Identification of Topological Entities and Naming Mapping for Parametric CAD Model Exchanges

Duhwan Mun* and Soonhung Han
Department of Mechanical Engineering, KAIST, Daejeon, Korea

Abstract

As collaborative design and configuration design gain increasing importance in product development, it becomes essential to exchange parametric CAD models among participants. Parametric CAD models can be represented and exchanged in the form of a macro file or a part file that contains the modeling history of a product. The modeling history of a parametric CAD model contains feature specifications and each feature has selection information that records the name of the referenced topological entities. Translating this selection information requires solving the problems of how to identify the referenced topological entities of a feature (persistent naming problem) and how to convert the selection information into the format of the receiving CAD system (naming mapping problem). The present paper introduces the problem of exchanging parametric CAD models and proposes a solution to naming mapping.

Key Words: CAD Model Exchange, Macro Parametric, Naming Mapping, Persistent Naming

1. Introduction

Many commercial CAD systems are feature-based modeling systems. Modeling with a feature-based CAD system includes features, constraints, and parameters in addition to B-rep data which explicitly represents the shape of the model. Features provide a higher level and domain-dependent vocabulary for shape-creating operations [1]. Each feature contains specific information such as parameters, attributes and the referenced topological entities. Feature-based modelers typically adopt a hybrid model to represent feature-based models. It contains both a procedural model and a B-rep model. The procedural model is represented in terms of the operation sequences used in a design session. In general, the feature tree of the CAD system corresponds to the procedural model. The B-rep model records explicit geometric information comprising bounding elements of a shape. B-rep elements are associated with the modeling operations [2].

There are two methods to exchange product data among different CAD systems, direct translation and through a neutral format. The method of using a neutral format starts from a pre-processor,

* Corresponding author:
Tel: +82-(0)42-869-3080
Fax: +82-(0)42-861-6080
Homepage: http://icad.kaist.ac.kr/~mun/
Email: mun@icad.kaist.ac.kr

which generates the neutral file from a native format. A post-processor receives the neutral file and converts it into the native format of the receiving CAD system. Examples of existing neutral CAD formats are STEP (Standard for Exchange of Product Model Data: ISO 10303), IGES (Initial Graphics Exchange Specifications), and DXF (Drawing Exchange Format).

The method of identifying the referenced topological entities of a feature in a neutral format is essential with respect to exchanging parametric CAD models. If CAD data include the modification history in addition to the construction history, a matching mechanism is also required to find the same entity in the new model (post-edit model) corresponding to the entity in the old model (pre-edit model). This problem is known as the persistent naming problem [3,4].

Fig. 1 shows an example of a persistent naming problem in SolidWorks. After the vertical slot feature is modified to penetrate the whole base feature, the edge referenced by the fillet feature is split into two edges in the final step. Contrary to expectations, the left edge is not rounded in the final result.

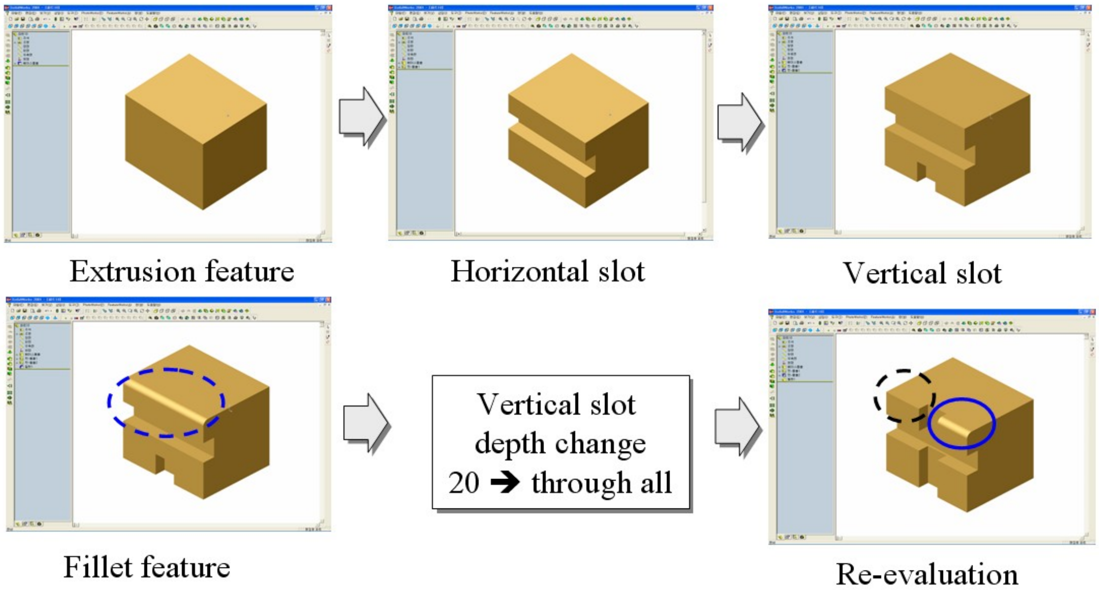

Fig. 1. A persistent naming problem in SolidWorks.

Commercial CAD systems have different naming schemes. In order to exchange parametric CAD models a naming mapping method should be provided between the naming method of the neutral file and the naming method of each CAD system.

2. Terminology and related works

2.1 Taxonomy of related technologies and terminologies

The problem can be divided into *persistent naming* (Problem 1) and *naming mapping* (Problem 2), as shown in Fig. 2-1.

Persistent naming (Problem 1) is further divided into *naming* (Problem 1.1) and *name matching* (Problem 1.2). In a feature-based CAD system, models are defined by a sequence of feature modeling steps. Each feature is generated implicitly by referencing topological entities of previously defined features, and later it is referenced to create other features. The *naming* problem (Problem 1.1) addresses how to uniquely name the topological entities that are created in each feature modeling step. The *name matching* problem (Problem 1.2) involves identification of an entity in the old model and finding the same entity in the new model, as illustrated in Fig. 2-2. *Naming* (Problem 1.1) is further divided into *basic naming* (Problem 1.1.1) and *ambiguity solving* (Problem 1.1.2), as shown in Fig. 2-1. Each step of the modeling history contains an operation and parameters. *Basic naming* (Problem 1.1.1) is naming the topological entities that constitute a feature, as illustrated in Fig. 2-2. *Ambiguity solving* (Problem 1.1.2) differentiates ambiguous topological entities when there are more than two topological entities having the same basic name. The main source of ambiguity is splitting or merging of topological faces during the process of attaching a feature to an existing model.

The *naming mapping* (Problem 2) problem occurs when CAD models with parametric information are exchanged. Since the methods to identify the referenced entity vary among different CAD systems, naming mapping is necessary so as to convert the entity names between different CAD systems as illustrated in Fig. 2-3.

The present paper introduces the problem of exchanging parametric CAD models and proposes a solution to the *naming mapping* problem (Problem 2). The detailed algorithms of *persistent naming* (Problem 1) have been described in [15].

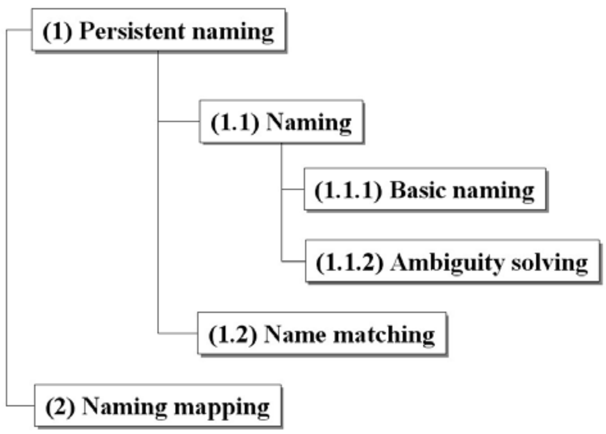

Fig. 2-1. Taxonomy of the problem.

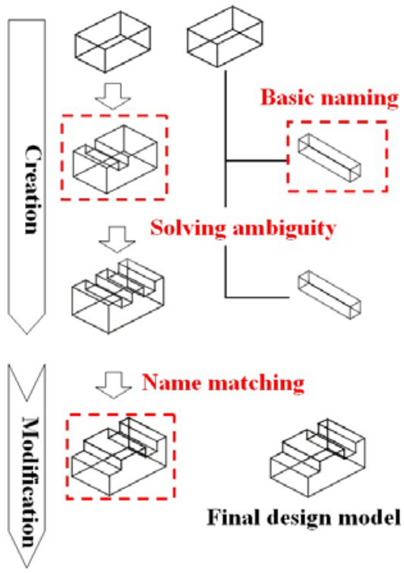

Fig. 2-2. Classification of the persistent naming problem.

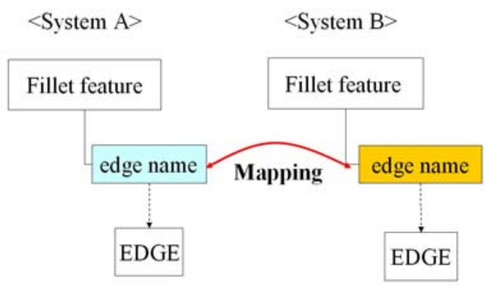

Fig. 2-3. Naming mapping problem.

2.2 Related works

Kripac [3] proposed a name matching algorithm using FaceIdGraph which is updated every time the topology of the model changes. However, Kripac's proposed algorithm is difficult to implement because details required for name matching such as the naming mechanism of faces are not addressed in the paper. Capoyleas [4] proposed a topological naming method that exploits feature specific information such as the profile and path of an extrusion feature. Based on the naming method of Capoyleas, Chen [5] proposed a name matching algorithm for vertices, edges, and faces.

Wu [6] proposed PSI (parametric space information), a geometry-based ambiguity solving method, to resolve the ambiguity problem that occurs in naming topological entities. PSI is calculated by *u*, *v* values of topological entities in parameter space and expressed as

PSI(TE) = [ON(f), Seq, Totle]

where *f* is one of the adjacent faces of the topological entity *TE* and *ON* is the basic name of *f*.

Seq is the sequence of *TE* on the face *f*, and *Totle* is the total number of topological entities that may generate ambiguity from having the same basic name. Wu did not address name matching, however. Yang [7] studied ID system implementation of a feature-based solid modeler based on Wu's naming method.

Agbodan proposed a naming method [8] and a name matching method [9] using a shell graph. The shell graph is similar to the FaceIdGraph of Kripac. However, it can reference several levels of granularity by introducing a hierarchical architecture to the graph. Hoffmann [10] proposed E-rep, a text-based high level representation method, as the main data structure for design sequences and features.

Tony Ranger [11] of the STEP parametrics group proposed a method to transfer the explicit geometry of a referenced topology in an effort to resolve the persistent naming problem that occurs during the exchange of CAD models. This approach is different from those of previous works.

Vendors of commercial CAD systems such as Pro/Engineer, SolidWorks, UG, and CATIA have studied the persistent naming problem in-house, but their implementation methods are not available to the public. The naming methods adopted in the macro file of CATIA, CATScript file, and the macro file of SolidWorks, swb file, are analyzed and used in the present study to exchange parametric CAD models. CATIA names topological entities using topological information [12], similar to the naming methods proposed in previous studies [4, 6, 8]. However, solutions for ambiguity that occurs in naming topological entities and a name matching method remain unknown. SolidWorks names topological entities using the entity types and 3D coordinates of referenced entities [13]. A solution for the name matching problem of SolidWorks has not been reported.

2.3 Comparison of persistent naming approaches

The previous approaches for the persistent naming problem can be summarized as follows. There are two approaches for the basic naming of topological entities of a feature: *topology*-based basic naming and *geometry*-based basic naming, as shown in Fig. 3. In the case of topology-based basic naming, the basic name is given to a face by utilizing input information such as sketch information and trajectory information, which are required to define features. The basic names of faces are then used to give names to edges and vertices. For example, names of two adjacent faces are used to give a name to an edge [3,6]. In Capoyleas' approach, edges and vertices are named directly using the input information instead of using face names [4]. The geometry-based basic naming method, meanwhile, identifies the selected entities by comparing the coordinates of the geometry, or the explicit geometry itself [11,13].

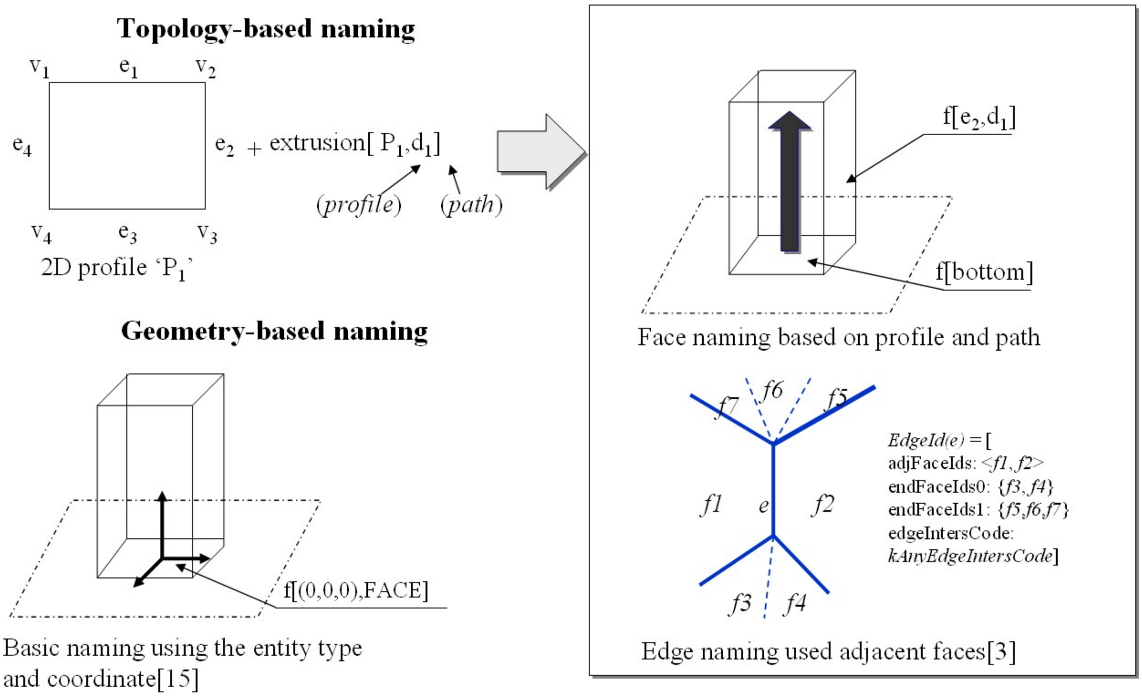

Fig. 3. Basic naming (*Problem 1.1.1*).

Topology-based basic naming may cause an ambiguity problem, whereas geometry-based basic naming does not lead to any ambiguity problem. Even if a topology-based basic naming method is applied, topological entities with the same basic name (*F1* and *F2* in Fig. 4-1, *e1* and *e2* in Fig. 4-2, and *f1* and *f2* in Fig. 4-3) can exist. An ambiguity problem arises from splitting or merging of topological entities during the process of attaching a feature to an existing model.

There are two approaches to solve this ambiguity. The first is a topology-based approach [4], shown in Fig. 4-2, which resolves the ambiguity using the order of adjacent faces to an edge. The second is a geometry-based approach [6], shown in Fig. 4-3, which resolves the ambiguity using PSI (parametric space information) (PSI of f1 = [ON(f1),1,2], PSI of f2 = [ON(f2),2,2]).

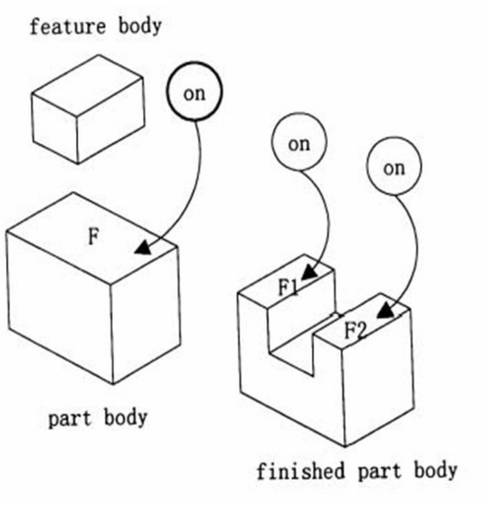

Fig. 4-1. Ambiguity problem [6] (*Problem 1.1.2*).

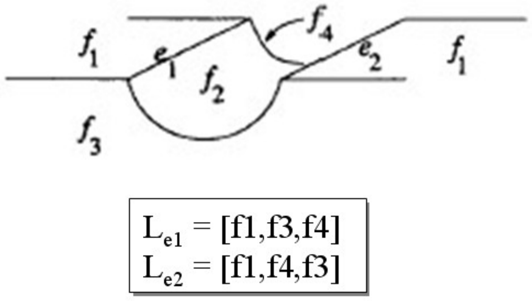

Fig. 4-2. Ambiguity solving based on topological information [4] (*Problem 1.1.2*).

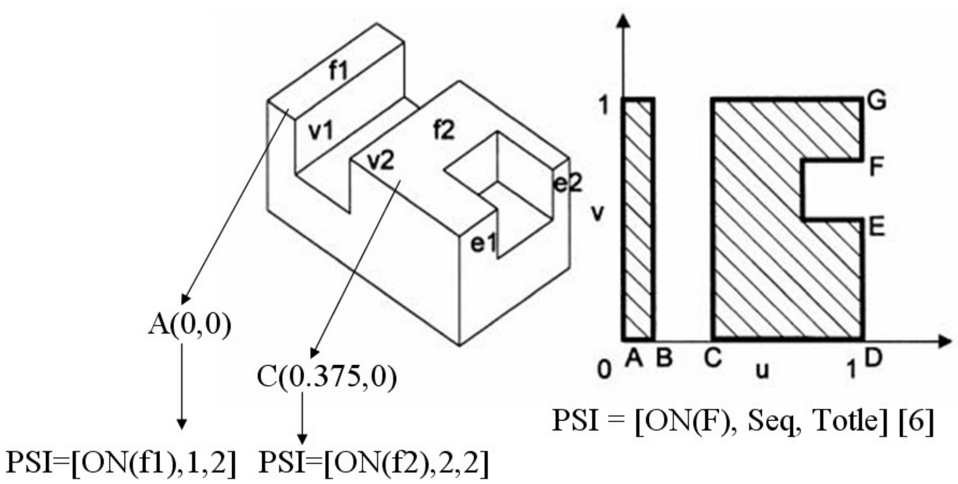

Fig. 4-3. Ambiguity solving based on geometric information [6] (*Problem 1.1.2*).

Solutions for the name matching problem (*Problem 1.2*) that occurs during re-evaluation of a model after modification can be broadly divided into *global* matching methods and *local* matching methods [14]. The local matching methods use a 1:N comparison that compares all the topological entities of the new model with the selected topological entity of the old model to find the topological entity of the new model that corresponds to the selected topological entity of the old model, as shown in Fig. 5-1. On the contrary, global matching saves the evolution history of the topological entities of the new model and the old models as shown in Fig. 5-2, and then performs a N:N comparison (the evolution history graph of the old model vs. the evolution history graph of the new model). Global matching methods are more expensive in terms of computing time, but can yield a more accurate naming mechanism [14].

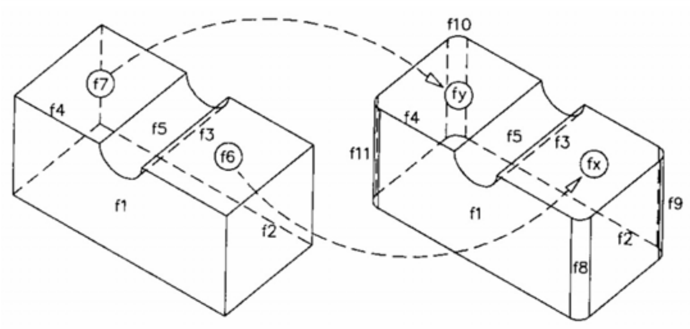

Fig. 5-1. Local matching (*Problem 2*).

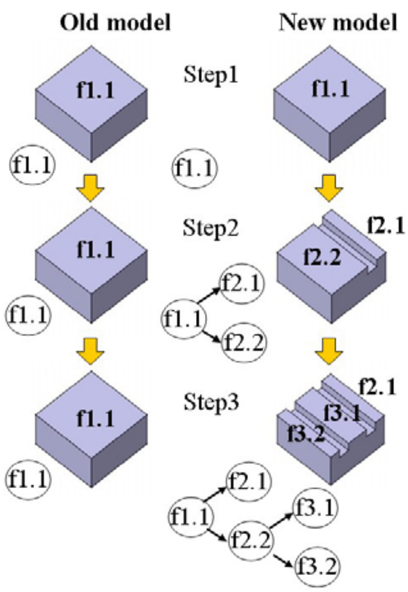

Fig. 5-2. Global matching (*Problem 2*) [3].

A series of basic naming, ambiguity solving, and name matching methods should be provided to solve the persistent naming problem. Previous studies are insufficient in this regard. Kripac and Agbodan did not address the basic naming of faces, and Capoyleas and Chen did not deal with the ambiguity problems arising from face merging or splitting. Meanwhile, Wu did not address the naming case where there are several closed loops in one profile, or the ambiguity problem arising from merging of topological entities. He also did not propose a name matching method.

3. Proposed persistent naming for the exchange of parametric models (*Problem 1*)

There are characteristics of the persistent naming problem that are specific to CAD model exchanges, and which are not found in CAD system development. Persistent naming should also be defined with the minimum data required for describing features, because limited data is available in the history-based exchange of parametric CAD models. Minimum data for the development of the parametric translator are 1) the type of feature; 2) attributes of the feature; 3) topological entities referenced by the feature; and 4) local coordinates for the placement of the feature [15].

There are two approaches for the history-based exchange of parametric CAD models: the feature construction method [16] and the macro-parametric method [17,18]. While only the construction history is exchanged by API (application programming interface) in the feature construction method, the modification history is also exchanged in the macro-parametric method. If the CAD model contains only the final version of the design, that is, the construction history, only the naming problem (*Problem 1.1*) needs to be solved. If the CAD model has the construction history and the modification history, both naming (*Problem 1.1*) and name matching (*Problem 1.2*) problems should be solved. Naming is more important than name matching.

Naming in a neutral file should also be generic and independent of CAD systems for parametric CAD model exchanges. In the case of the protrusion feature in Fig. 6, we can suppose that CAD system α names the faces in the order of start face, side faces, and end face while CAD system β does so in the order of start face, end face, and side faces. Then, the names of the second faces of the two systems will be different from each other, and a translator will not able to correctly identify the selected face. If the naming information of the neutral file depends on CAD system δ, a translator for other CAD systems will not be able to identify topological entities that are recorded in the neutral file, because the translator does not know the naming rules and algorithms of the CAD system δ.

This paper proposes a method of persistent naming and naming mapping for parametric CAD model exchanges. The scope of this study covers a feature-based solid modeling system. It is assumed that there is no modification of the profile of profile-based features, and the CAD model contains the minimum data described above.

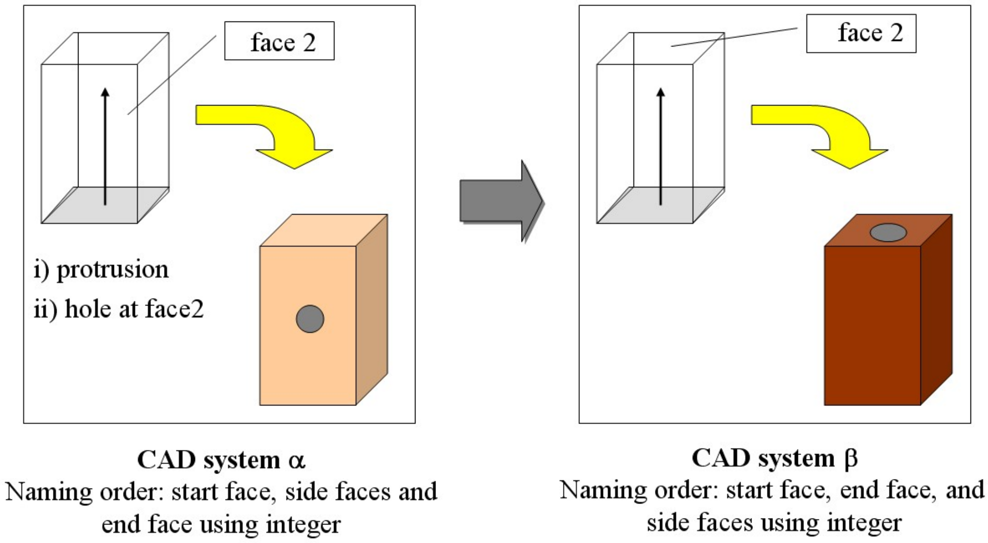

Fig. 6. Name mismatches arising from different naming order.

3.1 Proposed naming method (*Problem 1.1*)

3.1.1 Overview of the naming method

Naming, which records and retrieves topological entities in a feature-based CAD system, consists of basic naming (*Problem 1.1.1*) and ambiguity solving (*Problem 1.1.2*). The proposed naming is topology-based for basic naming and geometry-based for ambiguity solving. In the present study, Wu [6]'s method is referenced for basic naming, and a new ambiguity solving method is proposed on the basis of OSI (object space information) and SN (secondary name). The proposed naming method with OSI and SN records and retrieves topological entities as follows.

Naming of face

EN(f) = BN(f) : OSI : SN

=basic naming : OSI : merged face names

* If there is no ambiguity, [OSI] is [0,0].

* If there is no merged face, [SN] is [0;0].

Naming of edge

$EN(e) = EN(f_1)$ # $EN(f_2)$ # OSI

= name of adjacent face f_1 # name of adjacent face f_2 # OSI

Naming of Vertex

$EN(v) = EN(f_1)$ {# $EN(f_i)$} # OSI, where $i \neq 1$

= name of adjacent face f_1 {# name of adjacent face f_i} # OSI

BN(f) refers to the basic naming of each face of a feature obtained by the method explained in section 3.1.2. EN(f), EN(e), and EN(v) are the respective names of the face, edge, and vertex. The expression "**{ }**" indicates that the number of items in it can be zero, one, or more.

3.1.2 Basic naming for features (*Problem 1.1.1*)

The basic naming methods of Wu [6] and Yang [7] are used in the present study. Basic naming of faces of features is represented as follows:

BN(F) = [Feature id, id1, id2, id3, id4, id5, option]

Option represents the feature types. *id1 ~ id5* represent specific meanings that stem from the basic naming rules for features, for example, feature id, sketch id, and path id. The basic name of a sweep feature is defined as follows [6].

BN(F) =[Feature id, 0, -1, 0, 0] if F is the starting face

[Feature id, Sketch id, Sketch element id, Path id, Path element id]

if F is a side face

[Feature id, 0, -2, 0, 0] if F is the ending face

* id5=0, option = Sweep Feature

3.1.3 Solution to the ambiguity problem (*Problem 1.1.2*)

The ambiguity problem arises from merging or splitting of topological entities. An OSI (object space information) and SN (secondary name) based method is proposed to solve the ambiguity problem.

During the model creation stage, a form feature can be attached to existing features of a model. This might result in the merging of topological entities of the form feature with the existing model. SN is recorded in the form of

SN= [Total_Num, BN(f) {, BN(fi)}]

where *Total_Num* is the number of all the merged topological entities. *BN(fi)* is the basic name of a merged topological entity.

When a merging face *BN(f1_blank)* of an existing model with a merged face *BN(f2_tool)* of a new feature are merged into one face, the merged face name is recorded in SN and the value of *Total_Num* is increased by 1 [15]. After the face *BN(f1_blank)* merges with the face *BN(f2_tool)*, the merged face name becomes *BN(f1_blank):0,0:1;BN(f2_tool)*.

For the ambiguity problem arising from splitting of a topological entity, the present study proposes a method based on OSI (object space information). This method is similar to PSI, which was proposed by Wu [6]. However, OSI uses object space instead of parametric space for comparison of topological entities having the same basic name [15]. The object space refers to the local 3D Cartesian coordinates of a feature. In parametric form, each point on a face is expressed as a function of parameters u, v. The parameters act as the local coordinates at the point on the face [19]. PSI gives sequence numbers by comparing u and v values of the topological entities having the same basic name. OSI is the sequential number in the object space, which is calculated by comparing x, y, and z values of given topological entities having the same basic name. The concept for determining the sequential number of OSI is similar to that of PSI. The difference between OSI and PSI is the input information (object space vs. parametric space).

Extraction of the parametric space information from a CAD model depends on the file type of the CAD system and its API. For instance, CATScript file of CATIA and swb file of SolidWorks are used in the macro-parametric method [17, 18] and the Visual Basic API of CATIA can be utilized in the feature construction method. However, there is no parametric space information in the macro files and the Visual Basic API of CATIA does not provide a function to extract the parametric space information. If the parametric spaces vary among CAD systems, identical topological entities can have different PSI, as shown in Fig. 7. Because of the differences in the parametric space coordinates between system α and system β, the PSI value of *f1* in system α can be different from that of system β. Therefore, PSI is not suitable for the exchange of CAD models.

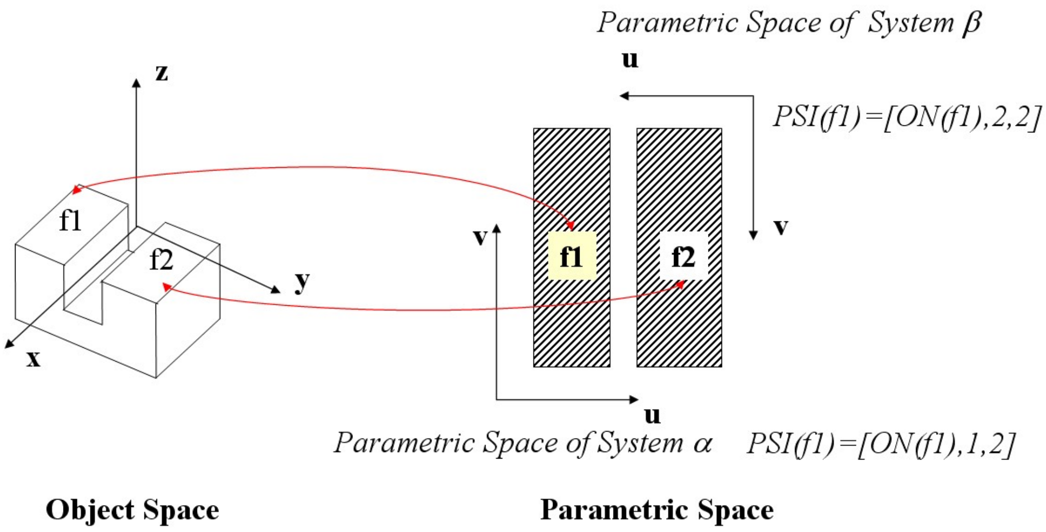

Fig. 7. Difference in PSI (parametric space information) among CAD systems [15].

OSI is expressed as

OSI= [Order, Total_Num]

where *Order* is the sequence of a given entity and *Total_Num* is the number of all the topological entities that may generate ambiguity.

OSI is calculated as follows:

- Determine reference points on the topological entities having the same basic name
- Calculate their *x, y,* and *z* values in the object space.
- Sort all the topological entities according to the *x*, *y*, and *z* values. Topological entities with larger *x*, *y*, and *z* values have priority. First, sort the topological entities by comparing *x* values. For topological entities having the same *x* value, sort them by comparing *y* values. For topological entities having the same *x* value and *y* value, sort them by comparing *z* values.

A detailed explanation is given by Wu [6] with regard to the selection algorithm of the reference point. While a 3D bounding box in the object space is used in the present paper to determine the reference point, a 2D bounding rectangle in the parametric space is utilized in Wu's method.

In Fig. 8, *f1* and *f2* have the same basic name. The bounding boxes of *f1* and *f2* do not overlap with each other and the *y* value of *f2* is larger than that of *f1*. Therefore, *f2* is the first, and thus OSI of *f2* becomes [1,2] and OSI of *f1* is [2,2].

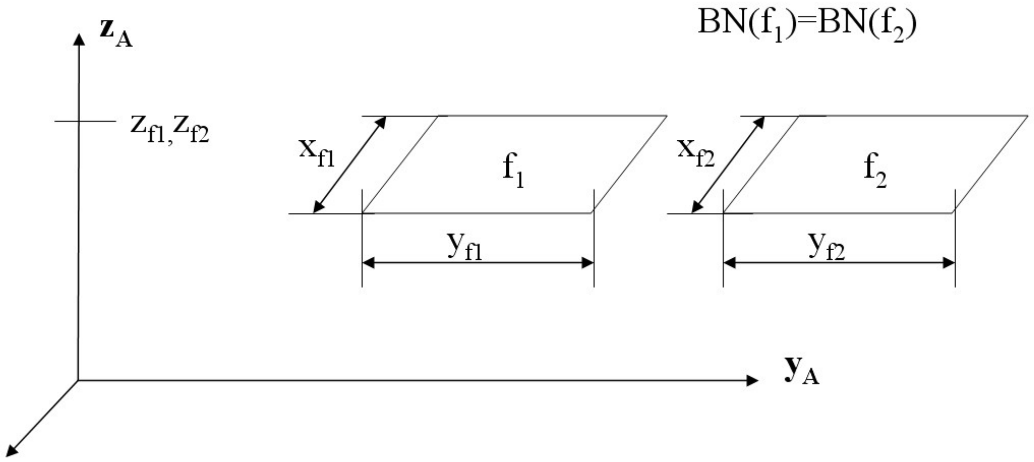

Fig. 8. Object Space Information (OSI) [15].

3.2 Proposed name matching method (*Problem 1.2*)

A new name matching method is proposed according to the proposed naming method explained in section 3.1.

3.2.1 Face matching

The basic name of a selected face of the old model (pre-edit model) is compared with those of all the faces of the new model (post-edit model) to find the face of the new model that corresponds to the selected face of the old model. The faces with the same basic name are added to a *candidate matching set.* If the new model has no face with the same basic name, SN (secondary name) of the faces of the new model is searched. When the SN of a face has an identical name with the basic name of the selected face, this face is added to the candidate matching set.

The entities of the candidate matching set have identical basic names with the selected face of the old model. After investigating the ranges of *x, y,* and *z* coordinates of the local coordinates of the selected face of the old model, a set of faces can be found from the candidate matching set, which has a similar range [15]. Here, the local coordinates of a face correspond with the local coordinates of the feature that contains the face. If a face A is one of the faces in a feature F, then the local coordinates of face A are the local coordinates of feature F. Comparison results belong to four categories between the entities of the candidate matching set and the selected face of the old model, as shown in Fig. 9. For each category, the matching criterion is given below.

If the object space of topological entity A of the new model is contained in that of topological entity B of the old model, as in Fig. 9(a), entity A is returned. The object space of another topological entity of the new model may exist inside that of entity B, as indicated by the dotted circle of Fig. 9(a). If the object space of topological entity A of the new model contains that of topological entity B of the old model, as in Fig. 9(b), entity A is returned. It is the only entity that matches the selected entity of the old model. If the object space of topological entity A of the new model overlaps partially with that of topological entity B of the old model, as in Fig. 9(c), entity A is returned when the ratio of the overlapping area to the whole area of entity A is larger than a predefined value. Finally, if the object spaces of both models do not overlap, as in Fig. 9(d), no topological entity of the new model is returned.

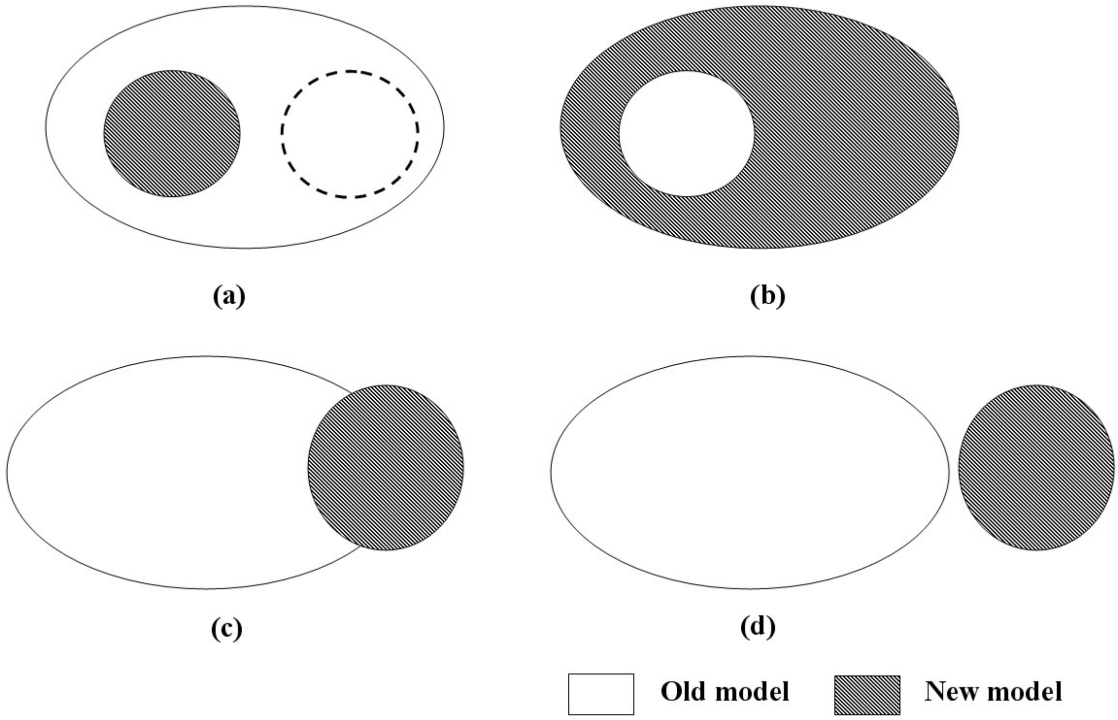

Fig. 9. Comparison between object space range of old model and new model [15].

Fig. 10 shows an example of a name matching problem that originates from a face splitting. The top face of the old model is split into three different faces in the new model. Faces having an identical basic name are first searched in the new model to match the upper face BN(f1):0,0:0;0 of the old model. These are the upper three faces (BN(f1):1,3:0;0, BN(f1):2,3:0;0, BN(f1):3,3:0;0). A candidate matching set is subsequently constructed. After the object space of the upper face of the old model is compared with that of the faces of the candidate matching set, all three faces are returned, as they all correspond to the case illustrated in Fig. 9(a).

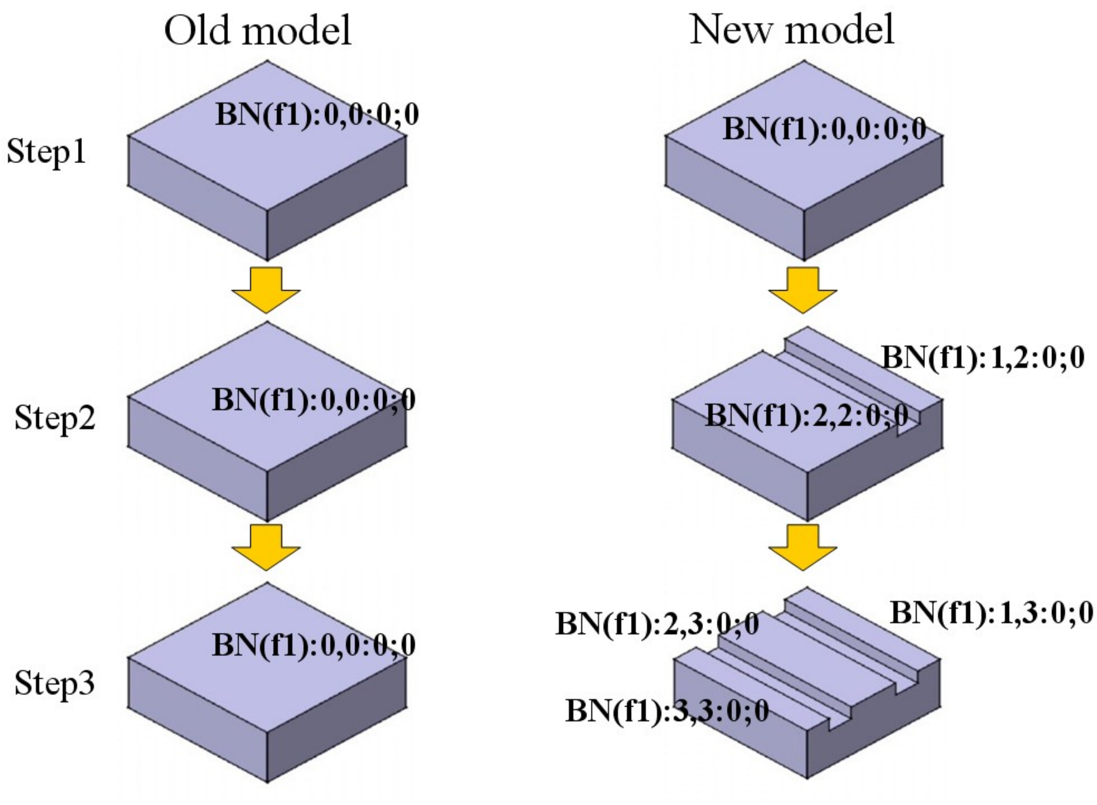

Fig. 10. Name matching problem arising from face splitting [15].

3.2.2 Edge and vertex matching

An edge or vertex name consists of adjacent face names and OSI, as explained in section 3.1. For edge matching and vertex matching, a face matching table is constructed for edge or vertex matching. The face matching table consists of faces of the new model that match with adjacent faces of the selected edge or vertex of the old model. From this table, edges or vertices of the new model are searched and added to a candidate matching set, which have the same adjacent faces with the selected edge or vertex of the old model. In the case of edge matching, the object space of the selected edge of the old model is compared with each edge in the candidate matching set. The edge of the new model, corresponding to the edge of the old model, is then returned. The category map shown in Fig. 9 is used to compare the similarity of the object space range between two edges. In the case of vertex matching, the returned vertex is determined by a comparison of closeness in the object coordinates between the selected vertex of the old model and each vertex in the candidate matching set, because the vertex does not have a range in object space.

Fig. 11 shows a matching example of edge *e1*. First, a face matching table is constructed for edge *e1*. *BN(f1):0,0:0;0* of the old model matches with *BN(f1):0,0:0;0* of the new model. *BN(f2):2,2:0;0* of the old model matches with 3 faces (*BN(f2):2,4:0;0*, *BN(f2):3,4:0;0*, *BN(f2):4,4:0;0*) of the new model. A candidate matching set is then constructed by finding the edges of the new model that have identical adjacent faces with *e1* of the old model. Edges *e11, e12,* and *e13* are found and added to the candidate matching set from this process. If the object space of *e1* is compared with that of *e11, e12,* and *e13*, then these edges belong to the category of Fig. 9(a), because the object spaces of *e11, e12,* and *e13* are contained in the object space of *e1*. Therefore, the edge matching indicates that *e11, e12,* and *e13* of the new model match with *e1* of the old model.

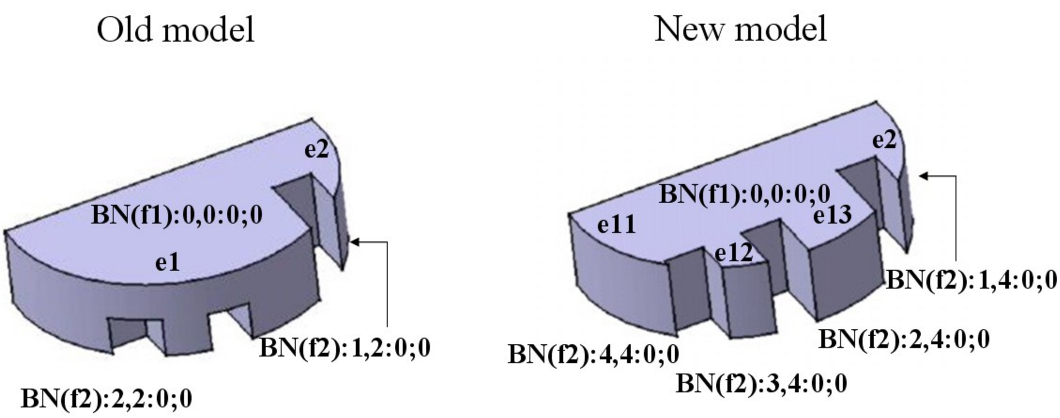

Fig. 11. Name matching problem arising from edge splitting [15].

4. Naming mapping between different naming methods (*Problem 2*)

In the present study, in order to translate CAD models, a macro file or a native part file generated by the sending CAD system is translated into a XML formatted neutral macro file [18] and then the neutral macro file is again translated into a macro file or a native part file of the receiving CAD system. The proposed naming method is used to identity the selected topological entities of a feature in the neutral macro file. Commercial CAD systems have different naming schemes. A naming mapping method should be provided between the proposed naming method and the naming method of each CAD system to exchange parametric CAD models.

Naming consists of basic naming and ambiguity solving. Naming mapping between different naming methods should have solutions for mapping between different basic naming methods and between different ambiguity solving methods.

Basic naming and ambiguity solving approaches are classified into geometry-based and topology-based methods. Naming mapping between different naming methods should be provided depending on these categories. The mapping cases between these naming methods are therefore as follows:

- Topology vs. Topology
- Geometry vs. Geometry
- Topology vs. Geometry

Mapping between a geometry-based method and a topology-based method is an operational mapping [20] in which operations are needed because of semantic and syntax differences between them. It is not possible to directly define or simply map relationships between two schemas in this case.

The present study proposes IGM–based naming mapping. IGM (internal geometric model) is a geometric model generated by a geometric modeling kernel such as ACIS [21] according to the design history in the form of the feature tree or the user-commands list of an input CAD model. The persistent names based on the proposed naming are attached to each face of the IGM. The IGM is generated at the same time as when CAD data are exchanged. The design history (modelling history) of a parametric CAD model contains feature specifications and selection information used to create each feature, such as an edge referenced by a chamfer feature. The name of the referenced topological entity is recorded in the selection information. The selection information should be converted into the format of the receiving CAD system for the exchange of parametric CAD models. The necessary information for naming mapping is retrieved or calculated from the IGM.

Basic naming of the proposed naming method is topology-based and ambiguity solving of that is geometry-based. Naming mapping between the proposed naming method and the topology-based naming starts with the generation of an IGM, as shown in Fig. 12. The main processes of naming mapping between the proposed naming method and the topology-based naming are 1) mapping of basic names; and 2) comparison or calculation of ambiguity solving information, that is, mapping of ambiguity solving information.

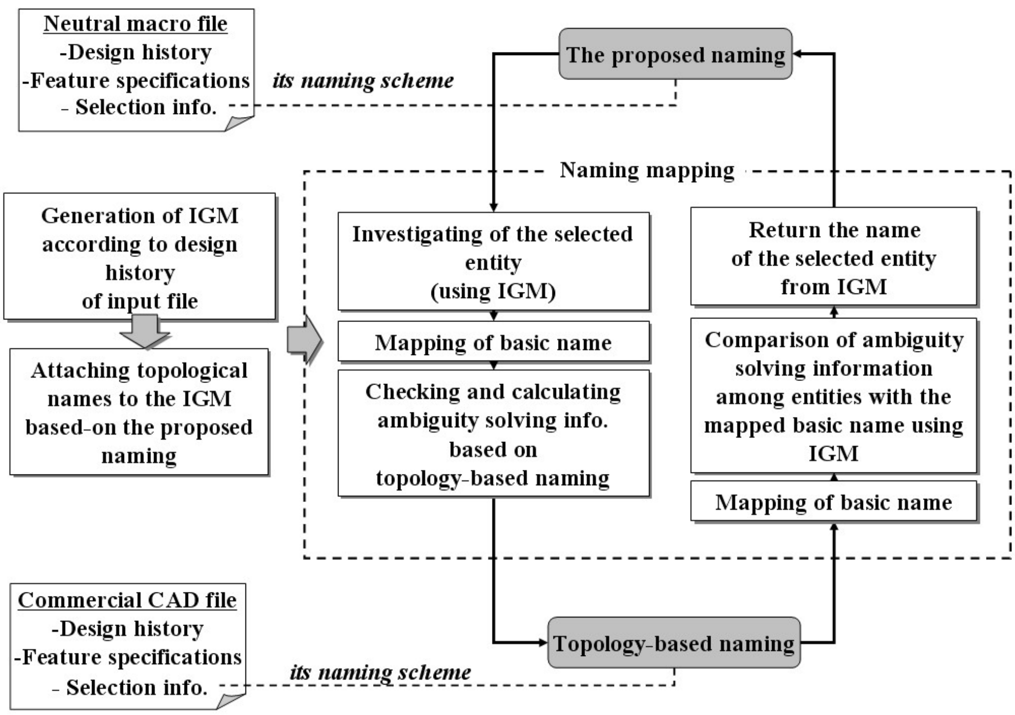

Fig. 12. Naming mapping between proposed naming and topology-based naming.

According to the modeling history of an input file, the IGM is generated using a geometric modeling kernel, and names are attached to faces of the IGM on the basis of the proposed naming.

In the case of mapping from a name with the proposed naming to a topology-based name, the entities having an identical name with the input name are searched from the IGM. A basic name based-on the target topology-based naming can be defined from feature specifications of the input file and IGM. After mapping of the basic name, checking and calculating ambiguity information according to the target topology-based naming follow.

In the case of mapping from a topology-based name to a name with the proposed naming, mapping of the basic name is conducted first. With the mapped basic name in the form of the proposed naming, a candidate set of entities having the same basic name is searched. The returned entity is determined by comparison between ambiguity solving information of topological entities in the candidate set and that of the input name using IGM.

Topology-based basic naming methods are divided into two categories. In *face-based basic naming* [3,6], the basic name is given to a face by utilizing input information such as sketch and trajectory information. The face name is then used to give names to edges and vertices. In *profile-based basic naming* [4], edges and vertices are named directly using the input information instead of using face name.

Mapping between topology-based basic naming methods is possible without an IGM. The basic approach of this type of mapping involves expressing a vertex in a sketch as the intersection of two edges. Vertex *v3* in Fig. 13 is the intersection of edge *e2* and edge *e3* and can be represented as *e2&e3*. On the basis of this relation, the profile-based basic naming can be mapped to the face-based basic naming. Edge *E*, "*e(v3)*" in Fig. 13, is an edge created by extruding the vertex *v3*. The basic name of edge *E* can be represented as *f(e2)&f(e3)* from the relation "*v3 = e2&e3*".

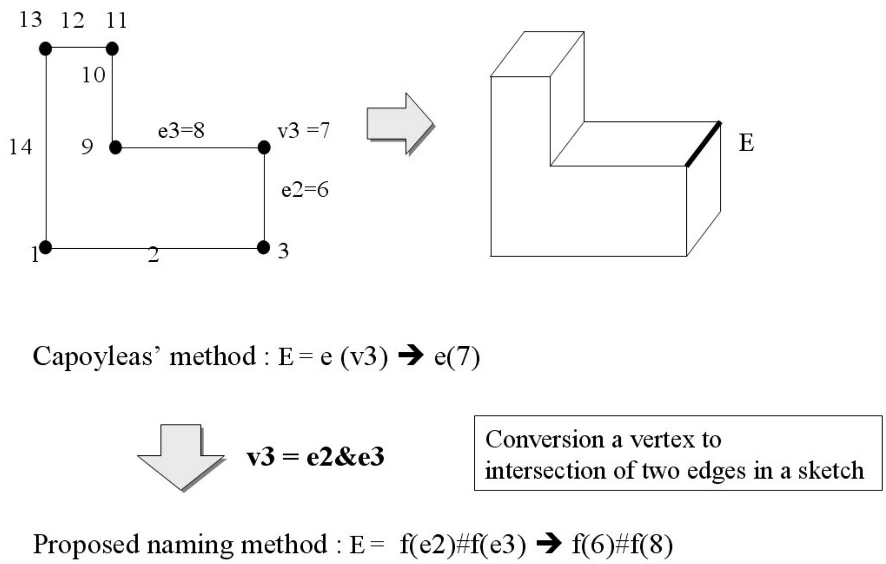

Fig. 13. Mapping of basic name.

Solutions to the ambiguity problem are divided into topology-based [4,6] and geometry-based methods [3,4,15]. Procedures for mapping of ambiguity solving information are as follows. After an IGM is generated according to the design history of an input file, names are attached to faces of the IGM based on the proposed naming method. A candidate set is constructed such that it consists of topological entities having the same basic name through the mapping of basic names. Ambiguity solving information of each topological entity in the candidate set is compared with that of input naming information using the IGM, and then the topological entity that will be returned is determined. Finally, ambiguity solving information of the returned topological entity is calculated or retrieved on the basis of the target naming method from the IGM.

An example of mapping the ambiguity solving information from Capoyleas's method (local orientation) to that from the proposed method (OSI) is illustrated in Fig. 14. The input naming information is *f1* as the basic name and $L_{f1}=[f3,f2,f4]$ as the ambiguity solving information, and names are given to the IGM in accordance with the proposed naming method. *Left face* and *right face* are selected to constitute the candidate set after mapping of the basic name. *Left face* is found to have the same ambiguity solving information with ($L_{f1}=[f3,f2,f4]$). This is found through searching adjacent faces of the given face by using the IGM. The ambiguity solving information of *left face*, "*OSI= [1,2]*", is then retrieved from the IGM.

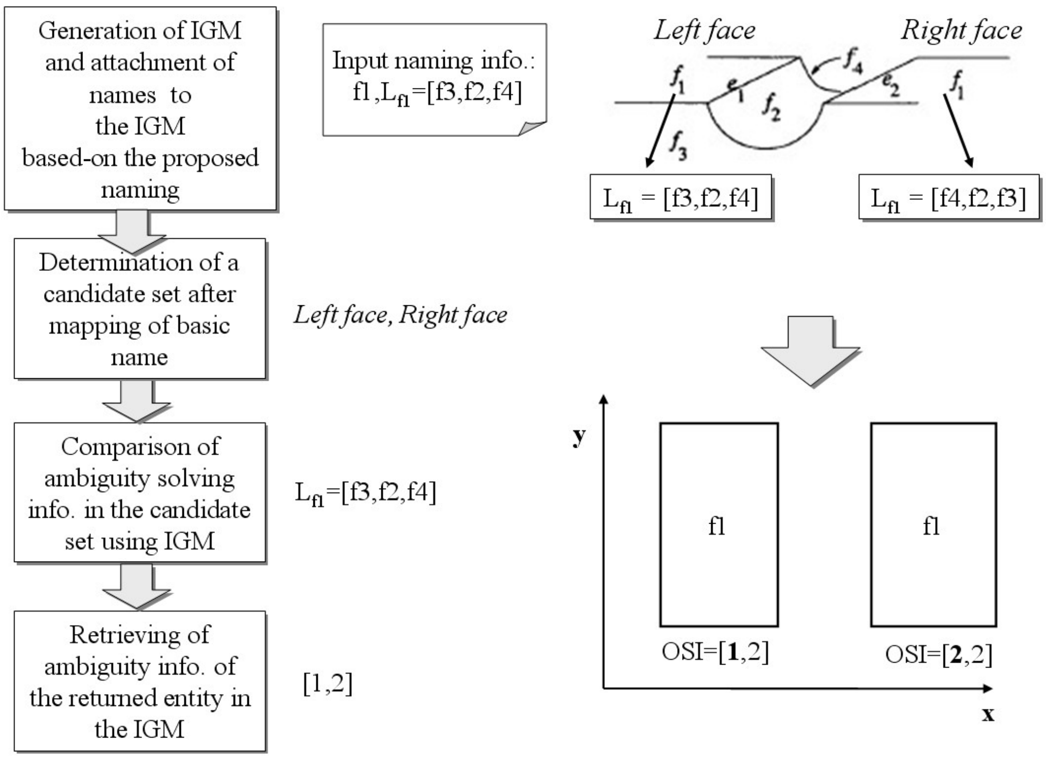

Fig. 14. Mapping ambiguity solving information.

Naming mapping between the proposed naming method and the geometry-based naming method starts with the generation of an IGM, as shown in Fig. 15. According to the modeling history of the input file, the IGM is generated using a geometric modeling kernel, and names are attached to faces of the IGM on the basis of the proposed naming. In the case of mapping from a name with the proposed naming to a geometry-based name, entities having an identical name with the input name are searched from the IGM. A geometry-based name can be generated using the geometric information of the selected entity. In the case of mapping from a geometry-based name to a name with the proposed naming, entities having identical geometric information with the input name are searched using the IGM. If an entity with the same geometric information exists, the entity name attached to the IGM is transferred.

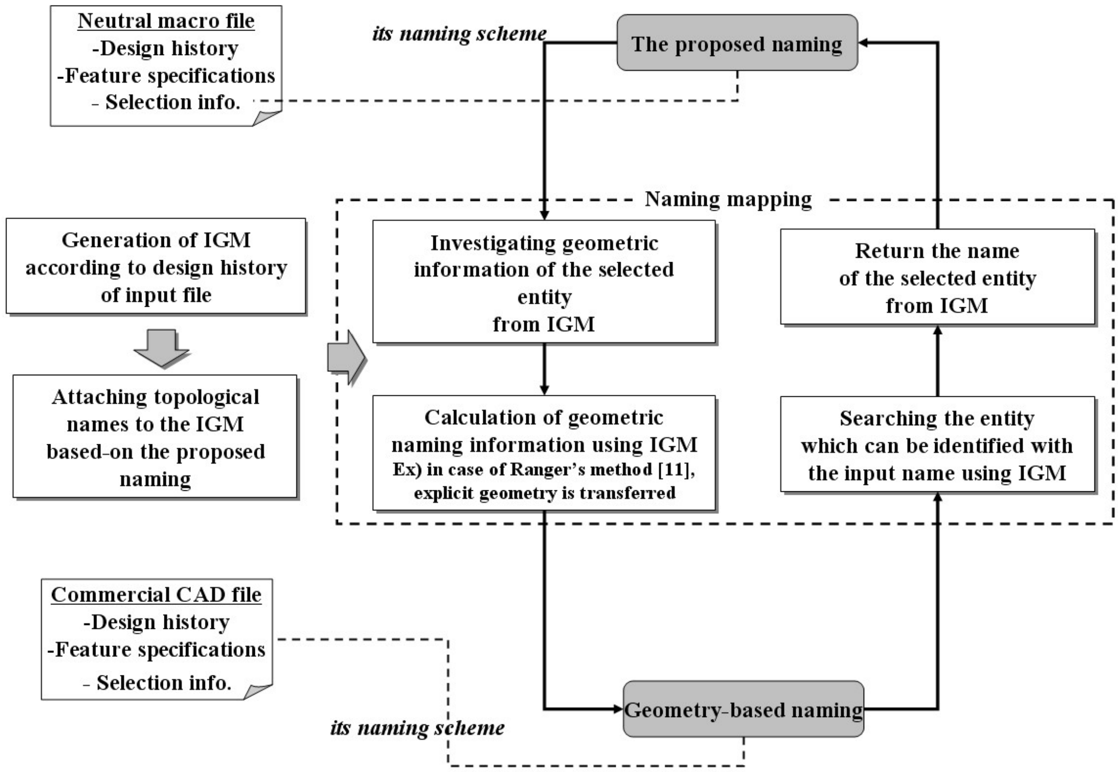

Fig. 15. Naming mapping between proposed naming and geometry-based naming.

An example of naming mapping from the naming of SolidWorks to the proposed naming is illustrated in Fig. 16. Cartesian coordinates of picking points and the types of picked entities are written in the macro file of SolidWorks while names of picked entities in accordance with the proposed naming are written in the neutral macro file. The bold edge is selected after a cube *pad1* is generated by extruding a sketch (an extrusion feature) in Fig. 16. To exchange a feature-based CAD model from SolidWorks to the neutral macro file, the naming mapping procedure for this edge with an IGM is as follows.

- The design history contained in the input SolidWorks macro file is 1) creation of *sketch1*; 2) creation of *pad1*; and 3) selection of the bold edge. The name of the selected edge is defined as ***"EDGE", 10,10,5*** according to the naming of SolidWorks.
- An IGM is generated according to the design history (A. creation of *sketch1*, B. creation of *pad1*) of the input SolidWorks macro file and by using a geometric modeling kernel. The

name based on the proposed naming is then attached to every face of the IGM. For instance, face *f1* of the IGM is named "***pad1, sketch1, line3, 0, 0, 0, Extrude_Feature : 0, 0 : 0; 0***" and face *f2* is named "***pad1, sketch1, line2, 0, 0, 0, Extrude_Feature : 0,0 : 0;0***". This case does not require ambiguity solving because there is no splitting or merging of faces.

- Find the edge that passes the Cartesian point (10, 10, 5) in the IGM and return the edge.
- The edge name *EN(e)* is defined as "$EN(f_1)$ # $EN(f_2)$ # *OSI*". Retrieve the names of two adjacent faces of the returned edge and check whether the returned edge has ambiguity, that is, whether there is another edge of which adjacent faces have same names with those of the returned edge. This case has no ambiguity and does not require ambiguity solving.
- The name of the returned edge is determined to be "***pad1, sketch1, line2, 0, 0, 0,Extrude_Feature : 0,0 : 0;0 #pad1,sketch1, line3, 0, 0, 0, Extrude_Feature : 0, 0 : 0; 0 # 0,0***" in accordance with the proposed naming method.

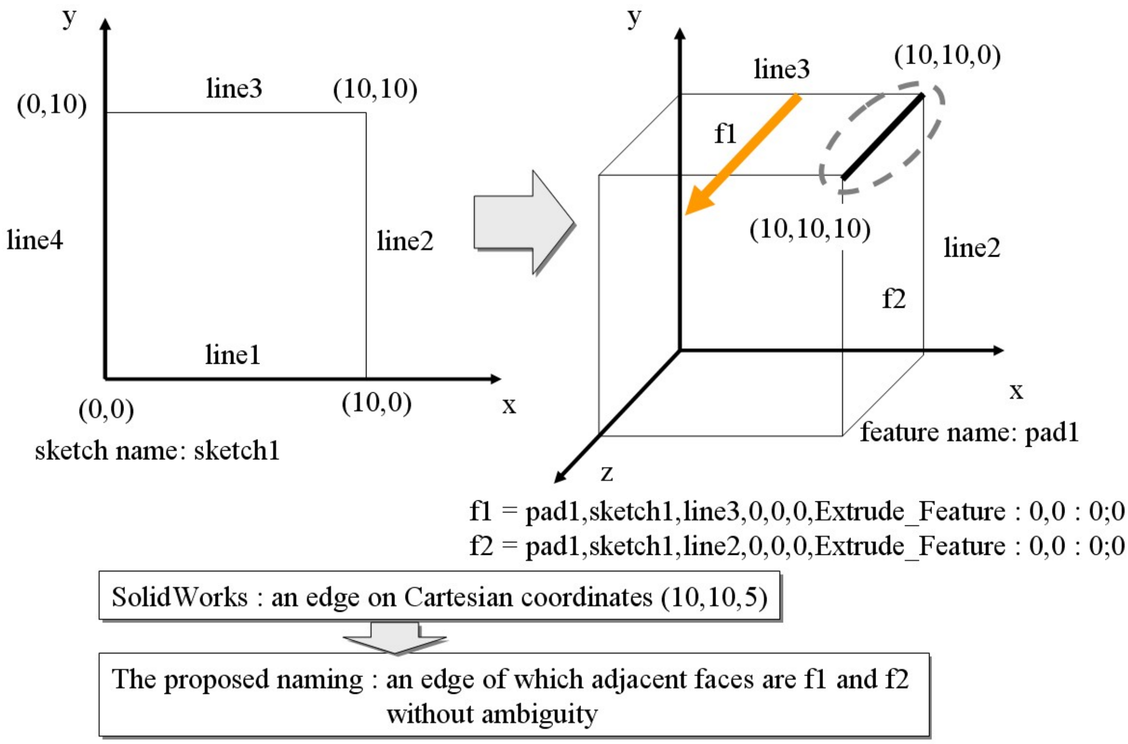

Fig. 16. Naming mapping between SolidWorks and the proposed naming.

IGM is used not only for naming mapping among different CAD systems, but also for extracting necessary geometric information for the mapping of different feature definitions. The proposed method has the disadvantage that an IGM should be generated and managed inside the translator such that it has the same geometry and topology as the input CAD data.

5. Conclusions

Persistent naming and naming mapping problems occur during the exchange of CAD models that have parametric information. The present study analyzes the characteristics of the persistent naming problem from the viewpoint of exchanging CAD models. Based on the analysis of related works, a persistent naming method and a naming mapping method are proposed. The proposed *persistent naming* method is based on OSI (object space information) and SN (secondary name). OSI is used to solve the ambiguity problem that arises from entity splitting. SN is used to solve the ambiguity problem that arises from entity merging. The proposed *naming mapping* method uses IGM. IGM is a geometric model generated by a geometric modeling kernel according to the design history. The design history is in the form of a feature tree or a macro file, and the necessary information for naming mapping is retrieved or calculated from the IGM.

The persistent naming method and naming mapping method proposed in the present study have been adopted into a macro parametric translator [17, 18]. Parametric information has been successfully transferred in an experimental implementation.

References

1. J. Rossignac and A. Requicha, "Solid modeling", in the Encyclopedia of Electrical and Electronics Engineering, Ed. J. Webster, John Wiley & Sons, 1999.
2. Hayong Shin, “Classification of geometric model”, IE752 geometric modeling class in KAIST, 2002
3. J. Kripac, "A mechanism for persistently naming topological entities in history-based parametric solid models", Computer-Aided Design, Vol. 29, No.2, pp 113-122, 1997.
4. V. Capoyleas, X. Chen and C. M. Hoffmann, "Generic naming in generative, constraint-based design", Computer-Aided Design, Vol. 28. No. 1, pp 17-26, 1996.
5. X. Chen and C. Hoffmann, "On editability of feature-based design", Computer-Aided Design , Vol. 27. No. 12, pp. 905-914, 1995.
6. Junjun Wu, Tianbing Zhang, Xinfang Zhang and Ji Zhou, "A face based mechanism for naming, recording and retrieving topological entities", Computer-Aided Design, Vol. 33, No. 10, pp 687-698, 2001.
7. Changyoon Yang and Kunwoo Lee, “Development of ID system for feature-based solid modeler (in Korean)”, Conference of the Society of CAD/CAD Engineers, Korea, 2003.
8. D. Agbodan, D. Marcheix and G. Pierra, "Persistent naming for parametric models", International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, 2000.
9. D. Agbodan, D. Marcheix G. Pierra and C. Thabaud, "A topological entity matching technique for geometric parametric models", International Conference on Shape Modeling and Applications, 2003.

10. C. M. Hoffmann and R. Juan, "Erep, an editable, high-level representation for geometric design and analysis" in Wilson, P, Wozny, M and Pratt, M (eds.) Geometric and Product Modeling, North-Holland, pp. 129-164, 1993.
11. ISO TC184/SC4/WG12 N1568, “Minutes of WG12 parametrics meeting from Seoul, KOREA”, 2002
12. Dassault systemes, “CATIA V5 R6 online manual”, 2001.
13. SolidWorks homepage, “http://www.solidworks.com/”, SolidWorks Corporation, 2002.
14. David Marcheix and Guy Pierra, "A survey of the persistent naming problem", 7th ACM Symposium on Solid Modeling and Applications, SM2002, 2002.
15. Duhwan Mun and Soonhung Han, “A persistent naming method for parametric CAD model exchanges”, Computer-Aided Design , Submitted, 2005.
16. Joe Greco, “Working magic with translation and healing”, CADENCEweb, http://www.cadenceweb.com/magazine, October, 2002.
17. Gukheon Choi, Duhwan Mun and Soonhung Han, “Exchange of CAD part models based on the macro-parametric Approach”, International Journal of CAD/CAM, Vol.2, No.1, pp.13-21, 2002.
18. Duhwan Mun, Soonhung Han, Junhwan Kim and Youchon Oh, “A set of standard modeling commands for the history-based parametric approach”, Computer-Aided Design, Vol. 35, No. 13, pp. 1171-1179, 2003.
19. Ibrahim Zeid, CAD/CAD theory and practice, McGRAW-HILL, 1991.
20. Youchon Oh, Soonhung Han and Hyowon Suh, “Mapping product structures between CAD and PDM systems using UML”, Computer-Aided Design, Vol. 33, No. 7, pp. 521-529, 2001.
21. Spatial homepage, “http://www.spatial.com/components/acis/”, 2005.