Proceedings of DETC'04
ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference
September 28-October 2, 2004, Salt Lake City, Utah USA

DETC2004/CIE-57703

AN APPROACH TO SUPPORTING SYNCHRONIZED COLLABORATIVE DESIGN WITHIN HETEROGENEOUS CAD SYSTEMS

Min Li
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, 310027, P.R.China
(Phone: +86-571-87951045)
limin@cad.zju.edu.cn

Shuming Gao
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, 310027, P.R.China
(Phone: +86-571-87951045)
smgao@cad.zju.edu.cn

Jie Li
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, 310027, P.R.China
(Phone: +86-571-87951045)
lijie@cad.zju.edu.cn

Youdong Yang
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, 310027, P.R.China
(Phone: +86-571-87951045)
yyoudong@cad.zju.edu.cn

ABSTRACT

This paper presents an approach to building up a synchronized collaborative design platform upon heterogeneous CAD systems. In the approach, Neutral Modeling Commands are used to achieve the real-time exchange of modeling operations between heterogeneous CAD systems. For the Neutral Modeling Commands, an object oriented representation is given and employed to support the translation between System Modeling Operations (SMO) and Neutral Modeling Commands (NMC). Moreover, two kinds of translators, SMO-to-NMC and NMC-to-SMO, are established, which can deal with modification/deletion operations and composite features and user-defined features besides common creation operations. The preliminary test results show that the proposed approach is promising.

KEYWORDS Neutral Modeling Commands (NMC), heterogeneous CAD systems, synchronized collaborative design

INTRODUCTION

Product development paradigm is changing with the increasing globalization of the economy and the rapid development of information technology. Nowadays, more and more complex products need to be collaboratively developed by multiple departments or groups geographically dispersed. It is well recognized that the new product development paradigm requires new CAD approaches and tools which can effectively support collaborative design.

Collaborative design is divided into two categories, synchronized collaborative design and asynchronous collaborative design. As far as the technical issues are concerned, synchronized collaborative design is more difficult to achieve than that of asynchronous one. Therefore, the research on synchronized collaborative design has been paid much more attentions is recent years, and the related research work in increasing. By analyzing the existing work, it is observed that, in all the existing work, synchronized collaborative design is achieved based on homogenous CAD systems. That means all the participants in the collaborative design have to use the same CAD system. Obviously, this is a limitation. As a matter of fact, since the designers in different companies or departments usually use distinct CAD systems and they prefer to work with their familiar CAD systems, the collaborative design tools based on heterogeneous CAD systems are required by more users. On the other side, it is much more difficult to develop a synchronized collaborative design tool based on heterogeneous CAD systems than that based on homogenous CAD systems. This is because the interoperability and communication between homogenous CAD systems is very easy to achieve for the homogenous CAD systems have the same data structure and operation commands, but it is a big challenging problem with heterogeneous CAD systems.

Copyright © 2004 by ASME

In this paper, we present an approach to supporting synchronized collaborative design within heterogeneous CAD systems. The approach attempts to achieve the real-time communication between heterogeneous CAD systems based on neutral modeling commands, and realized synchronized collaborative design by capturing each modeling operation executed in the local CAD system, transferring its converted neutral modeling command to every remote CAD system and execute the corresponding modeling operations there in real-time.

1. RELATED WORK

In last decade, quite a few researches have been conducted in synchronized collaborative design and several prototyping systems have been developed. According to the architecture, current synchronized collaborative design systems can be divided into two types: central and replicated. A central system includes a single or multi-server, and multi-site. The server is responsible for storing the kernel model and managing sessions. Each site only communicates with the server. The advantage of the central system is that the system is easy to achieve data consistency and perform concurrency control. The problem of the central system is that the performance of the system will descend when data interchange between site and server is frequent and the interchanged model is complex. The representative central systems include NetDraw [1], NetFeature [2], WebSPIFF [3], and so on.

For the replicated synchronized collaborative design system, there is no server. All sites are in charge of modeling and communication of themselves. Comparing with the central system, the replicated system has much more powerful modeling functions and quicker responsiveness. However, it is difficult for a replicated system to effectively maintain data consistency and perform concurrency control. The representative replicated systems include CoIIIDE [4], Syco3D [5, 6], TOBACO [7], etc.

Yang-Heon et al. developed a collaborative COCADCAM system that enables two geographically dispersed CAD/CAM users to co-edit CAD geometry dynamically [8]. The contribution of the work lies in successfully proposing and implementing the networking strategies and exploring a new data interchange format to extend a traditional single-location CAD/CAM application to a multi-location application. However, this work is based on homogeneous CAD systems.

Another class of related work is the work about the procedural exchange of parametric feature-based models from one format to another, which exchanges a parametric feature-based model by a sequence of parametric feature modeling operations issued to construct the model.

Guk-Heon Choi et al. [9] proposed a macro-parametric approach to exchanging CAD models. In their approach, parametric feature-based models are exchanged in the form of macro files including the history of a set of standard modeling commands are defined and used in their work [10].

Besides academic research, there are also some feature-based translators being developed by the companies such as ASPire3D [11], Proficiency [12], Theorem [13], TTI [14], and so on. Among them, Collaboration Gateway, the translator developed by Proficiency, is a representative one. According to [15], in the Collaboration Gateway, the Universal Product Representation (UPR) architecture is defined and adopted to provide universal support for all data levels employed by today's CAD systems. Currently, Collaboration Gateway has been able to support four high-end CAD systems including CATIA, I-deas, Pro/ENGINEER and Unigraphics. It should be pointed out that, in all the above work, just off-line exchange of a complete CAD models between two CAD systems is concerned.

2. OVERVIEW OF APPROACH

In this paper, we propose an approach to building up a synchronized collaborative design platform upon heterogeneous CAD systems [16]. The architecture of the platform constructed with the approach is shown in Figure 1.

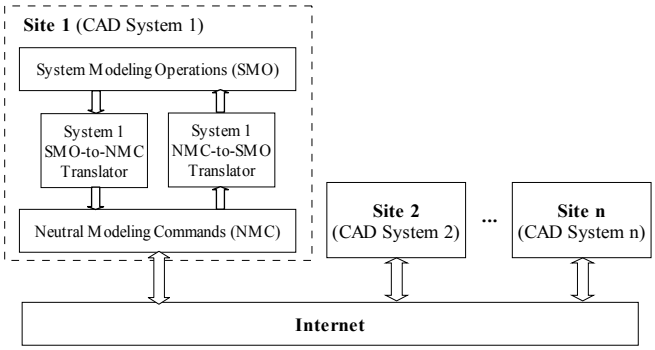


Figure 1: System structure

It is a replicated architecture with each site having a distinct CAD system performs product modeling. In each site, besides an independent CAD system, there are two translators. One is responsible for translating each SMO just carried out locally into a NMC that will be sent to other sites immediately, which is called SMO-to-NMC translator. Another translator called NMC-to-SMO translator, is in charge of translating each received NMC from other site into one or more corresponding SMOs of the CAD system. It is these two translators in each site that enable the real-time exchange of the modeling operations between heterogeneous CAD systems and make the platform able to support synchronized collaborative design. During the collaborative design, every user performed SMO is immediately translated into a NMC sent to other sites, and as soon as one NMC arrives, it is translated into corresponding SMOs to execute. Therefore, each CAD system only interacts with NMCs, independent of the CAD system on other sites. According to our current experiment, the implementation work of this architecture is almost linear in the number of CAD systems supported.

3. CONSTRUCTION PRINCIPLES OF NEUTRAL MODELING COMMANDS

NMC set plays an important role in achieving real-time communication within heterogeneous CAD systems. To guarantee the rationality and validity of the constructed NMC set, we construct the NMC set following the two principles described below.

Copyright © 2004 by ASME

3.1. Based On Parametric Feature Modeling Operations

Parametric feature modeling, as one of the most advanced ways for product modeling, can effectively support product modeling with parametric features, adapt for design practice, and support variational design and intelligent design. Accordingly, it is the most used product modeling technique in all of current commercial CAD systems. In the view that collaborative design should support parametric feature modeling, our NMC is constructed based on parametric feature modeling operations. That means, almost every NMC corresponds to some parametric feature modeling operations.

3.2. Uniting All Kinds of Parametric Feature Modeling Operations

We observe that the essential modeling operations provided by all commercial CAD systems are similar though their parameters of the corresponding operations may be different. To ensure that every SMO can be translated into a NMC easily, it is desirable to make NMC correspond to the union of parametric feature modeling operations of CAD systems (Figure 2). Similarly, the parameters of each NMC take the union of those of all corresponding operations (Figure 3). Taking extrusion as an example, there is a bi-extrusion attribute in Solidworks and UG while MDT only supports single-extrusion, so the parameters of NMC Extrusion will include bi-extrusion parameter. According to our principle, NMC will not only be applicable for all SMOs of all CAD systems, but also unify corresponding modeling operations that have functional specialty and semantic similarity.

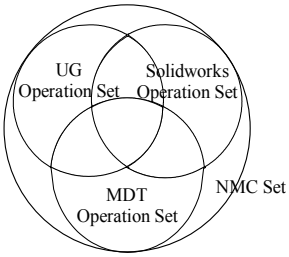


Figure 2:Union of operations

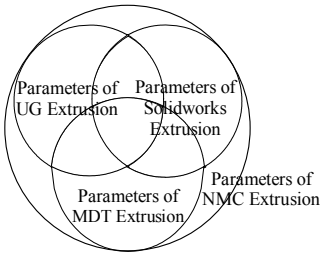


Figure 3:Union of operation parameters

4. REPRESENTATION OF NEUTRAL MODELING COMMANDS

To effectively support the realization of two translators, SMO-to-NMC and NMC-to-SMO, we represent NMCs in an object oriented way. Each NMC is represented as a class and can be instantiated to an object with some functions during collaborative design. In addition, each NMC has a string representation used for its transfer in network which consists of the name and all its attributes.

4.1. Class Diagram of Neutral Modeling Commands

The general UML [17] class diagram of the object oriented representation of NMCs is shown in Figure 4.

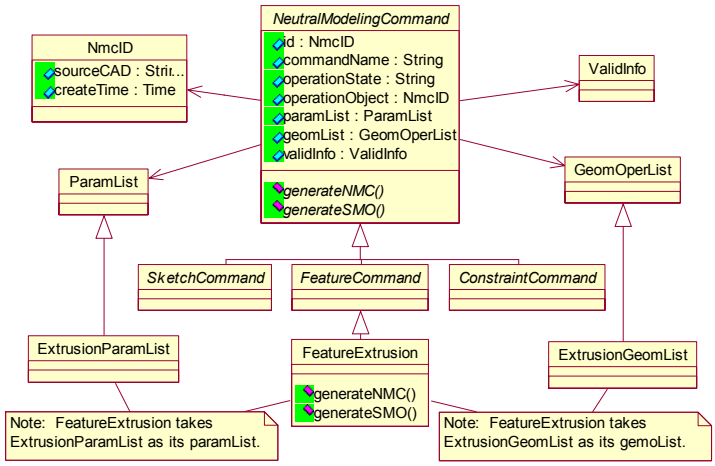


Figure 4: General UML class diagram of Neutral Modeling Commands

The class NeutralModelingCommand is the parent of all NMC classes. The first attribute id is an identifier that is composed of the local CAD system's name and the created time of the NMC. The commandName is the NMC's name and the operationState indicates one of three states: "Creation", "Modification" and "Deletion". For example, a modified extrusion operation for is translated into a NMC object whose commandName set to "FeatureExtrusion" and operationState set to "Modification". If operationState is "Modification" or "Deletion", operationObject is used to record the existing modified or deleted object, which is imperative for exchanging. The way to find out the existing object will be discussed in section 5.

Attribute paramList represents the parameters of NMC. For every class derived from NeutralModelingCommand, class ParamList provides a subclass to express its parameters. As shown in left-bottom of Figure 4, class FeatureExtrusion takes class ExtrusionParamList as its actual paramList. The corresponding geometric operations are also attached in the NMC class, which are useful only if there is no match between the NMC and any SMO in the remote site it is sent to. We use geomList to contain such geometric operations. Similar to ParamList's family, every NMC class takes a corresponding subclass of class GeomOperList to represent its specific geometric operations. The attribute validInfo is used to carry the validation information of the NMC.

There are two translation functions in every NMC class: generateNMC() and generateSMO(). The generateNMC() is used to translate a SMO into a NMC, whose input is a local operation and output is a NMC with corresponding parameters and geometric operations. The generateSMO() is in charge of converting a received NMC to one or more SMOs. In this work, these two functions of each NMC class are designed respectively so that they can work well. Also the design of them is tightly dependent on the CAD system which the NMC class is deployed.

Here we give an example of the parameters class of the NMC Extrusion. The parameter set of the class is the union of all the extrusion operations' parameters of three popular CAD systems, including MDT (Mechanical Desktop 6), SolidWorks

Copyright © 2004 by ASME

(SolidWorks 2003) and UG (Unigraphics V18.0). The primary UML class diagram of it is shown in Figure 5.

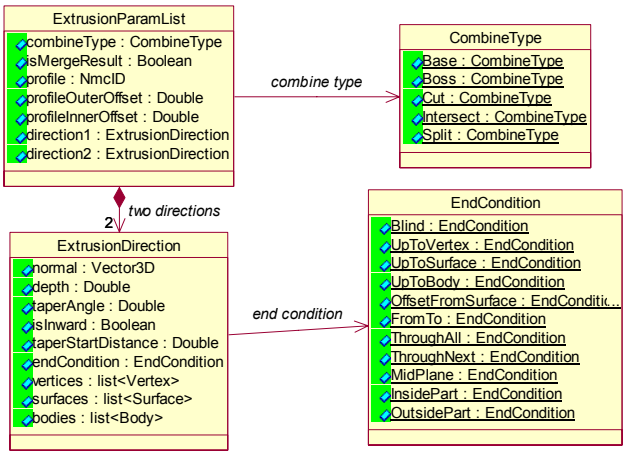


Figure 5: Primary UML class diagram of class ExtrusionParamList

Class ExtrusionParamList is the parameters class for FeatureExtrusion. Attribute profileOuterOffset and profileInnerOffset are particular parameters from UG. They are used to create extrusion features that are offset by a constant value from the profile. The direction1 and direction2 refer to two directions of a bi-extrusion feature. The taperStartDistance that indicates where the taper begins comes from UG too. Enumeration class EndCondition defines end conditions from three CAD systems. Among them, InsidePart and OutsidePart are MDT proprietary.

4.2. Expression of Topological Entity

For certain neutral modeling commands, their parameters include topological entities. For example, NMC Fillet needs one or more edges as its parameters. In homogeneous systems, transferred entities can be identified by their IDs since the same entity has the same ID. But it won't work for the heterogeneous systems because the IDs of the same topological entity in different CAD systems are usually different. To solve this problem, we directly use entity's type and its geometric information in the World Coordinates System (WCS) to express the topological entity in neutral commands. For instance, a linear edge is expressed by its type and midpoint. Similarly, a planar face is expressed by its type, normal and a point inside the face.

The feasibility of this method depends on whether the geometric information of the same topological entity in different CAD systems is equal or not. Fortunately, it is observed that the analyzed CAD systems' WCSs are all Cartesian coordinate systems with right hand rule, and never be changed in design process. Moreover, User Coordinate System (UCS) can be freely transformed to WCS and vice versa. In one word, no matter what UCS used, the geometric information of the same topological entity in WCSs of different CAD systems is always the same.

To match a curve or surface type at a CAD system which is not available at another system, the NMC also take the NURBS representation as auxiliary information to deal the incompatibilities. For the freeform curves or surfaces involved in the NMCs, their corresponding NURBS representations are also embedded in the NMCs.

5. TRANSLATION BETWEEN SYSTEM MODELING OPERATIONS AND NEUTRAL MODELING COMMANDS

In our approach, to effectively support synchronized collaborative design, we propose a NMC-based method for real-time exchanging modeling operations within heterogeneous CAD systems. The key issue here is how to effectively and efficiently make translation between SMOs and NMCs. In the following, we introduce the real-time translation from a SMO to a NMC and the real-time translation from a NMC to SMOs respectively.

5.1. Real-Time Translation from a SMO to a NMC

Figure 6 illustrates the process of the real-time translation from a SMO to a NMC. As soon as a user performs a parametric feature modeling operation locally, the operation is captured by SMO-to-NMC translator and matched with every NMC template in the local NMC library to find the corresponding one. Then a NMC object is created by the matched NMC template and its generateNMC() is invoked with the captured operation as its input to obtain all its parameters, the geometric operations and the validation information. At last, the SMO-to-NMC translator invokes the string serialization method to create a string representation of this NMC, which is sent to others sites immediately.

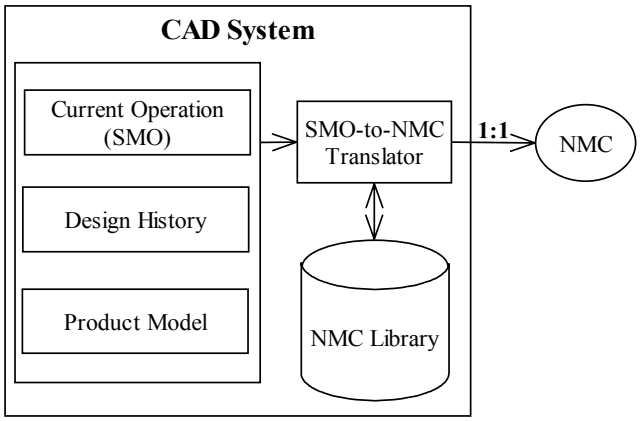


Figure 6: Translation from a SMO to a NMC

5.1.1. Design of generateNMC().

Function generateNMC() plays a key role in the process of translating a SMO into a NMC, which is responsible for generating all the parameters of the current NMC object as well as its geometric operations and validation information. Its main tasks are listed as follows:

1) Generating of all the parameters of the NMC object. For a SMO, its corresponding NMC's parameters consist of two parts, direct parameters and indirect parameters. Here direct parameters refer to those that are also the parameters of the SMO and hence can be obtained from the SMO directly. In contrary to the direct parameters, the

Copyright © 2004 by ASME

indirect parameters are calculated based on the product model.

2) Generating of the geometric operations corresponding to the NMC. The key step of generating the geometric operations is the generation of its parameters. Similarly, the parameters are also got from the SMO or calculated based on the product model.
3) Generating of the validation information of the NMC. In this work, the mass properties of the solid model updated by the performed operation are taken as the validation information of the NMC object.

The generateNMC() of each NMC is designed respectively. In addition, the design of generateNMC() of a NMC is dependent on the CAD system because the SMOs corresponding to the NMC of different CAD systems usually are not exactly the same.

To make the above method clearer, we show an example, the designing the generateNMC() of NMC Extrusion for MDT. For the single-extrusion operation with McadToFaceTerminator that means it is extruded to a surface, the generateNMC() fetches all direct parameters of extrusion from MDT system. As an indirect parameter, the depth is calculated with the distance between the selected face and the sketch plane. Meanwhile, some indirect parameters which make no sense to MDT, such as direction2 or profileOuterOffset, are set to null or zero. The geometric operations need not to be generated here since extrusion operation is common to all CAD systems. In the end the mass properties of the model are gained.

5.1.2. Handling of Modification and Deletion Operations.

Differing from data exchange systems, the synchronized collaborative design system must exchange modification and deletion operations in real time. Also, modification and deletion operations are different from creation operations in that they should find which object are modified or deleted before they are executed. Therefore, in our approach, every NMC has three states including creation, modification and deletion so as to distinguish different kinds of operations, and attribute operationObject is used to indicate the object to be operated.

To be able to effectively handle modification and deletion operations, we introduce an association mechanism from a NMC object to its corresponding SMO operated object. It is obvious that the NMCs sent by all CAD systems, which are sorted by time, form the design history of collaborative design. We call this sorted NMC list as NMC-based design history. Considering that only commands are transported and the transmission is supposed to be real-time, hence a NMC is always received instantly after it is sent. And all the NMCs sent and received in a site sorted by time are exactly the NMC-based design history. That also means among the NMC lists in all sites, the NMC with the same ID is identical. Based on the above observations in our approach, every CAD system sets up a NMC list, as shown in Figure 7. The pair of one NMC (whatever it is outcoming or incoming) and the operated object pointer of its corresponding SMO are stored into the NMC list as an item. Using the pair of the list item, one NMC and its corresponding SMO's operated object are associated each other.

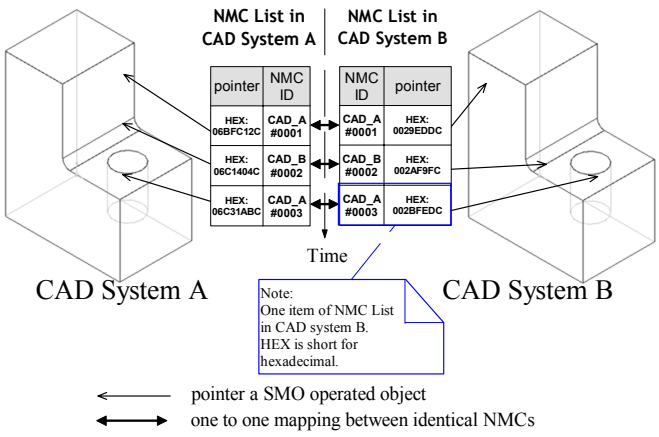


Figure 7: Association mechanism from a NMC to SMO operated object

With the help of NMC list, we can use the operated object of current SMO to find out the corresponding objects on the other sites, which should be operated synchronously. The process is described as follows:

1) In the local CAD system, by traversing the local NMC list from top to down, and matching current operated object pointer and the pointer stored in every item of the NMC list, we find out the item that contains the current operated object pointer.
2) And using the pair of the found item, we determine the associated NMC's ID of current operated object.
3) On a remote site, using the determined NMC's ID, we uniquely find out the corresponding item in the NMC list in that CAD system.
4) Using the pair of the found item in the remote site, we finally gain the pointer of the corresponding object that should be operated synchronously.

With the above-mentioned mechanism, we deal with modification operations as follows. First, the SMO-to-NMC translator captures a modification operation and finds out its operated object - a modified object. According to the type of found object, a corresponding NMC template is determined from the local NMC library by matching. Then using the matched template, a new NMC instance is created with its operationState set to "Modification" and its generateNMC() function is invoked. Since the memory location (namely pointer) of the modified object is usually changeless after it was created, generateNMC() can use the pointer of the modified object to look for the NMC which created it, in the local NMC list. The ID of the found NMC creation is assigned to attribute operationObject. The consequent steps such as generation of parameters and validation information are the same as those of a creation operation. Regarding deletion operation, its handling is similar to modification operation but the generation of parameters is not required.

5.1.3. Handling of Composite Features and User-Defined Features.

Composite features and user-defined features (UDF) need to be also specially handled. Composite feature is a complex

Copyright © 2004 by ASME

feature that is composed of several basic features, such as composite hole. As for UDF, it is a special kind of feature that is used to represent the form defined by user. Since the modeling operations related to composite feature and UDF are usually monopolistic modeling operations of CAD systems, to facilitate the translation of such operations, some auxiliary information is required.

To translate a composite feature operation, firstly we extract all the parameters of the composite feature itself in common with other SMOs. After that it is decomposed into basic features which are translated to their corresponding NMC objects later. The NMC objects related to basic features are stored in the NMC object for composite feature as the auxiliary information. As shown in Figure 8, we add an abstract NMC class called CompositedFeature to represent the parent of all composite features. Every concrete composite feature is represented by a subclass of class CompositedFeature. Finally those NMC objects corresponding to basic features are stored into a list named decomposedNmc.

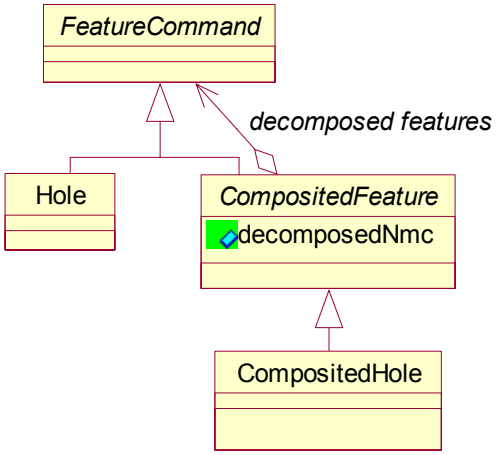


Figure 8: Primary class diagram of CompositedFeature

Similarly the UDF is also made up of several basic features. And some parameters of these basic features can be defined by users when the UDF is created, which are called driving parameters. The generateNMC() of UDF is designed as follows. First, following the name of current UDF operation, the UDF definition in local UDF library is found. Second, the current UDF is decomposed to several basic features based on the found UDF definition. And these basic features are translated into corresponding NMC objects as auxiliary information. Third, we find out all the driving parameters as auxiliary information too. Up to now, we gather enough useful information of UDF that will ensure correct restoration in the other sites. Similar to handling of composite feature, the UDF is represented by a class like CompositedFeature described above. The NMC objects related to the decomposed basic features are stored in a list similar to decomposedNmc. Also, the collection of the driving parameters is stored into another list.

5.2. Real-Time Translation from a NMC to SMOs

The following Figure 9 shows the process of real-time translation from NMC to SMO. As soon as a NMC string is received from the network, the local NMC-to-SMO translator find the corresponding NMC template in local NMC library by using command name of the received NMC. Note that for the same CAD system, the NMC library used in the SMO-to-NMC translator is the same as the one used in the NMC-to-SMO translator. Then a NMC object is created by the found NMC template and its generateSMO() is invoked with received NMC string as its input to generate one or more SMOs. After the generated SMOs are completed by local CAD systems, the verification of the product model is executed.

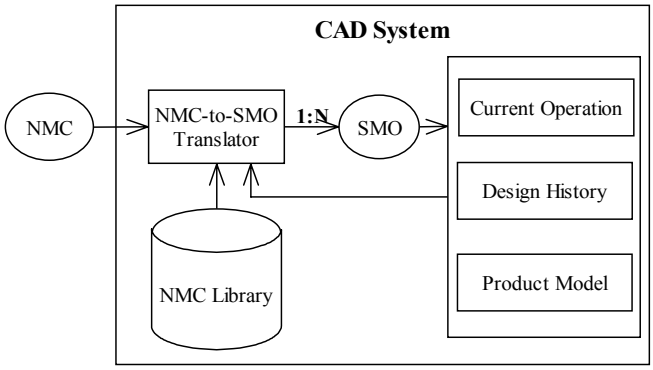


Figure 9: Translation from a NMC to SMOs

5.2.1. Design of generateSMO().

Function generateSMO() is responsible for translating one NMC to one or more SMOs according to the received NMC string. Its main tasks are listed as follows:

1) Parsing received NMC string and assigning parameters. Since the NMC string consists of NMC object attributes, all the parameters and validation information can be accessed from the parsed NMC string
2) Translating the NMC into one or a group of SMOs called SMO group hereafter. Having all its parameters, the NMC is translated into one or more local system modeling operations to execute.
3) Verifying the product model. After the SMO group is executed, some mass properties such as center of gravity and surface area are taken to verify the product model. If the difference of mass properties between local product model and validation information is beyond a given tolerance, the exchange of the current modeling operation is invalid and it should be noticed to users.

Similar to generateNMC(), the programmer should design the generateSMO() for each NMC respectively. And the design of generateSMO() is also dependent on the CAD systems.

When the received NMC does not match any SMO of the current CAD system, to enable the translation of the NMC, it is translated into several SMOs or its geometric operations are translated. As an example, for the received NMC Extrusion whose bi-extrusion attribute is true, the generateSMO() of the NMC Extrusion designed for MDT will translate it into two

Copyright © 2004 by ASME

single-extrusion operations based on the same sketch profile, due to the fact that there is no bi-extrusion operation in MDT.

When this one-more scenario occurs, it is noteworthy that the SMO-NMC translating for the modification or deletion on a portion of the operated objects of a SMO group is quite different to that on the operated object of a single SMO. Toward any modification or deletion on a portion of the operated objects of a SMO group, we treated it as the modification on the NMC corresponding to the whole SMO group. For instance, after the MDT system receives a NMC Extrusion that is a bi-extrusion from remote site, and translates it into two single-extrusions locally, when the designer on MDT deletes one of the two single-extrusions, the SMO-NMC translation generates a NMC modification for the bi-extrusion instead of a NMC deletion for the single-extrusion.

5.2.2. Handling of the NMC for Modification and Deletion.

If the NMC is a command for modification or deletion with operationState being "Modification" or "Deletion", the generateSMO() corresponding to it is different from others in that the local object to be modified or deleted should be found. The generateSMO() firstly finds out the previous NMC that created the required local object in the local NMC list with the help of the NMC's ID in the attribute operationObject. Following the associated object pointer of the found NMC creation, an operated object of creation SMO is found, namely the local object to be modified or deleted later. Then for the generateSMO() of the NMC for modification, it modifies the parameters of found object with the received parameters for modification, while the generateSMO() of NMC deletion removes the found local object directly. Other steps of them are similar to those of NMC creation.

5.2.3. Handling of the NMC for Composite Features and User-Defined Features.

Due to the specialty of composite feature and user-defined feature commands, the generateSMO() corresponding to them are also different from others. For a concrete composite feature command from a remote site, if there is the same composite feature operation in the local CAD system, it is translated into the corresponding operation directly. Otherwise, it is translated into a groups of SMOs based on the auxiliary information about the decomposed basic feature operations, which are contained in its decomposedNmc.

For a user-defined feature command, it is translated into a corresponding UDF operation of the local CAD system as follows: Firstly, according to the name, basic features and a list of driving parameters of the UDF, which is contained in the auxiliary information of the NMC command, a same UDF definition is created in the local UDF library. Secondly, an UDF object is made in the local product model with the UDF definition created just now. Thirdly, the same UDF is built after filling the driving parameters of the new created UDF object with the default values stored in auxiliary information. For the case that the local CAD system does not support UDF at all, similar to the handling of composite features, the received UDF will be translated into a group of SMOs according to the basic feature operations contained in decomposedNmc.

6. IMPLEMENTATION

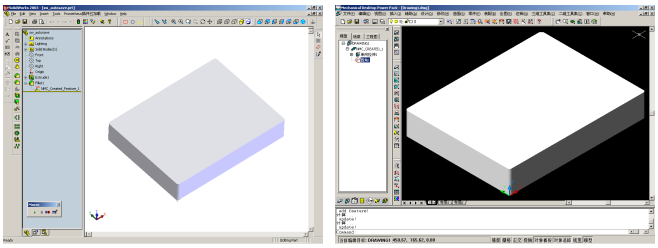
We have developed a preliminary prototype system of synchronized collaborative design based on SolidWorks 2003 and MDT6.0. For each of the two CAD systems, both SMO-to-NMC and NMC-to-SMO translators are implemented with Visual C++ 6.0 and the open programming APIs of the system. The translators are compiled into the plug-in of native CAD system, running as background process after system starts to work. The communication between CAD systems is achieved by Transmission Control Protocol (TCP), which can provide the reliable end-to-end data transferring across WAN.

In the developed prototype system, two geographically dispersed users, using Solidworks and MDT respectively, successfully completed the collaborative design of the test part shown in Figure 12. The detailed process of the collaborative design is described as follows.

The designer using Solidworks firstly creates a base feature. Then the local SMO-to-NMC translator is immediately triggered by modelChanged event and the information of the created extrusion feature inside the local CAD system is obtained, based on which the NMC template corresponding to the extrusion feature is found out in the NMC library by matching. An instance of the matched NMC template takes current extrusion object as input to generate all the parameters and validation information. Finally the NMC string generated by the NMC instance is delivered to other sites.

On the other site, the NMC-to-SMO translator in MDT system parses the NMC string as soon as it receives valid package from network. After generating a NMC object with the matched NMC template, MDT system executes an extrusion operation whose parameters are directly obtained from the NMC object.

Same as the exchange of base extrusion, after the designer on MDT then created one fillet feature to round the corners; the SMO-to-NMC translator inside MDT system translates it into one corresponding NMC, and delivers the NMC string to other sites. The views of the two systems are shown as Figure 10.

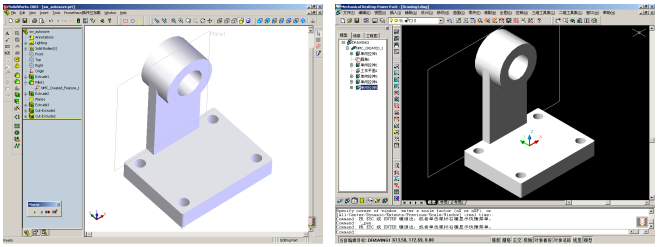


a) Solidworks 2003 View b) MDT 6.0 View

Figure 10: The views of the two systems after one base extrusion and one fillet feature are created

After that, the designer on Solidworks creates two boss extrusion features and the designer using MDT creates two cut extrusion features as holes. After all the sites complete the build of these operations, the views of the two systems are shown as Figure 11.

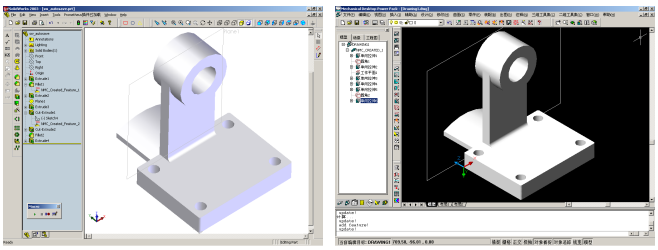
Copyright © 2004 by ASME



a) Solidworks 2003 View b) MDT 6.0 View

Figure 11: The views of the two systems after two boss extrusions and two cut extrusions are created

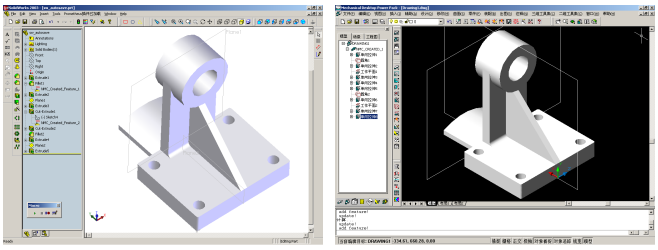
Then designer using Solidworks creates one fillet features to smooth a concave edge and the designer using MDT creates one boss features simultaneously. Figure 12 shows the results of two systems after the synchronized operations are completed.



a) Solidworks 2003 View b) MDT 6.0 View

Figure 12: The views of the two systems after one fillet feature and one extrusion feature are created.

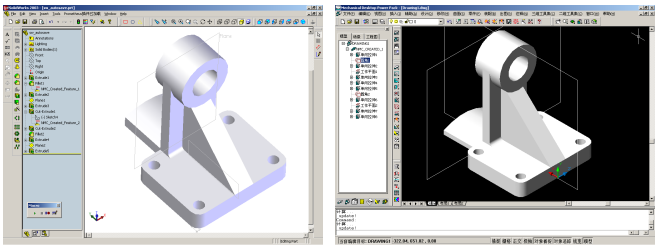
After that the designer on Solidworks creates a bi-extrusion feature as rib. When the NMC for bi-extrusion is received by MDT system, it is translated into two single-direction extrusion operations to be executed by local NMC-to-SMO translator since there is no bi-extrusion operation in MDT. Figure 13 shows the views of two systems after these boss features are built. In the feature tree of MDT view, there are two continuous single-direction extrusion features are created to represent one bi-extrusion feature in Solidworks.



a) Solidworks 2003 View b) MDT 6.0 View

Figure 13: The views of the two systems after the bi-extrusion feature is created.

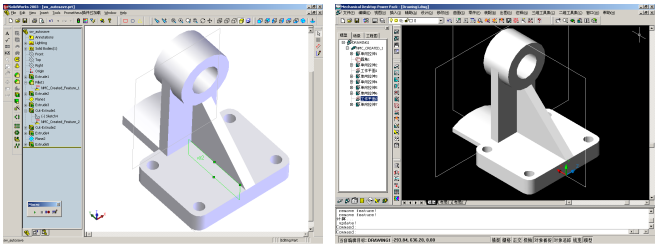
Then the designer using MDT modifies the radius of the fillet feature at corner and the designer on Solidworks makes the bi-extrusion widened in both directions. Figure 14 shows the results of two systems after the synchronized modification are completed.



a) Solidworks 2003 View b) MDT 6.0 View

Figure 14: The views of the two systems after the radius of fillet and the depth of bi-extrusion are modified.

In the last step, the designer using Solidworks deletes the concave edge fillet. At the same time, the designer on MDT deletes half portion of the bi-extrusion and leaves it as a single-direction extrusion. Figure 15 shows the views of two systems after these deletion operations are completed.



a) Solidworks 2003 View b) MDT 6.0 View

Figure 15: The views of the two systems after the fillet feature and the half of bi-extrusion are deleted.

7. CONCLUSION AND FUTURE WORK

In this paper, we present an approach to building up a synchronized collaborative design platform upon heterogeneous CAD systems. The characteristics of the approach include following four aspects:

1) The real-time exchange of the modeling operations between heterogeneous CAD systems is achieved based on Neutral Modeling Commands.
2) An object oriented representation of Neutral Modeling Commands is proposed, which can effectively support the translation between System Modeling Operations and Neutral Modeling Commands.
3) The translation method is able to achieve 1:N mapping between System Modeling Operations and Neutral Modeling Commands. Meanwhile it is capable of dealing with modification/deletion operations as well as composite features and user-defined features.
4) The approach can also be used to realize feature based product data exchange between commercial CAD systems.

Our future work will focus on following aspects. The first is to extend the prototype system to include more CAD systems such as UG, CATIA, Pro/E, etc. The second is to enabling the

Copyright © 2004 by ASME

handling of composite features and user-defined features in our system. The third is to develop a more comprehensive method to verify models.

ACKNOWLEDGMENTS

The authors are very grateful to the financial support from NSF of China (No.60273057 and No. 60021201) and the Trans-Century Training Programme Foundation for Talents by the Education Ministry of China.

REFERENCES

[1] Qian, D. and Gross, M. D. “Collaborative Design with NetDraw,” In Proceedings of Computer Aided Architectural Design Futures. 1999. 213--226.

[2] Lee J Y, Kim H, Han S B, Park S B. “Network-Centric Feature-Based Modeling,” Proceedings of Pacific Graphics '99, Kim, M.-S. and Seidel, H.-P. (Eds.), IEEE Computer Society, CA, pp. 280-289.

[3] Bidarra R, van den Berg E, Bronsvoort W F. “Collaborative modeling with features,” In: CD-ROM Proceedings of 2001 ASME Design Engineering Technical Conferences, Pittsburgh, Pennsylvania, DETC2001/CIE-21286.

[4] Van den Berg E, “Web-based collaborative modeling with SPIFF,” Ms Thesis, Delft University of Technology, September 2000, Delft.

[5] Nam TJ, Wright DK, 2001, “The Development and Evaluation of Syco3D: A Real-Time Collaborative 3D CAD System,” Design Studies, vol. 22, pp. 557-82.

[6] Nam TJ, Wright DK [1998] “CollIDE: A Shared 3D Workspace for CAD,” Paper for the 1998 Conference on Network Entities, October 1998, Leeds.

[7] Dietrich U, Von Lukas U, Morche I. “Cooperative modeling with TOBACO, Proceedings of the TeamCAD97 Workshop on Collaborative Design,” May 1997, Atlanta, pp. 115-122.

[8] Yung-Chou Kao, Grier C. I. Lin. “Development of a collaborative CAD/CAM system,” Robotics and Computer-Integrated Manufacturing 14(1998),55-68.

[9] Guk-Heon Choi, Duhwan Mun, Soonhung Han. “Exchange of CAD part models based on the macro-parametric approach,” International Journal of CAD/CAM 2(2002),23-31.

[10] Duhwan Mun, Soonhung Han, Junhwan Kim, Youchon Oh. “A set of standard modeling commands for the history-based parametric approach,” Computer-Aided Design 35 (2003) 1171-1179

[11] ASPire3D. http://www.aspire3d.com/

[12] Proficiency. http://www.proficiency.com/

[13] Theorem. http://www.theorem.co.uk/

[14] Translation Technologies Inc. http://www.translationtech.com/

[15] Ari Rappoport. “An Architecture for Universal CAD Data Exchange,” Proceedings, Solid Modeling '03, June 2003, Seattle, Washington, ACM Press.

[16] Min Li, Youdong Yang, Jie Li, Shuming Gao. “A preliminary study on synchronized collaborative design based on heterogeneous CAD systems," 8th International Conference on Computer Supported Cooperative Work in Design.

[17] UML Resource Page. http://www.omg.org/uml/

Copyright © 2004 by ASME