Computers in Industry 59 (2008) 894–904

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Maintaining time and space consistencies in hybrid CAD environments: Framework and algorithms

Liping Gao, Bin Shao, Lin Zhu, Tun Lu *, Ning Gu

School of Computer Science, Fudan University, 220 Handan Road, Shanghai, PR China

ARTICLE INFO

Article history:
Available online 3 September 2008

Keywords:
CSCW
Consistency maintenance
Total Data Model
Heterogeneous collaboration

ABSTRACT

Based on the analysis of two types of collaborations during engineering design, namely time collaboration and space collaboration, this paper introduces a Total Data Model (TDM) centered framework to support the maintenance of time and space consistencies in hybrid CAD environments. By separating data and view, the proposed framework supports view-wandering between different perspectives so that designers can adjust their behaviors to avoid conflicts. The maintenance of time consistency is achieved by utilizing the Address Space Transformation (AST) algorithm. In order to adapt AST to new large-scale design environments, an effective garbage collection strategy is proposed. As for space collaboration, a conflict detection algorithm based on feature face is introduced. A prototype system has been developed to prove the validity, efficiency, and feasibility of the proposed framework.
© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Globalization has brought new requirements to manufacturing enterprises in order for them to survive in an increasingly competitive world: shorter lead time, better product quality, and lower cost [1]. In order to meet these requirements, various computer-aided design (CAD) software tools have been used to reduce cost and augment efficiency. Traditional CAD software is mainly used during the process of detail design. As an auxiliary drawing and entity-modeling tool, it plays an important role in new products development. However, with the further development of computer hardware and software, people began to realize that the use of computers to model a product and plot design drawings is not enough. CAD should be extended to the entire product lifecycle including product data management and product lifecycle management. During the process of product design, designers from different disciplines or different teams in the same discipline are constrained by concrete engineering rules. Yet these design processes are often simultaneous, leading to the outcome of design contradicted with engineering rules.

In order to prevent that, communication and collaboration tools or strategies must be provided for designers to view, search, locate and switch during design processes, so that designers from different departments may share product information by *view-wandering* between different perspectives. However, single-user application tools used by designers with different perspectives are often dissimilar, so that real-time *view-wandering* is almost impossible to achieve. For example, in CAD pipe designers may wish to switch from a certain pipe of the 3D model to its corresponding arts flow sketch of 2D model. Besides that, not all designers from the same department are experienced with a particular application (or the same application but not the same version). Each participant may have his/her own choice of application and may not be familiar with those of others'. Since every application has its specific operation method, user interface, display mode, etc., one may become a second-class citizen in the collaboration if one's familiar application is not selected for sharing (as Grudin presented in [2] and Li presented in [11]). Therefore, the best choice to solve the problem is to provide an integrated framework to support the real-time communication among hybrid applications while allowing users to use their familiar applications.

Furthermore, the collaboration in the engineering design field is different from that of traditional group editing. In the process of group editing, data consistency can be achieved by only considering the relationship of data operations, which can be done by using time-related consistency maintenance algorithms such as Lock [3,4], Serialization [5], OT [14,17,18], or AST [15]. While in engineering design, data consistency embodies two aspects: time consistency and space consistency, which cannot be achieved by the only-time-related data maintenance algorithms. Time consistency means that the final status of data copy of every site

* Corresponding author. Tel.: +86 21 5566 4465; fax: +86 21 6564 3502.
E-mail addresses: lipinggao@fudan.edu.cn (L. Gao), lutun@fudan.edu.cn (T. Lu), ninggu@fudan.edu.cn (N. Gu).

0166-3615/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2008.07.005

should be kept consistent while space consistency means that data objects from different departments should be correlated and constrained by each other so that conflicts between different models can be detected, which include location conflict, topology conflict, project attributes conflict, and project rules conflict (for example, the distance between two objects cannot satisfy the requirements of the design). Moreover, there are often tens of thousands objects being manipulated simultaneously by designers from different departments during the design process, so that it is critical to provide strategies to improve the algorithm efficiency.

On the basis of the above analysis, we can find that in order to realize the real-time collaboration and *view-wandering* between different perspectives, a mediated model must be provided to negotiate between the hybrid data model of different applications so that the execution of operations requested by all designers can be reflected on that model and designers can get their views which they are interested in by data projection through data proxy. Once the Total Data Model (TDM) is designed, time collaboration can be achieved by mapping the document structure to a linear one which is modeled by consistency maintenance algorithm while the space consistency can be achieved by executing real-time space conflict detection on that model and detection results can be projected to user view by highlighting technology. Centered on the TDM, this paper proposes a layered transformation model to support heterogeneous collaboration between different applications.

The rest of this paper is organized as follows. Section 2 reviews the related work of heterogeneous collaboration between hybrid applications. Section 3 introduces a framework of the TDM centered model and introduces the TDM based on ACIS model. Section 4 realizes time consistency maintenance by modifying and extending the AST algorithm, and Section 5 presents a space conflict detection process. Section 6 describes the experiment results. Section 7 summarizes the paper and gives an outlook for the future work.

2. Related work

There have been many efforts in the research field of heterogeneous collaboration. The DistEdit [6] provides a set of primitives to be added to editors to provide collaboration support, without dealing with distributed system issues. The primitives provided by toolkit are generic enough to support editors with different user interfaces. However, it does not adopt the replicated architecture but the client/server one so that it can only support loosely coupled collaboration but not tightly coupled one. Besides that, it needs to modify the source codes of the applications which may not always be achievable. Dewan and Sharma [7] have done a series of experiments in order to develop protocols and techniques as for the interoperation between multiple users and proposed to use a language-independent system such as CORBA to allow different systems to communicate with each other. However, this strategy can only be used between applications that use the compatible communication protocols. The PSI [8] combines a simple data model with active sharing mechanisms to construct an active sharing infrastructure to support information sharing across a number of distributed applications. The platform is realized as an active layer on top of an extended tuple space. The Placement Documents [9] is somewhat similar to PSI in that it also needs a data repository to store the edited documents and a middleware to monitor the change of the common data. Both PSI [8] and Placement Documents [9] perform well in loosely coupled environments. However, in tightly coupled environments such as the CAD engineering process, where the awareness and control object should be as small as possible, they do not work well. Besides that, because of the common data repository, the replicated architecture cannot be supported naturally. The Disciple framework [10] presents a framework for building collaborative applications for clients with different displays and processing capabilities. However, its main purpose is to reduce the influence of the slow and/or unreliable connections of different clients and its structure is also based on client/server architecture. Li and Li [11] proposed a new approach called intelligent collaboration transparency, which is based on application black box assumption to transform heterogeneous single-user applications into collaborative groupware by a series of event capture, event reduction, consistency maintenance, event reproduction, and event replay. This method is interesting and can solve the conversion of some relatively easy application with simple function set. However, to realize the powerful functions of commercial off-the-shelf applications by event capturing is almost impossible. Later, Li and Lu [12] modified the former method by replacing the event capture strategy with state get and set one. However, not all applications support the CTRL +A, CTRL +C, CTRL +V or CTRL+SHIFT+HOME, CTRL+SHIFT+END keyboard shortcuts. Since many mature commercial applications have provided plenty of APIs, Sun [13] first introduced the idea of transparent leverage of single-user applications to multi-user applications. However, in his framework, only homogeneous applications are supported, no strategy for solving the incompatibility of data model of different applications is introduced. On the other hand, all the above research efforts mainly aimed at the field of group editing.

There has been no research work in heterogeneous collaboration in the field of engineering design, to the best of our knowledge. So based on the transparent approach introduced by Sun [13], we propose to build an adapter layer to transform current different CAD applications into a heterogeneous groupware system which uses the Total Data Model as the core mechanism to realize the compatibility of different applications and support both time and space collaborations.

3. Transparent transformation framework

3.1. Analysis of typical work scenario

Generally speaking, during the engineering design, designers are often divided into small groups according to their design phases, work spaces, roles and task fields, etc. Constraints are mutative according to different roles of different designers. For example, tabling designers can view the design result of building designers but not modify it, and vice versa. Fig. 1 shows a representative scenario of design collaboration with designers using different tools.

Geographically distributed designers construct a peer-to-peer network in their work teams. Group networks are then connected to realize the effect of *view-wandering* (which means that designers can view the total model or any subset of the model, which combines all data coming from all disciplines). For example, tabling designers can switch from a certain pipeline in 3D view to its corresponding arts flow sketch in 2D view. Also, one or several design servers are often equipped to provide the functions of access management, task division, project evaluation, version control or other whole-project related work. Supervisors can observe the development of the whole project through the large display with high resolution.

3.2. Definitions

Definition 1 (Time consistency). A collaborative system is time consistent iff: (1) the shared document is identical at quiescence (Convergence); (2) as for every operation O_a and O_b, if O_a is causally preceded O_b, then at every site, O_a must be executed before O_b

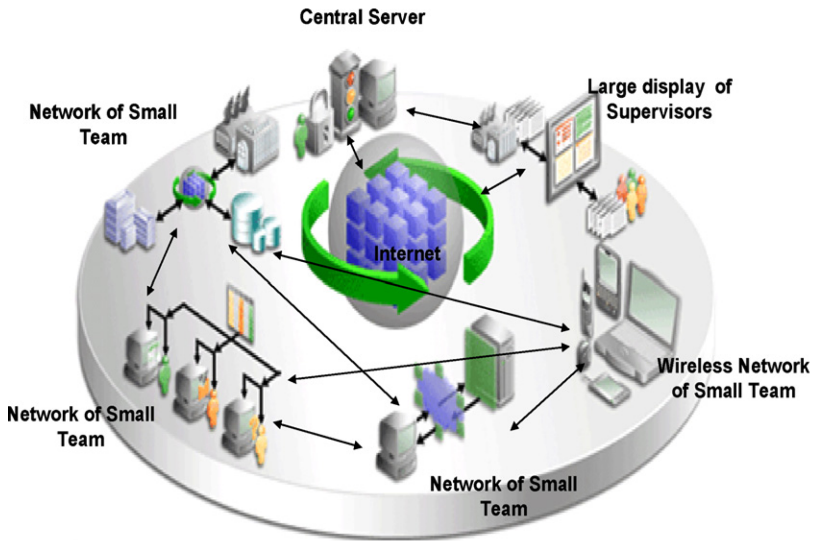


Fig. 1. Work scenario of the designers all over the world.

(Precedence); (3) for any operation O, the effects of executing O at all sites are the same as the intention of O, and the effect of executing O does not change the effects of independent operations (Intention) [14].

Definition 2 (Space consistency). A collaborative CAD system is space consistent iff: (1) the shared document is identical at quiescence (Convergence); (2) as for every entity E_1 and E_2, if E_1 and E_2 satisfy constraint set $R = \{R_1(E_1,E_2), R_2(E_1,E_2), \ldots, R_n(E_1,E_2)\}$ and after the execution of O with E_1 (without losing generalization, E_1 is taken as an example) as target object, the constraint set $R' = \{Ri_1(E_1,E_2), Ri_2(E_1,E_2), \ldots, Ri_n(E_1,E_2)\}$, where $i_1, i_2, \ldots, i_n \in 1, \ldots, n$, is no longer satisfied, user clue must be adopted to be presented to the end users or conflicts should be eliminated automatically (Constraint).

Definition 3 (BOS (Basic Operation Set)). Let $\widehat{A}$ denote all operations supported by application, then a set is called BOS of application A, notated as $\psi(A)$, iff: (1) $\forall o \in \psi(A),\ o \in \widehat{A}$; (2) $\forall O \in \widehat{A}$, O can be decomposed into a subset of $\psi(A)$; (3) $\forall o \in \psi(A)$, o cannot be decomposed into the elements of $\psi(A) - \{o\}$, where $\widehat{A}$ stands for all operations supported by application A.

Definition 4 (MCOS (Minimum Common Atomic-Operation Set)). MCOS of n applications indicated by $A_1, A_2, \ldots, A_n$ is defined as follows:

$$\bigcap_{1 \le i \le n} \psi(A_i) = \{x : x \in \psi(A_i), \quad 1 \le i \le n\}$$

Definition 5 (SOS (Super Operation Set)). SOS of n applications indicated by $A_1, A_2, \ldots, A_n$ is described as follows:

$$\bigcup_{1 \le i \le n} \psi(A_i) = \{x : \exists i \,\text{with}\, x \in \psi(A_i), \quad 1 \le i \le n\}$$

BOS of an application defines the minimal operation set of basic entities and every complex operation of applications can be decomposed into one or several BOS operations. Take AutoCAD as an example, the basic entity set is like that {LINE, PLINE, ARC, RECTRANGELE, BOX, CONE, CYLINDER, MTEXT, ...} while minimal operation is like that {CREATE, DELETE, ZOOM, PAN, ALIGH, BREAK, ...}. The MCOS of applications defines the common basic operations of different applications while the SOS defines the super operations of them (Fig. 2).

Definition 6 (OMM (Operation Mapping Matrix)). OMM of application A_i is defined as

$$\mathrm{OMM}(A_i) = \left\{ \begin{matrix} O_{i1,1} & O_{i1,2} & \cdots & O_{i1,n} \\ O_{i2,1} & O_{i2,2} & \cdots & O_{i2,n} \\ & \cdots & & \\ O_{im,1} & O_{im,2} & \cdots & O_{im,n} \end{matrix} \right\}$$

where n is the dimension of application sets $(A_1, A_2, \ldots, A_n)$ with A_k representing the kth application and $O_{ij,k}, j, k = 1, \ldots, n$ representing the corresponding operation of O_{ij} to application A_k and m is the operation number which application A_i supports. The mapping operation of an operation is defined as Definition 7.

Definition 7 (MO (Mapping Operation)). The MO of an operation O from application A to B is defined as: $\mathrm{MO}(O) = O_k$ if O_k satisfies any of the following three conditions:

(1) $O_k \supseteq O$ ^ ¬ exit O_m satisfies: $O_k \supseteq O_m \supseteq O$;
(2) $O_k \subseteq O$ ^ ¬ exit O_m satisfies: $O_k \subseteq O_m \subseteq O$;
(3) $O_k \supseteq O$ and $O_k \subseteq O$ but $O_k \cap O \neq \phi$ and there doesn't exit O_m satisfies that $|O \cap O_m| > |O \cap O_k|$.where $O = \{x: x \in \mathrm{SOS} \wedge x \in O\}$ and $O_k = \{y: y \in \mathrm{SOS} \wedge x \in O_k\}$

Definition 8 (TDM (Total Data Model)). A TDM of application $A_1, A_2, \ldots, A_n$ is defined by a structure of the form $\mathrm{TDM} = \langle \mathrm{DM}, O \rangle$ where

DM is a set of data models which include all the data models of application $A_1, \ldots, A_n$:
$O = \mathrm{SOS}\ (A_1, A_2, \ldots, A_n)$.
Total Data Model is defined to combine all the data models of different applications used in a specific field. It supports the operation on the SOS of heterogeneous applications.

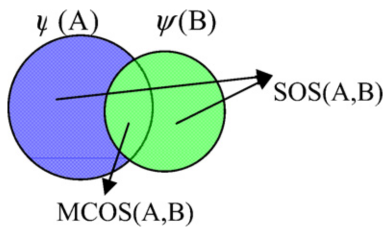


Fig. 2. The MCOS and SOS of application A and B.

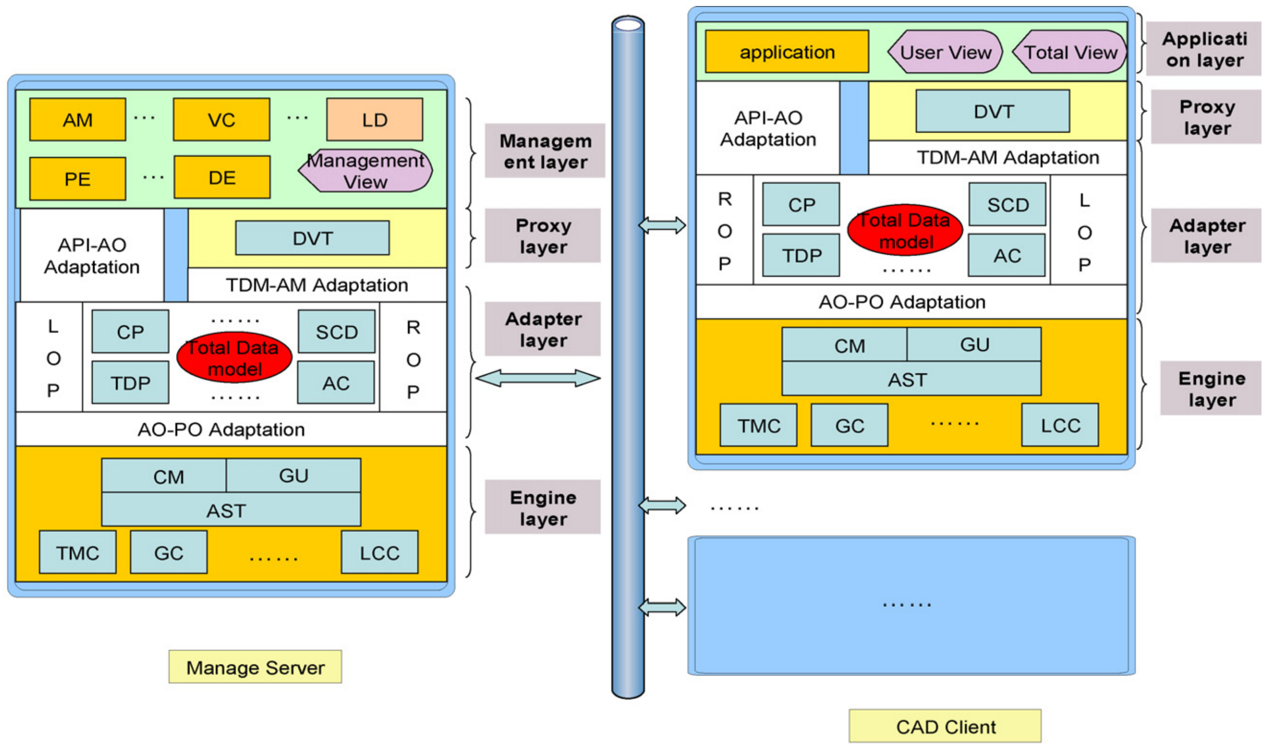
3.3. TDM centered layered architecture

The system allows multiple designers to edit the same TDM at the same time over the Internet as shown in Fig. 3. A collaborative editing session consists of multiple CAD client instances. Each site maintains and manipulates a local copy of the shared TDM document. Shared TDM documents and design rule databases can be located in any collaborating designer's local database with corresponding design right. This framework adopts the combination of C/S architecture and P2P architecture. The organization between server and client is achieved by using the architecture of C/S while that between different clients is achieved by architecture of peer-to-peer. Both management server and CAD client maintain the same copy of the TDM while the data proxy module will provide different views to different users. The implementation of client and server is realized by a layered architecture. Four layers of client are named as application layer, proxy layer, adapter layer and engine layer, while four layers of server are named as management layer, proxy layer, adapter layer and engine layer. More detailed description will be given in the following sections.

3.3.1. CAD client

CAD client achieves the transparent and convenient transition of single-user CAD applications to heterogeneous multi-user groupware by constructing proxy layer, adapter layer and engine layer, based on APIs of single-user applications. TDM combines all the information of all correlated fields (its detailed description will be given in Section 3.4), which is loaded from the module of version control of the management server when a CAD client first joins. Proxy layer is introduced to separate data model from user view, providing the convenience of multi-view wandering and visual clue of space conflict by highlighting techniques.

The operations from user interfaces will be captured by the LOP module of the adapter layer. First of all these operations will be tested whether it is allowed by the user right of the current designer, and then the allowed operation will be explained and decomposed into BOS operations. The BOS operations will be transmitted to the engine layer to be attached by timestamps. Then the Ops attached with timestamp are returned to the adapter layer and be broadcast to other clients and server by ROP module. Meanwhile, the ROP module of the adapter layer receives remote



AM: Access Management
PE: Project Evaluation
VC: Version Control
DE: Designer Evaluation
LD: Large Display
DVT: Data-View Transformation
TDM-AM Adaptation: Total Data Model-Application Model Adaptation
AO: Adapted Operation
API: Application Programming Interface
TDM: Total Data Model
LOP: Local Operation Processing
ROP: Remote Operation Processing
CP: Collaborative Policies
TDP: Total Data Processing
SCD: Space Conflict Detection
AC: Access Control
PO: Primitive Operation
CM: Consistency Maintenance
GU: Group Undo
AST: Address Space Transformation
TMC: Time Stamp Control
GC: Garbage Collection
LCC: Late-Comer Control

Fig. 3. TDM centered layered architecture to support heterogeneous collaboration.

operations, passes them to the engine layer to accomplish time consistency maintenance by the AST algorithm presented in Section 4. The operations are then recomposed and will be mapped into corresponding operations based on the local application by querying Operation Mapping Matrix. The API will be revoked to execute the recomposed operation and when it is executed, the TDM is also changed, which will trigger the SCD module to process space conflict detection using Procedure 4 discussed in Section 5. The check result will be highlighted on the users' view through the projection of data proxy.

The engine layer is responsible for judging the causal relationship of operations based on the attached timestamps and completing operation transformation, to guarantee the consistency of model copy. Traditional OT algorithm (such as dOPT, adopted, GOT, and LBT), Mu3D, dARB, treeOPT, etc., have some deficiency in solving the problem of copy consistency maintenance [15]. The AST algorithm [15], as the previous research result of our group, has some advantages in efficiency and maturity, so our work is based on this AST algorithm. Moreover, in order to promote space utilization, we introduce an efficient garbage collection mechanism to support the work of large amounts of designers with tens of thousands objects, which will be described in details in Section 4.3.

3.3.2. Management server

The project supervisors execute access management, task management, rule management, project evaluation, efficiency assessment as well as version management through the management layer of the server port. Access management is responsible for role creation, right deployment and user identification validation. Task deployment divides design region, design task and design flow, with a tree described workflow as the storage structure. Information about project's design quantity and design cost is maintained by the project evaluation module. Supervisors can get the information about workers' efficiency and quantity according to the real-time observation of the whole work group and the session logs analysis tools. Adjustment of the roles, tasks, and efficiency assessment of the engineers will be propagated to corresponding clients.

The function of other layers is similar to that of client port and will not be described in details in this paper.

3.4. TDM based on ACIS model

We use boundary representation or B-rep modeling of ACIS [16] to construct the 3D model of TDM, which provides comparatively full functionalities to cover most data models used in the CAD field and convenient tools to determine the relationship between different objects that are integrated in the data model. B-rep modeling allows TDM to integrate the three models of wireframe surfaces and solids so as to represent sophisticated objects. Geometry, topology and attribute constitute the three factors of TDM. They are all derived from the abstract ENTITY class, which defines the common data and methods such as save, insert, delete, and retrieve. The class hierarchy of the TDM based on ACIS [16] is shown in Fig. 4. The following sections give a brief description of the three factors: geometry, topology, and attribute. A more detailed description can be found in [16].

3.4.1. Geometry

Geometry defines generic geometrical elements such as curve, surface and entity. It defines the physical attributes of entities, without considering their relationship. For example, a POINT object holds a count of the number of vertices that refer to the point, as well as the coordinates of the point. Geometry is divided

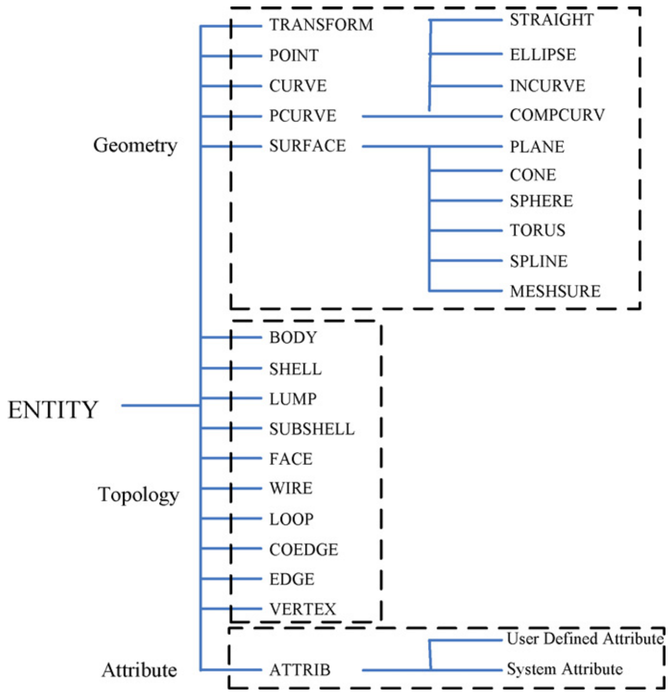


Fig. 4. Relationship of layered class of TDM [16].

into two layers: generic layer and higher layer, with the former defining the basic functionalities and attributes of the geometry and the latter defining the specific functionalities and attributes of the specific geometry type. Geometry class includes CURVE, LINE, ELLIPSE, PCURVE, SURFACE, CONE, etc.

3.4.2. Topology

Topology describes the relationship or the connectivity of different entities. Entities of TDM can be decomposed into different layers according to the topological relationship. The topology includes BODY, LUMP, SHELL, SUBSHELL, FACE, LOOP, WIRE, COEDGE, EDGE and VERTEX. Because of the definition of topology, TDM can describe sophisticated objects through the combination of simple objects. For example, a BOX is composed of one body, which includes six faces and each face is composed of one coedge with four vertexes. TDM integrates WIREFRAME, SURFACE and SOLID into unified data structure so that we can use the Boolean operation to judge the relationship of different objects conveniently.

3.4.3. Attribute

Attribute is attached to entities to describe system-defined or user-defined characters of them. One entity may have zero or more attributes with each of them representing comments, and design rules of a specific object. In our system we extend it to describe four constraint rules: size constraint, location constraint, edge constraint, and intersect constraint.

4. Time consistency maintenance

4.1. Basic AST

AST [15] is the abbreviation of address status transformation. It is a consistency maintenance algorithm used in group editors which was first proposed by Gu et al. in Group'05 [15]. As for consistency maintenance algorithm, two methods should be mentioned here: OT and AST. OT is a traditional consistence maintenance technique. In the OT method, due to the existence of concurrent operations, a remote operation should include or exclude some operations' effects [17,18] and find the correct operation position or area in the local site. To maintain the CCI model [18], in the past decade, the OT method has been gradually improved. However, the OT algorithm is still intricate and vulnerable in some situations [15]. The AST algorithm based on Mark & Retrace is a new method. Different from OT, it retraces the document's address space to the state at the time of the operation's generation. In that state, execution position of the operation can be discovered immediately. The retrace process does not affect the order of characters in the current document. It only needs to determine the position of a new operation. It can maintain the CCI model in replicated architecture and is easier to support group Undo [19,20] and the execution efficiency as it can reach complexity of $O(\log n)$.

The AST algorithm assumes a document as a linear structure composed of characters. Each character may have several operations targeting itself while each operation targets only one character. The operation together with its timestamp is saved in the character's linear node. Apart from the information of characters and corresponding operations, effective/ineffective mark information is also added to every node in the linear structure. The document structure is modeled as a linear structure and the control process is described in Procedure 1 where Doc_s is the linear document of site s, O is a remote operation and SV_s is the state vector at site s.

```
Procedure 1. Control-Algorithm ($\text{Doc}_s$, $O$), execute $O$ on $\text{Doc}_s$
1 Retracing ($\text{Doc}_s$, $\text{SV}_o$)
2 Execute the operation $O$
3 Attach the operation with its timestamp to the character mode
4 $\text{SV}_s[R] = \text{SV}_s[R] + 1$
5 Retracing ($\text{Doc}_s$, $\text{SV}_s$)
```

4.2. Applying AST to new environments

The AST algorithm is used in linear documents, in which precedence relationship of each node is maintained. However, objects in CAD documents are out-of-order and a newly inserted object can be added in anywhere of the object list. The parameters of every operation in CAD are different from those of group editors. For example, the insert operation in CAD is like "Circle(position (0, 0, 0), 5)" while that in group editor is like "Insert ("A", 1)" where "(0, 0, 0)" represents the absolute coordination while "1" represents the relative position of the newly inserted character "A". Therefore, in order to apply the AST algorithm, the random address space of CAD must first be projected into linear address space with basic operations and the absolute coordination will be encapsulated into a parameter string. The index value of the operation in the linear document is produced by the site which creates the operation and the index is the sequence number of all the operations created at the site. For example, if site i has sent out N create operations, then the index value of the newly created operation will be set to N (see Fig. 5).

Moreover, the AST algorithm does not provide an efficient garbage collection strategy to process the invalid node and redundant operation link of every node, which becomes a great obstacle in real engineering design process with huge amounts of objects to be handled. Besides that, the AST algorithm does not consider the role partition, which leads to named out-of-order operation of each other. The work responsible by a certain designer would not to be modified or deleted by other designers at some certain design phases.

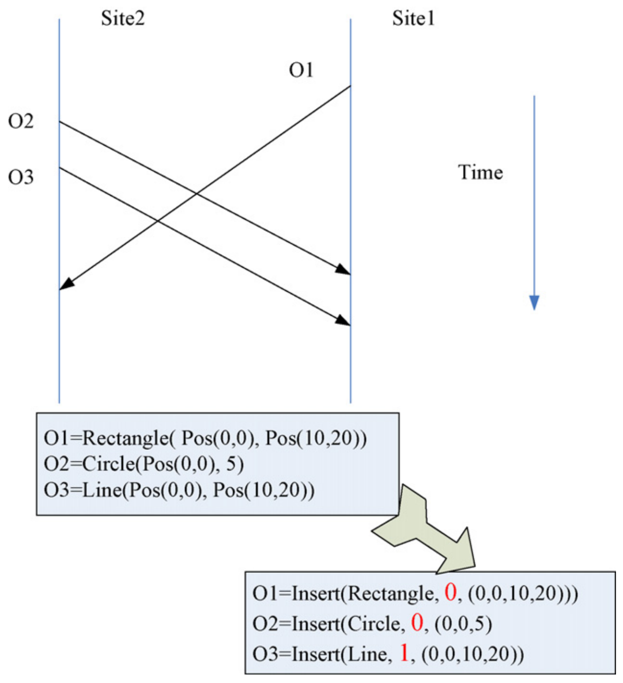

Site2
Site1
O1
O2
O3
Time
O1=Rectangle(Pos(0,0), Pos(10,20))
O2=Circle(Pos(0,0), 5)
O3=Line(Pos(0,0), Pos(10,20))
O1=Insert(Rectangle, 0, (0,0,10,20)))
O2=Insert(Circle, 0, (0,0,5)
O3=Insert(Line, 1, (0,0,10,20))

Fig. 5. Projection of random address space to linear address space.

4.3. Effective garbage collection strategy

During the design process, designers insert or delete objects frequently, which lead to large amounts of *forever invalid nodes*. These nodes make no effect to future work once its delete effect has been reflected at all sites. We consider that in fact the designer, who has deleted an object, will generally undo that delete operation in limited steps and that after having committed N_{undo} operations after that deletion, he would not like to undo it anymore because there already have been many midst operations between his current operation and that deletion operation or even if he wants to undo that deletion, he can remedy that by re-insert it.

The resolution is to introduce a garbage collection process to check the validity of the nodes of the document. Maintain a state vector table (SVT) at every site and suppose the state vector table of the sth site is SVT_s. Initially, $\text{SVT}_s[i][j] = 0$ for all $i, j \cdot 1, \ldots, N$ (N denotes the number of clients who joined in the design work). After executing an operation O from a remote site r, time stamped by SV_r, the maintenance of SVT is described by Procedure 2 and the new garbage collection method is described by Procedure 3.

```
Procedure 2. Maintain_SVT ($\text{SVT}_s$, $\text{SV}_r$), maintain the SVT
after the execution of Op timestamped by $\text{SV}_r$
1:  for $i = 1$ to $N$ do {//updates all the elements of the vector of
    remote site $r$.
2:      $\text{SVT}_s[r][i] = \text{SV}_r[i]$
3:  }
```

```
Procedure 3. Garbage_Collect ($\text{Doc}_s$, $\text{SVT}_s$), collect forever invalid
node on $\text{Doc}_s$
1: for all $i = 1, \ldots, N$ //get the minimum of the vectors to judge
   whether all the sites have executed the delete Op {
2:     $\text{MSV}_s[i] = \min(\text{SVT}_s[1][i], \text{SVT}_s[2][i], \ldots, \text{SVT}_s[N][i])$
3: }
4: for all character node $\text{CN}_i$ of linear structure of $\text{Doc}_s$ do {
```

```
5:   Consider the last $O$ of $CN_i$
6:     If $O$.type = 'Insert' or "update" then {
7:         Goto Line 16
8:     }
       //The last Op of CNi is delete operation
9:     if $O.SV > MSV_s[i]$ //not all the sites have executed the
   $O$ then
10:        Goto Line 16
11:    step = $SVT_s[r][r] - O.SV[r]$ //$r$ indicates the site id of the
       birth of $O$
12:    if step > = $N_{undo}$ then {// remote $r$ has created at
       least $N$ operations after that deletion
13:      Delete Node $i$ and its corresponding operation table
14:      Continue
15:    }
16:    for all $O$ of CN$i$ which $O$.type = "update" do{// to compress
       update $Op$
17:      step = $SVT_s[r][r] - O.SV[r]$ // $r$ denotes the site id of
         the $O$ generated
18:      if step > = $N_{undo}${
19:        Delete $O$ from the operation table of $CN_i$
20:      }
21:    }
22:}
```

4.4. Support access control during pursuing consistency among peer-to-peer people

The AST algorithm supports the out-of-order collaboration and communication in peer-to-peer architecture. However, practical engineering design process is apt to partition different regions and distinguish different permissions in order to clarify the responsibilities of different designers. For example, designers of conduit department can only view but not modify or delete the objects created by designers of structure department. Based on the above analysis, we introduce a permission vector (PV) on every site to support access control. PV is a vector like $\langle R_1, R_2, \ldots, R_N \rangle$, $R_i = 0$ or 1, denotes whether current site has the modification permission of the ith object node counting from left to right on the linear structure of Doc_s. N is the length of the linear structure of Doc_s. Initially, the value of PV is obtained from the server according to the login role of the current designer. During the operation process, once a new object is inserted into the DTM, only the same group members have the permission to modify it, others can only view the operation effect. Certainly the constraint can be changed by the re-allocation of the server manager. Once an interface operation is captured by the adapter layer, access control will be checked to identify whether the current designer has the permission to do that. If not, the operation will be abandoned and view clues will be given to him or her.

5. Space consistency maintenance

The hard-conflict based on design rules (e.g. intersects or segment of two objects) and soft-conflict (e.g. distance between several objects) should be detected as soon as it occurs and clues should be displayed to the end users. The detection of space conflicts is based on specific design rules. Different design processes or different design environments may have different rules. For example, during detail design process, the water and electricity pipe should not be intersected by steel tube of the building structure.

The introduction of TDM realizes the combination of different models from different perspectives and the time consistency maintenance strategy provides the possibility of real-time detection. Then real-time space consistency can be provided to maintain space consistency based on that. In order to achieve the goal, design rules should first be established according the real requirements of the design environment. Here four constraint rules are proposed as an example and based on the rules and a real-time detection process is provided. The constraint rules are as follows:

(1) Size constraint: To define the size range of an object, or to construct equation or in-equation relation of one object with other feature size.
(2) Location constraint: To define the location relation with other features. It points to the feature face of other feature by persistent ID; there are three kinds of Location Constraint: = ,$\langle\ \rangle$ and dif.
(3) Edge constraint: To define that the feature face should not be split.
(4) Intersect constraint: To define that a certain feature face should not be interacted with other ones.

Following Bronsvoort [21], we also use feature face to maintain the persistent, steady information of geometry face. The feature face will not be deleted, split or united during the engineering process. Every entity maintains the list of the feature face and constructs constraint relationship with other entities by the list. m_ID is the unique identifier in the world. The class structure of feature face is as follows:

```
ENTITY FeatureFace
  m-ID : Integer; //Persistent ID
  m-Face :FACE; // Pointer to the geometry face
  m-Owner :Feature; // the owner feature
  m-Driven :Feature; // the driven feature
  m-Origion :FeatureFace; // the owner feature face
END ENTITY
```

Taking Fig. 6 as an example to describe the relationship of geometry face and feature face, after the size modification of slot, geometry FACE will be split into geometry FACE1 and FACE2, where FACE and FACE1 point to the same face, while FACE2 points to the newly split face. As for FeatureFace2, m_Face points to geometry FACE2, m_Owner points to the base face, m_Driven points to the slot face. m_Owner and m_Driven describe the constraint relationship of FACE2 with base feature and slot feature. m_Origin points to the feature face which geometry face FACE1 represents. Therefore, we can find geometry FACE1 by the m_Origin pointer of FeatureFace2 easily, which provides great convenience for us to define rule constraints between FACE1 and FACE2.

Then the constraint based on feature face is described as follows:

```
ENTITY.Attribute DesignConstraint
  m_Name: String; //constraint name
  m_Identifier: UUID; //constraint identifier
  m_Timestamp: TIME; //construction time
  m_Status: BOOLEON; //whether satisfied?
```

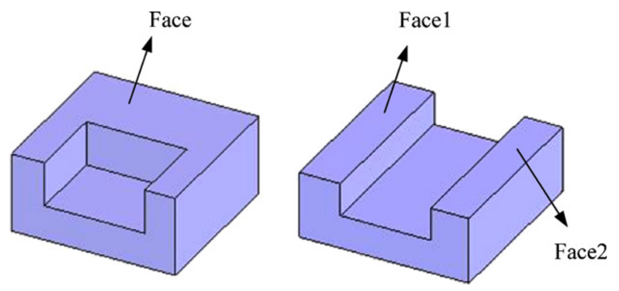


Fig. 6. Relationship of geometry face and feature face.

```
    m_Nature: ENUM; //constraint type
    m_Domain: STRING; //constraint field
    m_Variable: STRING; //Variables to be constrained
    m_protection: STRING; //Open to where?
    m_Weight: INTEGER; //weight value (from 1 to 10)
    m_Owner: BODY; //owner entity
    m_Creator: STRING; //Creator of the constrain
    m_Specification: STRING; //detailed comment of constraint
END ENTITY.Attribute
```

The conflict detection procedure is described in Procedure 4.

```
Procedure 4. SpaceConflictDetect (TDM)
for each entity in TDM do {
   Cons = entity.Attribute.DesignConstraint
   for each constaint in Cons do {
      Select constraint.m_Nature
         Case "Size-Constraint":
            for each wire in constraint.m_Varible do{
               api_wire_length(wire,len)
            if not Match(len, constraint.m_Specification) then
               HighLight( wire, SIZE_CONSTRAINT)
               }
         Case "Location-Constraint":
            for each featureface in constraint.m_Varible do {
               api_entity_entity_distance(entity, featureface, pos1,-
   pos2,dis,NULL,NULL)
            if not Match(dis, constraint.m_ Specification) then
               Highlight( entity, featureface, LOCATION_CONSTRAINT)
               }
         Case "Edge-Constraint":
            for each featureface in constraint.m_Varible do{
               for each ff in entity of TDM do{
               if ff.m_Origin = featureface then
               Highlight( fetureface, ff, EDGE_CONSTRAINT)
               }
            }
         Case "Intersect-Constraint":
            for each face in constraint.m_Varible do {
            Boolean flag= api_intersect (entity, face)
            If flag = true then
               HighLight(entity, face, INTERSECT_CONSTRAINT)
            }
         End Select
      }
}
```

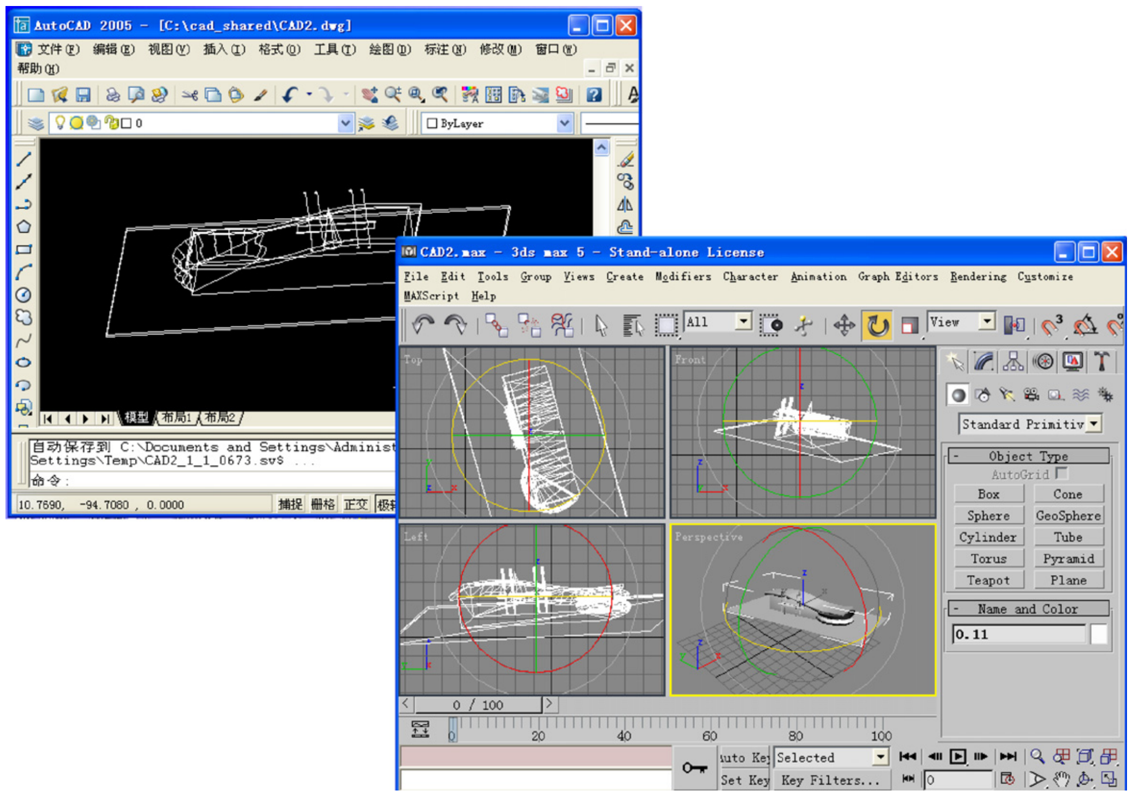


Fig. 7. Different applications sharing the same model.

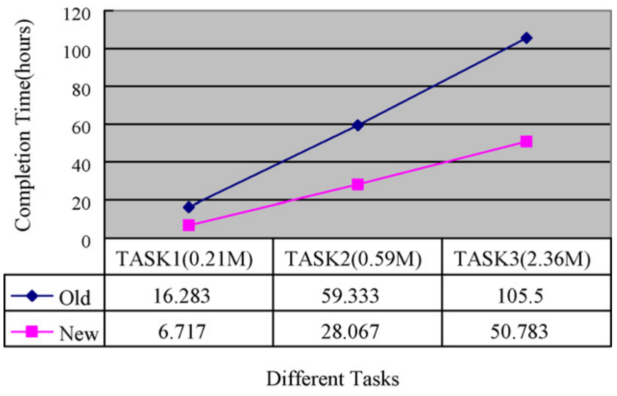
6. Experiments

In order to validate the efficiency of the proposed framework and find the value of N_{undo} in different environments, we had experiments on Microsoft Windows XP platform with two applications AutoCAD 2005 and 3DMax 5, using Microsoft Visual Studio 2003 as the development tool. The hardware components of this experiment include 6 Dell PCs (each with P2.4 CPU, 1 GB memory and 160 GB hard disk) and the network bandwidth is 100MPS and with network delay is simulated to be between 50 ms and 250 ms. We have selected six participants working in the simulation experiment. The participants are college students and each of them is only familiar with only one of the two tools. The prototype system has achieved generic Engine component based on AST and solved the problem of time consistency, implemented the TDM based on ACIS to cover most operations used in AutoCAD and 3DMax (see Fig. 7).

The first experiment is done to evaluate the validity of the proposed framework by comparing the completion time of the proposed framework with single-user one with three design tasks (2.36 M volume, 0.59 M volume, 0.21 M volume, respectively). During the design process, the designers are divided into three small groups with two persons per group. Before the experiments start, all the participants are given five minutes' demonstration of the final product and are required that their design results should not be the same with the demo, but have similar functionalities. Fig. 8 shows the contrast results related to different tasks.

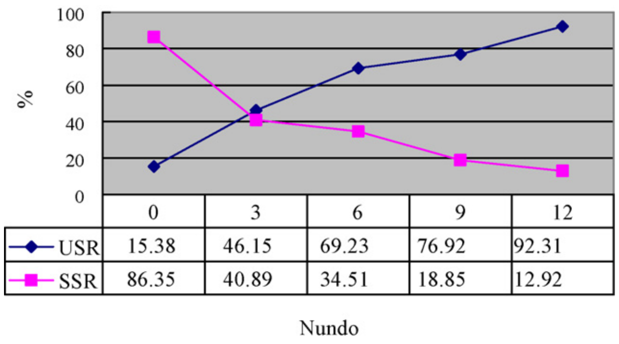
From Fig. 8, we can see that the completion time of both simple and complex tasks can be reduced by using the integrated framework since the new environment can provide support for convenient and real-time collaboration.

In order to find the value of N_{undo}, we simulated the operations sent by all participants at all sites. When undo operations cannot be executed because of the limitation of N_{undo}, we repair it by doing the reverse operation of the undo targeted operation. That's to say, if a delete operation is called to be undone, we do a corresponding insert operation. In order to find the reasonable value, we first define two parameters which are named as undo successful ratio (USR) and space saving ratio (SSR). USR is defined as the percentage of successful undo operations (SUO) to total undo operations (TUO) and SSR are defined as the percentage of the difference of max space utilization (MSU) and space utilization (SU) to MSU (see following formula). USR reflects the convenience degree designers may feel about the system while SSR reflects the space saving degree of the new garbage collection strategy. Different participant number and different network delay may have different experiment results. Fig. 9 shows the variation of USR and SSR related to N_{undo} with the participant number being 6 and network delay being between 50 ms and 250 ms and the interval between the calculations of SSR is set to 20 min.



	TASK1(0.21M)	TASK2(0.59M)	TASK3(2.36M)
Old	16.283	59.333	105.5
New	6.717	28.067	50.783

Fig. 8. Completion time of different tasks.



	0	3	6	9	12
USR	15.38	46.15	69.23	76.92	92.31
SSR	86.35	40.89	34.51	18.85	12.92

Fig. 9. USR and SSR related to N_{undo}.

$$\text{USR}\,(\%) = \frac{\text{SUO}}{\text{TUO}} \times 100$$

$$\text{SSR}\,(\%) = \frac{\text{MSU} - \text{SU}}{\text{MSU}} \times 100$$

From Fig. 9 we can see that with the increasing of N_{undo}, the value of USR increases and reaches 100% when N_{undo} is set to a big value. The value of USR reflects the influence of limited undo on user actions and should be as big as possible. However, with the increasing of N_{undo}, SSR decreases since more invalid nodes cannot be collected in time. So we should select a mediated value of N_{undo} by generally considering both USR and SSR. In order to find the appropriate value, we define the following formula and the value of N_{undo} should be set to be the one which leads to the max value of the fitness function.

$$\text{Fitness} = \left(\frac{\begin{array}{r}\alpha(\text{USR} - \text{Min}(\text{USR})/(\text{Max}(\text{USR}) - \text{Min}(\text{USR}))) \\ +\,\beta(\text{SSR} - \text{Min}(\text{SSR})/(\text{Max}(\text{SSR}) - \text{Min}(\text{SSR})))\end{array}}{2} \right)$$

where $\alpha, \beta \in [0, 1] \wedge \alpha + \beta = 1$, referring to the influence power of two factors, which can be determined based on the real requirements of a solid task. For example, if the hard wares assigned in a design task are little old with low performance, the value of β may be set a little bigger. If we set $\alpha = 0.6$ and $\beta = 0.4$, we will get values listed in Table 1 and from the table, we can conclude that the value of N_{undo} should set to be around 12.

Here, we must emphasize that with different design tasks, different designers and different network environments, the experiment results may be different. Our experiments were just to describe one possible solution to find the best choice of N_{undo}. Another possible solution is that before the collaboration begins, N_{undo} is set to a solid value which will be informed of all the participants and then the value will be adjusted dynamically according to three factors: tasks, designers, and network environments.

Table 1
Fitness of different N_{undo}

	0	3	6	9	12
USR	15.38	46.15	69.23	76.92	92.31
SSR	86.35	40.89	34.51	18.85	12.92
Fitness	0.2	0.196173598	0.268800404	0.256135838	0.3

7. Conclusions and future work

This paper presents a TDM centered framework to support heterogeneous collaboration of hybrid CAD applications by transforming different single-user applications into a groupware system without modifying the source code of the applications. ACIS-based TDM is introduced to combine not only the information of entities but also relationships between the entities with a data proxy to separate data model from user view so that different users will have different views according to different roles. Based on the framework, two critical problems are solved: time consistency and space consistency. As for time consistency, we introduced the AST algorithm to the new environment by mapping the random address space to liner one. In order to adapt to the new application environment with large-scale objects being edited concurrently, we introduced a garbage collection algorithm to compress *forever-invalid* nodes efficiently. As for space consistency, since the TDM model gives the basis to integrate different models from different perspectives, a conflict detection algorithm based on feature face is introduced to realize real-time inspection. Experiments have been done to evaluate the validity of the proposed framework and to analyze how to get the appropriate value of N_{undo} which is an important factor used in the garbage collection strategy and is vital for the AST algorithm to be used in large-scale environments.

As for future work, we will research into the automatic explanation of the captured operation and the automatic construction of the mapping matrix between different applications. Based on the Total Data Model, the model-view projection is also a research direction which is taken by our industrial partner. Currently, the proposed framework supports the space consistency only by detecting and highlighting strategy. We will investigate new solutions to support automatic modification of constrained objects based on design rules by modifying the AST algorithm to support rule automation.

Acknowledgements

The work is supported by National Natural Science Foundation of China (NSFC) under Grant Nos. 90612008 and 60736020, National Grand Fundamental Research 973 Program of China under Grant No. 2005CB321905, the Open Subject Fund of Jiangsu Provincial Key Laboratory of Network and Information Security of China under Grant No. BM2003201 and Shanghai Leading Academic Discipline Project under Grant No. B114.

References

[1] H. Liu, X. Liu, L. Ma, Supporting evolution in a computer-aided design environment, Journal of Computer-Aided Design & Computer Graphics 15 (2) (2003) 167–173.
[2] J. Grudin, Why CSCW applications fail: problems in the design and evaluation of organizational interfaces, in: Proceedings of ACM Conference on Computer Supported Cooperative Work, Portland, OR, USA, (1988), pp. 85–93.
[3] I. Grief, R. Seliger, W. Weihl, Atomic data abstractions in a distributed collaborative editing system, in: Proceedings of the 13th Annual Symposium on Principles of Programming Languages, 1986, pp. 160–172.
[4] M. Stefik, G. Foster, D.G. Bobrow, K. Kahn, S. Lanning, L. Suchman, Beyond the chalkboard: computer support for collaboration and problem solving, Communications of the ACM 30 (1) (1987) 32–47.
[5] K.A. Lantz, An experiment in integrated multimedia conferencing, in: Proceedings of ACM Conference on Computer Supported Cooperative Work, 1986, pp. 267–275.
[6] M.J. Knister, A. Prakash, DistEdit: a distributed toolkit for supporting multiple group editors, in: Proceedings of ACM Conference on Computer Supported Cooperative Work, Los Angeles, CA, USA, (1990), pp. 343–355.
[7] P. Dewan, A. Sharma, An experiment in interoperating heterogeneous collaborative systems, in: Proceedings of the Sixth European Conference of Computer Supported Cooperative Work (ECSCW'99), Copenhagen, Denmark, (1999), pp. 371–391.
[8] K. Palfreyman, T. Rodden, J. Trevor, PSI: a platform for shared interaction, in: Proceedings of the Sixth European Conference of Computer Supported Cooperative Work (ECSCW'99), Copenhagen, Denmark, (1999), pp. 351–370.
[9] A. LaMarca, W.K. Edwards, P. Dourish, J. Lamping, I. Smith, J. Thornton, Taking the work out of workflow: mechanisms for document-centered collaboration, in: Proceedings of the Sixth European Conference on Computer-supported Cooperative Work (ECSCW'99), Copenhagen, Denmark, (1999), pp. 1–20.
[10] A.M. Krebs, M. Ionescu, B. Dorohonceanu, I. Marsic, The DISCIPLE system for collaboration over the heterogeneous web, in: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, Hawaii, USA, (2003), pp. 11–21.
[11] D. Li, R. Li, Transparent sharing and interoperation of heterogeneous single-user applications, in: Proceedings of the ACM Conference on Computer-supported Cooperative Work, New Orleans, Louisiana, USA, (2002), pp. 246–325.
[12] D. Li, J. Lu, A lightweight approach to transparent sharing of familiar single-user editors, in: Proceedings of ACM Conference on Computer-supported Cooperative Work, Banff, Alberta, Canada, (2006), pp. 139–148.
[13] S. Xia, D. Sun, C. Sun, D. Chen, Leveraging single-user applications for multi-user collaboration: the CoWord approach, in: Proceedings of ACM Conference on Computer-supported Cooperative Work, Chicago, Illinois, USA, (2004), pp. 162–171.
[14] C.A. Ellis, S.J. Gibbs, Concurrency control in groupware systems, in: Proceedings of the ACM Conference on the Management of Data, Portland, OR, USA, (1989), pp. 399–407.
[15] N. Gu, J. Yang, Q. Zhang, Consistency maintenance based on the mark & retrace technique in groupware systems, in: Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work, Sanibel Island, FL, USA, (2005), pp. 264–273.
[16] http://www.spatial.com/products/acis.html.
[17] C. Sun, C.A. Ellis, Operation transformation in real-time group e*ditor: issuer, algorithm, and achievement, in: Proceedings of ACM Conference on Computer-supported Cooperative Work, Seattle, WA, USA, (1998), pp. 59–68.
[18] C. Sun, X. Jia, Y. Zhang, Y. Yang, D. Chen, Achieving convergence, causality-preservation, and intention-preservation in real-time cooperative editing systems, ACM Transactions on Computer–Human Interaction 5 (1) (1998) 63–108.
[19] J. Yang, N. Gu, X. Wu, A document mark based on method supporting group undo, in: ACM CSCW 2004 Workshop on Collaborative Editing Systems, Chicago, Illinois, USA, 2004.
[20] J. Yang, Q. Zhang, N. Gu, G. Yang, Z. Liu, The multi-version and single-display strategy in undo scheme, in: Proceedings of the fifth International Conference on Computer and Information Technology, Shanghai, China, (2005), pp. 290–296.
[21] B. Bidarra, W.F. Bronsvoort, Semantic feature modeling, Computer-Aided Design 32 (2000) 201–225.

Liping Gao received her BSc in Computer Science from Shandong Normal University, China in June 2002, and the master degree in Computer Science from the same university in June 2005. She is currently a PhD student in Laboratory of Cooperative Information and Systems of Computer Information and Technology Department of Fudan University. Her current research interests include CSCW, heterogeneous collaboration, consistency maintenance and collaborative engineering.

Bin Shao received his BSc in Computer Science from Shandong University, China in June 2005. He is currently a PhD student in Laboratory of Cooperative Information and Systems of Computer Information and Technology Department of Fudan University. His current research interests include concurrent control algorithms and web-based groupwares.

Lin Zhu received his BSc in Computer Science and Engineering from Fudan University, China, July 2006. He is currently Master student in Laboratory of Cooperative Information and Systems of Computer Information and Technology Department of Fudan University. His current research interests include web-based collaboration and collaborative image annotation.

Tun Lu graduated from Sichuan University, China with a BEng in 2000, a MEng in 2003 and a PhD in 2006, all in Computer Science. He is doing his postdoctoral research at the Department of Computing and Information Technology at Fudan University now. His research interests include grid computing, collaborative computing, and software engineering.

Ning Gu is a Professor of Computing and Information Technology at Fudan University, Director of Cooperative Information and Systems Laboratory, Director of CSCW Committee of Shanghai Computer Society, ACM Member and IEEE Member. His main research areas are CSCW, Web Service and data and knowledge management.