Open C++ or KF and the UI's interface was essentially the breadth of public routines available in the application libraries. Therefore, it is essential that rigorous standards be developed and followed for creating the JA API so that

1. The API is consistent, robust, and easy to understand for customers, and

2. The functionality will journal properly when record is turned on.

1.5 Journamation in NX

Figure 1.2 is an idealized (and simplified) view of the NX Architecture once the JA API is fully implemented. Note the following while examining this diagram:

- The NX UI and the various language APIs all share a common entry point into the application: the JA C API.

- The dashed line represents code that is generated entirely by processing a set of interface definition files, called JA files. The JA API is fully defined by what is in these JA files.

- The JAX C API layer lies immediately underneath the JA C API. This is the layer that contains the implementations of the methods defined in the JA files.

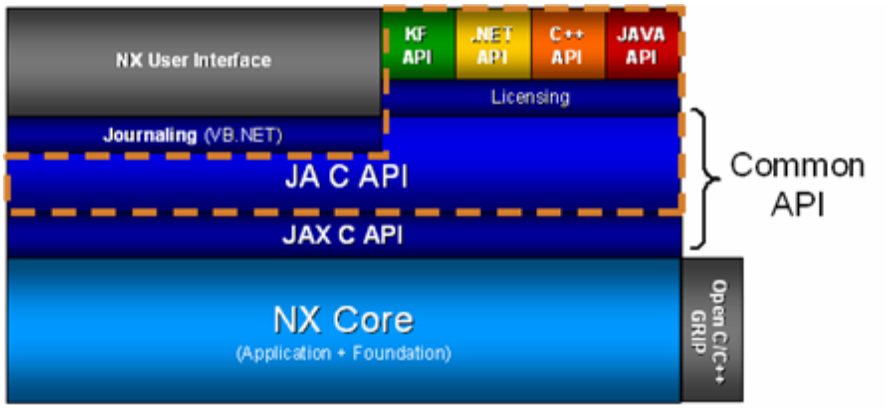

Figure 1.2: Idealized NX Architecture

1.6 Introduction to Copy and Paste

1.6.1 Problem Definition

Design and knowledge reuse is very significant to design process. Feature copy and paste is essential to design reuse. However, feature copy and paste as a general tool has robustness issues.

The robustness issues can be clarified in the following three areas which are stated as follows.

1. It is related to feature robustness to deal with parent changes and their response to input flipping when their output direction is sensitive to the input direction.

2. It is also depends on the code robustness in the copy and paste framework.

3. Building robust models is an important one as we saw in a lot of Catia user cases, designers trying to use body and feature input rather than individual output entities.

This project is mainly to improve the quality of the first two. We may need to provide smarter rules, or educate users to create robust models. But it is out of the scope of this project.

1.6.2 Methodology

1. Study of NX Modeling Software.
2. Study of NX framework, geometric kernel Parasolid, OM, and CMOD.
3. Study of Feature code.
4. Improving the direction sensitive issues related to any feature..
5. Adding robust input references and updating of the feature as expected.
6. Returning of the meaningful error messages on failure of any feature to successfully copy and paste.
7. Editing of the failed pasted features.

1.6.3 General Code Flow Of copy and Paste

Whenever user selects any single feature or multiple features to copy, that corresponding features are simply kept into the memory clipboard.
When user pastes them, we first get the dialog up. The following figure shows different blocks of the paste dialogue.

In the paste dialogue, we list all the feature references that user need to map in the dialog in order to copy these features correctly. There are two types of feature references. One is the direct parents and the other is indirect parents. The direct parents are those parents that not created by other pasted features and directly referenced by the features. The indirect parents are for those features that reference the outputs of other pasted features. As these feature's direct parents are not created yet while the UI needs up-in-front feature reference matching, we need to trace back what is used to label the direct parents until we find the labeling entities that are not outputs of

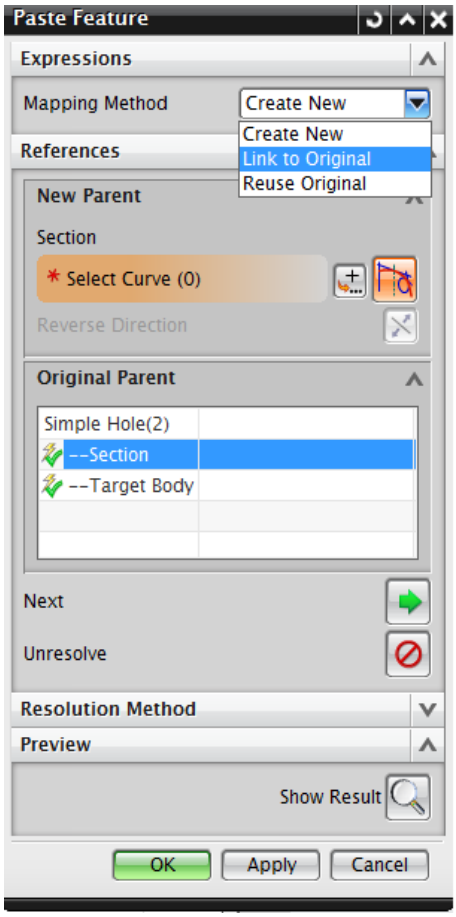

Figure 1.3: Paste Dialogue

any pasted features. To get correct feature references, we depend on feature's AskUiParents and AskLabelingEntitites methods.

After user matches all feature references that we ask them to match and Ok the dialog, we paste features one by one. For each feature, we create a new feature record and deep copy the parms. Then we invoke feature's copy method to replace the reference of the parms to those mapped. Then we update the feature to allow the outputs to be created. After a feature is updated, we map the outputs based on the input map and other criteria before we copy the next feature. The paste process depends on feature's copy method and most importantly the update method (CreateBody and apply methods). To map the outputs correctly, we also need or may need the

AskLabelingEntities andAskRmChildren methods correctly implemented.

1.6.4 Mapping Methods

Create New

The create new method creates the whole new feature records and deep copy the parms. It will ask to map all the basic as well as necessary feature references required to create the feature. As for example, when a three point arc is copy pasted, the create new method will ask for all the three references required to create the arc. The following figure describes the behavior of the create new method.

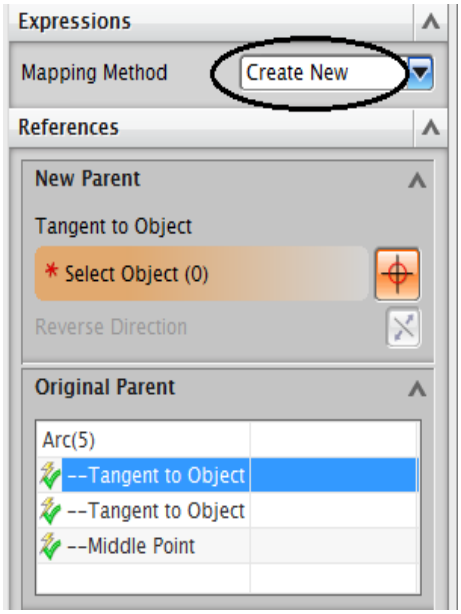

Figure 1.4: Create New

Link to Original

This mehtod will ask for only the necessary feature references to map instead of asking all the basic feature references. For the above example it will not ask for the middle

point of the arc and will only ask the two lines required as tangents.

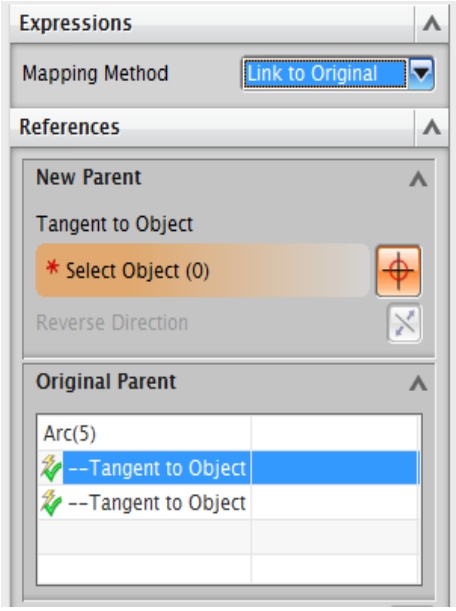

Figure 1.5: Link to Original

Reuse Original

This option simply copy pastes the copied features without asking any kind of feature references to map. The only thing it will ask is the coordinate system as to where to paste the copied entity.

Chapter 2

Literature Review

2.1 Solid Modeling

Programs that are capable of solid modeling can be much more powerful than simple wireframe modelers. These programs are used to build parts that are actually solid objects instead of simply a wireframe outline of the part. Since these parts are represented as solids, they have volume, and if given a density can have a weight and mass as well. The computer can calculate many physical properties of these parts, such as center of gravity and moments of inertia. These calculations can even be performed for irregularly shaped parts, for which manual calculations would be extremely difficult. Finite Element Analysis techniques can also be used to perform stress analysis of these parts. [1 3]

The weakness of wireframe and surface modeling is that they have Ambiguous and incomplete geometric description, lack topological information, tedious modeling process and awkward user interface. Solid modeling is based on complete, valid and unambiguous geometric representation of physical object.

1. Complete: points in space can be classified. (Inside/ outside)

2. Valid: vertices, edges, faces are connected properly.

3. Unambiguous: there can only be one interpretation of object

Analysis automation and integration is possible only with solid models. Solid models have properties such as weight, moment of inertia, mass. Solid model consist of geometric and topological data.

2.1.1 Geometry

Shape, size, location of geometric elements.

2.1.2 Topology

Connectivity and associativity of geometric elements. Non graphical, relational information Topology: The study of geometric properties of an object's shape those are unchanged when it is deformed according to some simple rules: Bending, twisting, and stretching are allowed, but cutting and gluing are not. Connectivity of the geometric elements of a model help to characterize its topology. For example, a polyhedron's topology is defined by specifying how the vertices are connected to form edges.

2.1.3 Connectivity

A topological property indicating how geometric elements of a model are interconnected.

2.2 Solid Modeling Methods

There are two basic methods used to create solid models. They are Constructive Solid Geometry (CSG) methods, and Boundary Representation (Brep) methods. CSG uses solid primitives (rectangular prisms, spheres, cylinders, cones, etc.) and Boolean operations (unions, subtractions, intersections) to create the solid model. B-rep methods start with one or more wireframe profiles, and create a solid model by extruding,

sweeping, revolving or skinning these profiles. The Boolean operations can also be used on the profiles themselves and the solids generated from these profiles. Solids can also be created by combining surfaces, which often have complex shapes, through a sewing operation [1 3]

These two methods can often be combined in order to create the desired parts. Each of these methods has its limitations, and parts which are very difficult to create using just one or the other method can be created much more easily using a combination of both methods. Thus, most commercial solid modeling systems are hybrids using both CSG and Brep methods.

2.2.1 Constructive solid geometry (CSG)

In CSG objects are represented as a combination of simpler solid objects (primitives). The primitives are cube, cylinder, cone, torus, sphere etc. Copies or instances of these primitive shapes are created and positioned. A complete solid model is constructed by combining these instances using set specific, logic operations (Boolean).

Each primitive solid is assumed to be a set of points, a boolean operation is performed on point sets and the result is a solid model. Boolean operations include union, intersection and difference. The relative location and orientation of the two primitives have to be defined before the boolean operation can be performed. Boolean operation can be applied to two solids other than the primitives. [1 3]
Boolean operations includes -Union, Difference, Intersection.

Union The sum of all points in each of two defined sets. (Logical OR).Also referred to as Add, Combine, Join, Merge.

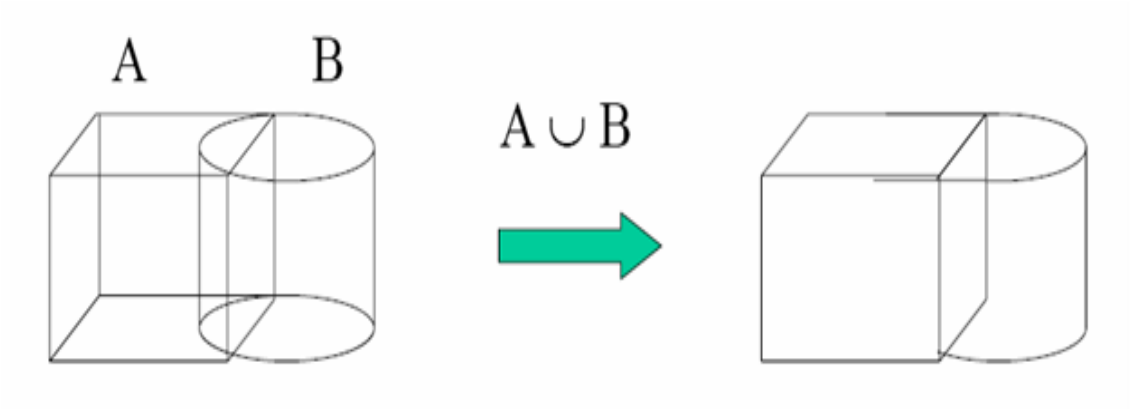

Figure 2.1: CSG Union

Difference The points in a source set minus the points common to a second set. (Logical NOT) Set must share common volumeAlso referred to as subtraction, remove, cut.

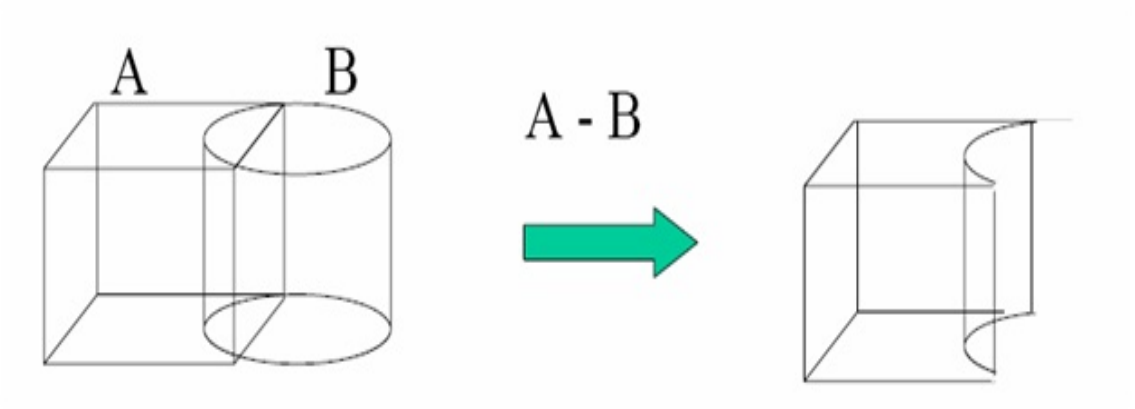

Figure 2.2: CSG Difference

Intersection Those points common to each of two defined sets (logical AND).Set must share common volume. Also referred to as common, conjoin Boolean operation is intuitive to user, easy to use and understand and provide for the rapid manipulation of large amounts of data. Data structure does not define model shape explicitly but rather implies the geometric shape through a procedural description. Object is not defined as a set of edges and faces but by the instruction: union primitive1 with primitive 2 .This procedural data is stored in a data structure referred to as a CSG tree. The data structure is simple and stores compact data and is also easy to manage.

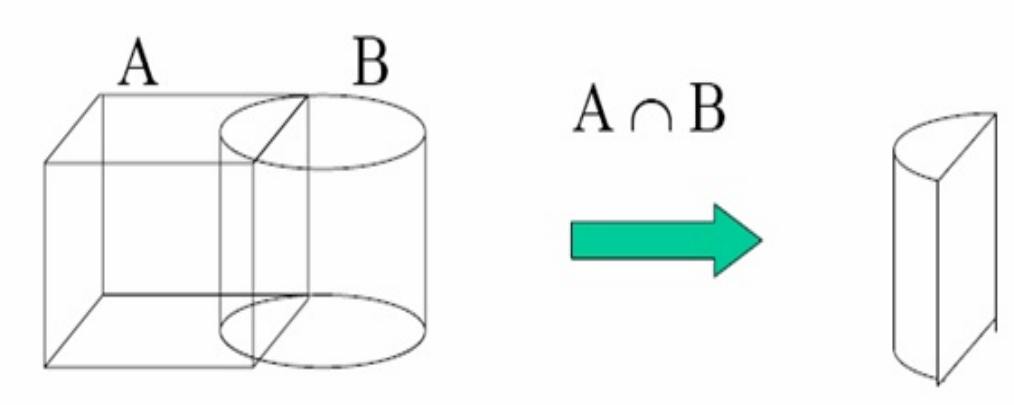

Figure 2.3: CSG Intersection

2.2.2 CSG tree

CSG tree stores the history of applying Boolean operations on the primitives. Stores in a binary tree format. The outer leaf nodes of tree represent the primitives. The interior nodes represent the Boolean operations performed. More than one procedure (and hence database) can be used to arrive at the same geometry.

Advantages of CSG

1. CSG representation is unevaluated .Faces, edges, vertices not defined in explicit
2. CSG model are always valid Since built from solid elements.
3. CSG models are complete and unambiguous
4. Easy to construct a solid model minimum step.
5. CSG modeling techniques lead to a concise database so less storage.
6. Complete history of model is retained and can be altered at any point.
7. Can be converted to the corresponding boundary representation.

Limitations of CSG

1. Only Boolean operations are allowed in the modeling process...With Boolean operation alone, the range of shapes to be modeled is severely restricted thus not possible to construct unusual shape.
2. Requires a great deal of computation to derive the information on the boundary, faces and edges which is important for the interactive display/ manipulation of solid.

Solution

1. CSG representation tends to accompany the corresponding boundary representation...Hybrid representation.
2. Maintaining consistency between the two representations is very important.

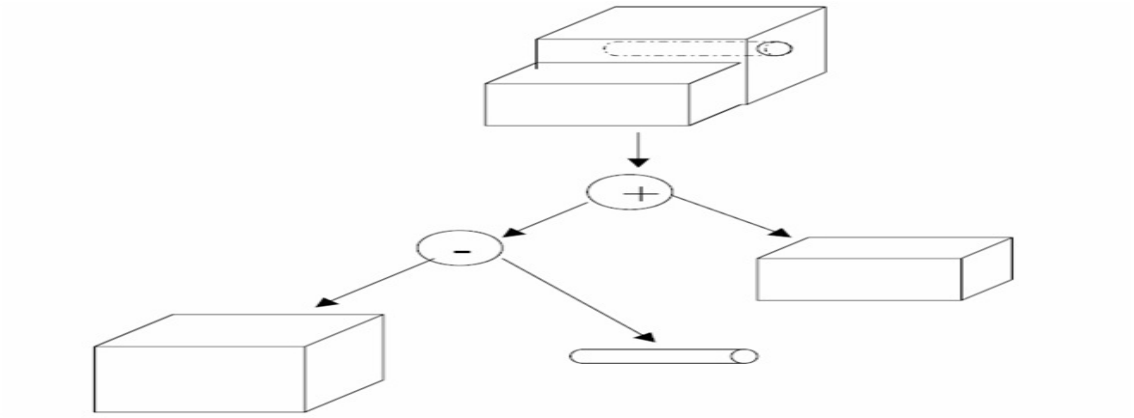

Figure 2.4: CSG Tree

2.2.3 Boundary representation (B-Rep)

Solid model is defined by their enclosing surfaces or boundaries. This technique consists of the geometric information about the faces, edges and vertices of an object with the topological data on how these are connected-Rep includes such topological information because a solid is represented as a closed space in 3D space (surface connect without gaps).The boundary of a solid separates points inside from points outside solid. [1 3]

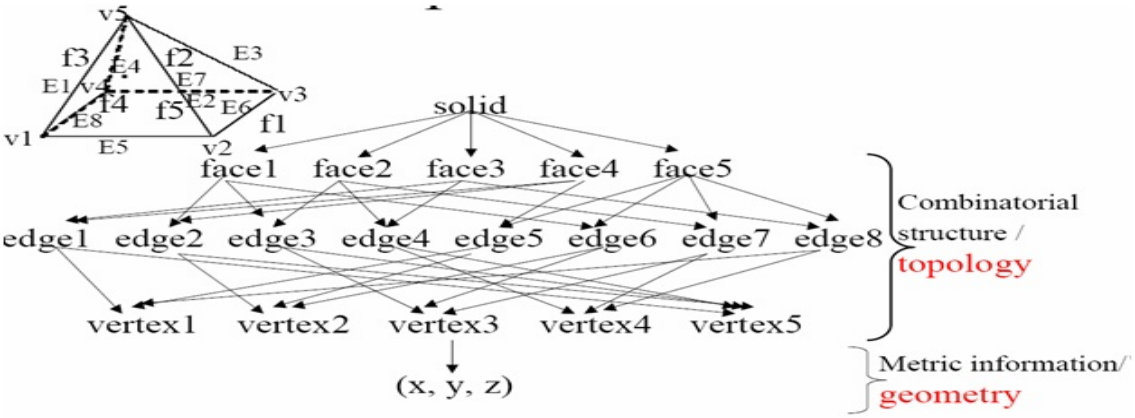

Figure 2.5: B-rep Data Structure

B-Rep vs. surface modeling Surface model is a collection of surface entities which simply enclose a volume lacks the connective data to define a solid (i.e topology).B-Rep model is a technique guarantees that surfaces definitively divide model space into solid and void, even after model modification commands.

B-Rep graph store face, edge and vertices as nodes, with pointers, or branches between the nodes to indicate connectivity.
Boundary representation-validity System must validate topology of created solid.B-Rep has to fulfill certain conditions to disallow self-intersecting and open objects. This condition include

1. Each edge should adjoin exactly two faces and have a vertex at each end.
2. Vertices are geometrically described by point coordinates.
3. At least three edges must meet at each vertex.
4. Faces are described by surface equations.
5. The set of faces forms a complete skin of the solid with no missing parts.
6. Each face is bordered by an ordered set of edges forming a closed loop.
7. Faces must only intersect at common edges or vertices.
8. The boundaries of faces do not intersect themselves.

Validity also checked through mathematical evaluation. Evaluation is based upon Eulers Law (valid for simple polyhedral no hole) **figure B-Rep Euler Law to be inserted here**
Expanded Eulers law for complex polyhedrons (with holes) Euler-Poincare Law: H number of holes in face, P-number of passages or through holes, B-number of separate bodies. **figure B-Rep Euler-Poincare Law to be inserted here**
Advantages of B-Rep

1. Capability to construct unusual shapes that would not be possible with the available CSG. e.g. aircraft fuselage, swing shapes.

2. Less computational time to reconstruct the image.

Disadvantages of B-rep

1. Requires more storage.

2. More prone to validity failure than CSG.

3. Model display limited to planar faces and linear edges, complex curve and surfaces only approximated.

2.3 Feature Based Modeling and Parametric Modeling

2.3.1 Feature Based Modeling

Dimension-driven design refers to a collection of solid-modeling capabilities that include parametric, and feature based. Feature-based modeling has, among engineers, rapidly become the preferred method of constructing solid models. In feature-based packages, solid models are constructed from geometric features such as slots, shells, bends, drafts, rounds, and so forth. The alternative is to construct models using mathematical geometric entities such as unions of spheres, cylinders, and boxes. One advantage of features is that they provide dimensions that correctly define how the feature behaves when dimensions change.[2]

For hole drilled through a plate, in a feature-based modeling system, the geometry has enough embedded intelligence to know that the hole should go all the way through the plate, regardless of how thick the plate is. Thus, even if the designer decides to

increase the plate thickness tenfold, the hole will still go through to the other side.

In a model defined with older solid geometry schemes, the designer would have to manually lengthen the hole if the plate became thicker. Otherwise, the hole would stop within the plate. The formal way of referring to this property is that a feature is capable of producing many different geometric instances, depending on the dimension values that the designer spells out. The most important aspect of feature-based techniques is that they capture design intent. In the drilled hole example, the designer intended to put a hole through the plate. This intent was maintained regardless of what changes were made in the plate dimension. Another important property of feature-based modelers is the ability to let a feature reference the geometry of various models in an assembly. This referencing allows changes made in one model to propagate to other affected models. One example is where a metal housing has features that are dimensioned from other parts mounted to the frame. When these parts move or change shape, the housing updates as well.

Feature-based models have been likened to a recipe approach to building solid models. Once a design has been specified, it is possible can be created in simple tables. This offers the opportunity to create generic designs. For example, adding a macro program that prompts for inputs would enable the creation of a customized solid-model assembly, complete with drawings and related data such as a bill of materials, cutter paths, and so forth. Feature-based modelers allow operations such as creating holes, fillets, chamfers, bosses, and pockets to be associated with specific edges and faces. When the edges or faces move because of a regeneration, the feature operation moves along with it, keeping the original relationships. The choices made developing these models are very important. If the features aren't referenced correctly, they may not end up in the correct place if the model is regenerated. A feature that is located at an X and Y offset from a corner of the face instead of at the center of the face will not remain at the center of the face when the model is regenerated unless constraints

are added to the model that will change the X and Y offsets to keep the feature at the center of the face.[2]

2.3.2 Parametric Modeling

Parametric methods depend on the sequence of operations used to construct the design. The software maintains a history of changes in specific parameters. The point of capturing this history is to keep track of operations that depend on each other, so that, whenever it is told to change a specific dimension, the system can update all operations referenced to that dimension.

For example, a circle representing a bolt hole may be constructed so it is always concentric to a circular slot. If the slot moves, so does the bolt circle. [1] Parameters are usually displayed in terms of dimensions or labels and serve as the mechanism by which geometry changes. The designer can change parameters manually by changing a dimension, or by referencing them to a variable in an equation that is solved by the modeling program itself. Parametric modeling is most efficient in working with designs where changes are likely to consist of dimensional changes rather than grossly different geometries.

Important feature of modern CAD systems is the ability to create parametric models. In a parametric model, each entity, such as a Boolean primitive, a line or arc in a wireframe, or a filleting operation, has parameters associated with it. These parameters control the various geometric properties of the entity, such as the length, width and height of a rectangular prism, or the radius of a fillet. They also control the locations of these entities within the model.[2]

These parameters can be changed by the operator as necessary to create the desired part. Parametric modelers that use a history-based method keep a record of how the model was built. When the operator changes parameters in the model and regen-

erates the part, the program repeats the operations from the history, using the new parameters, to create the new solid. There are many uses for this type of modeling. Designers can test various sizes of parts to determine which is the "best" part for their use by simply adjusting the model parameters and regenerating the part. Some parametric modelers also allow constraint equations to be added to the models. These can be used to construct relationships between parameters.

2.4 Overview of Parasolid

Parasolid offers high performance modelling in a broad range of areas, giving us the ability to create new models or edit existing models using an unparalleled range of tools and performance. The major areas of functionality supported includes

1. Complex blending: Parasolid provides a wide range of blending on complex geometry with unmatched reliability. Capabilities include rolling ball, variable-radius, face-face, cliff-edge, overlapping, curvature continuous, disc, conic-section, and conic hold line.

2. Hollowing, shelling, offsetting and thickening of surfaces: These invaluable techniques create thin-walled parts. They are conceptually simple to the CAD user but the topological and geometric changes required are a rigorous test of a modeler's reliability.

3. Tapering and parting line calculations: These operations primarily serve mold and die designers. Parasolid can apply a taper that follows complex parting lines. These capabilities and non-uniform scaling make Parasolid a powerful mold-design tool.

4. Local operations: Many Parasolid operations may be applied to specific areas of a body, such as faces or groups of faces. Parasolid ensures the integrity of models to which these operations are applied remains intact, by making appropriate

changes to surrounding topology and geometry. Supported local operations include replacing and transforming specific topologies, hollowing and offsetting, and model simplification via the identification and removal of features such as holes and blends.

5. Complex modeling using B-surfaces: Parasolid contains fully integrated B-curves and B-surfaces using industry standard NURBs. The kernel also simplifies geometry to analytic surfaces (planes, cylinders, cones, spheres, and tori) whenever possible to optimize reliability and performance. If more free-form construction techniques are required, NURBS of any shape can be arbitrarily trimmed to meet a designer's concept. Such surfaces can be sewn to construct a solid or a sheet model as required. Rendering support for large models: Parasolid supports rendering large models, including the ability to preserve memory usage by only rendering areas of interest, or by ignoring small features that may not be relevant in a particular context.

6. Application support: A major area of functionality is dedicated support for application-specific functionality, such as the ability to store attribute information on modelling entities, comprehensive rollback and partitioning capabilities, debugging facilities for the developer, and enhanced performance for multi-processor machines.

Parasolid also offers the developer unprecedented accuracy. Parasolid's default resolution is 10e-8 in a world size of 10e3. This gives an accuracy ratio of 10e11, which is an ORDER OF MAGNITUDE more accurate than any other kernel modeler's.[10]

2.4.1 Relationships between Parasolid entities

The major modeling entities that Parasolid uses to create its model structure and how they are combined so as to construct a Parasolid model are explained below. The entities available fall into three broad categories: topological, geometric, and

associated data. Their general relationships are shown in the following diagram [10].

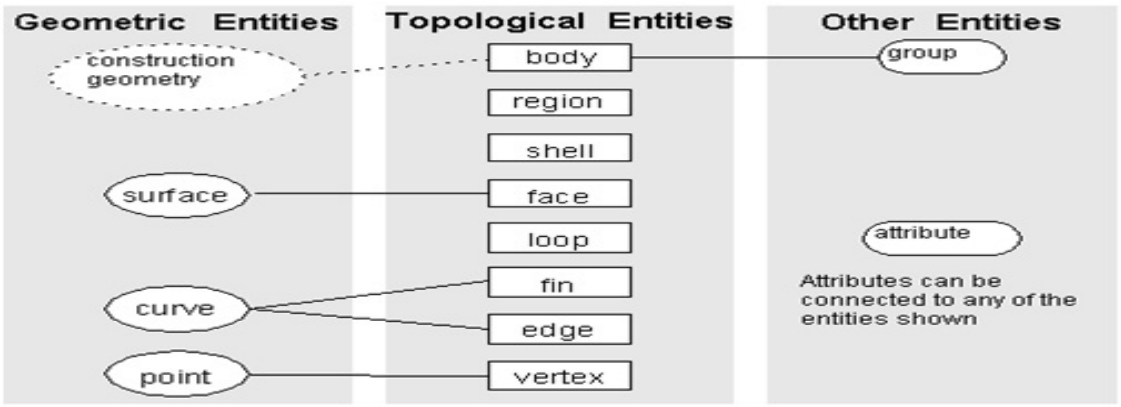

Figure 2.6: Relationship Betwen Parasolid entities

Identifying Entities: It is essential for both Parasolid and for our application to be able to identify different entities in a model. Parasolid provides two methods for doing this:

Tags are used to identify particular entities within a Parasolid session. Every entity in a model has an associated tag that is created when the entity is created or enquired about. Each tag is unique within a session. Tags are held in integer variables which Parasolid functions use as pointers to entities.

Identifiers (IDs) can be used to identify particular entities between Parasolid sessions. Parasolid automatically attaches unique IDs to entities in a model because Parasolid does not preserve tags between different sessions: when the same part is loaded into two different sessions, its entities may have different tags.

Topological Entities Topological entities comprise all the entities that are used to construct the structure or skeleton of a model.

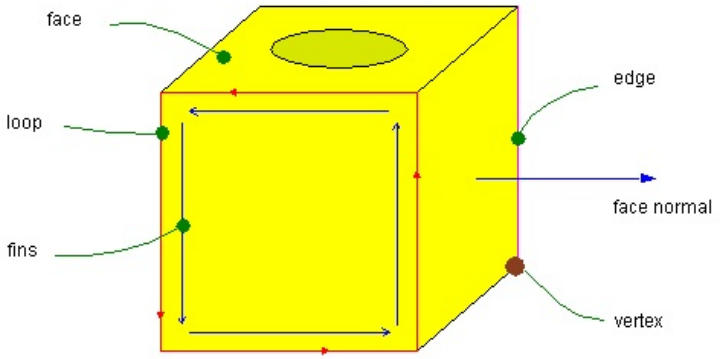

Figure 2.7: Topological entities in a body

Bodies Bodies are fundamental to modeling with Parasolid. A body is composed of one or more connected entities, or components. Bodies can contain the topological entities shown

Topology Description

1. Face-A face is a bounded subset of a surface, whose boundary is a collection of zero or more loops. A face with zero loops forms a closed entity, such as a full spherical face.

2. Loop- A loop is a connected component of a face boundary. A loop can have:an ordered ring of distinct fins, and a set of vertices.

3. Fin-A fin represents the oriented use of an edge by a loop.

4. Edge-An edge is a bounded piece of a single curve. Its boundary is a collection of zero, one or two vertices.

5. Vertex-A vertex represents a point in space. A vertex has a single point, which may be null

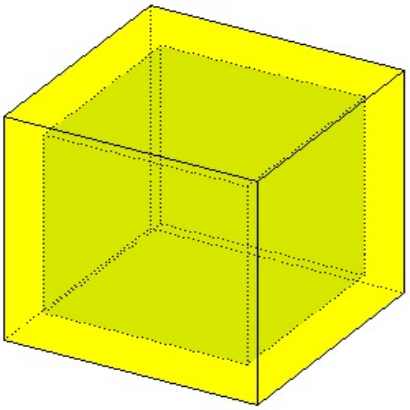

Figure 2.8: Hollow cube consisting of three regions

For a fully defined, valid model, every face, edge and vertex in a body requires an attached geometric entity: a surface, curve or point. **Regions** A region is a connected subset of 3-dimensional space bounded by a collection of vertices, edges, and oriented faces. Regions are either solid (contain material) or void (empty).
Bodies always have an infinite void region, which we can think of as all the space outside of the body itself. The sum of all regions in a body comprises the whole of 3D space. For example, the Figure shows a hollow cube that contains three regions:

1. The void region in the center of the cube.
2. The solid region occupied by the material of the cube.
3. The (infinite) void region outside the cube.

Regions contain a set of shells that define the topological structure of the parts in the body. Note: A face is a two-dimensional analogy of a region. An edge is a one-dimensional analogy of a region. A vertex is the zero-dimensional analogy of a region.
Shells A shell is a connected collection of oriented faces (each face used by the shell on one or both sides of the face) and edges. The shells in a region do not overlap or share faces, edges or vertices. Note: A loop is the two-dimensional analogy of a shell.

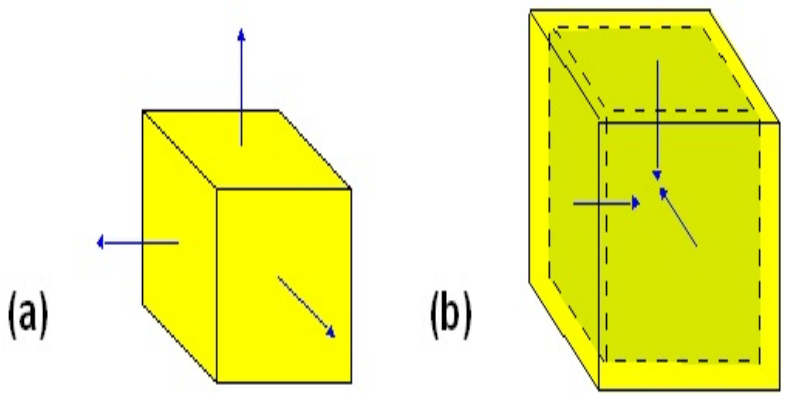

Figure 2.9: Face normals of solid bodies

Face normals In a solid body, the normals of each face must always point away from the solid region:

1. For an outer shell, the normals point away from each other.
2. For an inner shell (i.e. a void inside a solid) the face normals point towards each other.

Sometimes, a modeling operation may create a negative body, that is, a body whose face normals point the wrong way. Such bodies can easily be negated.
For sheet bodies, face normals must also point away from the body. The face normals in a sheet body must be consistent, as shown in Figure.

Direction of loops, fins, edges Loops, fins and edges have directions, and Parasolid defines the following conventions for their relationships to face normal and to each other:

1. The forward direction of a loop has the face on the left of the loop, when viewing the face down the face normal.

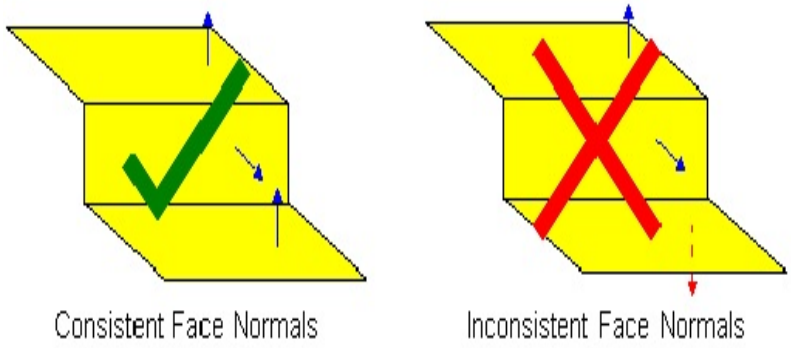

Figure 2.10: Face normals of sheet bodies

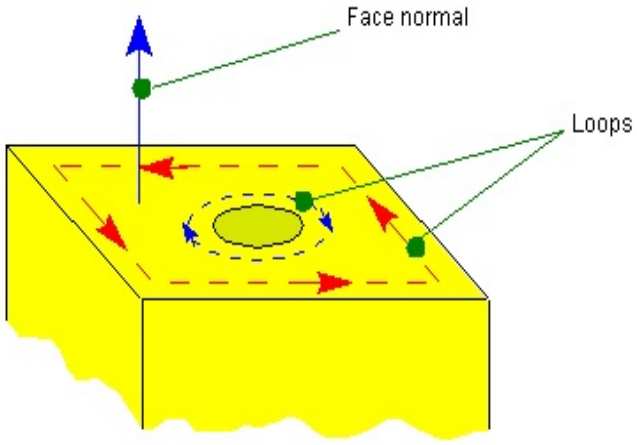

Figure 2.11: Loop Directions of Top face of a cube with cylindrical hole

2. A loop represents one boundary of a face as a closed set of fins, therefore the direction of the fin is the same as that of the loop that contains the fin.

3. An edge, which (on a manifold solid) has a left and right fin, takes its direction from the left fin (which takes its direction from its loop)

Parasolid Geometry: The geometric relationships between the different components of a body are known as its principle geometry. In a completely defined part, geometric entities are attached to topological entities to represent these relationships. We can also attach geometry directly to a body to form construction geometry, as a means of keeping relevant entities with the body. For example, we might want to attach a point to a body that represents its center of gravity.
Parasolid also supports orphan geometry, which is not attached to any topological entity. This is useful if we want to create geometry prior to using it in a body, for example, to create a surface before incorporating it into a body.
There are three main classes of geometric entity:

1. **Point**- Points are principally attached to vertices. All points are Cartesian.

2. **Curve**- Curves are principally attached to edges or fins of the model. There are several subtypes of curve, such as circle, ellipse, and line.

3. **Surface** -Surfaces are principally attached to faces. Normally every face has an attached surface, but it may be detached temporarily as the model is being built or modified. There are several subtypes of surface, such as plane, sphere, and cone.

All of these classes may also be attached directly to bodies as construction geometry. In order to reduce the overall size of a model, geometric entities can be shared where appropriate. Geometry sharing only occurs in a single body and works for the foll. cases:

1. Edges or fins with a common direction can share a common curve.

2. Faces can share a common surface.

2.4.2 Accuracy of Parasolid models

A Parasolid model is very precise. All calculations are performed to fixed accuracies called the session precision and session angle precision.

1. Session precision - The linear precision of the modeler. Parasolid session precision is 1.0e-8 units. Distances less than this value are treated as zero and distances that differ by no more than this value are treated as equal.

2. Session angle precision - The smallest angle (in radians) that is treated as different from zero. Parasolid session angle precision is 1.0e-11 units. Angles less than this value are treated as zero and angles that differ by no more than this value are treated as equal.

To handle precision correctly, all parts of a body must lie inside the Parasolid size box. This is a box, 1000 units on each side and centered at the origin, that represents the whole of model space. Typically, the default unit in your application is set to one meter, giving 1 kilometer as the maximum size, in any one direction, which can be modeled. [10]

Chapter 3

Copy Paste Issues and Feature Improvement Guidelines

3.1 Common Issues in Features

3.1.1 Direction Sensitivity

Some features creates outputs that have directions which would affect how the result of children. These sensitive outputs can be sheet bodies, curves or others. Though most features have been already modified to have correct correlation of input and output in terms of directions and also allow user to modify the output directions based on the input directions, there are features that still cannot support direction flipping or does not have persistent correlation to allow users to flip the input direction in paste to get the result that user would like. The following example shows the direction sensitive problem related to the copy pasting of arc feature.

The figure shows a three point arc that is created by taking the two lines as tangent and providing a mid point. When such an arc is copy pasted, while mapping the feature references if the direction of the lines selected as tangents are reversed, the

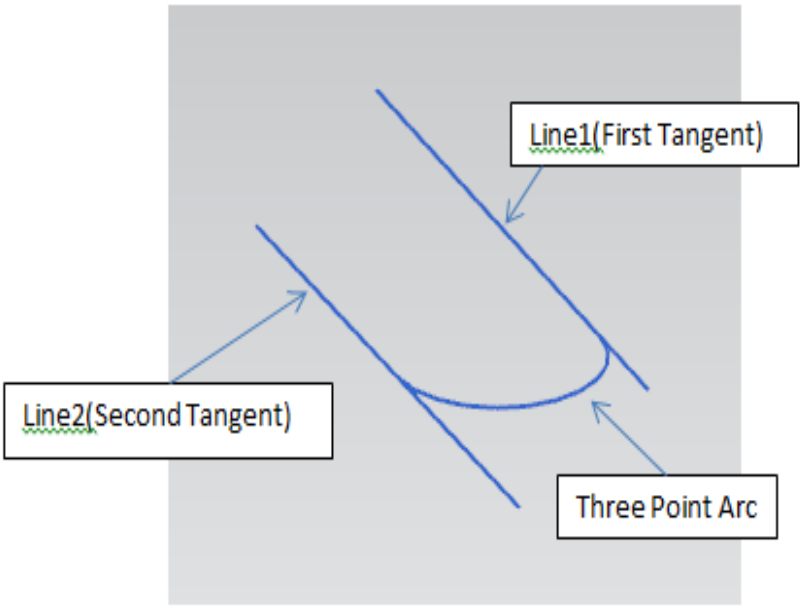

Figure 3.1: Direction Sensitive Problem in Three Point Arc

arc will fail to paste and it will raise an error. Such direction sensitive problem persists with other feature also which needs to be made robust.

3.1.2 No robust input

There are features dependent on some information in the selection such as pick point to decide how their features' outputs should be constructed. They do not have persistent data saved in the feature to track input changes so paste can have random results that user cannot predict. We had this issue in trim sheet before with the "retain" region. Though we do not have the issue for trim sheet, there are other features like hole and intersection where we have to resolve this issue.

3.1.3 Unnecessary errors

The copy and paste feature depends on some methods in features and people are not implementing them robustly and sometime raises errors unnecessarily in their copy method. This will prevent all features to be pasted rather than let this feature to fail to update and still let others to be pasted correctly. This will also give users a change to edit the failed feature later. In such a case, the feature input is still a face while the feature expects a planar face. This type of error should not be raised in the copy paste as there you should not use the geometry yet.

3.1.4 Failure messages not clear

Failure messages in the copy time or update time sometimes are not clear partially due to features do not protect their code to return meaningful error messages when the function they call throw an error. Where user should get a meaningful error message such as "A planar face is expected", but instead, it gets an error of "unable to paste".

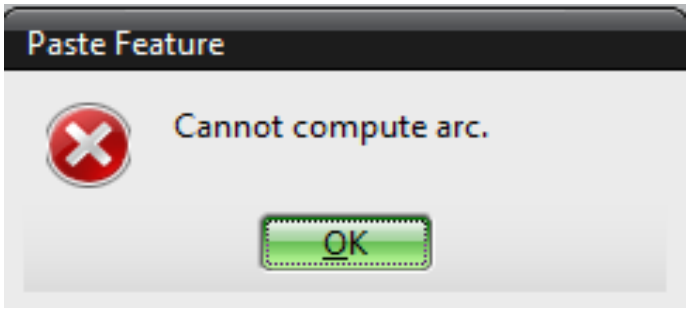

Figure 3.2: Improper Failure Message

The above figure also show the improper error raise on failure of the arc to paste. It simply states that the arc cannot be computed but it doesn't give proper reason as to why the arc is not computed.

3.1.5 Feature reference name is improper

The feature reference name is not correctly returned. This makes the general code hard to attach other information on it. The string returned from feature should never include verbs such as "select face". It should be "draft face" for example. Similar such example is shown in the following figure. It has just return face and the section. It should actually return "Section to Project" and "Face to Project".

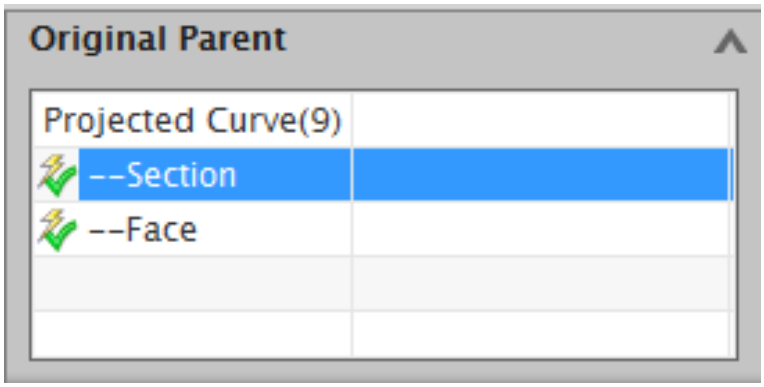

Figure 3.3: Unnecessary error

Another example of improper feature reference name is as shown below.

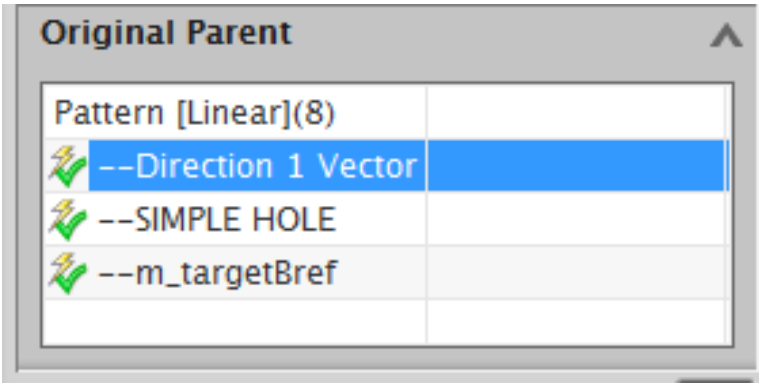

Figure 3.4: Unnecessary error

The figure show another example of the improper feature reference names. Instead of target body, it had incorrectly returned targetbref. So such things needs to be take care of while giving names to the feature references.

3.1.6 Lack of copy and paste testing

The positioning dimensions are used to position a feature relative to the geometry on model. The feature is then associated with that geometry and will maintain those associations whenever the model is edited. The position of a feature can be edited by changing the values of the positioning dimensions.

3.2 Feature Improvement Guidelines

3.2.1 Feature Improvements

1. Any feature that has direction sensitive output that should depends on the direction of input needs to make sure the output direction will follow the input direction persistently.

2. If the feature's output direction can persistently depends on the input direction, the feature should let user to have the option ip the output direction that is the opposite of the input direction. This need to be persistently remembered in the feature so when the input is change, this option will apply on the new input.

3. Features do not have robust input references needs to improve to have robust inputs so when parents are changes to be totally different (do not think about edit, but it can be changed by any other operations), the feature would update as expected.

4. Features have to have the ability to let user edit failed features when the feature references are not alive or are deleted entities.

5. Features need return meaningful error messages.

6. Features should provide short and clear strings related to their referenced entities without attach any verbs.

7. Features need to correctly implement some methods related to copy/paste.

8. Fix any update issues and code quality issues.

3.2.2 Copy and paste framework improvements

1. Reorganize and clean up some large core functions related to copy and paste framework. This will reduce the future cost of maintaining them and make it easier to implement new functionalities on top of them.

2. We should remove the direction handle for indirect parents in the UI as it does not take any effect but only confuse user.

3. We need to address the issues related to split count in the UI and during mapping propagation.

4. We need to fix issue of missing target so we will not get failures due to target is not selected.

5. We should improve the mechanism copy suppressed features to allow the copy and paste with suppressed features to have a high successful rate.

6. We have to ensure the correct map propagation so features copied with their parents will have higher successful rate.

7. Paste should never overwrite the error messages returned by the features as it can mask the real meaning of the error message. If the error does not make sense, let the feature owner to fix it rather than mask it.

8. We have to use examples in geometry instance and pattern features to make sure copy features with slave features created the real geometry is handled correctly.

9. In copy and paste, some objects such as sketch, expressions and measures are using the transfer mechanism to paste. We need to make sure that these objects are copied correctly with the transfer mechanism.

10. Currently, the indirect parent string using a very general string. This can be improved when the feature's reference strings are improved. We may need to do something about it.

Chapter 4

Hole and Line Feature Enhancements

4.1 Simple Hole Feature

4.1.1 Hole create dialogue

The simple hole feature is the design feature used for creating holes in the solid or sheet body. Following is the dialogue of the hole feature in the NX. The dialogue contains different blocks that user need to give as input to successfully create a hole. There are different types of hole as General hole, drill size hole, screw clearance hole, threaded hole, etc.

Though there are many different types of hole that one can create in NX, we are concerned about only Simple Hole. TO create a Simple hole the three general parameters required are the position, direction and expression. Again to give the expression there are four different methods namely value, until selected, until next and through. The functionality of the different expression methods is self explanatory from their names itself. Other blocks of the dialogue are for giving different functionality to the

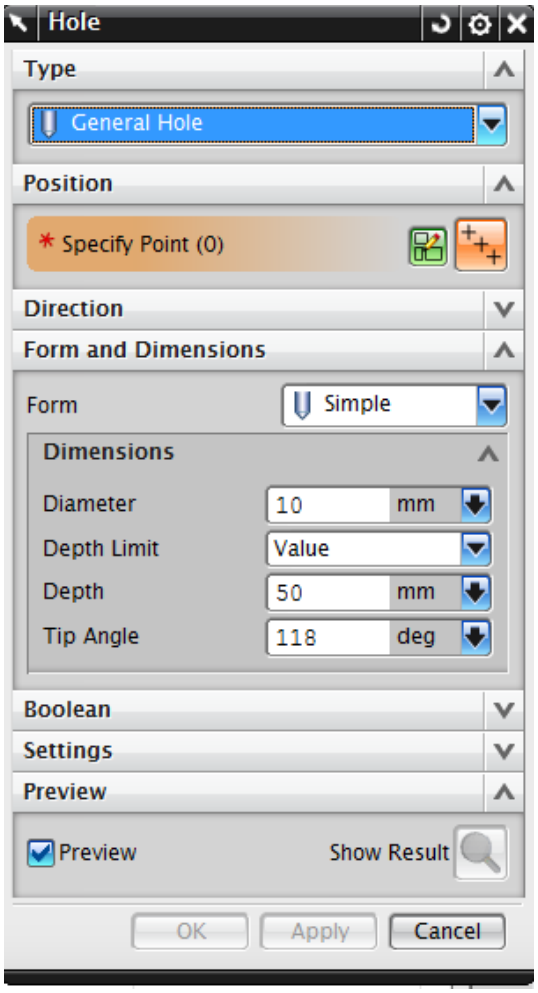

Figure 4.1: Simple Hole Dialogue

hole feature to which we are not concerned as we want to study the copy paste of the hole feature.

4.1.2 Hole Paste dialogue

When user selects the hole feature to copy and then pastes it, the following dialogue pops up. The references described are the required references that the user need to map in order to successfully paste the hole. Section and the target body are the two basic references that user needs to map in case of hole feature.

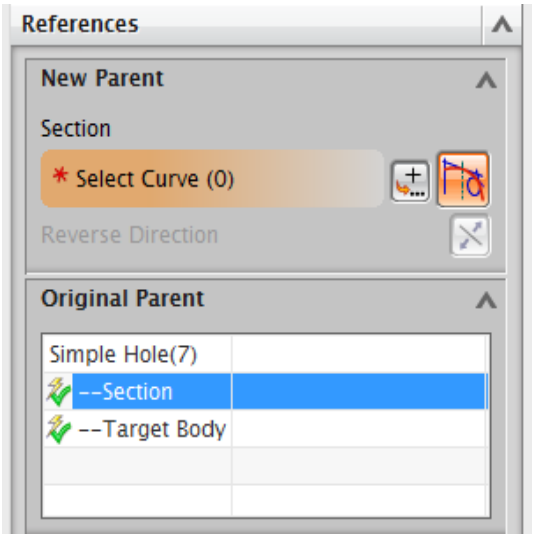

Figure 4.2: Hole Paste Dialogue

While creating the hole if user has given value as expression, then while pasting it will not ask for the value and will only ask for the section and target body which is similar to position. Also it will take the same direction as was taken while creating the hole.

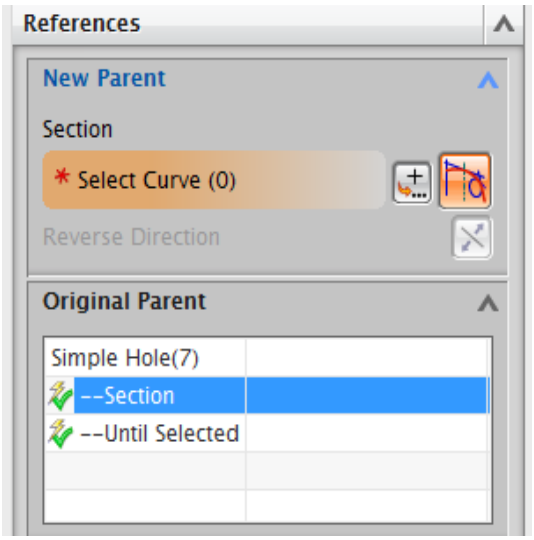

Figure 4.3: Until selected references

If while creating the hole, if the expression was taken as Until Selected, then the paste dialogue will show the feature references as Section and Until Selected to map. It will not ask for the target body and will assume the target body from the face selected for the until selected option. For options like Until next and through, it will again show section and the target body as the references required to map.

4.1.3 Feature Interface methods

Feature interface method for asking UI parents is one method that used by copy and paste to show user what need to be matched when paste features.

Before you fill up the data structure for one of your UI parents, you need to initiate it by calling initialization method so you do not need to set all the fields if you do not need to and the framework can add new attributes in it to enhance it to have more functionality.

The most important attribute in the data structure is the "parent". Feature should return the entity that user selected when the feature is created as UI parent rather than something that the feature generated internally in general.

Collectors and sections can be returned as UI parents but you need to set some properties in the data structure related to the selection options.

Smart object (SO) that created on fly in UI can also be UI parents now. But you should call SO UI parents method for these objects as not all the SO is displayable or selectable. Features should never try to call ask parent method of SO to get SO's UI parents as this method is defined to get update dependencies rather than the parents user selected when the object is created on fly.

The string (name in the data structure) of the UI parents that a feature returns will be displayed in the paste dialog to give user a clue of what it is. The string returned with the UI parent need to be short, clear and need to be a noun as the framework code can attach other string on it. Never add a verb in front of it such as "select face". The scope decides whether the parent in the working part or in the

assembly.

Triple is the one that controls what user can select in the paste dialog for the UI parent. It has limitations and cannot ever be the same as a filter function. But it can save multiple triples depending on what the feature allows for this parent.

Most features using selection triple method to have the default behavior where the scope would be work part if the feature and the UI parent are in the same part or in assembly otherwise. The default triple will only allow user select the same type. For example, if the ui parent is a face, this triple indicates that user can only select a face.

There is other information in the data structure that features can overwrite. We basically treat everything referenced in the parms as ui parents. But features save their outputs there too such as those in the label tables. For all those that save other references in the parms, this method needs to be overwritten.

4.1.4 Issues in UI parent method of Hole Feature

The ui parent method of the hole feature was asking the copy and paste framework to map the entities that are copied. Hence the direct and the indirect feature references in the paste dialogue gets populated from the copy paste frame work. Due to this reason the feature references names were getting incorrectly returned in the paste dialogue as and when user selects hole feature to copy or those feature of which hole is a child feature.

Figure 4.4 shows such a problem. In the following example, a pattern is created using hole as a child feature. Hence hole becomes the child feature of pattern which makes pattern the parent of the hole feature.

Whenever a an interdependent child and parent features are created, the direct and the indirect parents of the parent and child features were asked to map in the references block of the paste dialogue.

Figure 4.4: hole as child of pattern

So when this pattern feature is selected for copy and pasting which has hole as its child feature, following references will get listed in the paste dialogue for mapping. As these references get populated in the paste dialogue from the copy and paste framework, one of the references is get wrongly returned in the dialogue as shown in the figure. Instead of returning target body it mistakenly returned targetbref. Such a problem may occur in case of other features as it is getting returned from the copy and paste framework. So whenever target body is required to map, it will show targetbref instead. Hence this needs to be improved at individual feature level.

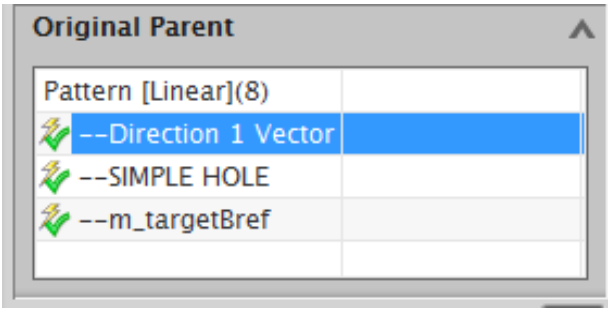

Figure 4.5: Wrong Feature reference name

For this purpose, the whole of hole's ask ui parents method need to be overridden. Earlier in this method the framework method was called which actually does the work of populating the paste dialogue. So the call of that framework in the hole ui parent method has been removed and the code for populating the references in the paste dialogue is implemented in hole method. Hence in this method, two large

code pieces for populating two feature references i.e. "Target Body" and "Until Selected" has been implemented. Also there used to be different kind of hole before the NX5 version. Hence for such kind of Pre-NX5 holes, a separate code needs to be implemented in this method. "Placement Face" was the feature reference name which used to appear in case of pre-NX5 holes. Again the code of the "Section" is not required as this is already implemented in the ui parent method hole package methods. The hole package methods are parent methods of the hole features. Hence it will take the section from those methods and hence no separate code needs to be implemented here. Figure 4.6 shows the paste dialogue feature references after implementing the code in the ui parent method of the hole method.

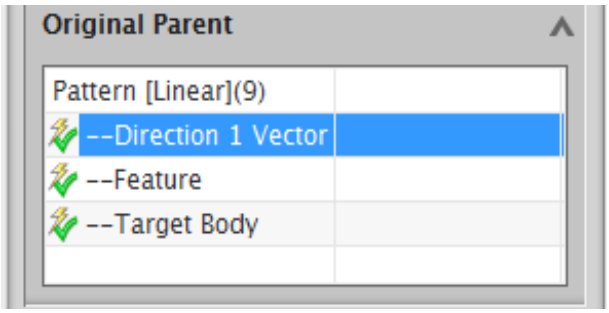

Figure 4.6: Improved Feature reference name

Hence after implementing the code, whenever a hole feature is selected for copy and pasting or any other features is selected of which hole is the child feature, proper feature references for mapping is shown in the paste dialogue.

4.2 Line Feature

4.2.1 Line Create Dialogue

The line feature comes under the curves feature and is used for creating a straight line between two points. This is the most general form of the curve feature and is

the most extensively used feature as any geometric entity has a line as its base entity. Hence, problem in such a soft feature tremendously affects the performance of the software. Hence this needs to be enhanced at the top priority level. Though generally everything has been taken care of for this feature though there are some issues which needs to be improved.

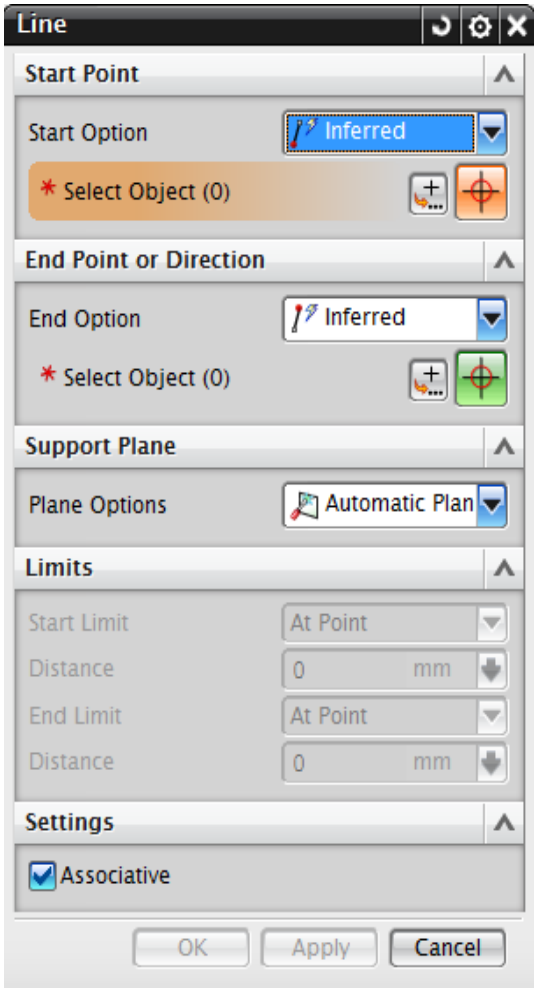

Figure 4.7: Line create Dialogue

Figure 4.7 shows the basic line create dialogue consisting of two basic blocks as start point and end point. The support plane allows user to create the line on a specific plane. Even if the support plane is not selected, the feature takes XY plane as the

default plane and creates the line. There is an option known as associative in the settings block. if this option is turned on, the feature record and the parms of the line feature will get created and every bit of information related to the feature is kept in the memory. but in case it is turned off, then no feature record or parms will get created and the feature will create only a curve and not a single bit of information will get saved in the memory. The non associative line will be shown in the part navigator.

4.2.2 Issues in Line Feature

In case of the line features, the interdependency of the parent and child features were not handled correctly because of which the feature was failing to get properly pasted. Also the feature references names in this feature also was not correctly implemented though there was already the code implemented in its ui parents method. Hence modification in the existing method of the ui parent method of the line needs to be done.

Figure 4.8: time stamp order of line

Figure 4.8 shows the creation of the line feature in the time stamp order by picking two point as the child features. Hence, the line feature consists of two child point features. Here point(3) was taken as the start point and point(4) was taken as the end point. Also the line created here is the associative line as it got listed in the part navigator.

If user selects this line feature for copy and paste it, the paste dialogue should show two feature references for mapping which should be "Start Point" and "End Point" which was kind of obvious as the line was created by selecting two points. And if the user correctly picks two points from the UI, the line will get successfully pasted.

Now if by chance the user deletes one of the point from the part navigator, then the created line feature will get failed as it will fail to find one of its dependent child feature. Such a scenario is shown in figure 4.9. On deleting one of the point say point(4) which was selected as the end point while creating the line, Line(5) will fail to update.

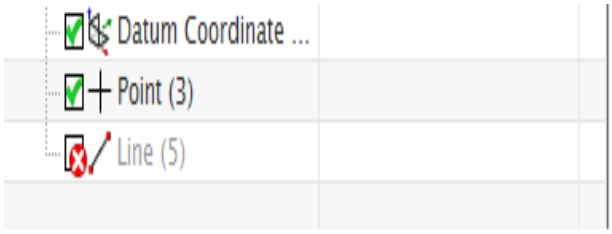

Figure 4.9: Failed Line Feature

Now ideally if user selects such a fail line to paste, the paste dialogue must provide two references for mapping irrespective of whether the line is a fail line or passed line. If such a thing is not implemented then, copy pasting of such a failed line will always create a failed line and user, in any way will not be able to make that line pass. But due to the poor code implementation for the interdependent child feature, the copy pasting of such a failed line shows only one feature reference for mapping as shown in figure 4.10.

Again, such thing needs to be improved only in case of create new option in the paste dialogue, as for the link to original and reuse original options, the line that

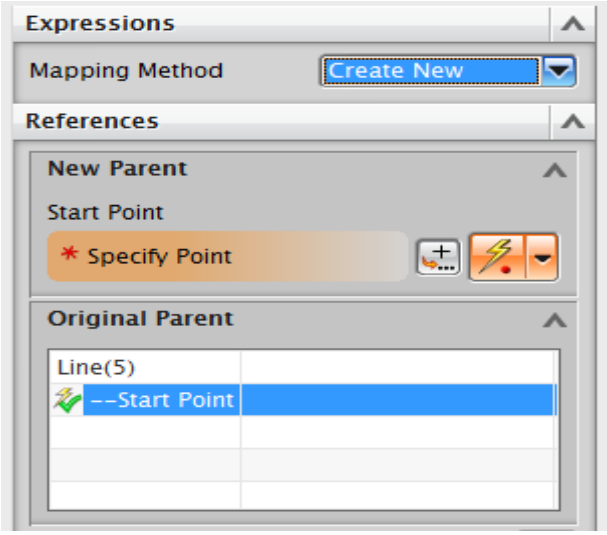

Figure 4.10: missing second reference for mapping

needs to be pasted has to be linked or reused with the created line and so for these two options it should only show one feature references to map and it will create the failed line as it is while creation. Hence while modifying the code, we have to take care of this criteria.

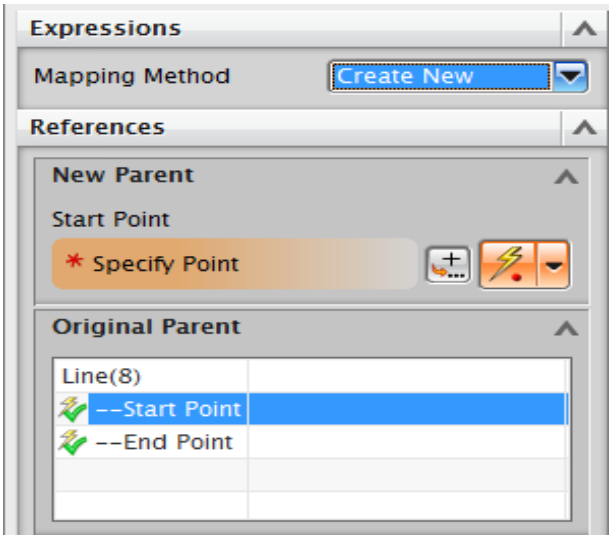

Figure 4.11: required feature references

So the paste dialogue was showing only one reference for mapping was because in the ui parent method, the end point was skipped to show for mapping if the point selected for creating the line was not the SO. Now if the point is deleted, it becomes the dumb point and hence for that point the paste dialogue was not showing that point for mapping. Hence a check has to be put in the ask ui parent method of the line feature that even if the point is not an SO, then also it should get listed in the paste dialogue for mapping. Hence figure 4.11 shows the paste dialogue with two feature references even after selecting the failed line after the implementation of the code.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The copy and paste works with features that are directly or indirectly dependent on one another providing different behaviors to it while pasting was done by a user for a particular feature. Hence issues regarding failure of the paste, improper feature reference name, incorrect error messages, direction sensitivity and no robust output need to be addressed for effective functioning of copy and paste. Hence, each issue is fixed by applying a proper foolproof solution, thereby enhancing the copy and paste feature.

Reorganized and cleaned up some large core functions related to copy and paste framework. This will reduce the future cost of maintaining them and make it easier to implement new functionalities on top of them.

Removed the direction handle for indirect parents in the UI as it does not take any effect but only confuse user.

Fixed issue of missing target so we will not get failures due to target is not selected. Ensured the correct map propagation so features copied with their parents will have higher successful rate.

5.2 Future Scope

In NX, copy and paste has nearly all features using it and providing consistency across these features is a big challenge. Hence, it is liable that some features may have some issues regarding copy and paste which need to be targeted. For e.g. Copy and Paste behavior needs more consistency across other features too apart from the issue for Boolean which has been solved. Another such example is Feature Faces rule that does not work well for inter-part cases as feature is never queried for faces to avoid confusion because a face can have more than one feature. Hence, in future, efforts can be made to make copy and paste more effective and corrective across existing as well as new clients.

References

[1] Mortensen.M.E., Geometric Modeling, New York, Jhon Wiley and Sons, Inc.1985

[2] Shah J.J., Mantyla M. Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications: Wiley-Interscience 2004

[3] Zeid. I. , CAD/CAM, Theory and Practice, McGraw-Hill, New York, topology and basic algorithms, 1991

[4] Bruce Eckel, Thinking in C++, MindView Inc., Volume 1, 2nd Edition Jan 2000

[5] Herbert Schildt , C++: The Complete Reference, Osborne/McGraw-Hill; 3rd edition 1998

[6] Duhwan Mun and Soonhung Han, "Identification of Topological Entities and Naming Mapping for Parametric CAD Model Exchanges", Department of Mechanical Engineering, KAIST, Daejeon, Korea.

[7] Mehdi Baba-Ali and Xavier Skapin, "A method to improve Matching Process by shape characteristics in Parametric system", Computer Aided Designing, Vol 6 Issue 3, 2005

[8] Hilderick A. Van der Meiden and William F. Broneroort, "Tracking topological changes in parametric models",Computer Aided Design, Vol 27 Issue 3, Pages 281-293, March 2010

[9] Tamas Varady, Balaz S. Goal and Graham E.M Tared, "Identifying Features in solid modeling", Computer Aided Design, Vol 14 Issue 1-3 Pages 43-50, May 1990

[10] Jose Encarnaco, Jairo Cote-Munzo, Dieter Eckardt, User Interfaces to Support the Design Process, Computers In Industry, Volume 17, Issue 4, Dec 1991.

[11] Umberto Cugini, The problem of user interface in geometric modeling, Computers in Industry, Volume 17, Issue 4, Dec 1991.

[12] Rest of the Information is taken from the material given during training, and Legacy codes, and