Journal of Computational Design and Engineering, Vol. 1, No. 2 (2014) 103~115
www.jcde.org

A multi-user selective undo/redo approach for collaborative CAD systems

Yuan Cheng[1], Fazhi He[1,*], Bin Xu[1], Soonhung Han[2], Xiantao Cai[1] and Yilin Chen[1]

[1] *School of Computer Science and Technology, Wuhan University, Wuhan, China*
[2] *Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea*

(Manuscript Received September 16, 2013; Revised October 31, 2013; Accepted November 1, 2013)

Abstract

The engineering design process is a creative process, and the designers must repeatedly apply Undo/Redo operations to modify CAD models to explore new solutions. Undo/Redo has become one of most important functions in interactive graphics and CAD systems. Undo/Redo in a collaborative CAD system is also very helpful for collaborative awareness among a group of cooperative designers to eliminate misunderstanding and to recover from design error. However, Undo/Redo in a collaborative CAD system is much more complicated. This is because a single erroneous operation is propagated to other remote sites, and operations are interleaved at different sites. This paper presents a multi-user selective Undo/Redo approach in full distributed collaborative CAD systems. We use site ID and State Vectors to locate the Undo/Redo target at each site. By analyzing the composition of the complex CAD model, a tree-like structure called Feature Combination Hierarchy is presented to describe the decomposition of a CAD model. Based on this structure, the dependency relationship among features is clarified. B-Rep re-evaluation is simplified with the assistance of the Feature Combination Hierarchy. It can be proven that the proposed Undo/Redo approach satisfies the intention preservation and consistency maintenance correctness criteria for collaborative systems.

Keywords: Undo/Redo; Collaborative CAD; Intention preservation; Configuration management

1. Introduction

When exploring a new system for the first time, a user needs to execute and cancel operations repeatedly to fully understand each function. The idea that "any system with a complex interaction interface should offer Undo/Redo support" was proposed in the late 1980s [1]. The fact that "no one will doubt the importance of offering Undo/Redo function in interactive systems" was also pointed out [2].

Collaborative CAD systems are an important platform for geographically dispersed designers working together. As an important feature in collaborative systems, Undo/Redo can help in reducing errors and eliminating misunderstandings [3-4]. Several kinds of group Undo/Redo models have been proposed, and the selective Undo/Redo model is the one most adopted. Supporting Undo/Redo in a collaborative CAD system is considered to be more complex and technically challenging than in a single-user environment.

Artifacts in collaborative design have a complex structure. Features are combined to form the boundary representation of a CAD model. The creation of a new feature depends on the existing features. When an operation is chosen as the Undo target, operations that depend on it must be obtained. In this circumstance, the structure of the complex design artifacts should be clearly presented.

Basic correctness criteria for a generic collaborative environment should be satisfied, such as intention preservation and consistency maintenance. The intention of an Undo and Redo commands is to eliminate and re-create the feature created by a certain modeling operation. Operations are interleaved at different sites, and an error can be detected after a number of operations are performed after it. The Undo/Redo implication should be interpreted unambiguously.

As a continuation of our previous work [5- 7], a multi-user selective Undo/Redo method for a 3D collaborative CAD environment is proposed. Within our method, the Undo/Redo target can be uniquely identified at each site. We also studied the structure of the complex design artifacts, and a tree-like structure called a Feature Combination Hierarchy is constructed to present the decomposition of a CAD model. Based on this structure, the dependency relationship among features and operations is clarified. The boundary model must be refined each time an Undo or Redo is issued, and this can be

*Corresponding author. Tel.: +86 18986211761
E-mail address: fzhe@whu.edu.cn

doi: 10.7315/JCDE.2014.011

achieved with the assistance of the proposed structure.

This paper is extended on the basis of our conference work [8]. In this paper, the construction of the Feature Combination Hierarchy is presented in more detail (section 3), the Undo/Redo algorithm is optimized against our conference work (section 4), and; a more detailed example is given (section 5).

The remainder of this article is organized as follows. In Section 2, the basic concepts of Undo/Redo operations and research fields are introduced. In section 3, the feature combination hierarchy is introduced. In section 4, our Undo/Redo method for a 3D collaborative system is introduced. In section 5, test results of the proposed method are presented. Finally, a summary of major contributions are given in section 6.

2. Related works

An Undo/Redo model determines the number and sequence of the undoable operations. There have been many efforts in the research field of Undo/Redo models. The initial research of Undo/Redo models are in single-user environments [9-10]. The Undo/Redo models in single-user environments are classified into 4 categories: 1) the single-step Undo/Redo model [2], 2) the linear Undo/Redo model [3], 3) the US&R (Undo, Skip, Redo) model [11], and 4) the history Undo/Redo model [12]. However, in a multi-user collaborative editing system, operations are interleaved at different sites. The last operation at the local site is not necessarily the last executed operation at other sites. Therefore, the selective Undo/Redo model is the most adopted multi-user Undo/Redo model. Its flexibility is limited by the Undo/Redo scope; namely, local Undo/Redo and global Undo/Redo [11, 15- 17]. AnyUndo is a framework which separates an Undo policy from the Undo mechanism. It allows users to devise a single Undo algorithm to support both local and global Undo/Redo and multiple models mentioned above [4, 13]. Moreover, it allows for undoing any operation at any time. This idea has been extended in the literature [7, 14].

Table 1 gives a detailed description of the features of the selective Undo/Redo model and the representative prototype. Different application systems have different characteristics. There are also different methods in the Operation ID, correlativity analysis and processing.

Operation ID is required in every selective Undo/Redo model. The symbol "√" means the operation ID is adopted by the model, but no implementation method is described. Detailed implementation methods are given in the literature [2, 18, 19, 21].

The Multi-user Undo/Redo model from Abowd [2] and Choudhary [18] do not take correlativity among operations into consideration, but the Selective Undo model in GINA [19] and Any Undo Model [4] do. However, in the selective Undo model, an operation cannot be undone or redone when it has a relationship with operations executed afterwards. In the last two models, operation parameters are transferred when making a Redo operation. When confronting unresolved conflicts, they adopt a multi-version method as a solution. The Multi-level model [20] and cascading selective Undo model [21] are designed for the single-user environment. The solution for the correlativity among operations is based on the task ID and task correlativity. That is, when undoing an operation, all tasks related to it are undone. Edwards's model supports the Redo, while Cass's model does not.

There has been little research work in collaborative CAD systems in the field of multi-user Undo/Redo, to the best of our knowledge. There are still defects in our previous research on multi-user Undo/Redo solutions [7]. At first, when clarifying the dependency relationship among operations, every attached feature is checked to see if it references the topological entities created by the Undo target. This will cause low efficiency. Secondly, model restoration is done by storing the model state after the execution of every modeling operation. The effect of the Undo target is eliminated by obtaining the model state at its execution first and re-executing all the valid operations. If the Undo target is at the front end of the history buffer and there are fewer operations that depend on it, the restoration efficiency will be very low.

Yet, there are still two other ways for model restoration [23]. One is by re-executing all the valid operations in the

Table1. Undo/Redo implementation in multi-user collaborative systems.

Model	Multi-user Undo	Selective Undo	Selective Undo	Any Undo	Multi-level Undo	Cascading Selective Undo
Literature	[18]	[11, 19]	[15]	[4]	[20]	[21]
Operation Location	User ID	√	User＋ Object ID	√	√	Task ID
Dependency Process	×	×	√	√	√	√
Object	Data Record	Text	2D Graphic	Text	Strokes	Slides
Prototype	Suite	DistEdit	GINA	Web-based REDUCE	Flatland White-board	Little-JIL-based Application

history buffer. Despite huge computational workload, it is still feasible because of hardware speed acceleration. The second way is by storing interim geometry models or incremental values between neighboring modeling operations. This method was adopted by Wang from the National CAD Engineering Center, HUST [22].

3. Feature combination hierarchy in 3D collaborative CAD systems

In a collaborative CAD system, functions and data are replicated at every site. A modeling operation is executed at the local site immediately before it is sent to other remote sites. A CAD model is a complex design artifact combined by features created by collaborative sites. The feature modeling operations that aid in the designing of complex artifacts consist of geometric operations and Boolean operations. Geometric operations choose topological entities from an existing solid model as the operation target. An example would be to choose a topological edge to fillet. There are three types of Boolean operations, namely Union, Subtraction and Intersection. By executing Boolean operations, a new solid is generated from two existing solids. Typical examples include protrusion and hole-attachment operations. Although the history buffer can honestly record the arriving sequence of all modeling operations at a collaborative site, it is still inadequate to represent the structure of the design artifact and interdependency among features. We use the boundary representation of ACIS for representing the shape of the CAD model. In ACIS, an attribute is attached to entities to describe their system-defined or user-defined characteristics. Given this, every entity created by a certain modeling operation is attached with a CREATE_ATTRIB in the form of (Create_SiteID, Create_SEQ) as auxiliary information of its ID. Therefore, whenever a topological entity is located, its generation operation can be obtained no matter which persistent naming mechanism is employed. Details of the workflow in a collaborative CAD system are presented in subsection 4.1. In this section, we study the structure of the complex design artifact, and the Feature Combination Hierarchy data structure, denoted as FCH for brevity, is introduced.

3.1 A brief view of the feature combination hierarchy

An FCH is a tree-like data structure to represent how 3D objects are combined and their relations in a CAD model. Objects, also called sub-configurations, created during the collaborative design process are categorized as primitive objects and composite objects. A primitive object cannot be logically decomposed into any primitive objects or constituents. Features created by inter-dependent modeling operations can be combined into composite objects using Boolean operations. It can also be decomposed into its component parts which in turn may be decomposed recursively so as to be addressed separately.

The CAD model created in a collaborative CAD system is constructed in a manner that a base feature is created firstly and other features are attaching to it incrementally. Based on this, a CAD model is decomposed into a primitive object representing the constructive base feature and several sub-configurations depending on the base feature. The special primitive object delegating the constructive base feature of a sub-configuration is called Base Object. There two types of composite objects: the complex CAD model and sub-configurations built intermediately by combining several inter-dependent features. Figure 1 illustrates how a CAD model, called PART in this example, is decomposed into different sub-configurations. The BaseCylinder feature is the constructive base feature of the PART model. So, the corresponding BaseCylinder object is noted as the Base Object of the PART object. The CylinderChamfer sub-configuration is a primitive object even though it is constructed by two modeling operations. That means a cylinder needs to be created first, and the edge of the cylinder's top face is filleted afterwards. The CylinderUnion sub-configuration is a composite object which can be decomposed further. CylinderBoss1 is the base object of the CylinderUnion sub-configuration.

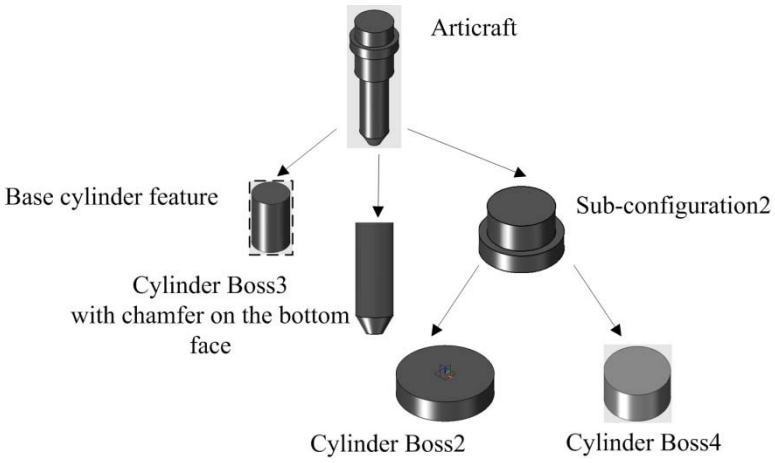


Figure 1. An example of complex artifact decomposition.

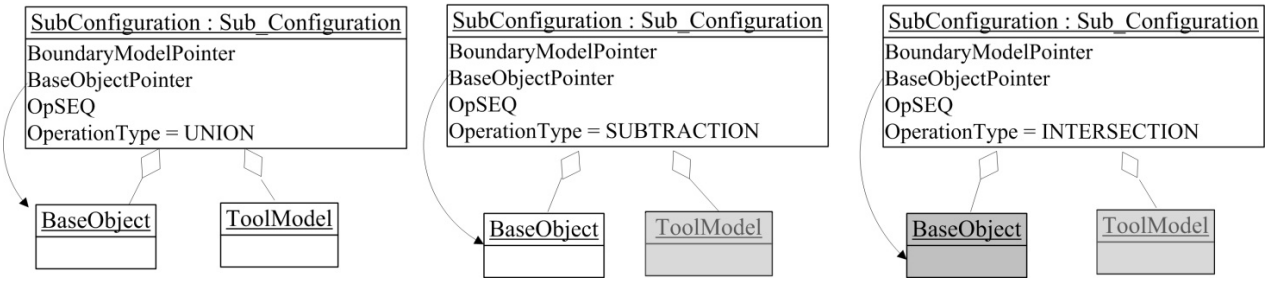
The tree-like data structure and main characteristics are summarized as follows:

1) The root of the hierarchy is a composite object delegating the complex design artifact. A class Artifact is declared to represent the root. The root node has several *branches*. Each *branch* is a decomposed sub-configuration from the complex design artifact. The first *branch* is always the constructive base feature of the part. The other *branches* are sub-configurations depending on the base feature. The root owns pointers to its branches. Different *branches* are in a non-interacting relationship.
2) The object that an intermediate node in the hierarchy delegates can be divided into three types: a) a composite object created by a Boolean union operation which can be decomposed further; b) a primitive object created by a Boolean subtraction or intersection operation; and c) a primitive object created by executing geometric operations.
3) The leaf node is an additive feature volumetrically added onto the complex design artifact, or a subtractive feature volumetrically removed from the artifact.
4) In an FCH, a composite object is always created by a Boolean Union operation. It should be decomposed in the way the root is decomposed. However, as far as a primitive object is concerned, its evolving process is also clarified if it is created by a number of modeling operations. This will be illustrated in subsection 3.2.
5) Given a sub-configuration SC decomposed into sub-configurations SC_1 and SC_2, The Base Object of SC is also the Base Object of SC_1 and SC_2.
6) During the evolving process of a part, the part is updated after the execution of every modeling operation. Therefore, the corresponding FCH is updated as well. With the consistency maintenance mechanism of our previous work, the CAD models at every site are consistent. Therefore, the FCHs at every site are also consistent.

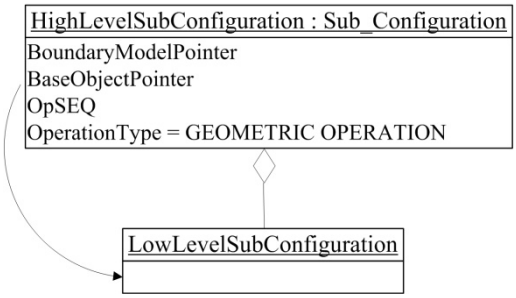
3.2 Basic feature modeling operation representation

Separate classes are declared for the ultimate CAD model and primitive/composite objects separately, as illustrated below:

```
Struct
{
    long int BoundaryModelPointer;
    // Pointer to the boundary model of the sub-
    // configuration

    long int BaseObjectPointer;
    // Pointer to the base object of the sub-configuration

    int   OpSEQ;
    // The sequence number of modeling operation
    // creating this sub-configuration from its previous
    // version

    CString   OperationType;
```



(a)



(b)

Figure 2. Decomposition of the object created by different types of modeling operation: (a) decomposition of the object created by Boolean operations, (b) decomposition of the object created by geometric operations.

```
    // A geometric operation or Boolean Union,
    // Subtraction or Intersection operation

    int    SubConf_No;
    // The branch the sub-configuration is in

    std::list<Sub_Configuration>m_listSubConfPointer;
    // Pointers to the decomposed sub-configurations if
    // a composite object or empty if a primitive object

    Sub_Configuration *next;
    // Pointer to the evolved version of this sub-
    // configuration after executing a consequent
    // modeling operation

} Sub_Configuration ;

Struct
{
    long int BoundaryModelPointer;
    // Pointer to the boundary model of the sub-
    // configuration

    long int BaseFeaturePointer;
    // Pointer to the base feature of the part

    std::list<Sub_Configuration> m_listSubConfPointer;
    // Pointers to the decomposed sub-configurations

} Artifact;
```

Based on the two classes, the decomposition of an object obtained by executing different types of feature modeling operations are illustrated in Figure 2. The node at the higher level delegates the renewed sub-configuration with whichever kind of feature modeling operation is executed. Figure 2(a) illustrates the decomposition of an object obtained by executing Boolean union, subtraction and intersection. The object obtained after executing a Boolean union operation is a composite object that is decomposed into a Base Object and an additive feature. The object obtained after executing a Boolean subtraction or intersection operation is a primitive object. However, the primitive object obtained is also illustrated. The grey-shadowed node means the feature is volumetrically removed from the sub-configuration. Figure 2(b) illustrates the evolving process of a primitive object created by the geometric operation.

4. Undo/Redo method in replicated collaborative modeling systems

Before the Undo/Redo method is proposed, it is necessary to discuss the consistency maintenance mechanism we adopted in our collaborative CAD system. A collaborative CAD system requires the execution of a modeling operation to satisfy causality preservation and intention preservation. All replicas reach the state of convergence at the end of a collaborative task. In order to identify the sequencing of simultaneous operations issued by collaborative sites in a replicated collaborative CAD system, a timestamp ordering technique based on the Lamport State Vector [25] is employed. A State Vector is an N component vector, where N represents the total number of all the collaborative sites, and each site has a unique ID ranging from 0 to N-1. Every site keeps a State Vector where the i-th component indicates how many operations from site i have been executed at the site. Two State Vectors, SV_i and SV_j, are compared in a way that: 1) $SV_i = SV_j$ *iff* each element of SV_i is equal to the corresponding element in SV_j; 2) $SV_i < SV_j$ *iff* each element of SV_i is equal to or less than the corresponding element in SV_j and at least one component of SV_i is less than that in SV_j; 3) $SV_i > SV_j$ *iff* each element of SV_i is greater than the corresponding element in SV_j. Whenever a modeling operation is sent to the remote sites, a State Vector at its generation moment is attached to it. An operation can only be executed in the causality-ready condition, such as when its State Vector is *not greater* than the State Vector kept at that remote site. The goal of intention preservation is reached by adopting an optimistic serialization concurrency control method proposed by our research group [26, 27].

When a feature is eliminated from a boundary model, features that depend on it are meaningless. The Undo/Redo method in a collaborative modeling environment needs to take the dependency relationship among operations into consideration. As illustrated in subsection 3.2, objects in a composite sub-configuration all depend on its Base Object. Even a primitive object is constructed on the basis of some primitive object. When a modeling operation is chosen as the undo target, the first step is to check if the corresponding feature is the Base Object of some sub-configurations. Afterwards, all executed operations that depend on the undo target should be undone altogether.

When an undo command is issued, its intention is to eliminate the effects of one or more operations and restore to some previous state. The requirements include: 1) to correctly locate the undo target operation; and 2) to successfully eliminate the effects of the undo target and the operations that depend on it without affecting the non-interacting sub-configurations. When a Redo command is issued, its intention is to undo the most recent undo issued by the same user. In this section, we propose a local Undo/Redo method where the user can only undo operations issued by him/her from back to forth. This is more similar to the single-user environment. This method has a pre-condition that the collaborative sites have reached a state of convergence when an Undo command is invoked.

4.1 Locating Undo/Redo target

In order to satisfy intention preservation requirement 1, the Undo/Redo target should be correctly located. Due to network latency and the causality preservation based on State Vectors, the operation history at each collaborative site has the following characteristics:

1) Modeling operations from the same site are executed in the same order at all sites.
2) Modeling operations from different collaborative sites are interleaved and executed in different order at different sites.

Since the Undo target that a user is aiming at is not necessarily the last operation in the execution list at each site, when locating the Undo target, there are 2 aspects we should take into consideration: 1) locating the Undo target at the local site; and 2) locating the Undo target at remote sites.

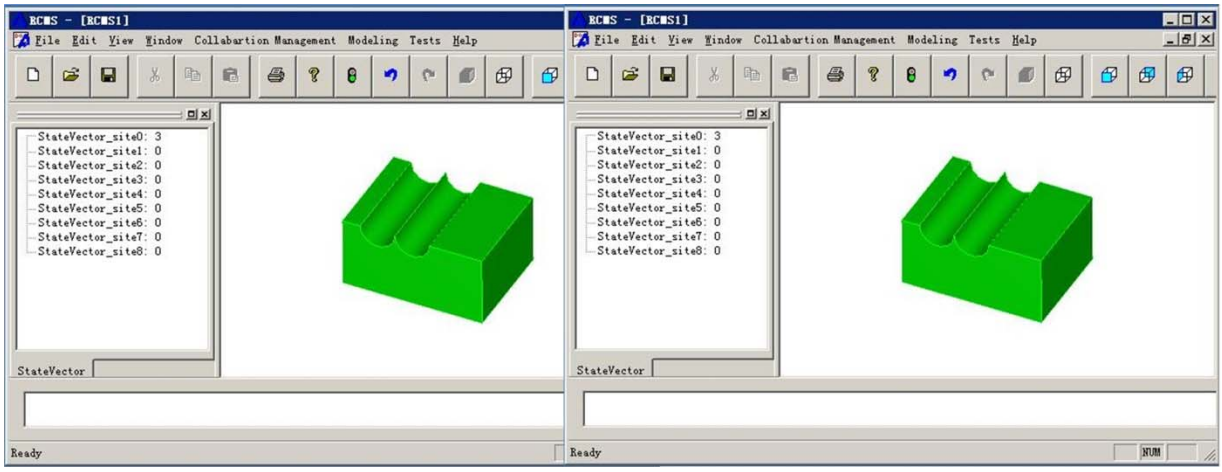
A modeling operation is executed immediately at its generation site immediately after it is issued. Thus, the Undo object merely exists in the execution list at the local site.

Consider a collaborative CAD system CS as an example. There are N collaborative sites in CS. $A_{i,j}$ ($0 \le j \le M-1$) means the modeling operation sent from site S_i, and M is the total number of operations. For S_i, all the operations it sends are put in its execution list $ExecuteList_i$ from $A_{i,0}$ to $A_{i,M}$ one by one.

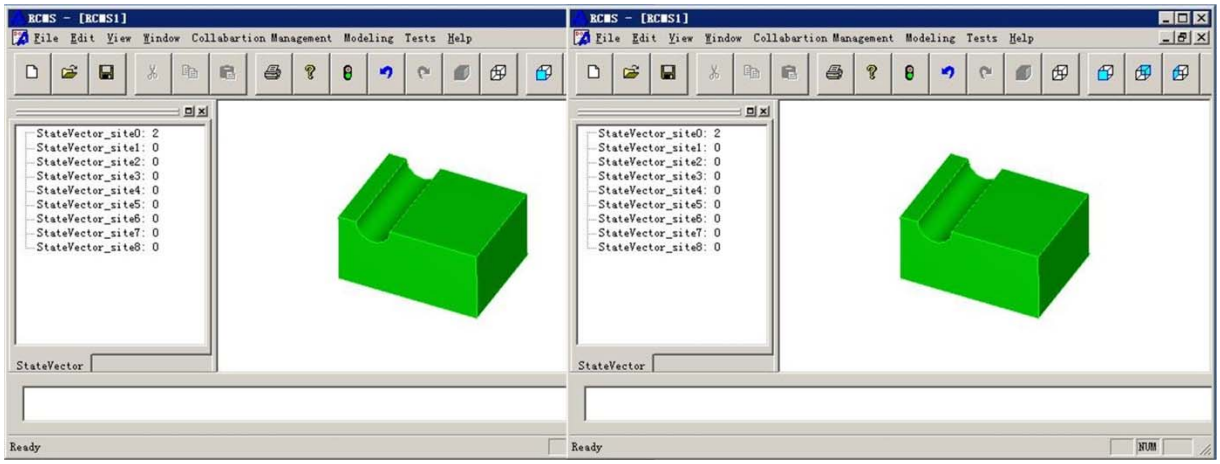
As soon as S_i issues an Undo command, it is easy to search through its $ExecuteList_i$, and the last operation satisfying A_i.siteId == i is the Undo target.

Given network latency and the summarized characteristic (2), the Undo target may exist in two lists at any remote site. One possible list is the execution list, and the other one is the waiting list keeping waiting operations that are not causality ready.

When some random remote site S_j receives the Undo request, the possible identified Undo target may not be the last executed operation. So, it can only be located in the $ExecuteList_j$ or $WaitList_j$. In order to locate the Undo target, the combination of site ID and operation's State Vector are adopted. An operation from S_i is sent to S_j in the format of A_i (SiteId, StateVector). When A_i arrives at S_j, $ExecuteList_j$ will be searched first. If no operation in this list satisfies O.siteId == i and O.StateVector==A_i.StateVector, then $WaitList_j$ will



(a)



(b)

Figure 3. Location of Undo object at remote site: (a) the initial state before applying the undo command, (b) the model state after applying the undo command.

Algorithm 1: Obtaining the dependency operation set of the Undo target.

```
Input: the identified Undo target operation Op, current FCH
Output: DOS(Op)
Op_Node = the object node of Op in FCH;
DOS(Op) = Empty;
i = Op_Node.SubConf_No;
root = root node of FCH;
SubConf_Node = root.m_listSubConfPointer[i];
Temp_Node = Op_Node.next;
while (Temp_Node != SubConf_Node)
    if (Temp_Node->BaseModelPointer == Op_Node)
        Temp_Node is put into DOS(Op);
        if (Temp_Node->BaseModelPointer != Op_Node)
            Temp_Base_Node = Temp_Node->BaseFeaturePointer;
            while (Temp_Base_Node != NULL )
                if (Temp_Base_Node->BaseFeaturePointer == Op_Node)
                    Temp_ Base_Node is put into DOS(Op);
                else
                    Temp_Base_Node=Temp_Base_Node->BaseFeaturePointer;
                endif
            endwhile
        endif
    endif
    Temp_Node = Temp_Node->next
endwhile
```

be searched in order to locate the Undo target.

Figure 3 shows an example of applying operation ID and State Vector to the Undo process. In the example, CubidSlot(1) is from site 1 and CubidSlot (2) is from site 2. The Undo command is issued by site 2.

4.2 Undo targets preservation

Algorithm 2: Undo method at a collaborative site$_i$.

```
Input: Undo request, history buffer HB_i at site_i, current FCH
Output: Re-evaluated geometry model
Op = the identified undo target in history buffer;
DOS(Op) = Op's dependency operation set;
Op_Node = the object node of Op in the FCH;
//obtaining the re-newed sub-configuration without Op's effect
i = Op_Node.SubConf_No;
root = root node of FCH ;
SubConf_Node = root.m_listSubConfPointer[i];
Base_Ob = the base object of the SubConf_Node;
Temp_Node = Base_Ob->Next;
Temp_SubConf = the boudary model Base_Ob delegates;
while (Temp_Node != SubConf_Node)
    if (Temp_Node is in DOS(Op || Temp_Node == Op))
        delete Temp_Node from FCH;
    else
        Temp_SubConf = the re-newed boundary model obtained by executing the feature modeling operation
                       saved in Temp_Node on Temp_SubConf;
        creating a new node according to Temp_SubConf and insert the node into FCH;
    endif
    Temp_Node = Temp_Node->Next;
endwhile
root.m_listSubConfPointer[i] = Temp_Node;
combine all the existing sub-configurations to obtain the re-evaluated boundary model;
```

Algorithm 3: Redo method at a collaborative site$_i$

```
Input: Redo request, history buffer HBi at sitei, current FCHi
Output: Re-evaluated geometry model
 1:  if (Op.ReferenceEntityList == NULL)
 2:          Ob_Base = new SubConfiguration();
 3:          Ob_Base.BaseModelPointer = NULL;
 4:          Ob_Base.BoundaryModelPointer = the physical address of base feature model
 5:          Ob_Base.ModelingCommandString = the advanced modeling command of Op;
 6:          Ob_Arti = new Artifact();   //creating the root node
 7:          Ob_Arti.BaseModelPointer = Ob_Base;
 8:  endif
 9:  if (Op.ReferenceEntityList ! = NULL)
10:          identify the operation creating the referenced topological entities;
11:          Temp_Node = the object nodes of the operation;
12:          Temp_Command_String = the advanced modeling command of the operation;
13:          Temp_Type_String = the operation type obtained by parsing Temp_Command_String;
14:          Ob_New = new SubConfiguration();
15:          Ob_New.BaseModelPointer = Temp_Node;
16:          Ob_New.BoundaryModelPointer = pointer points to the new subconfiguration;
17:          Ob_New.ModelingCommandString = the advanced modeling command;
18:          Ob_New.OperationType = Temp_Type_String;
19:          if (the topological entities of Op.ReferenceEntityLis are from the base feature)
20:                  i = m_listSubConfPointer.count() ;
21:                  Ob_Arti.the_i+1th_Configuration _Pointer = new SubConfiguration();
22:                  Ob_Arti. the_i+1th_Configuration _Pointer = Ob_New;
23:          else
24:                  Ob_Arti.This_SubConfigurationPointer = Ob.New;
25:          endif
26: endif
```

The intention of an Undo/Redo command is to eliminate/re-create the effect of a certain feature. This means the B-Rep re-evaluation is an unavoidable step during the Undo/Redo process. As far as intention preservation requirement 2 is concerned, whenever an Undo or Redo command is issued, only the *branch* that the corresponding feature is in is effected in an FCH, while the rest of the branches remain unchanged. Based on the preparations made in subsections 4.1 and 4.3, two algorithms are presented to introduce our Undo and Redo methods separately. Algorithm 2 illustrates the Undo implementation at a collaborative site and Algorithm 3 describes the Redo implementation at a collaborative site. The Redo method is proposed with the pre-condition that an operation undone because of a dependency relationship cannot be redone.

5. Implementation and test results

To demonstrate the effectiveness and feasibility of the proposed Undo/Redo solution, we have made several experiments within the prototype we built using ACIS 6.0 and Visual C++ 6.0. There are 3 collaborative sites, denoted as Site0, Site1 and Site2, involved in the experiments. The experiments are initiated with a collaborative modeling process illustrated in Figure 4.

During the collaborative modeling process, a feature modeling command is executed immediately after its generation at the local site. It is then sent to the other two remote sites for execution. How many operations are generated at a site and the execution sequence of all received operations are illustrated in Table 2.

Three Undo commands are issued and executed in the following way:

(Step 1) Site0 sends an Undo command to Undo the last operation it issued. The target operation O_4 is identified and the corresponding node denoted as Sub_Configuration4 in the site's feature combination hierarchy is obtained. The identified node represents the one object in the 4th sub-configuration, and DOS(O_4) is empty. Therefore, the sub-configuration is skipped during the re-evaluation process. Eventually, the other four sub-configurations are recomposed as illustrated in Figure 6.

Table 2. Operation generation and execution at each site.

Site ID	Operation generated at the site	Execution sequence of operations at the site
Site0	O_0:BaseBlock(0) O_1:Cylinder(0) O_4:PolygonExtrusion(0)	$O_0,O_1,O_4,O_2,O_5,O_3,O_6$
Site1	O_2:Cylinder(1) O_5:RoundHole(0)	$O_0,O_1,O_2,O_5,O_4,O_3,O_6$
Site2	O_3:PolygonExtrusion(1) O_6:Extrusion(0)	$O_0,O_2,O_1,O_3,O_4,O_5,O_6$

(Step 2) Site1 sends its second Undo command. The target operation O_2 is identified and the corresponding node, Cylinder2, is obtained in the current Feature Combination Hierarchy. Since $DOS(O_2) = O_6$, the 5th sub-configuration is skipped during the re-evaluation process. The result is illustrated in Figure 7.

(Step3) Site2 sends an Undo command to Undo the last operation it issued. The target operation O3 is identified and the corresponding node denoted as Sub_Configuration2 is obtained. The identified node represents the one object in the 2*th* sub-configuration branch, and DOS(O3) is empty. Therefore, the sub-configuration is skipped during the re-evaluation process. Eventually, the other two sub-configurations are recomposed as illustrated in Figure 8.

6. Conclusions

We proposed a selective multi-user Undo/Redo method in replicated collaborative 3D modeling systems. It identifies the Undo/Redo target operation uniquely at a local site and remote sites by site ID and a Lamport State Vector so as to preserve a user's Undo/Redo intention. With the assistance of a Feature Combination Hierarchy, the structure of the design artifact and feature relations are clarified. In this way, operations that depend on the Undo target can be easily obtained and will be undone altogether as well. Finally, the correctness of our algorithms was proven with experiments executed within a prototype that we built.

Acknowledgments

This paper is supported by the National Science Foundation of China (Grant no. 61070078).

References

[1] Yang Y. Undo support models. International Journal of Man-Machine Studies.1988; 28(5): 457-481.

[2] Abowd G, Dix AJ. Giving undo attention. Interacting with Computers.1998; 4(3): 317-342.

[3] Berlage T. A selective undo mechanism for graphical user interfaces based on command objects. ACM Transactions on Computer-Human Interaction (TOCHI). 1994; 1(3): 269-294.

[4] Sun CZ. Undo any operation at any time in group editors. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work; 2000; Philadelphia, PA; p.191-200.

[5] Jing SX, He FZ, Liu HJ. Collaborative naming for replicated collaborative solid modeling system. In: ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference; 2008 Aug 3-6; NewYork, NY; p. 141-150.

[6] He FZ, Jing SX. A naming and corresponding mechanism of topological entities for replicated collaborative solid modeling. 2008; P.R.China Patent. Application Number: 200810047976.9.

[7] Cheng Y, He FZ, Cai XT, Zhang DJ. Group undo/redo method in 3D collaborative modeling systems with performance evaluation. Journal of Network and Computer Applications. 2013; 36(6): 1512-1522.

[8] Cheng Y, He FZ. A multi-user selective undo/redo approach for collaborative CAD systems. In: Proceedings of the 2013 Asian Conference on Design and Digital Engineering (ACDDE 2013); 2013 Aug 12-14; Seoul, Korea; p. 593-603.

[9] Archer Jr JE, Conway R, Schneider FB. User recovery and reversal in interactive systems. ACM Transactions on Programming Languages and Systems. 1984; 6(1): 1-19.

[10] Kontogiannis T. A systems perspective of managing error recovery and tactical replanning of operating teams in safety critical domains. Journal of Safety Research. 2011; 42: 73-85.

[11] Prakash A, Knister MJ. Undoing actions in collaborative work: Framework and experience. Computer Science and Engineering Division. Dept. of Electrical Engineering and Computer Science. University of Michigan. 1994.

[12] Lanvin DF, Castnedo RL. Extending object-oriented languages with backward error recovery integrated support. Computer Languages, Systems & Structure. 2010; 36: 123-141.

[13] Sun CZ. Undo as concurrent inverse in group editors. ACM Transactions on Computer-Human Interaction. 2002; 9(4): 309-361.

[14] Weiss S. Logoot-undo: Distributed collaborative editing system on P2P networks. IEEE Transactions on Parallel and Distributed Systems. 2010; 21(8): 1162-1174.

[15] Young R, Whitington J. Using a knowledge analysis to predict conceptual errors in text-editor usage. In: SIGCHI Conference on Human Factors in Computing Systems: Empowering People; 1990 Apr 1-5; Seattle, WA; p.91-98.

[16] Prakash A, Knister MJ. Undoing actions in collaborative work. In: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work (CSCW '92); 1992; p. 273-280.

[17] Ressel M, Gunzenhauser R. Reducing the problems of group undo. In: International ACM SIGGROUP Conference on Supporting Group Work; 1999 Nov 14-17; Phoenix, AZ; p.131-139.

[18] Choudhary R, Dewan P. A general multi-user undo/redo model. In: Proceedings of European Conference on Computer Supported Cooperative Work; 1995 Sep 11-15; Stockholm, Sweden; p.231-246.

[19] Prakash A, Knister MJ. A framework for undoing actions in collaborative systems. ACM Transactions on Computer-Human Interaction. 1994; 1(4): 295-330.

[20] Edward WK, Igarashi T. A temporal model for multi-level undo and redo. In: ACM Symposium on User Interface Software and Technology; 2000 Nov 5-8; San Diego, CA; p. 31-40.

[21] Cass AG, Fernandes CST. Using task models for cascading selective undo. LNCS. 2007; 4385: 186-198

[22] Wang TY, Wu JJ. Research on undo/redo technology in CAD/CAM. Engineering Journal of Wuhan University. 1998; 31(3): 65-66.

[23] Bidarra R, Bronsvoort W. Semantic feature modeling. Computer Aided Design. 2000; 32(3): 201-225.

[24] Jing SX, He FZ, Liu HJ. A survey of persistent naming problem for topological entities. Journal of Computer Aided Design & Computer Graphics. 2007; 19(5): 545-552.

[25] Lamport L. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM. 1978; 21(7): 558-565.
[26] Jing SX, He FZ, Han SH. A method for topological entity correspondence in a replicated collaborative CAD system. Computers in Industry. 2009; 60(7): 467-475.
[27] Cheng Y, He FZ, Cai XT, Cheng Y. A method for object reference in collaborative modeling systems. Journal of Computer Research and Development. 2011; 48(11): 2031-2038.

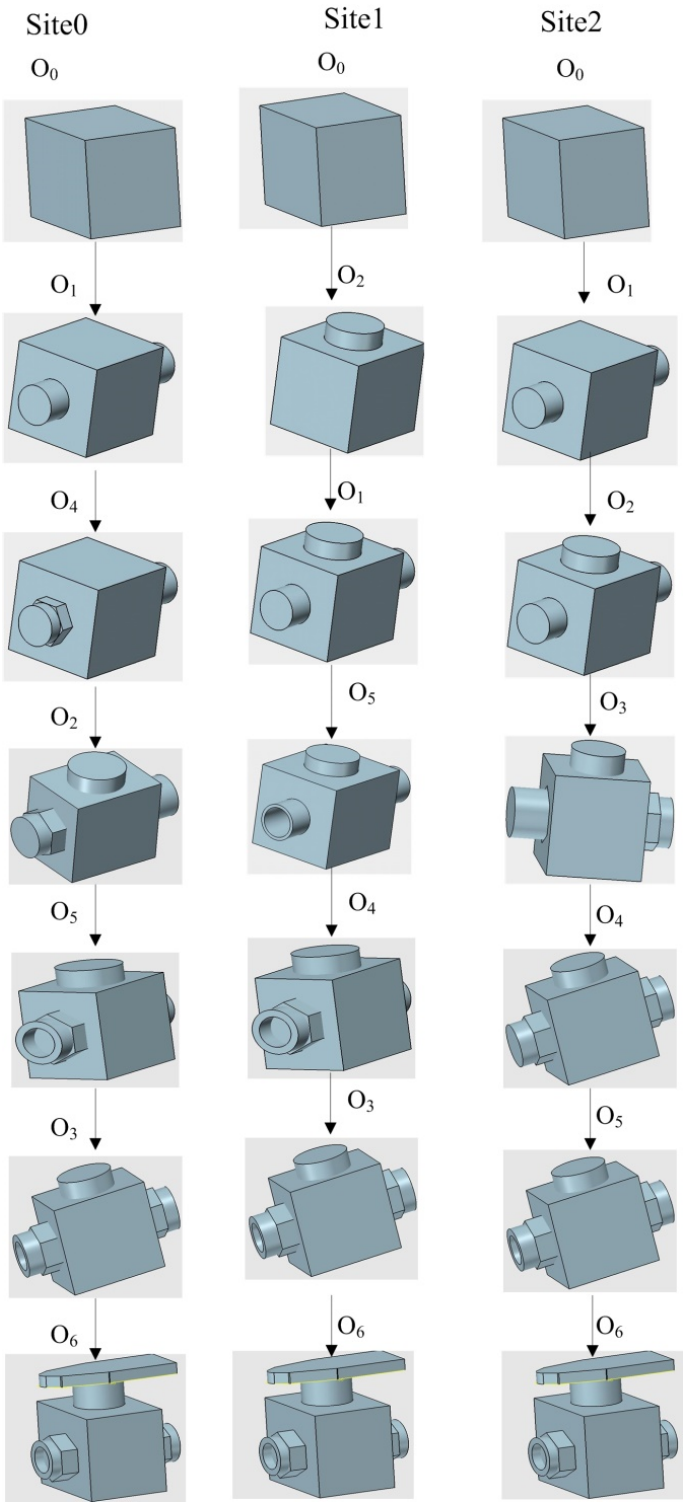


Figure 4. An example of the collaborative modeling process.

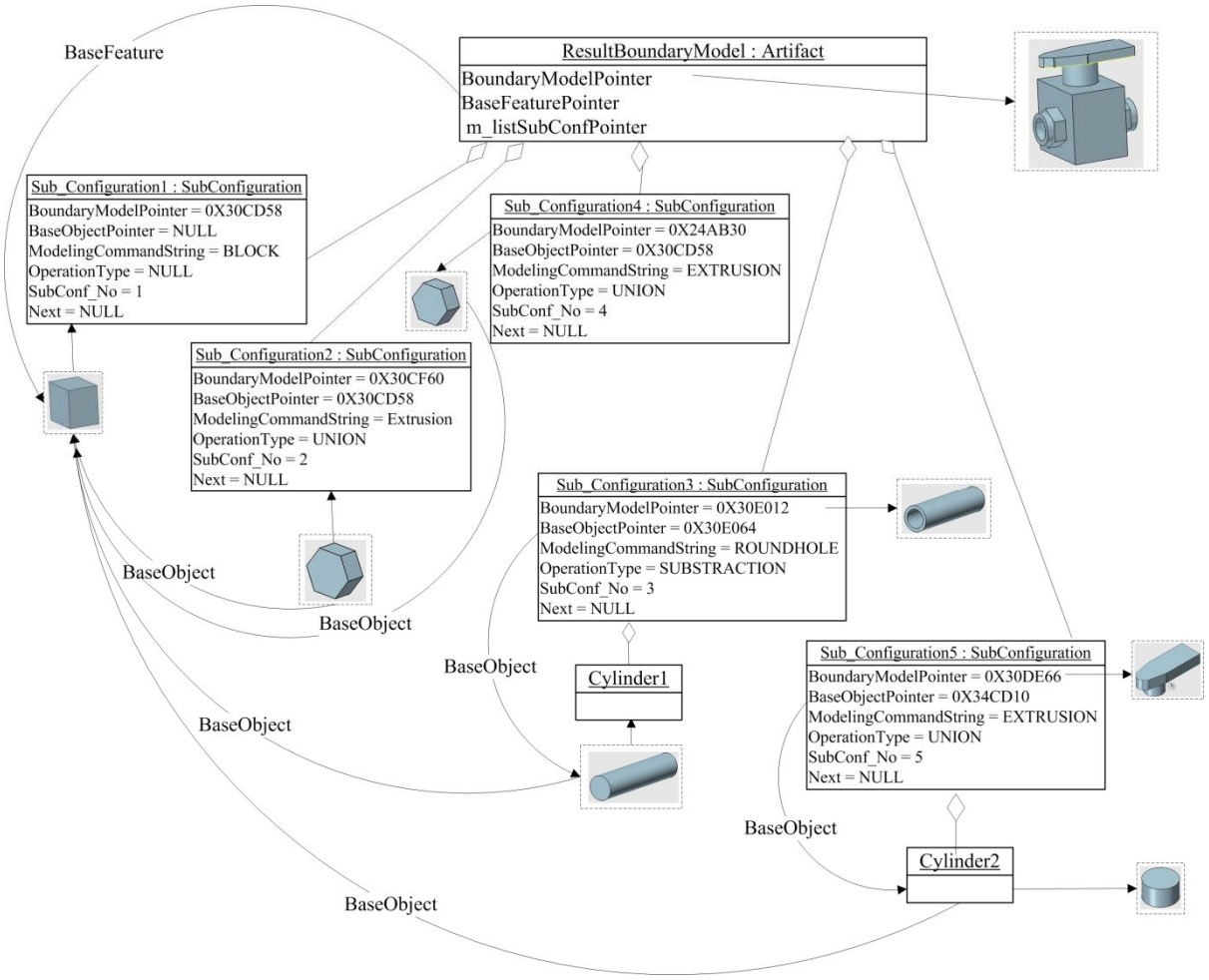


Figure 5. The feature combination hierarchy at each site after the collaborative modeling process.

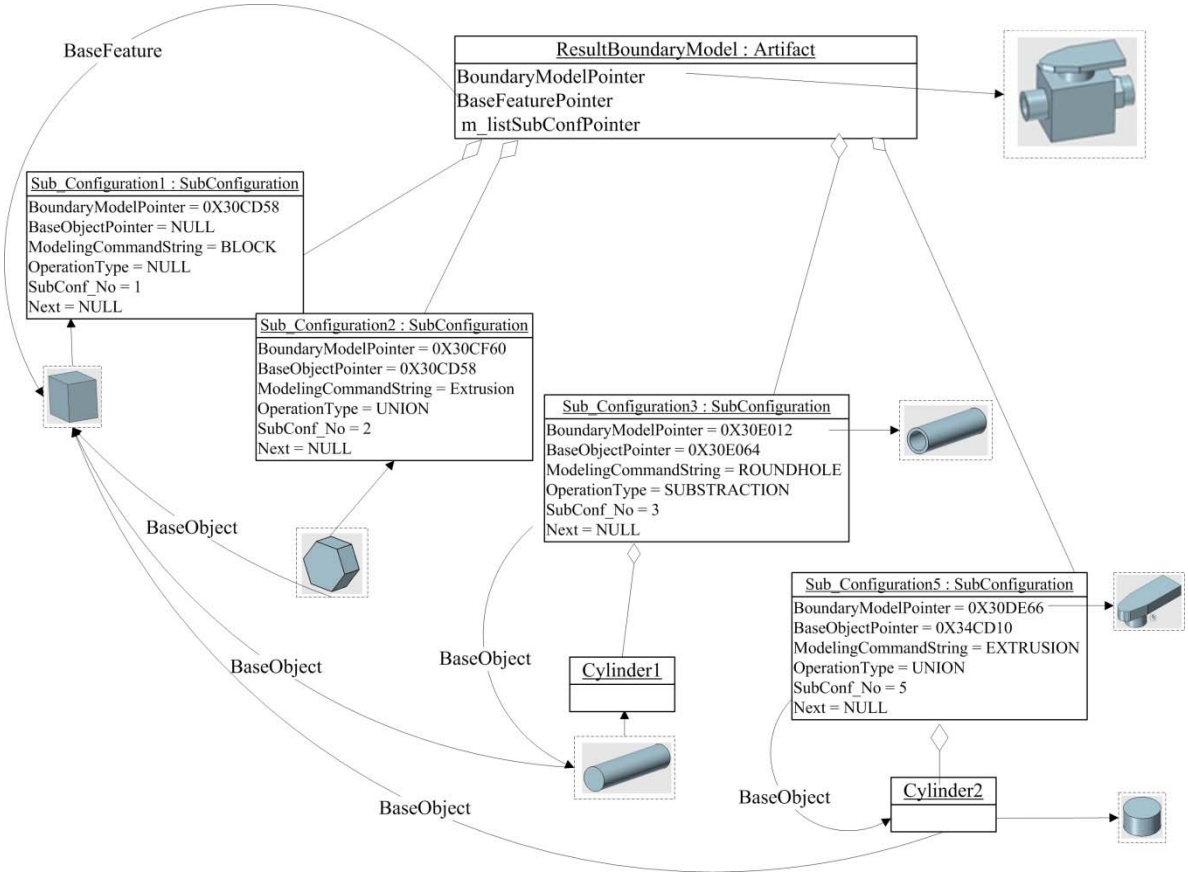


Figure 6. The Feature combination hierarchy after the first undo command.

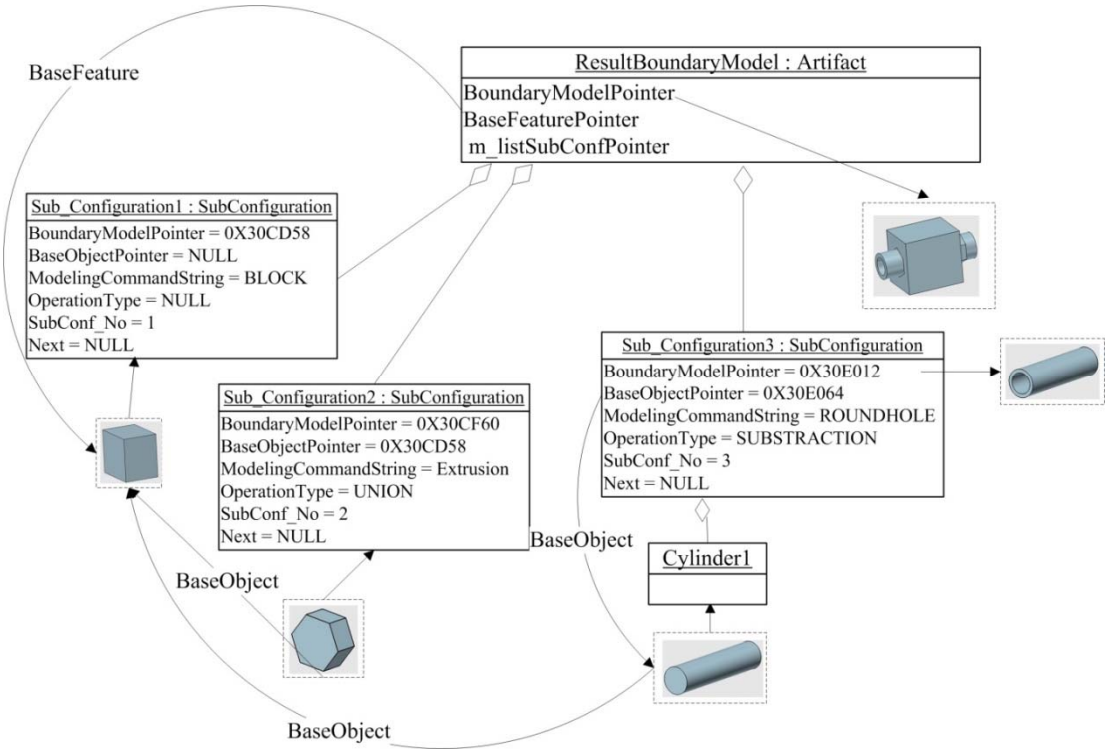


Figure 7. The feature combination hierarchy after the second Undo command.

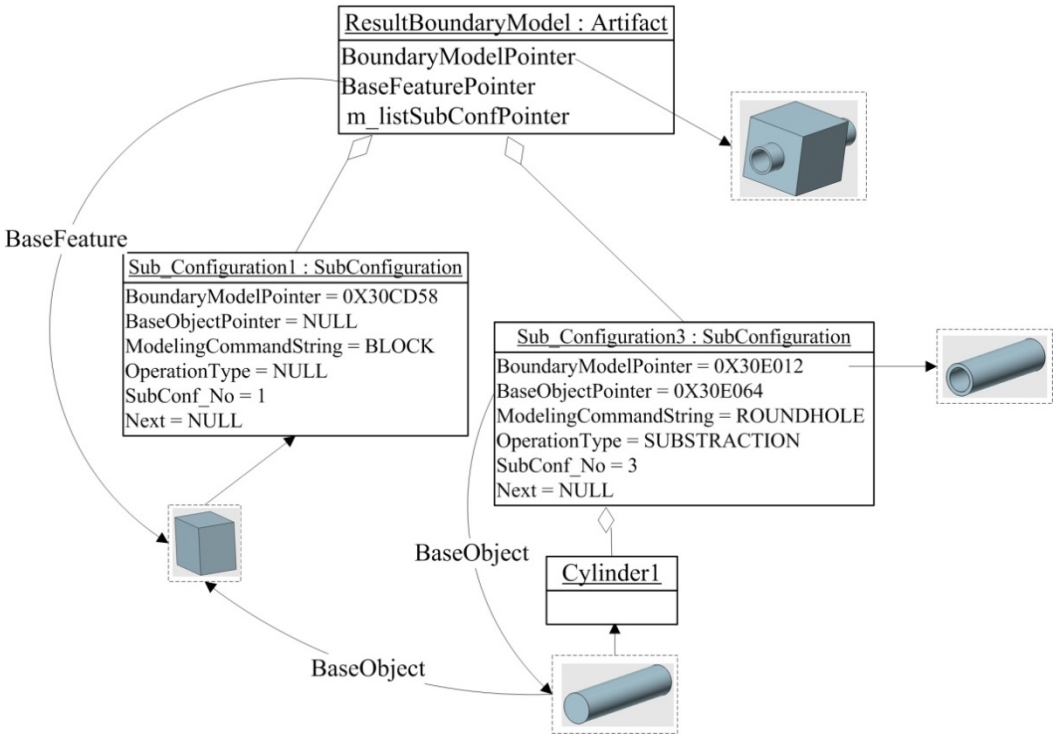


Figure 8. The feature combination hierarchy after the last Undo command.