**Min Li**
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, 310027, P.R.China
(Current address:
Department of Mechanical Engineering
National University of Singapore
10 Kent Ridge Crescent,
Singapore 119260
e-mail: limin@nus.edu.sg)

**Shuming Gao**
**(Corresponding author)**
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, 310027, P.R.China
e-mail: smgao@cad.zju.edu.cn

**Charlie C. L. Wang**
Department of Automation and
Computer-Aided Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong, P.R.China
e-mail: cwang@acae.cuhk.edu.hk

# Real-Time Collaborative Design with Heterogeneous CAD Systems Based on Neutral Modeling Commands

*This paper presents an integration-based solution for developing a real-time collaborative design (co-design) platform on heterogeneous computer-aided design (CAD) systems. Different from the visualization-based approaches, the product models are allowed to be constructed and be modified from various sites together in the proposed collaborative design platform. Our approach is based on a mechanism for the translation between system modeling operations (SMO) and neutral modeling commands (NMC). Every operation given by a user on one site is translated into an NMC and transmitted to all the other sites through network, and then the received NMC is converted into corresponding SMOs on every other site instantaneously. Since only the commands but not the product data are transferred, the data size under transmission is greatly reduced, so that a real-time synchronization can be achieved with a standard network bandwidth. In addition, by developing system-dependent SMO↔NMC translators on different client CAD systems, users on different sites could join the collaboration by using their familiar CAD systems; this is the benefit that cannot be offered by the homogeneous co-design systems. The prototype implementation proves that our approach works well for integrating various current popular commercial CAD systems into a real-time collaborative design platform.*

*Keywords: command-based, real-time synchronization, collaborative design, CAD systems, heterogeneous structure, feature-based modeling, interoperability*

## 1 Introduction

The paradigm of product development is changing with the increasing globalization of the economy and the rapid development of information technology. In recent years, more and more complex products need to be collaboratively developed by multiple departments or groups geographically dispersed. It is well recognized that this new product development paradigm requires new computer-aided design (CAD) approaches and tools which effectively support collaborative design activities. For example of the enterprises in Hong Kong, the customers are mainly from US and Europe, the design centers are usually located at their headquarters in Hong Kong, and most of them have their manufacturing facilities in mainland China. Therefore there is a growing demand to enable collaborative product development linking the overseas customers, the Hong Kong headquarters, and the manufacturing plants. The Internet is an ideal platform to articulate such development. However, general CAD software cannot support the requirement of an instantaneous collaborative design task, especially in the sense of *instantaneous* and *collaborative design*.

In current CAD systems, the design behavior of parts, assemblies, and manufacturing planning only supports a single user. However, in practice several engineers are usually involved in the development of a product. It is true for not only complex products but also relatively simple products. Moreover, collaboration among team members shows an increasing importance in solving design conflicts as early as possible in the design stage. Thus, a platform supports collaborative design with current popular CAD systems is a desideratum. The major requirements of such a platform are:

- Not only viewing operations but also modeling functions should be enabled for the development of product models, so that different users who are involved in the design activity and located at different sites can modify the product data together online;
- The size of data transferred should be reduced as much as possible, for the bandwidth is still a bottleneck of current Internet;
- Users could use their familiar systems during the design procedure.

Based on these requirements, an integration-based method is given in this paper for constructing a real-time collaborative design platform within heterogeneous CAD systems. Our method is command-based, so that the amount of data transmission is greatly limited. Different from the visualization-based approaches, models can be constructed and modified synchronously from various sites in the proposed collaborative design environment. Based on a translation mechanism between *system modeling operation*s (SMO) and *neutral modeling command*s (NMC), every operation given by a user on one site will be translated into an NMC and be sent to all the other sites through the network. When the other sites receive this command, it is converted into corresponding SMOs on the local system. The whole collaborative design platform is constructed in an integrated manner by developing a central management server, and several client-side system-dependent manager applications, which are usually in the form of add-ons. The mechanism and structure of our integration approach are shown in Fig. 1. On every client site, the CAD system is equipped with a manager add-on, which takes the role of SMO↔NMC translators, the sender and receiver of NMCs, and the coordinator for the modification permission. The translated NMCs are sent to the central server and then forwarded to all the other sites. For the security reason, the NMCs are usually encrypted and compressed before transmission.

Compared with other collaborative design solutions that can be found in literature [1-17], our integration-based method for

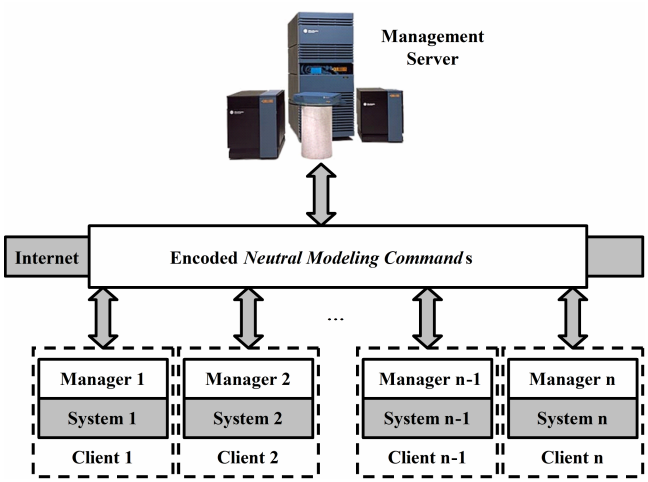



**Fig. 1 Structure of the proposed collaborative design environment**

developing a collaborative design platform within heterogeneous CAD systems has the following contributions:

- A new method for developing an online collaborative design platform is presented;
- Our collaborative design platform is based on an integration approach with heterogeneous CAD systems. Thus users on different sites can still manipulate models by using their familiar CAD systems during the design procedure. This is the benefit that cannot be offered by the homogenous collaborative design (co-design) systems;
- Not only the visualization but also the real-time manipulation of models is supported by the developed platform, which is the urgently requested function by industrial users;
- The creation and modification of models could be given online collaboratively by several users in real-time. Only the neutral modeling commands but not the models are transferred, the data size under transmission is very limited, so that an instantaneous synchronization can be achieved by a standard network bandwidth.

The rest of the paper is organized as follows. After reviewing the related works in collaborative design, the methodology of our proposed approach is introduced in section 3, where the mechanism of collaborative design platform, the selection criteria of client CAD systems, and the construction principles of neutral modeling commands are presented consecutively. Section 4 will focus on the representation of NMCs and the translation between NMCs and SMOs. Results of our current implementation on top of several commercial CAD systems are given in section 5, and the limitations are also discussed in this section. Finally, our paper ends with the conclusion section.

## 2 Related Works

In last decade, quite a few pieces of research have been investigated in synchronized collaborative design and several prototype systems have been developed. Following the classification given in [1], the approaches can be divided into two types: 1) visualization-based design systems, which support the function of viewing, annotating and inspecting design models in a Web or a CAD environment; and 2) co-design systems, which provide users the function of modeling and modifying models interactively and collaboratively online.

The visualization-based CAD systems usually have the functions supporting visualization, annotation and inspection of models. They are implemented either in plug-ins of Web browsers or as add-ons in some CAD systems. Among the visualization-based collaborative design platforms, the most famous one is SolidWorks eDrawing™ [2] which is a viewer for SolidWorks files. The eDrawing is equipped with viewing, marking-up, 3D pointing and animation tools. The product data in eDrawing is delivered in a save-and-download manner, so that it is in fact an offline approach. In order to deliver and manipulate interactive 3D objects effectively through the Internet, a variety of 3D streaming-based communication methods for collaborative design [1, 3-5] have been developed. In [1] and [3], the authors developed a geometric model simplification approach to exploit trimming information in CAD models while preventing the distortion of design features. Their work aims at supporting visualization of multiple CAD models in a distributed CAD environment. Wu and Sarma in [4] introduced a mechanism to trace the update of facet models, where a changed portion of a model is encoded in an incremental editing manner, transmitted to other sites in a distributed environment, and finally embedded into the associated faceted models at other sites. Two benefits are given by their approach: 1) the editing activity is encoded incrementally, so that the complex reconstruction after each operation is avoided; and 2) only updated portion of a model is transmitted for synchronization, therefore, the bottleneck of repeatedly transferring a large amount of facet data over networks is prevented. The approach in [5] presented a similar idea to [4] but focused on the real-time transmission of the boundary representation models (B-reps). The algorithm consists of three steps: identifying and encoding the incremental model of the B-rep once a modeling operation is performed; then transmitting the incremental model as well as the related geometric information to other remote sites; finally, decoding the received codes of the incremental model and directly embedding the restored entities into the local B-rep. Since the B-rep models rather than the facet models are supported by [5], the technique can be conducted to develop the geometric modeling kernel of co-design systems. There are some commercial viewers based on 3D streaming technologies available in the market (e.g., Cimmetry Systems Autovue™ [6], ConceptWorks™ [7], and Autodesk Streamline™ [8]).

As mentioned in [1], the co-design systems usually can effectively support collaborative modeling and collaborative modifying functions among designers. According to the architecture, the co-design systems can be divided into two types: homogeneous and heterogeneous. A centralized homogeneous platform usually acts in the mode of fat-server and thin-clients. The clients are light-weight and they primarily support visualization and interactive function such as selection, transformation, changing visualization properties of displayed parts, etc. The main modeling activities are taken in a common workspace in the server side (e.g., Alibre Design™ [9], OneSpace™ [10], the framework of Bidarra et al. [11], and the approach of Wang and Wright [12]). The advantage of a centralized system is that the system is easy to achieve the synchronization of data and perform the concurrency control. Their major problem is that the response speed of a system will be slowed down when the data exchange between clients and server becomes frequent and the interchanged model becomes complex. Therefore, some systems are developed in the mode of thin-server and strong-clients, where a server only plays as an information exchanger to broadcast CAD files or commands generated by client sites [1]. The implementations in this architecture include CollabCAD™ [13], IX Design™ [14], and the approach of Tay and Roy [15]. However, for all above co-design platforms, users must use the same CAD system which is distributed among the client/server structure – it means that they have to move from their accustomed design systems into the new system, and some additional cost for this new system is also applied to enterprises adopting it. Thus, the following question arises: could we find a

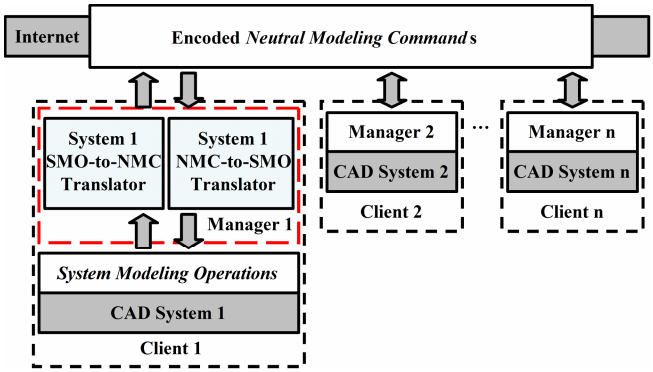



**Fig. 2 Structure of a client-side add-on application**

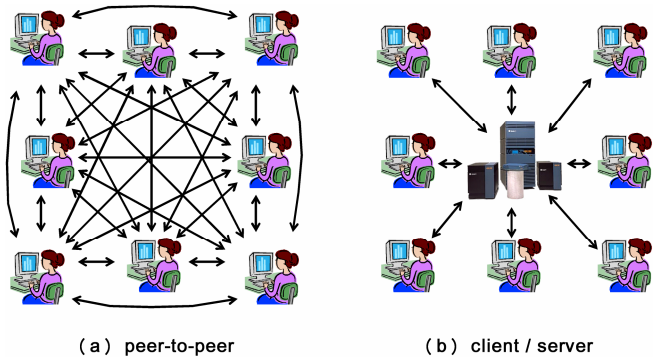



**Fig. 3 Peer-to-peer topology vs. client/server topology**

way to support collaborative design activities while still adopting the systems we used to? This question leads to a co-design platform supporting heterogeneous CAD systems proposed in this paper.

Another piece of related works is about the procedure-based data exchange of parametric feature-based models. Some research focuses on the translation of parametric feature-based models from one format into another. Choi et al. [16] proposed a macro-parametric approach to exchange CAD models. This approach is further extended to the feature-based macro-file format supporting the representation of the history-based parametric design [17]. We borrow some idea from [17] to define the set of *neutral modeling command*s. Besides academic research, there are also some feature-based translators being developed by industries such as ASPire3d [18], Proficiency Collaboration Gateway™ [19], Theorem Solutions [20], and Acc-u-Trans™ [21]. Among them, Collaboration Gateway, the translator developed by Proficiency, is a representative one. According to [22], in Collaboration Gateway, the Universal Product Representation (UPR) architecture is defined and adopted to provide universal support for all data levels employed by present CAD systems. Currently, the newest version of Collaboration Gateway supports five high-end CAD systems including CATIA V4 and CATIA V5, I-Deas, Pro/ENGINEER, and Unigraphics. However, all these translators [18-22] concern about the offline exchange of CAD models. Simply extending them into a distributed environment is not feasible.

Recently, Li et al. in [23] also conducted a feature-based approach to develop a distributed and collaborative environment. Based on feature-to-feature relationships, they proposed a distributed feature manipulation mechanism to filter the varied information of a working part during a co-design activity to avoid unnecessary transfer of the large size complete CAD files each time when any interactive operation is imposed on the model by a client. However, their system is still in the mode of homogeneous platform with the modeling activities are given on the server side. Our approach is different: the server only manages the command transmitting events; and the modeling activities are performed in real-time on every client sites by their own CAD systems. Since only commands are transferred, real-time responses can be achieved on the Internet with standard bandwidth. Details are presented in the following sections.

## 3 Methodology

### 3.1 Mechanism.

The major idea of our integration-based solution for developing a real-time collaborative design platform on heterogeneous CAD systems is to integrate existing CAD systems into a distributed design framework that supports real-time collaborative design activities. The structure of our proposed framework is shown in Fig. 1, where the manager applications are developed as add-ons on selected commercial CAD systems. These client-side manager add-ons take the duty of capturing the operations given by users, converting the *system modeling operation*s (SMOs) into *neutral modeling command*s (NMCs), sending NMCs, receiving NMCs, and decoding the received NMCs into corresponding SMOs. A client-side manager application developed for our proposed environment should follow the structure shown in Fig. 2. Since our platform is in a distributed mode with heterogeneous CAD systems, every client site has a distinct CAD system performing the activities of product modeling. On top of an independent CAD system is a system-dependent manager add-on consisting of two translators. The SMO-to-NMC translator captures and encodes each locally performed modeling operation into an NMC; then this NMC will be sent to the central management server through the Internet. Another translator plays the role of NMC-to-SMO translation, which is in charge of decoding every NMC that is received from the central server into one or more corresponding SMOs. These two translators are the kernel technologies to enable the real-time exchange of modeling operations between heterogeneous CAD systems, so that the synchronized collaborative design is supported. In the environment equipped with the SMO↔NMC translators, every user-performed SMO is immediately translated into an NMC being sent to other sites; while as soon as one NMC arrives, it is decoded into corresponding SMOs to be executed on the local system. Based on the proposed mechanism, every CAD system only interacts with NMCs. Therefore, one CAD system is independent of those CAD systems on the other sites. According to our experiments, the time taken to implement this platform is approximately linear to the number of CAD systems integrated.

#### 3.1.1 Topology of Sites.

When conducting communication among sites to transfer NMCs, there are two basic ways to structure the messaging topology of communication: peer-to-peer or client/server (shown in Fig. 3). The implementation of the peer-to-peer is simple; however, the client/server mode is more efficient than the peer-to-peer mode, especially for the case there are a large amount of clients involved in the design. In particular, they scale much better than the peer-to-peer mode because additional users only cause a linear increase in the message traffic. The weak processing power of a user's computer will greatly influence the response speed in the peer-to-peer mode, but have almost no effect in the client/server mode. Therefore, we suggest the client/server topology.

In most cases of client/server topology, the servers having modeling functions or data accessing functions must face the problem that performance of these servers will decline when the number of clients is increasing. Nevertheless, in our proposed platform, the functions of the central management server are

limited to receiving incoming commands and forwarding them to the other sites. Hence, unlike those "heavy" servers with modeling functions, our central server works as a "thin" one due to its limited functions and lower performance requirements. Furthermore, equipped with server-side multi-threading technique, the response time could be obviously shortened and the thin server could be speeded up to overcome the problem of performance bottleneck.

### 3.1.2 Initialization.

The central management server can be physically located in the same computer of a user (i.e., the project manager). We define *coordinator* as the user who initializes a collaboration session, and define *modifier* as the user who is authorized to modify the product data at some time current. Among *n* users involved in a collaboration session, there is only one *coordinator* and one *modifier* at any time. During the process of design, the *modifier* can be shifted to different users by gaining the modification permission from their *coordinator*, while the *coordinator* cannot be changed.

The first initialization method is that, when a *coordinator* creates a collaboration session, the client manager add-on on the *coordinator*'s computer delivers the existing product data to all the other *n*-1 users' sites through the management server. In detail, the initial product data is transferred from the *coordinator*'s computer to the server first; and then the server transfers the data to the computers of the other *n*-1 users. In our approach, the initial existing product data is represented by a list of NMCs encoding the design history of the parametric product model, which is the result of last collaboration design session.

The second method to initialize product models across sites is to let the *coordinator* open the legacy or saved native CAD file using "File→Open" menu of the local CAD system. The *fileOpened* event on the *coordinator*'s site triggers the local manager add-on to firstly traverse the feature tree of the opened file in a top-to-bottom manner, and then translate every feature into its corresponding NMC, and finally send them out. In this way, after all translated NMCs have been sent out from the *coordinator*'s site, parametric product models with the identical feature semantics are disseminated among the other client CAD systems and ready for a new collaboration session. Chen et al. [24] describe the capture of *fileOpened* event and the traversing mechanism of feature-trees in detail.

Compared with the first initialization method using an NMC list, a limitation of the second method is that, some feature created in last collaboration session, which is supported by our proposed co-design platform but is incompatible with a certain native CAD file format, will be filtered out and lost after such a CAD file is saved. As a result, the re-opened model is inconsistent with the model saved in the last collaboration session. Therefore, we recommend the first initialization method.

### 3.1.3 Concurrency Control.

In order to avoid concurrent modification conflicts, we introduce a token-based locking mechanism, which is implemented by transferring the modification permission (token) among all users. In detail, on every site, the manager add-on keeps a flag indicating whether the user on this site is the only *modifier* or not. If the user is the *modifier*, the modeling operations performed by this user are converted into NMCs and delivered to the management server. Otherwise except for system viewing operations, every modeling operation given by this user will be automatically rejected by the local manager add-on. By this locking mechanism, only the *modifier* can modify the product model at any time. In this way, the write-after-write conflicts are avoided in our proposed co-design platform. Considering assignment of the *modifier*, it is the duty of the *coordinator*, who creates the collaborative design session and controls the whole design procedure. Usually the *coordinator* is the project manager. When the *coordinator* wants to assign the modification permission to a user, the manager add-on of the *coordinator*'s site will send a command to the user's site to let its manager add-on activate the SMO-to-NMC translator. At the same time, commands will be sent to all the other sites to let their manager add-ons to serve as NMC-to-SMO translator only – i.e., the SMO-to-NMC translation is disabled. The modification permission can be assigned and withdrawn by the *coordinator* (i.e., the project manager) at any time.

Anyone wants to modify the model can request the modification permission from the *coordinator* through a chatting channel. If the modification permission is authorized, the *modifier* – the user who got the authorization, can modify the product model on his or her site. The NMCs generated on the site of the *modifier* will be firstly sent to the central management server, and forwarded to the *coordinator* by the server. If the modeling operations were rejected by the *coordinator* (i.e., the project manager), the manager add-on on the *modifier*'s site will undo these operations automatically, so that the product models on all the sites are consistent. If the *coordinator* confirmed this modification, the server delivers these NMCs to all the other sites. When the manager add-ons on other sites receive these NMCs and finish corresponding updates of their local models, every add-on will highlight these modifications to its user in a visual manner and send an acknowledgement message to the server automatically. If the server did not receive the acknowledgement from a site within an expected time, these NMCs will be re-sent to that site for two more times. If the server still did not receive any reply from that site, the user on that site is assumed to have aborted the collaboration session, and the aborted user needs to join the collaboration session again sometime later. Once a user requests to join the collaboration session, the current parametric design history in the form of an NMC sequence stored in the coordinator's computer is generated and sent to the new user to initialize the product data.

Another method of concurrency control is named as token-ring algorithm, which passes the modification permission (token) among the members along with a logical ring. However, the token-ring algorithm would perform very poorly in lightly loaded cases just like our co-design environment, mainly because a site may have to wait through many unused token passes for a turn. Moreover, the token-ring algorithm is known to be less scalable. Therefore, we adopt the token-based locking method as our concurrency control mechanism owing to its simplicity, stability, and scalability.

## 3.2 Selection Criteria of Client CAD Systems.

According to the framework introduced above, no special CAD systems need to be developed for the proposed collaborative design environment. Only add-ons are to be developed on each selected CAD system. Of course, not every CAD system can be integrated into such a collaborative design environment, thus the selection criteria of client CAD systems are given below. For the candidate CAD systems, they must satisfy the following two major criteria:

- The systems should provide the ability for developing add-ons;
- Each operation applied to the product model in a CAD system is able to be tracked instantaneously.

For the first criterion, most modern CAD systems support it. There are usually two ways to program on CAD systems: by script language and by C++ API (application programming interface). The script languages are often interpreted languages which must be checked for errors at run-time; while an add-on written in C++ API is compiled from source codes to native machine instructions. Thus, an add-on in a script language runs much slower than the

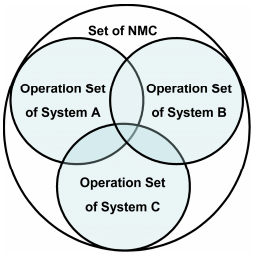



**Fig. 4 Union of parametric feature modeling operations**

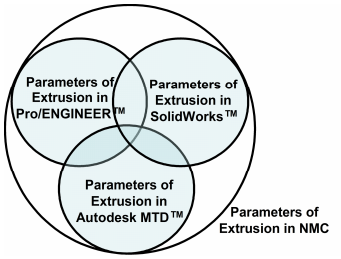



**Fig. 5 Example union of parameters for extrusion**

equivalent add-on written in C++ API. In addition, an add-on written in C++ can be equipped with the existing network communication libraries [25-27]. Therefore, in our solution of collaborative design with heterogeneous CAD systems, all add-ons are written in C++ APIs.

The second criterion requests that the add-on program on one site is able to instantaneously trace operations performed on the local CAD system. According to our mechanism, every SMO performed on an arbitrary local CAD system should be captured and translated into an NMC. Thus, the selected CAD systems must provide their manager add-ons with the ability to trace all operations of the CAD system as well as their parameters in real-time. In addition, the NMC that corresponds to each SMO should also has a number of corresponding API functions for all selected CAD systems, so that each sites can apply its corresponding API functions to update its product model accordingly.

Our preliminary investigation shows that the following commonly used CAD systems satisfy the above two selection criteria: SolidWorks™ [28], Autodesk Mechanical Desktop™ [29] (known as MDT), Pro/ENGINEER™ [30] (known as ProE), CATIA™ [31], NX™ [32] (previously known as Unigraphics), and PowerShape™ [33]. We then select the first three systems to implement a prototype of the proposed real-time collaborative design platform which will be demonstrated later. For other CAD systems, since most of them are developed on an open architecture in current fashion, they can be easily integrated into our platform as long as they provide necessary API functions.

### 3.3 Construction Principles of Neutral Modeling Commands.

Neutral modeling commands play an important role in achieving real-time synchronization for the collaborative design among heterogeneous CAD systems. To guarantee the rationality and validity of the NMC set, the set should be constructed following the two principles below:

- Based on parametric feature modeling operations and their parameters;
- As a union of parametric feature modeling operations and their parameters on all integrated client systems.

Parametric feature modeling (PFM), as one of the most advanced ways for product modeling, can effectively support geometric modeling with parametric features. PFM is adaptive to design practices, and the environment of variational design and intelligent design can be developed based on parametric features. Accordingly, PFM is the most popular product modeling method provided in all of current commercial CAD systems. Therefore, a successful collaborative design platform must support PFM, and the NMC set is constructed based on the activities of feature-based parametric design (i.e., every NMC corresponds to a number of PFM operations).

We observe that the essential modeling operations provided by all commercial CAD systems are similar, although some equivalent operations may differ slightly from one another in their

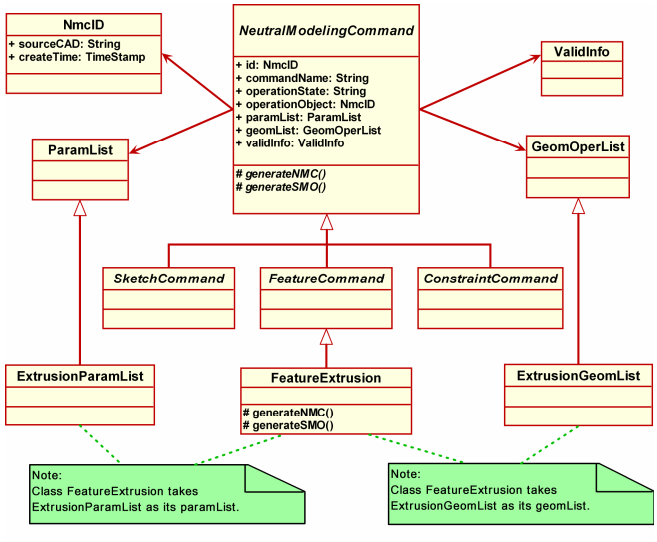



**Fig. 6 General UML class diagram of NMCs**

detailed parameters. In order to ensure that every SMO can be translated into an NMC, the NMC set is desired to be the union of all PFM operations of the integrated CAD systems (see Fig. 4). Similarly, the parameters of each NMC take the union of parameters of all equivalent operations with the same design semantics. As shown in Fig. 5, taking extrusion operations as an example, all investigated CAD systems support extrusion in one direction which is called *one-side-extrusion*, while there is a *bi-extrusion* option provided in SolidWorks and Pro/ENGINEER to enable users to extrude the profile in both directions from the sketch plane. Thus, the parameters of the extrusion command will also include the *bi-extrusion* attribute. For a modeling operation that cannot find corresponding operations in another system, we convert it into a sequence of geometric operations in the local system.

## 4 Representation and Translation of Neutral Modeling Commands

Serving as the key technique in our integration-based solution for developing a real-time collaborative design platform on heterogeneous CAD systems, the representation method of NMCs is firstly detailed in this section. In the following, the translation mechanism between SMOs and NMCs is described.

### 4.1 Representation.

For supporting the implementation of two translators – SMO-to-NMC and NMC-to-SMO effectively, we represent all NMCs in an object-oriented manner, where each type of NMC is a class and can be instantiated into an object with functions during the collaborative design. In addition, each NMC has a string representation for transmission through the network, and the string includes the name and all attributes of the NMC. To facilitate interoperability between heterogeneous systems, extensible markup language (XML), the de facto open standard for information exchange, is adopted as the format of the string representation for NMCs. The general unified modeling language (UML) class diagram of the object-oriented representation of NMCs is shown in Fig. 6.

The class *NeutralModelingCommand* is the root of all NMC classes, where its first attribute *id* is an identifier consisting of the local system's name and the creation time of the NMC. The attribute *commandName* is the NMC's name and the *operationState* indicates one of three states: *creation*, *modification* and *deletion*. For example, a modification operation about an

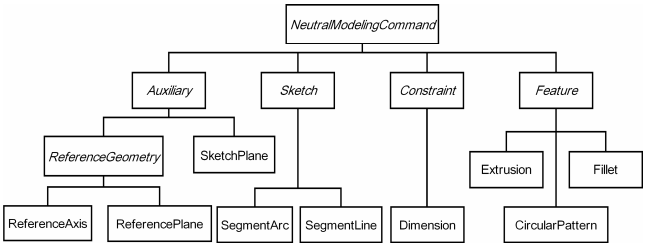



**Fig. 7 Hierarchical tree of the identified neutral modeling commands (NMCs) used in the paper**

extrusion feature is translated into an NMC object with whose *commandName* being "FeatureExtrusion" and its *operationState* being "modification"; and the modified object is indicated by *operationObject*. Similarly, the removed object after a deletion operation is referred in *operationObject*, too. The attribute *paramList* represents the parameters of an NMC. For each class derived from *NeutralModelingCommand*, class *ParamList* provides a concrete subclass to express its specific parameters. As shown in the left-bottom of Fig. 6, class *FeatureExtrusion* takes class *ExtrusionParamList* as the type of its *paramList* attribute. In the case if no match is found between an NMC and the SMOs at the remote site, the NMC will be converted into a sequence of geometric operations to be applied at the remote site. The *geomList* is conducted to store this sequence of geometric operations. Similar to the family of *ParamList*s, every NMC class takes a corresponding subclass of class *GeomOperList* to represent its specific sequence of geometric operations. The attribute *validInfo* carries the validation information of an NMC.

There are two translation functions in each NMC class: *generateNMC*() and *generateSMO*(). The *generateNMC*() is used to translate an SMO into an NMC, whose input is a local operation and output is an NMC with corresponding parameters and geometric operations; while *generateSMO*() is in charge of converting a received NMC to its corresponding SMOs on the local system. The implementation of these two functions of each NMC class is CAD system dependent (i.e., in the add-ons on different systems, they should be developed respectively).

For certain NMCs, their parameters include some topological entities (e.g., fillet operation needs one or more edges as its parameters). In homogeneous systems, such entities can be identified by their IDs or pointers in the local machine where they are created. However, this does not work for the communication among the distributed heterogeneous systems since the IDs or the pointers of the same topological entities may vary in different CAD systems. To overcome this difficulty, we adopt one entity's type and its geometric information in the world coordinate system (WCS) to express this topological entity in an NMC. For instance, a linear edge is expressed by its type and the coordinates of two endpoints in WCS. Similarly, a planar face is expressed by its type, its unit normal in WCS and the WCS coordinate of a point on the face, and freeform curves and surfaces are presented in Non-Uniform Rational B-Spline (NURBS). The feasibility of this method depends on whether the geometric information of the same topological entities in different CAD systems is equal or not. Fortunately, from our experiments, the WCSs of all investigated CAD systems are Cartesian coordinate system following the right-hand rule. Since any user coordinate system (UCS) can be freely transformed to WCS and vice verse, no matter what UCS is adopted, the geometric information of the same topological entities under the WCSs of different CAD systems is consistent.

All identified neutral modeling commands in this paper are shown as leaf nodes in Fig. 7. The non-leaf nodes represent the semantic abstraction of NMCs.

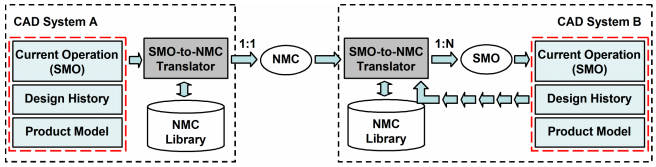



**Fig. 8 Translation between SMOs and NMCs**

## 4.2 Translation.

In order to support the synchronized collaborative design, an NMC-based method for the real-time exchange of modeling operations among heterogeneous CAD systems is proposed, where the primary issue is how to effectively and efficiently implement the translation between SMOs and NMCs.

### 4.2.1 Real-Time Translation between SMOs and NMCs.

Fig. 8 illustrates the process of the real-time translation between SMOs and NMCs. As soon as a feature-based parametric design operation is applied on a local CAD system, the operation is captured by the SMO-to-NMC translator in the local add-on (shown in the left of Fig. 8). The SMO-to-NMC translator matches the operation in the local NMC-library to find a corresponding NMC-template. The NMC-library is system-dependent – in other words, an NMC will have different NMC-templates in the NMC libraries for various client CAD systems, and the NMC libraries for the same system should be identical even if they are located on different sites. An NMC object is then created by the matched NMC-template and its system-dependent *generateNMC*() function is invoked with the captured operation as its input to determine necessary information of the NMC including all parameters, validation information, and optional geometric operations. Finally, the SMO-to-NMC translator calls the string serialization method to create a string representation of this NMC, which is sent to the other sites immediately. The process of the real-time translation from an NMC to SMOs is illustrated in the right of Fig. 8. As soon as an NMC string is received from the network, the local NMC-to-SMO translator finds out its corresponding NMC-template in the local NMC-library through the command name of the received NMC. Then, an NMC object is created by the matched NMC-template and its system-dependent *generateSMO*() is invoked with the received NMC string as input to generate related SMOs. After the SMOs are completed by the local CAD system, the validation of the product model is executed.

### 4.2.2 Implementation of *generateNMC*() and *generateSMO*().

The function *generateNMC*() plays a principal role in the process of encoding an SMO into an NMC, which is responsible for generating all the necessary information of a matched NMC-template. Its detailed tasks are:

- Generating all parameters of an NMC object by an SMO – the corresponding NMC parameters of an SMO usually consist of two parts: *direct* and *indirect* parameters, where direct parameters refer to those are also parameters of the SMO and hence can be directly obtained from the SMO, and the indirect parameters are calculated from the product model or the design history by the *generateNMC*() function (e.g., for the SMO of extruding from a sketch to a face, the extrusion depth is implicitly defined by the position of the selected face, which is an indirect parameter to be computed by *generateNMC*());
- Constructing the sequence of geometric operations of the NMC, where both the types and the parameters of

each geometric operation are computed from the SMO applied on the product model;

- Creating the validation information of the NMC – the mass properties of the updated solid model (e.g., the surface area and the center of gravity) are taken as the validation information.

The implementation of *generateNMC*() for each NMC is system-dependent.

Similarly, the other system-dependent function *generateSMO*() is responsible for decoding an NMC string into a sequence of system-dependent SMOs which is called *SMO-group* when receiving the NMC string, where the function's tasks include:

- Parsing the received NMC string and accessing its parameters – all parameters and validation information should be easily accessed;
- Converting the NMC into an *SMO-group* – with all the parameters involved in the NMC as well as the parameters calculated from the product model and design history for certain cases. All the modeling operations involved in an *SMO-group* are executed on the local CAD system consecutively;
- Verifying the updated product model – two mass properties: 1) the surface area and 2) the center of gravity are computed to verify the updated product model, where if the difference of mass properties between the local product model and the received validation information is greater than a given tolerance, the NMC is considered as being applied incorrectly.

One received NMC may be decoded into more than one SMOs, for example, for the received NMC of extrusion whose *bi-extrusion* attribute is *true*, the *generateSMO*() defined on the system not supporting bi-extrusion will translate the bi-extrusion NMC into two one-side-extrusion operations based on the same sketch. If a received NMC does not match any NMC-template in the local NMC-library, the local product model will be updated by the geometric operations involved in the received NMC.

### 4.2.3 Processing Method for Modification and Deletion.

Differing from data exchange systems, the synchronized collaborative design system must support the function of real-time modification and deletion. The synchronized modification and deletion of a feature are more difficult to be achieved than the synchronized creation because they depend on some existing features. Before any modification or deletion operation is performed, we need to determine these depended features. The attribute *operationState* of each NMC is set to *creation*, *modification* or *deletion* to distinguish the three states, and the attribute *operationObject* indicates the feature of product model to be manipulated.

Before presenting the method to process modification or deletion requests, let us introduce a concept about *NMC-based design history*, which associates each NMC object to its affected portions of the product model. The design history of collaborative design can actually be presented in a list of NMCs received by the management server and sorted by the receiving time. Similarly, the NMCs being sent out and received at every client site also form a design history of the local product model if these commands are sorted chronologically. The list of sorted NMCs on one site is maintained to be consistent with the lists of NMCs recorded in the other sites, where the NMCs having the same ID are identical (e.g., as shown in Fig. 9, the lists of NMCs in two different CAD systems have the same order of commands). The list of sorted NMCs on one site is called the local *NMC-based design history* of this site. In a local *NMC-based design history*, every NMC is stored together with the pointers to one or more affected features in the model (shown in Fig. 9). Therefore through one-to-one mapping between the identical NMC IDs, the

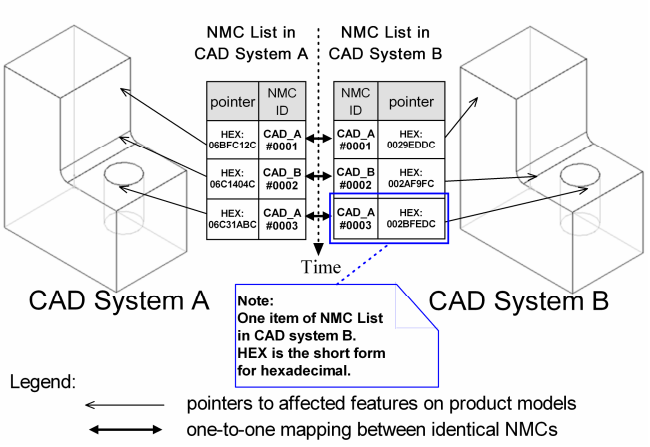



**Fig. 9 The NMC-based design history concept for associating affected portions of heterogeneous product models**

corresponding features on heterogeneous product models in different CAD systems can be associated effectively. An illustration of such association is shown in Fig. 9. Note that, for the same NMC in the *NMC-based design histories* recorded on different sites, the number of the pointers stored with the NMC may be different. For instance, for the bi-extrusion NMC on one site with a CAD system supporting bi-extrusion, there is only one pointer stored with it; while the bi-extrusion NMC on another site with a CAD system not supporting bi-extrusion has two pointers stored with it, each of which points to a one-side-extrusion feature.

With the help of *NMC-based design history* concept introduced above, when a feature is modified by an SMO locally, it is easy to find its corresponding features on the other sites with following four steps:

i. In a local CAD system where an SMO is applied, by traversing the local NMC list, we find out the NMC whose pointer also points to the feature modified by the SMO just applied;
ii. The found NMC's ID is then determined;
iii. On a remote site, using the determined ID, we locate the corresponding NMC in the local NMC list;
iv. Using the pointers stored together with the located NMC, the affected features of the product model on the remote site is also determined.

If a deletion request is given, all corresponding features should be deleted on an arbitrary remote site could also be determined using the similar steps listed above in a real-time manner.

By this associating mechanism of features, we deal with modification requests as follows. First, the SMO-to-NMC translator captures a modification operation and finds out its modified feature on the product model. According to the type of the modified feature, a corresponding NMC-template is chosen by searching the NMC-library and a new NMC instance Ψ is created with its *operationState* set to "modification". The *generateNMC*() function of Ψ is then invoked, where the modified feature of the captured SMO is used to find out a creation-NMC (i.e., the NMCs with operationState as "creation") that created this feature in the local *NMC-based design history*. The ID of the found NMC is assigned to the attribute *operationObject* of Ψ. The consequent steps such as generation of parameters and validation information are the same as generating a creation-NMC. Regarding a deletion operation, the processing method is similar to that of processing a modification operation but with all parameters of the NMC being null.

Because the features to be modified or deleted need to be determined firstly, unlike a creation-NMC, the *generateSMO*()

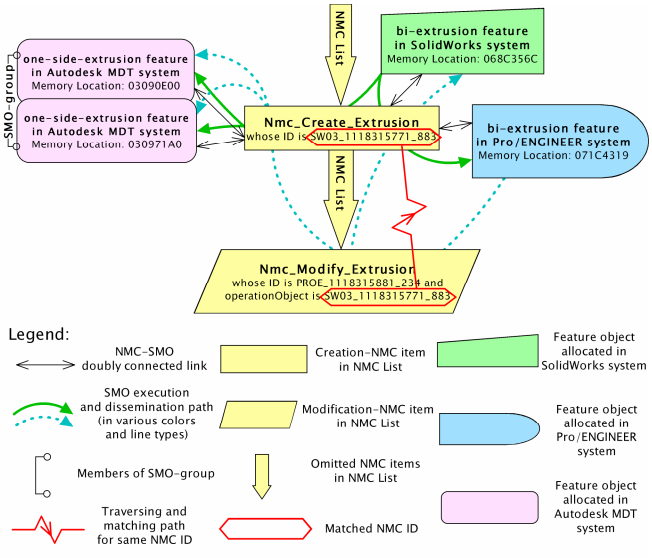



**Fig. 10 An example for illustrating the association mechanism of SMO-group, the SMO execution and dissemination path among heterogeneous CAD systems, and the NMC traversing and matching path in the NMC-based design history**

function's implementation of a modification-NMC (i.e., the NMC with *operationState* as "modification") or a deletion-NMC (i.e., the NMC with *operationState* as "deletion") is quite different. With the help of ID stored in the *operationObject* parameter of an NMC, the previous creation-NMC that constructed the requested feature is found out from the local *NMC-based design history*. Following the associated pointers of the found creation-NMC, the features to be modified or deleted are determined on the local site so that they can be finally updated. It is worth noting that for the case that the found creation-NMC has more than one associated pointers, all the features referred by the associated pointers are modified or deleted, i.e. multiple SMOs are generated and executed. The NMC-to-SMO translation of the modification-NMC of a bi-extrusion on the site with a CAD system not supporting bi-extrusion is an example of such case.

Fig. 10 illustrates an example of creating and modifying a bi-extrusion feature collaboratively using various CAD systems some of which support bi-extrusion feature while the other ones do not. After one SMO "bi-extrusion feature" has been executed locally in SolidWorks system and disseminated remotely (indicated in Fig. 10 as bold arrows), the identical replicas of an NMC item (shown as rectangles in the middle of Fig. 10) are appended to the NMC list of every site. Every replica binds with one or more affected feature objects in the local system with the help of pointers (shown as slim doubly connected links) stored in this NMC item. The bound affected feature objects are called as *SMO-group* of this NMC. From this figure, it can be clearly seen that, for the same NMC, the *SMO-group* of Autodesk MDT contains two one-side-extrusion features, whereas there is only one bi-extrusion feature in SolidWorks and Pro/ENGINEER. Once Pro/ENGINEER issued a modification-NMC (shown as oblique rectangles), which is received by the other two systems and appended to their local NMC lists, a traversing and matching routine (shown as zigzags) uses the *operationObject* parameter of the modification-NMC as input and starts to find out the local feature objects. By traversing the local NMC list, an NMC can be matched with the given *operationObject* parameter. Using this matched NMC, the local affected feature objects within its *SMO-group* could be determined using the NMC-SMO doubly connected links of this NMC, and then they are manipulated

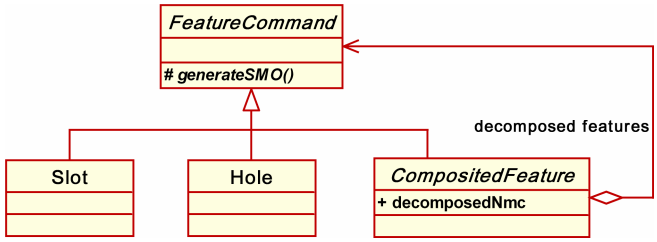



**Fig. 11 UML class diagram of CompositedFeature**

further. Here we can see that two one-side-extrusion features within the *SMO-group* in Autodesk MDT are coherently determined and updated as one.

### 4.2.4 Composite Features and User-Defined Features.

Composite features and user-defined features (UDF) need to be specially processed in our proposed collaborative design environment, where a composite feature is a complex feature composed of several basic features (e.g., a composite hole) and a UDF is a special kind of features defined by users.

To translate a composite feature operation, we decompose it into several basic features which are going to be translated into their corresponding NMCs. The NMCs for basic features are stored in the NMC for the composite feature as the auxiliary information. The abstract NMC class called *CompositedFeature* that represents the parent of all composite features is shown in Fig. 11. Every concrete composite feature is represented by a subclass of class *CompositedFeature*, with the NMC objects corresponded to the decomposed basic features stored in the list named *decomposedNmc* (refer to the right-bottom part of Fig. 11).

Similarly, a UDF also consists of several basic features. However different from composite features, for constructing itself, a UDF usually has some user-defined parameters, which are called *driving parameters*. Therefore, the NMC class for a UDF will have another list to store the *driving parameters* besides *decomposedNmc*.

## 5 Results and Discussions

Based on the proposed approach for integrating heterogeneous CAD systems into a collaborative design environment, we have implemented a prototype collaborative design platform with SolidWorks 2003, Autodesk MDT 6.0 and Pro/ENGINEER Wildfire 2. For each of these three systems, both SMO-to-NMC and NMC-to-SMO translators are implemented with Visual C++ 6.0 and the C++ API functions of the selected CAD systems. The translators are complied into add-ons of each CAD system, running as a background application after the host CAD system starts to work. The program serving as a management server is also written in C++, and it communicates with client sites using Transmission Control Protocol/Internet Protocol (TCP/IP).

Our first example is a bracket model which is shown in Fig. 12. With our prototype platform, three geographically dispersed users, using SolidWorks, Autodesk MDT and Pro/ENGINEER respectively, build and refine the bracket model in a collaborative manner. The designer using SolidWorks acts as both the *coordinator* and the *modifier* initially. At the beginning of the co-design session, the user using SolidWorks creates a base extrusion feature. Instantaneously, the SMO-to-NMC translator stationed in the SolidWorks is triggered by the *featureCreated* event. Based on the captured information about the extrusion feature, the NMC-template that corresponds to the extrusion feature is found out from the NMC-library and an instance of it is created. Using the instance's *generateNMC*() function, all parameters and the mass properties for validation are computed, which are finally serialized into an XML string and transmitted to all the other sites through the management server. After receiving the XML string of this

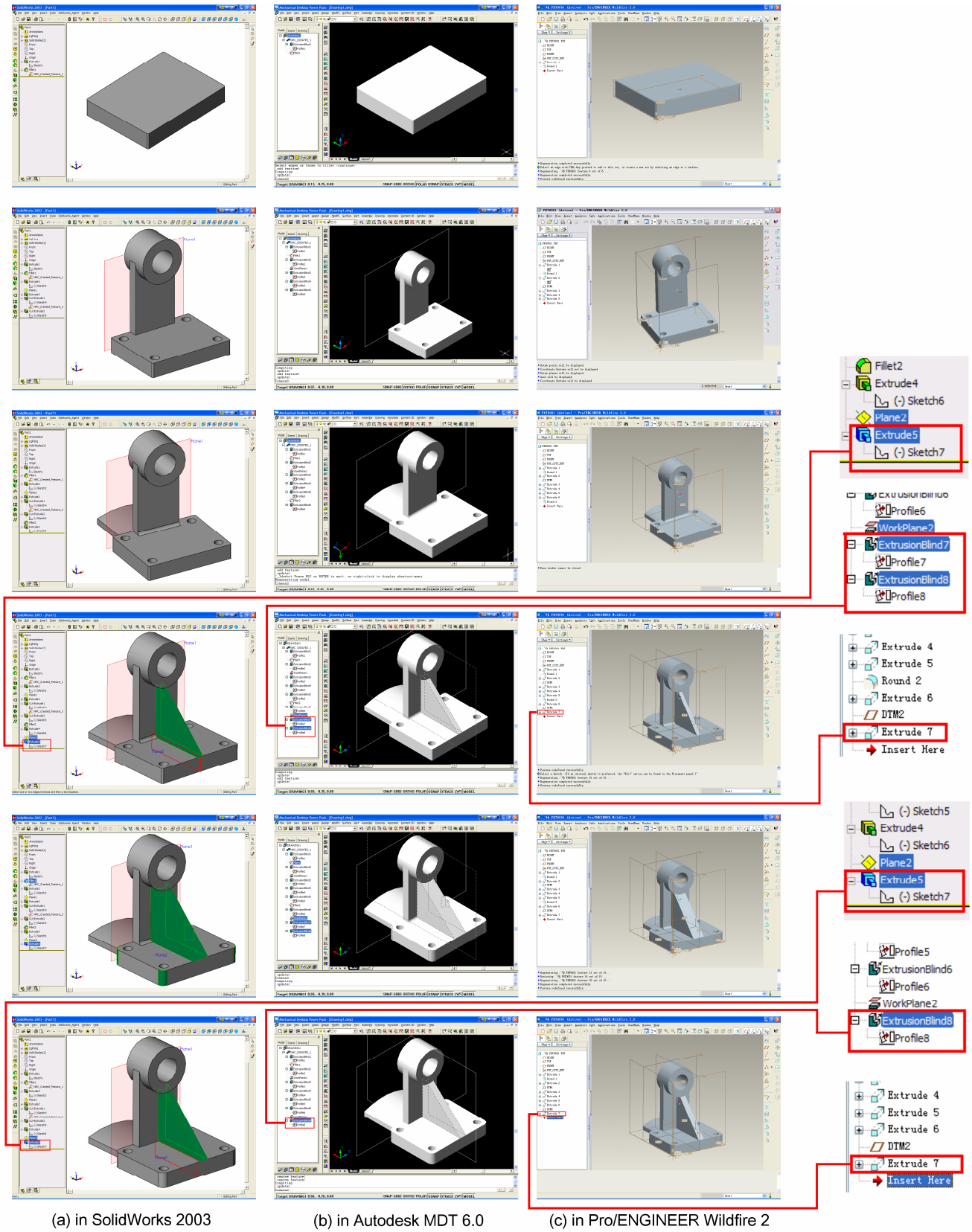



**Fig. 12 Example I: collaborative modeling of a bracket model using SolidWorks, Autodesk MDT and Pro/ENGINEER**

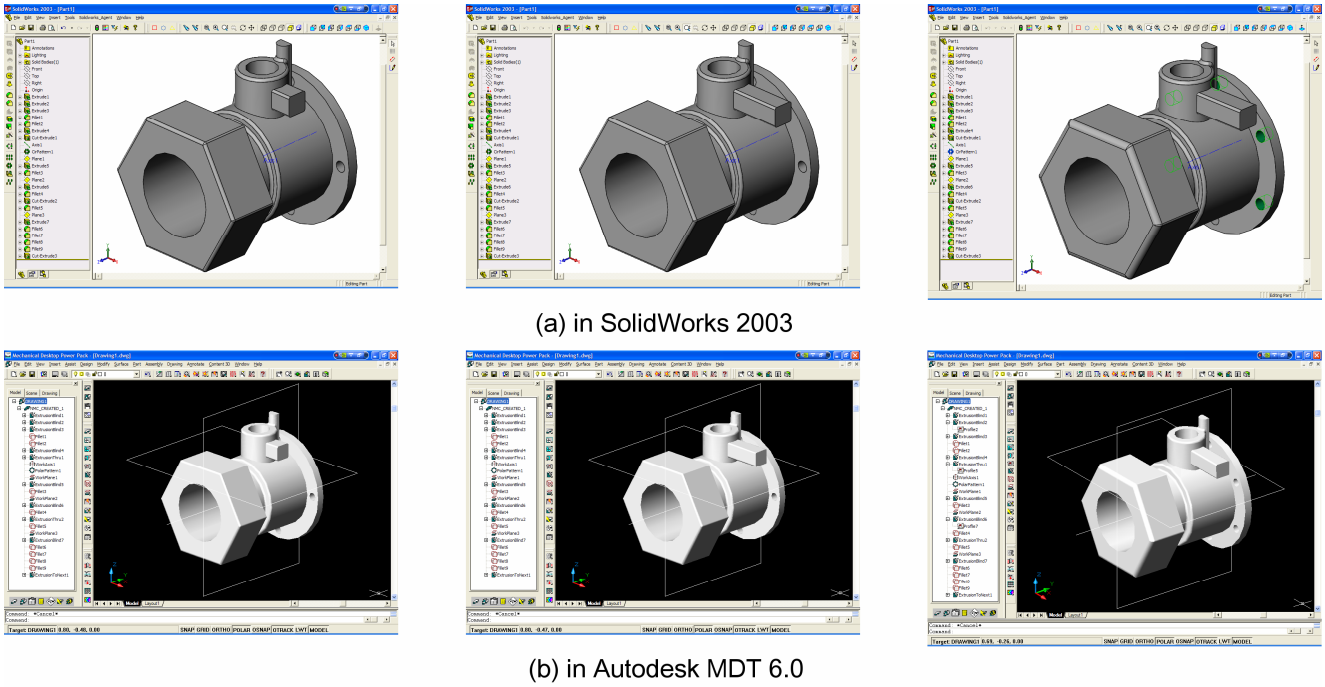



**Fig. 13 Example II: collaborative modification of a real mechanical part using SolidWorks and Autodesk MDT**

NMC on the sites of MDT and Pro/ENGINEER, the NMC-to-SMO translators stationed on those sites convert the command into an SMO that is going to be performed on the local client CAD systems. Then, the user on the MDT site feels that the extruded base part needs to have its four corners filleted, so this user asks for the modification permission from the coordinator by sending a request via a chatting channel. Being the *modifier*, the user on the MDT site fillets the base extrusion feature and the corresponding NMCs are generated and broadcasted to all the other sites after being confirmed by the coordinator. After fillets added, the modification permission is passed back to the coordinator (the user of SolidWorks). After the user of SolidWorks extrudes a block on top of the base part, the user of Pro/ENGINEER is authorized to create a reference plane based on the block and extrude a cylinder which is perpendicular to the block's back face. In the following, the user on the MDT site drills a through hole on the cylinder and the user of SolidWorks drills four small through holes on the base part consecutively. Go on the modeling, the bracket model is finally constructed by three users in a collaborative manner. Fig. 12 shows the progressive results.

Note that, after the user on the SolidWorks site constructs a bi-extrusion feature as a rib, the NMC for bi-extrusion is translated into two one-side-extrusion operations in the MDT system since there is no bi-extrusion operation supported by MDT (see the 4$^{th}$ row in Fig. 12 and its corresponding feature trees in the middle-right of Fig. 12 – two extrusion features are shown in the feature tree of MDT system). Pro/ENGINEER has the bi-extrusion operation, so just one bi-extrusion SMO is conducted synchronously. After modeling the rib and changing its thickness (shown in the 5$^{th}$ row in Fig. 12), the user of MDT deletes one of the two one-side extrusions that correspond to the bi-extrusion feature created in SolidWorks. The final result is as shown in the last row of Fig. 12 and its corresponding feature trees are shown in the bottom-right of Fig. 12.

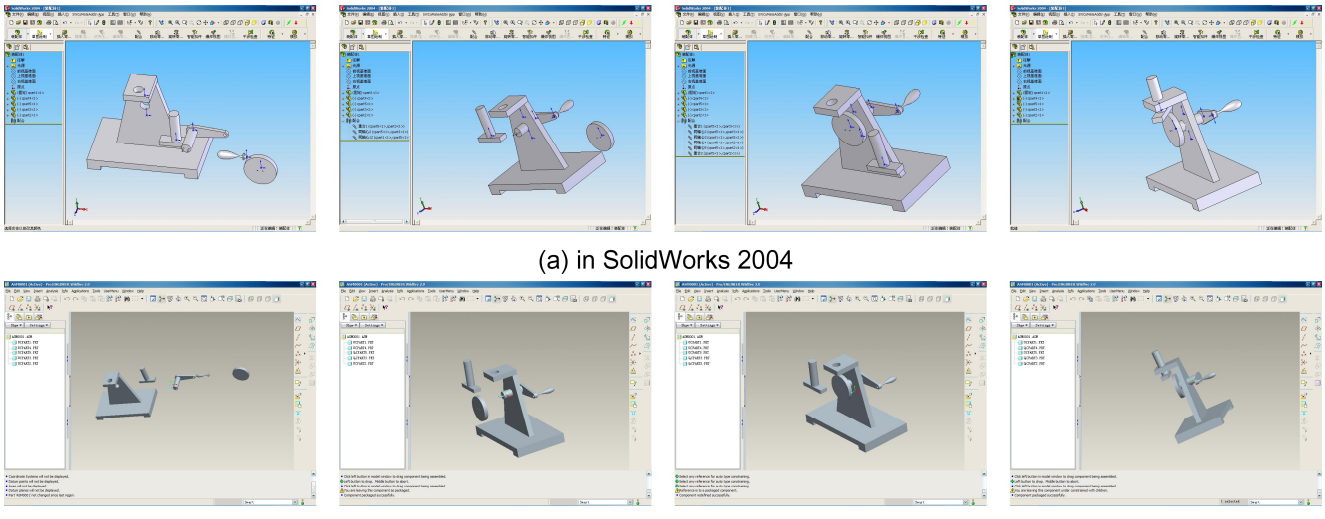



**Fig. 14 Example III: collaborative assembly of a mechanical structure using SolidWorks and Pro/ENGINEER**

```
- <NeutralModelingCommand class_id="0" class_name="NmcFeatureExtrusionData"
    tracking_level="1" object_id="_0">
  - <nmcData class_id="1" tracking_level="0">
    - <identifier class_id="2" tracking_level="0">
        <identifier.systemId>1</identifier.systemId>
        <identifier.timeStamp>1118315771</identifier.timeStamp>
        <identifier.milliSecond>883</identifier.milliSecond>
      </identifier>
    - <operationState class_id="3" tracking_level="0">
      - <NamedEntity class_id="4" tracking_level="0">
          <nameEntity.name>Creation</nameEntity.name>
        </NamedEntity>
      </operationState>
    - <operationObject>
        <operationObject.systemId>0</operationObject.systemId>
        <operationObject.timeStamp>0</operationObject.timeStamp>
        <operationObject.milliSecond>0</operationObject.milliSecond>
      </operationObject>
    </nmcData>
    <nmcFeatureExtrusionData.direction1>1</nmcFeatureExtrusionData.direction1>
    <nmcFeatureExtrusionData.direction2>-1</nmcFeatureExtrusionData.direction2>
    <nmcFeatureExtrusionData.draftAngle1>0</nmcFeatureExtrusionData.draftAngle1>
    <nmcFeatureExtrusionData.draftAngle2>0</nmcFeatureExtrusionData.draftAngle2>
    <nmcFeatureExtrusionData.depth1>0.01</nmcFeatureExtrusionData.depth1>
    <nmcFeatureExtrusionData.depth2>0.01</nmcFeatureExtrusionData.depth2>
    <nmcFeatureExtrusionData.combineType>3</nmcFeatureExtrusionData.combineType>
    <nmcFeatureExtrusionData.terminator1>1</nmcFeatureExtrusionData.terminator1>
    <nmcFeatureExtrusionData.terminator2>1</nmcFeatureExtrusionData.terminator2>
  </NeutralModelingCommand>
```

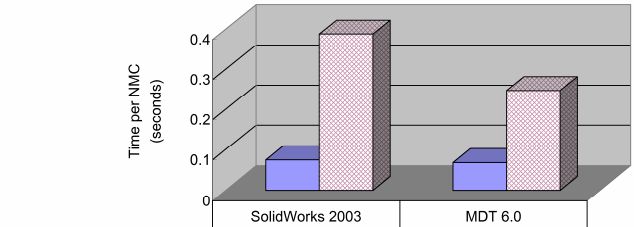
**Fig. 15 An example for a bi-extrusion NMC in XML format**

The second example is a real mechanical part shown in Fig. 13, which is recently used in [34] to demonstrate the functionality of their approach. Since the end of rectangle extrusion should align with the outer borders of disk base and hexagonal cap due to the fact that three points or components should be coplanar, the user of MDT modifies the length of rectangle extrusion to meet this requirement. After enlarging the radius of hexagonal face fillet, the user on the site of SolidWorks finally modifies the circular pattern from quarter-instances to hex-instances to adapt other connectors.

Based on the NMC-based approach, we have also implemented a collaborative assembly module in our co-design platform. The third example shown in Fig. 14 demonstrates this function, where two users work together to assemble five parts – a bracket, a cam, a hand knob, a cam follower lever, and a camshaft into a mechanical structure.

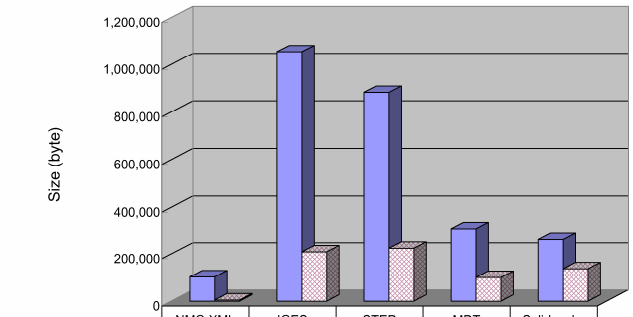
The data size per NMC and the data exchange rate have also been tested on our prototype platform using Pentium™ IV 2.6GHz PCs with 512 MB RAM running Windows XP Professional SP2. The average size per NMC (represented in XML format – e.g., Fig. 15) is 1843 bytes which is sampled by using 50 NMCs (including all NMCs identified in Fig. 7). Based on this data size, the network transmission time under different network speeds could be calculated as:

- Using a local area network (LAN) with 100Mbps (million bits per second) bandwidth, the transmission time is (1843×8bit)/(100×1024$^2$ bit/second) = 0.141×10$^{-3}$second;
- Using a standard dial-up modem with 56Kbps (kilobits per second) bandwidth, the transmission time is (1843×8bit)/(56×1024 bit/second) = 0.257 second.




| | SolidWorks 2003 | MDT 6.0 |
|---|---|---|
| SMO-to-NMC time per NMC | 0.0743 | 0.0678 |
| NMC-to-SMO time per NMC | 0.388 | 0.247 |

**Fig. 16 Average computing times on SolidWorks and MDT**




| | NMC-XML | IGES | STEP | MDT | Solidworks |
|---|---|---|---|---|---|
| Original Size (byte) | 101,829 | 1,054,930 | 882,991 | 306,068 | 261,632 |
| Compressed Size (byte) | 4,605 | 208,729 | 223,903 | 99,493 | 136,961 |

**Fig. 17 The size comparison of datasets between NMCs in XML format and other offline schemes**

Considering about the SMO-to-NMC and the NMC-to-SMO translation time, the client add-on program on SolidWorks 2003 spends 3.672 seconds to generate 50 NMCs and 19.421 seconds to execute 50 NMCs. The same tests also have been performed on MDT 6.0 system. The average computing times of SMO-to-NMC translation and NMC-to-SMO translation on these two systems are shown in Fig. 16. Therefore, even using a modem, a common "SMO(SolidWorks)→NMC→SMO(MDT)" exchange will take 0.577seconds (0.0734+0.257+0.247), and a common "SMO(MDT)→NMC→SMO(SolidWorks)" exchange takes 0.713seconds (0.0678+0.257+0.388). This data exchange rate can be considered as a real-time one.

At the same time, when comparing to those offline product data exchange schemes, the size of data exchanged in our approach is much smaller. Based on a model having 34 features, the size comparisons between the NMCs in XML format (NMC-XML) and the initial graphics exchange specification (IGES) file (text-based *.igs* file), the standard for the exchange of product data (STEP) file (text-based *.step* file), the MDT file (binary-based *.dwg* file), and the SolidWorks file (binary-based *.sldprt* file) are shown in Fig. 17. The differences are even much greater after these files are compressed by GNU zip with default compression level. NMC-XML performs better in both the compressed size and the compression ratio due to the higher percentage of descriptive tags than the other four. Note that the whole list of NMCs is only sent in the initialization phase. During the collaborative design, the NMCs are delivered progressively, so that the actual size of data transferred on the network is even less.

## 5.1 Limitations.

The integration-based approach presented in this paper has several limitations:

- First of all, the approach relies on the mapping between SMOs and NMCs. If no such correspondence is found, we need to either expand the set of NMCs or define the corresponding geometric operations for each NMC. For this reason, every time when the C++ API version of an integrated CAD system is updated, we need to upgrade the client add-on program of this CAD system to reflect the change.
- Secondly, the concurrency control method of our current implementation is relatively simple, where only one user is allowed to modify the product data in a specific time period. A more efficient way for the concurrent product development is to let multiple users work in their own portions of the product. In our future work, we are going to find a method to separate the whole product dataset into different critical sections, so that multiple *modifier*s can be assigned to different portions to work concurrently. For this issue, our current solution is to

subdivide the product dataset into several isolated subsets in advance; we can then start several collaborative design sessions for each of the portions.

- Furthermore, the current approach does not consider about the security problem. Every user in the collaborative design session can access every detail of the product data. This is not always allowed. Thus, we will further develop our current approach into a hierarchical structure in the future work, so that the product data is organized with layers and different users can only access and view the different authorized portions of product data in particular layers.
- In our current implementation, the synchronization for the operations on composite features or user-defined features is primal. Therefore more studies will be conducted in the areas about how to efficiently and effectively decompose them into basic features provided by each system.
- Another drawback of our approach is the method of verifying the results from SMOs because using mass properties is not an ideal solution. In some extreme cases, incorrect operations may also cause the same values of mass properties. We need to find some better methods to verify models. Our first thought is to adopt the implicit representation of product models.
- Lastly, only three CAD systems have been investigated and integrated, more systems (e.g., CATIA, PowerShape and NX, etc.) need to be considered in our future work.

Although the current implementation shows the above limitations, our contribution to the methodology for developing online collaborative design systems is significant – this will be summarized in the following conclusion section.

## 6 Conclusion

In this paper, an integration-based solution for a real-time online collaborative design platform is proposed. The platform developed by this method can consist of heterogeneous CAD systems. Different from the visualization-based approaches, product models are allowed to be constructed and modified from various sites using different client CAD systems synchronously. In addition, our approach is different from the homogeneous co-design systems because users on our proposed platform can manipulate product models by the CAD systems they used to. Our approach is based on a mechanism for the translation between *system modeling operation*s (SMO) and *neutral modeling command*s (NMC). Every operation given by a user on one site will be translated into an NMC and transferred through the network to all the other sites after being confirmed, and then the NMC are converted into corresponding SMOs on every other site. Since only commands but not model data are transmitted, the size of data under transmission is very limited. As a result, the real-time synchronization can be achieved with a standard bandwidth. The prototype implementation of the proposed environment proves that our approach can integrate various current popular commercial CAD systems into a real-time collaborative design environment. In summary, compared with the other existing collaborative design solutions in the literature, our approach contributes a novel method for integrating heterogeneous CAD systems into a collaborative design platform, which shows the following advantages:

- Not only the visualization but also the real-time manipulation of models is supported by our platform, which is the urgently requested function by industrial users;
- The creation and modification of models could be performed collaboratively online in real-time;
- Our collaborative design platform is based on an integration approach with heterogeneous CAD systems, so users on different sites can still manipulate models by using their familiar systems;
- The architecture of our platform is open. In other words, it provides the possibility for more and more client systems to be integrated incrementally.

In summary, the approach presented in this paper provides a new methodology to the research and development for online collaborative design systems with heterogeneous CAD systems.

## Acknowledgements

The work is supported by the NSF of China (No.60273057, No.60574061 and No.60021201), 973 Plan of China (2002CB312106) and the Trans-Century Training Programme Foundation for Talents by the Education Ministry of China. We also would like to thank Li Jie, Chen Xiang, Yang Youdong and Yang Fanqin for the part of the implementation they did.

## References

[1] Li, W.D., Lu, W.F., Fuh, J.Y.H., and Wong, Y.S., 2005, "Collaborative Computer-Aided Design - Research and Development Status," Computer-Aided Design, **37**(9), pp. 931-940.
[2] SolidWorks eDrawing™, http://www.solidworks.com/edrawings/
[3] Qiu, Z.M., Wong, Y.S., Fuh, J.Y.H., Chen, Y.P., Zhou, Z.D., Li, W.D., and Lu, Y.Q., 2004, "Geometric Model Simplification for Distributed CAD," Computer-Aided Design, **36**(9), pp. 809-819.
[4] Wu, D., and Sarma, R., 2004, "The Incremental Editing of Faceted Models in An Integrated Design Environment," Computer-Aided Design, **36**(9), pp. 821-833.
[5] Li, J., Gao, S.M., and Zhou, X., 2003, "Direct Incremental Transmission of Boundary Representation," *Proc. 8th ACM Symposium on Solid Modeling and Applications*, ACM, New York, USA, pp. 298-303.
[6] Cimmetry Systems Autovue™, http://www.cimmetry.com
[7] ConceptWorks™, http://www.realitywave.com
[8] Autodesk Streamline™, http://www.autodesk.co.uk/streamline/
[9] Alibre Design™, http://www.alibre.com
[10] OneSpace™, http://www.cocreate.com
[11] Bidarra, R., Kranendonk, N., Noort, A., and Bronsvoort, W.F., 2002, "A Collaborative Framework for Integrated Part and Assembly Modeling," Transactions of the ASME - Journal of Computing and Information Science in Engineering, **2**(4), pp. 256-264.
[12] Wang, F., and Wright, P., 1998, "Internet-Based Design and Manufacturing on An Open Architecture Machine Center," *Japan-USA Symposium on Flexible Automation*, Otsu, Japan, pp. 221-228.
[13] CollabCAD™, http://www.collabcad.com
[14] IX Design™, http://www.impactxoft.com/products/design.asp
[15] Tay, F.E.H., and Roy, A., 2003, "CyberCAD: A Collaborative Approach in 3D-CAD Technology in A Multimedia-Supported Environment," Computers in Industry, **52**(2), pp. 127-145.
[16] Choi, G.-H., Mun, D., and Han, S., 2002, "Exchange of CAD Part Models Based on the Macro-Parametric Approach," International Journal of CAD/CAM, **2**(1), pp. 13-21.
[17] Mun, D., Han, S., Kim, J., and Oh, Y., 2003, "A Set of Standard Modeling Commands for the History-Based Parametric Approach," Computer-Aided Design, **35**(13), pp. 1171-1179.
[18] ASPire3d, http://www.aspire3d.com
[19] Proficiency Collaboration Gateway™, http://www.proficiency.com
[20] Theorem Solutions, http://www.theorem.co.uk
[21] Acc-u-Trans™, http://www.translationtech.com
[22] Rappoport, A., 2003, "An Architecture for Universal CAD Data Exchange," *Proc. 8th ACM Symposium on Solid Modeling and Applications*, ACM, New York, USA, pp. 266-269.
[23] Li, W.D., Ong, S.K., Fuh, J.Y.H., Wong, Y.S., Lu, Y.Q., and Nee, A.Y.C., 2004, "Feature-Based Design in A Distributed and Collaborative Environment," Computer-Aided Design, **36**(9), pp. 775-797.
[24] Chen, X., Li, M., and Gao, S.M., 2005, "A Web Services Based Platform for Exchange of Procedural CAD Models," *Proc. 9th International Conference on Computer Supported Cooperative Work in Design*, IEEE, Piscataway, NJ, USA, **1**, pp. 605-610.
[25] Winsoft, http://www.winsoft.sk
[26] Microsoft DirectPlay, http://www.microsoft.com
[27] ACE™, http://www.cs.wustl.edu/~schmidt/ACE.html
[28] SolidWorks™, http://www.solidworks.com
[29] Autodesk Mechanical Desktop™, http://www.autodesk.com
[30] Pro/ENGINEER™, http://www.ptc.com
[31] CATIA™, http://www.3ds.com/products-solutions/plm-solutions/catia/
[32] NX™, http://www.ugs.com/products/nx/
[33] PowerShape™, http://www.delcam.co.uk
[34] Rappoport, A., Spitz, S., and Etzion, M., 2005, "One-Dimensional Selections for Feature-Based Data Exchange," *Proc. 2005 ACM Symposium on Solid and Physical Modeling*, ACM, New York, USA, pp. 125-134.