A New Mechanism of Naming Topological Entities for Semantic Feature Operations

Xue-Yao Gao
School of Computer Science and Technology, Harbin University of Science and Technology, Harbin, China
Email: gaoxueyao@hotmail.com

Chun-Xiang Zhang
School of Software, Harbin University of Science and Technology, Harbin, China

Ming-Yuan Ren
School of Software, Harbin University of Science and Technology, Harbin, China

Shang-Min Gao
School of Computer Science and Technology, Harbin University of Science and Technology, Harbin, China

Yong Liu
School of Computer Science and Technology, Heilongjiang University, Harbin, China

***Abstract*—Naming topological entities and the validity maintenance of model are two important issues in Computer Aided Design, which can make a semantic feature model conform to all constraints correctly. In this paper, a new mechanism of naming topological entities based on face features is proposed. Based on the corresponding mechanisms for naming topological entities, the methods of coding entities, sub-entities and virtual entities are given. The mechanisms for maintaining the validity of semantic feature operations are described. Sub-edges, sub-faces, virtual faces and virtual sub-edges are applied to process the problem of face obliteration. Sub-edges and virtual edges are used to process the problem of edge obliteration. After the mechanisms for naming topological entities and the methods of validity maintenance are applied to HUST-CAID system, its modeling performance is improved and the feasibility of the proposed mechanisms and methods above is also proved.**

***Index Terms*—topological entities, validity maintenance, semantic feature model, virtual entities, face obliteration, edge obliteration**

I. INTRODUCTION

Naming topological entities and the validity maintenance of semantic feature model are two important issues in Computer Aided Design(CAD) and Computer Aided Manufacturing(CAM).

Marcheix identifies five common concepts that may be found in studies of naming topological entities, and two orthogonal criteria for classifying approaches of naming entities persistently are proposed[1]. Wu analyzes the design history of parts, and presents a face-based mechanism for naming, recording and retrieving topological entities. At the same time, parametric space information is utilized to solve the ambiguity when entities are recorded and retrieved[2]. This mechanism provides a solid basis for replaying the design history precisely. Jing analyzes four fundamental conditions to solve the problems of naming topological entities persistently and their internal relationships, based on the model representation used in 3D parametric CAD systems[3]. The problems of naming entities persistently can be classified into two categories. The first category includes two aspects. The first one is how to associate the original name with the B-rep model, and the second one is how to construct the reference name. The second category involves two processing methods according to the reference purposes. The first one is referring to the geometric information in entities, and the second one is referring to the entity itself. Chen proposes a new approach to name and match topological entities, based on the affected features and affected faces in topological entities[4]. The usual topological entity is named by using ID of the feature which it belongs to, and its index in the B-rep of the feature. For the referenced topological entity, the local topology information is added into the name. The local topology information consists of the adjacent faces relevant to the topological entity and the in-out relationship among them. Zheng designs a face-based and history-based naming mechanism which includes entity naming and entity retrieval[5]. The necessary history information of faces is kept by face's name. All topological entities are managed by three kinds of tables which are face tables, edge tables and vertex tables. Kripac gives a topological ID system, which systematically assigns IDs to topological entities including faces, edges and vertices in solid models[6]. When the solid model is edited and is re-evaluated automatically from the history of modeling operations,

doi:10.4304/jsw.7.3.705-711

IDs of topological entities in old version of the model are mapped to IDs of topological entities in new version of the model. This mapping defines the correspondence between topological entities in old model and topological entities in new model. Marcheix considers the naming problem of aggregates like shells which are connected sets of faces[7]. A complete framework for extending a naming model of atomic entities is proposed in order to identify and match any kinds of shells based on their underlying topological structures. Yan develops a semantic ID scheme based on geometry continuity to solve the problem of naming persistency and improve interoperability of CAD feature modeling[8]. Hierarchical namespaces localize entity creation and entity identification. All geometric and topological entities are referred uniformly based on IDs of the surfaces, and topology semantics is retained in its ID. During the process of re-evaluation, topological entities may be split, merged and obliterated. Wang introduces a data structure called as name propagation graph(NPG) to represent the identity propagation of topological entities, in order to trace such alterations of topological entities[9]. Rules and algorithms are also presented to identify the genetic entities, which originate from the entities on the original version of a part model. Identifications of sketch share and regeneration are realized by appending additional index and flag, and the presented naming system has been implemented in a commercial feature-based modeling system[10]. The naming techniques build a bridge between design variants and design intents. Wu presents a graph-based coding and decoding method with parameter information to maintain the design intent in constraint-based variational design system[11]. At the same time, parameter vector field is used to describe a mathematical definition on comparability between original model and regenerated model. Bidarra proposes a cellular representation for feature models that contains all relevant information to solve effectively a variety of current problems in feature modeling[12]. Much benefit is gained from a coherent integration between shapes of a feature model and cells in the cellular model. At the same time, methods for modifying and querying the cellular model are given, and their applications for feature validity maintenance, feature interaction management, feature conversion between multiple views, and feature visualization are illustrated. Bidarra proposes the mechanisms to maintain the validity of the semantic feature model, in which each invalid situation is detected and is reported to the user with appropriate explanation on its causes and effects[13]. At the same time, the user is provided with a convenient choice of reaction hints. Chen proposes a representation for feature validity condition based on the extended attributed adjacency graph, and gives a novel approach for maintaining feature validity using local feature recognition technique. The approach can not only automatically detect whether the feature validity is destroyed, but also determine accurately the reason why the feature becomes invalid and the state of the destroyed feature. Furthermore, the approach can renovate the feature model automatically based on the user's intent[14].

In this paper, we propose a new mechanism for naming topological entities based on face features. The methods of coding topological entities, sub-entities and virtual entities are given. At the same time, the maintenance method of semantic feature model is given. Sub-edges, sub-faces, virtual faces and virtual sub-edges are applied to process the problem of face obliteration. Sub-edges and virtual edges are used to process the problem of edge obliteration. Experimental results show that after the new naming mechanism and the maintenance method of semantic feature operations are applied to HUST-CAID system, the feasibility of the presented mechanism and method is proved.

The rest of this paper is organized as follows: the mechanism of naming topological entities based on face features is described in section II. When the model is edited and re-evaluated, the validity maintenance method of semantic feature operations is proposed in section III. Experimental results are given in section IV. Conclusions of this paper are given in section V.

II. Mechanism of Naming Topological Entities Based on Face Features

When the semantic feature model is edited and re-evaluated automatically, its topological structure may change and the enumeration of topological entities becomes invalid. So, the fundamental problem in process of semantic feature modeling is how to name and identify topological entities, in such a way that these entities can still be identified even after the model has been re-evaluated. Topological entities should be named, including faces, edges and vertexes in model. When the semantic feature model is edited and re-evaluated, these names of topological entities could be maintained correctly. Edge can be considered as the intersection of two faces and vertex is also viewed as the intersection of three faces. In this paper, we firstly name topological faces. Then based on the names of face features, topological edges and topological vertexes are named.

A. Naming Topological Faces

When the topological face is named, the name of feature which it belongs to and its ID in this feature will be considered together. The name of topological face is expressed as feature_name.face_ID. Here, feature_name is the name of feature which this face belongs to and face_ID is the face's ID in this feature. For example, there are three topological faces in a cylinder, including cylinder.top, cylinder.side and cylinder.bottom, as described in Figure 1.

Features can also be generated by sweeping operation and rotating operation. So, one feature may contain multiple side faces. So, these side faces should be distinguished correctly.

Because a sweeping path is made up of multiple line segments, every contour edge in a sketch will be corresponded with a side face. The steps of naming side

faces in feature generated by sweeping operation are shown as follows:

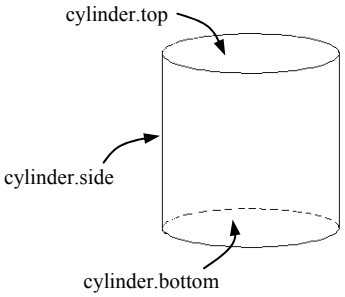

Figure 1. Naming topological faces in a cylinder.

(1) Number contour edges in a sketch serially.

(2) The sweeping path is divided into multiple line segments, and these path segments are numbered serially.

(3) Based on the number of contour edges and the number of path segments, side faces are named.

If a square is swept along a 3-dimension polyline, top face, bottom face and 12 side faces will be gotten in Figure 2. Based on the above method, the side face determined by No.i path segment in sweeping path and No.j contour edge in sketch is named as sweep.sideij.

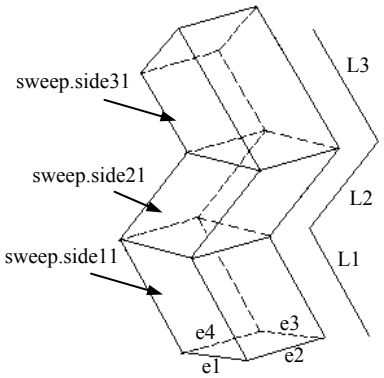

Figure 2. Naming side faces in a feature generated by sweeping operation.

Features generated by rotating operation are determined by the contour and the axis, in which the contour rotates in some angle along the axis. Contour edges will be numbered serially. Based on the numbers of contour edges, all side faces are named. When a contour which has N contour edges rotates in some angle along the axis, the steps of naming its side faces are shown as follows:

(1) From all contour edges, select edge *m* whose starting point is the nearest from the axis.

(2) Contour edge *m* is used as the datum edge.

(3) According to datum edge *m* and the trend of the contour, all contour edges will be numbered serially.

(4) If the datum edge is not coincided with the axis, side faces are named based on the numbers of contour edges. Otherwise, the front N-1 side faces will be named serially.

A feature generated by rotating operation is shown in Figure 3. The names of its side faces are respectively rotate.side1, rotate.side2, rotate.side3 and rotate.side4.

According to the method of naming faces, the topological face is coded as follows:

Face_ID=(FeatureID, type, index, PID, bsplit, addi)

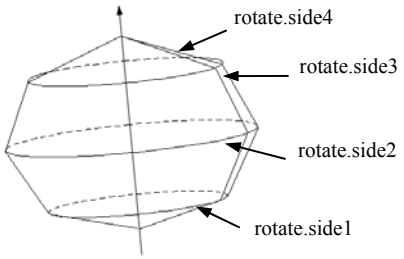

Figure 3. Naming side faces in a feature generated by rotating operation.

Here, FeatureID is the ID of feature which this face belongs to and type is its category. When the value of type is -1, it's a bottom face. When the value of type is 0, it's a top face. When the value of type is 1, it's a side face in a feature generated by sweeping operation and PID is the number of sweeping path segment. When the value of type is 2, it's a side face in a feature generated by rotating operation and PID is the rotating angle. When there are serial side faces, index is the number of side face. For top face and bottom face, the value of index is 0. We use bsplit to describe whether the face has been split or not. When the model is edited, some topological faces maybe disappear and they will not appear in the boundaries of the model. These faces will become virtual ones and they will be invisible for user. We use addi to describe the states of faces. When the face becomes a virtual one, the value of addi will be set to 0. Otherwise, the value of addi is set to 1.

When the model is edited, some topological faces will be split. These split sub-faces maybe merge in subsequent feature operations. So, sub-faces should be coded in a different way. Sub-face is coded as follows:

subFace_ID=(FaceID, type, index, PID, bsplit, fID, addi)

Here, FaceID is its father face's ID, and fID is the number of sub-face. At the same time, addi is used to explain whether the sub-face is a virtual one or not.

B. Naming Topological Edges

When the topological edge is named, face features relevant to this edge will be considered together. The name of topological edge is expressed as ((adjacent_feature_faces), (end_faces)).

Here, adjacent_feature_faces are the adjacent faces relevant to this edge and end_faces are the starting face and ending face in which two endpoints of this edge are located. A block slot is opened in base block. Then edge *e1* is gotten as shown in Figure 4. Edge *e1* is named as ((block.top, block.side1), (block.side4, blindslot.side1)).

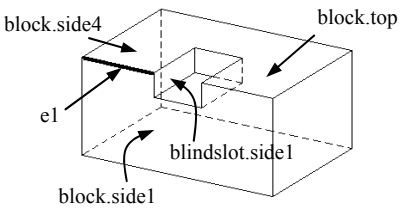

Figure 4. Naming edge e1.

If edges determined by surfaces are named based on the above method, some edges will not be distinguished correctly. Reference line need be introduced in order to distinguish these edges. So, edges determined by surfaces are named as ((adjacent_feature_faces), (end_faces, reference.orientation)). Here, reference is a reference line and reference.orientation is the orientation of the reference line. It is used to describe that edge is located in left side or right side of the reference line.

When we open a round slot on top face of a cylinder, parallel edges *e1* and *e2* are obtained. In order to name parallel edges correctly, a reference line is introduced to describe the positions of two parallel edges. Reference line reference1 is defined, which is located on top face of cylinder and is paralleled with edge *e1* and *e2* as shown in Figure 5.

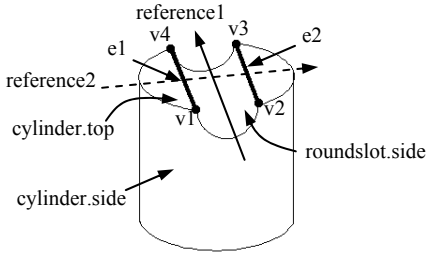

Figure5. Apply reference line to distinguish parallel edges.

Then edge *e1* and *e2* are named as follows:

e1: ((cylinder.top, roundslot.side), (cylinder.side, roundslot.side, reference.positive))

e2: ((cylinder.top, roundslot.side), (cylinder.side, roundslot.side, reference.negative))

Sometimes, the reference line maybe closed. When a round slot is opened on top face of base block, two parallel edges *e1* and *e2* are gotten. In order to name *e1* and *e2* correctly, a closed reference line reference3 is introduced here. It is shown in Figure 6.

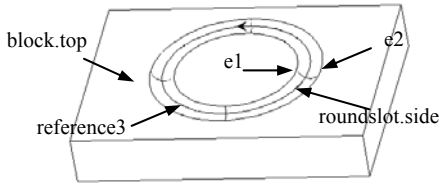

Figure 6. Closed reference line.

Then edge *e1* and *e2* are named as follows:

e1: ((block.top, roundslot.side), (reference.positive))

e2: ((block.top, roundslot.side), (reference.negative))

According to the method of naming edges, topological edges are coded as follows:

Edge_ID=(adjFaceIDs, endFaceIDs, RefOriCode, addi)

Here, adjFaceIDs are IDs of two adjacent faces whose intersection is this edge, and endFaceIDs are IDs of the starting face and ending face in which two endpoints of this edge are located. If the edge is closed, the value of endFaceIDs will be set to 0.

RefOriCode is used to describe that the topological edge is located in left side or right side of the reference line. When the value of RefOriCode is -1, the topological edge is located in right side of reference line. When the value of RefOriCode is +1, the topological edge is located in right side of reference line. If the reference line is not needed, RefOriCode will be set to 0. We use addi to describe whether the topological edge is a virtual one or not. When the value of addi is 0, this edge does not belong to the model boundary and it is a virtual edge. When the value of addi is 1, this edge belongs to the model boundary.

When the model is edited and re-evaluated, some topological edges will be split. These split sub-edges maybe merge in subsequent feature operations. So, sub-edges should be coded in a different way. Sub-edge is coded as follows:

subEdge_ID=(EdgeID, adjFaceIDs, endFaceIDs, addi)

Here, EdgeID is ID of the topological edge which these sub-edges belong to, and adjFaceIDs are IDs of two adjacent faces relevant to this sub-edge. We use endFaceIDs to describe the starting face and ending face in which two endpoints of this sub-edge is located, and at least one of these faces is the starting face or ending face in which its father edge is located. At the same time, addi is used to explain whether the sub-edge is a virtual edge or not.

C. Naming Topological Vertexes

A vertex can be viewed as intersections of three adjacent faces, which describes its topology property. The topological vertex is named as ((adjacent_feature_faces), (reference.orientation)).

Here, adjacent_feature_faces are adjacent faces relevant to this vertex and reference.orientation is the orientation of the reference line. Intersections of cylinder.top, roundslot.side and cylinder.side are vertexes *v1*, *v2*, *v3* and *v4* as shown in Figure 5. Reference line reference2 vertical to reference1 is introduced to distinguish these vertexes. Then vertexes *v1*, *v2*, *v3* and *v4* are named as follows:

v1: ((cylinder.top, roundslot.side, cylinder.side), (reference1.positive, reference2.negative))

v2: ((cylinder.top, roundslot.side, cylinder.side), (reference1.negative, reference2.negative))

v3: ((cylinder.top, roundslot.side, cylinder.side), (reference1.negative, reference2.positive))

v4: ((cylinder.top, roundslot.side, cylinder.side), (reference1.positive, reference2.positive))

According to the method of naming vertexes, topological vertexes are coded as follows:

VertexID=(adjFaceIDs, RefOriCode, addi)

Here, adjFaceIDs are three adjacent faces which are relevant to this vertex. RefOriCode is used to describe that the vertex is located in left side or right side of the reference line. When the value of addi is 1, this vertex is located in the model boundary, and when the value of addi is 0, it is not located in the model boundary and it is a virtual vertex.

III. VALIDITY MAINTENANCE OF SEMANTIC FEATURE MODEL

Model validity maintenance is a process of monitoring each feature operation in order to ensure that all features

and topological entities conform to the semantics specifications. When semantic feature model is edited and re-evaluated, some topological entities including faces and edges will be split, emerged and obliterated. In order to maintain the validity of semantic feature model, faces, edges and vertexes should be managed efficiently in feature operations. Here, virtual faces, sub-edges and virtual vertexes are used to process the problem that topological entities are split, emerged and obliterated. In the process of semantic feature modeling, three mechanisms are given to manage the topological entities, including inheritance of topological entities, split of topological entities and obliteration of topological entities.

A. Inheritance of Topological Entities

All topological entities have specific semantics which are unchangeable in the process of modeling. When a face or an edge splits, sub-face or sub-edge will inherit the property of its father feature and the property of feature which it belongs to.

To figure axis labels, use words rather than symbols. Do not label axes only with units. Do not label axes with a ratio of quantities and units. Figure labels should be legible, about 9-point type.

Color figures will be appearing only in online publication. All figures will be black and white graphs in print publication.

B. Split of Topological Entities

The split of topological entities consists of face split problem and edge split problem. Sub-face or sub-edge inherits all properties of its father feature, and are coded as described in section II. Face split table(FST) and edge split table(EST) are used to manage the split topological entities. In FST and EST, there are records for describing the topological entity which has been split in the process of modeling and the pointer of chain in which codes of the spit entities are stored. The split topological entity is changed into a virtual one. FST and EST can help to describe the correspondence between topological entities in new model and topological entities in old model.

C. Obliteration of Topological Entities

Because vertex obliteration is viewed as the obliteration of faces and edges, we only consider the face obliteration and edge obliteration.

The steps of processing face obliteration are shown as follows:

(1) Based on constraints in model, decide whether there is obliteration part between face *f1* and face *f2* or not. If the obliteration part is empty, the codes of *f1* and *f2* are kept unchangeable. Otherwise, go to (2).

(2) Split boundary edges into sub-edges and code these sub-edges as described in section II. At the same time, sub-edges in obliteration part between face *f1* and face *f2* are coded as virtual sub-edges. The codes of boundary edges and sub-edges are processed and inserted into edge split table(EST).

(3) The sub-faces which are enclosed by these sub-edges in obliteration part between *f1* and *f2*, are coded as virtual faces. The codes of this face and sub-faces are processed and inserted into face split table(FST).

The steps of processing edge obliteration are shown as follows:

(1) Based on constraints in model, decide whether there is the obliteration part between edge *e1* and edge *e2* or not. If the obliteration part is empty or a vertex, the codes of *e1* and *e2* are kept unchangeable. Otherwise, go to (2).

(2) Decide whether starting face and ending face of *e1* are the same as those of *e2*. If the starting and ending faces of *e1* are the same as those of *e2*, *e1* or *e2* is coded as a virtual edge. Otherwise, go to (3).

(3) Based on constraints in model, decide whether there is intersection between edge *e1* and edge *e2*. According to the intersection, *e1* and *e2* are divided into several sub-edges and these sub-edges are coded as described in section II. One of sub-edges in obliteration part between edge *e1* and edge *e2* is kept, and others are coded as virtual edges. Codes of *e1*, *e2*, and sub-edges are processed, and inserted into edge split table(EST).

IV. Experiment

The mechanisms of naming topological entities and the methods of validity maintenance are applied to HUST-CAID system, in order to solve the problem that topological entities are split, merged and obliterated during the process of modeling. Here, HUST-CAID is a semantic feature modeling system which is developed by Research Institute of Computer Applied Techniques in Harbin University of Science and Technology. The methods of coding topological entities are defined and integrated into data structure in HUST-CAID system. The steps of processing face obliteration and edge obliteration are implemented and integrated in HUST-CAID system.

Then we model a part in improved HUST-CAID system. There are slot1, slot2, cylinder and block in this part model. Firstly, we open two slots in base block. Here, slot1 is a round slot and slot2 is a block slot. Secondly, we append a cylinder on slot2. At the same time, cylinder is not intersected with slot1. Initial model of this part is shown in Figure 7.

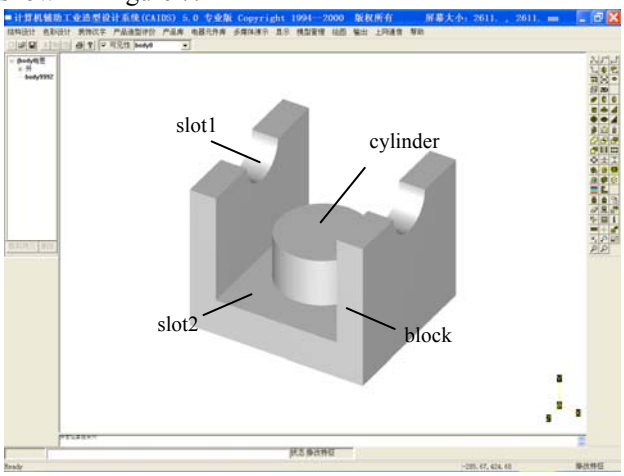

Figure 7. Initial model.

When the part model is edited, the height of slot2 is decreased and intersections between cylinder and slot1 are two parallel edges. One of these two parallel edges is edited by filleting operation. In HUST-CAID system, the methods of naming topological entities and validity maintenance proposed in this paper are used to code topological entities and maintain them correctly in part. So, we can distinguish these two parallel edges. After the part is edited by filleting operation, the model is shown in Figure 8.

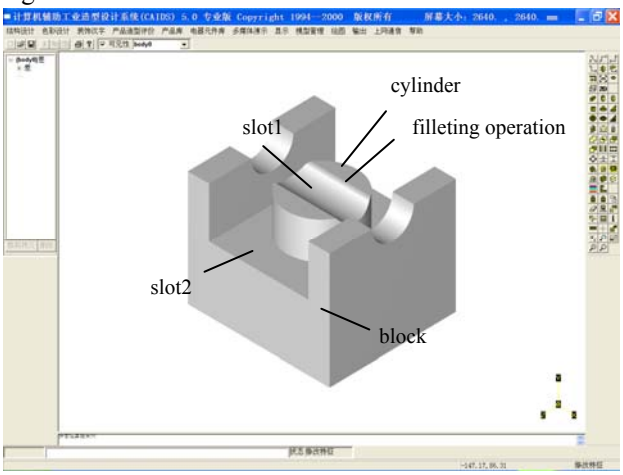

Figure 8. The modified model.

V. CONCLUSIONS

In this paper, the problems of naming and identifying topological entities are analyzed. The new mechanisms of naming topological entities based on face features are proposed. At the same time, the methods of coding topological entities, sub-entities and virtual entities are given. In order to maintain the validity of the model, three mechanisms including inheritance of topological entities, split of topological entities and obliteration of topological entities in semantic feature operations are described. Sub-edges, sub-faces, virtual faces and virtual sub-edges are applied to process the problem of face obliteration. Sub-edges and virtual edges are used to process the problem of edge obliteration. After the mechanisms of naming topological entities and the methods of validity maintenance are applied to HUST-CAID system, its modeling performance is improved and it can process entities' split, emerge and obliteration. At the same time, the feasibility of the proposed mechanisms and methods above is proved.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China under Grant Nos. 60903082, 60975042, Chun-Hui Cooperated Project of the Ministry of Education of China under Grant Nos. S2009-1-15002, and Science and Technology Research Funds of Education Department in Heilongjiang Province under Grant Nos. 11541045. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers.

REFERENCES

[1] D. Marcheix, and G Pierra, "A survey of the persistent naming problem," *Proceedings of the Symposium on Solid Modeling and Applications*, pp. 13–22, 2002.
[2] J. Wu, T. Zhang and X. Zhang, "A face based mechanism for naming, recording and retrieving topological entities," *Computer-Aided Design*, 33(1), pp. 687–698, 2001.
[3] S. X. Jing, F. Z. He, and H. J. Liu. "A survey of persistent naming problem for topological entities," *Journal of Computer Aided Design and Computer Graphics*, 19(5), pp. 545–552, 2007.
[4] Z. M. Chen, S. M. Gao, and F. J. Zhang, "Approach to naming and identifying topological entities," *Chinese Journal of Computers*, 24(11), pp. 1170–1177, 2001.
[5] J. J. Zheng, M. Fan, and R. F. Tong, "A mechanism for persistently naming topological entities," *Proceedings of the 8th International Conference on Computer Supported Cooperative Work in Design*, pp. 84–89, 2004.
[6] J. Kripac, "A mechanism for persistently naming topological entities in history-based parametric solid models," *Computer-Aided Design*, 29(2), pp. 113–122, 1997.
[7] D. Marcheix, "A persistent naming of shells," *Proceedings of the Ninth International Conference on Computer Aided Design and Computer Graphics*, pp. 259–265, 2005.
[8] Y. Wang and B. O. Nnaji, "Geometry-based semantic ID for persistent and interoperable reference in feature-based parametric modeling," *Computer-Aided Design*, 37(17), pp. 1081–1093, 2005.
[9] Y. W. Wang, J. J. Wu, and L. P. Chen, "Identity propagation method for tracing alterations of a topological entity in a history-based solid modeling system," *Journal of Advanced Manufacturing Technology*, 27(3–4), pp. 305–312, 2005.
[10] Y. W. Wang, J. J. Wu, and L. P. Chen, "Naming of sketch entities in feature based modeling system," *Journal of Engineering Graphics*, 28(1), pp. 66–71, 2007.
[11] T. Wu, Y. S. Xi, and Z. Li, "Coding and decoding of topological entities in constraint-based variational design," *Proceedings of the 2008 International Conference on Advances in Product Development and Reliability*, pp. 201–208, 2008.
[12] R. Bidarra, K. J. Kraker, and W. F. Bronsvoort, "Representation and management of feature information in a cellular model," *Computer-Aided Design*, 30(4), pp. 301–313, 1998.
[13] R. Bidarra, and W. F. Bronsvoort, "Validity maintenance of semantic feature models," *Proceedings of the Symposium on Solid Modeling and Applications*, pp. 85–96, 1999.
[14] Z. M. Chen, S. M. Gao, and Q. S. Peng, "Feature validity maintaining approach based on local feature recognition," *Journal of Software*, 13(4), pp. 552–560, 2002.

Xue-Yao Gao is Ph.D. and graduates from School of Computer Science and Technology, in Harbin University of Science and Technology. She is also a lecturer in Harbin University of Science and Technology. Her research interests are computer graphics, CAD, and natural language processing. She has authored and coauthored more than ten journal

and conference papers in these areas.

Chun-Xiang Zhang is Ph.D. and graduates from MOE-MS Key Laboratory of Natural Language Processing and Speech, School of Computer Science and Technology, in Harbin Institute of Technology. He is also an associate professor in Harbin University of Science and Technology. His research interests are computer graphics, CAD, and natural language processing. He has authored and coauthored more than twenty journal and conference papers in these areas.

Ming-Yuan Ren is Ph.D. candidate in School of Astronautics, in Harbin Institute of Technology. He is also a lecturer in Harbin University of Science and Technology. His research interests are computer graphics, CAD, and natural language processing. He has authored and coauthored more than ten journal and conference papers in these areas.

Shang-Min Gao is an associate professor in Harbin University of Science and Technology. His research interests are computer graphics, CAD, and natural language processing. He has authored and coauthored more than ten journal and conference papers in these areas.

Yong Liu is Ph.D. and graduates from School of Computer Science and Technology, in Harbin Institute of Technology. He is also a lecturer in Heilongjiang University. His main research interests include computer graphics, data mining and graph data management. He has authored and coauthored more than ten journal and conference papers in these areas.