Repairing CAD model errors based on the design history

Jeongsam Yang (jyang@ajou.ac.kr)
Industrial & Information Systems Engineering, Ajou University, Korea
and Soonhung Han (shhan@kaist.ac.kr)
Department of Mechanical Engineering, KAIST, Korea
ABSTRACT

For users of CAD data, few things are as frustrating as receiving unusable, poor quality data. Users often waste time fixing or rebuilding such data from scratch on the basis of paper drawings. While previous studies use the boundary representation (B-Rep) of CAD models, we propose an approach to repairing CAD model errors that is based on the design history. CAD model errors can be corrected by an interdependency analysis of the feature commands or of the parametric data of each feature command, as well as by a reconstruction of the feature commands through rule-based reasoning of an expert system. Unlike other correction methods based on B-Rep models, our method repairs parametric feature models without translating them to a B-Rep shape, and it also preserves parametric information.

Keywords: Boundary representation; CAD model; Correction; Design history; Interoperability
1. Introduction

Because of competition in the market, the lead time for designing and manufacturing products is decreasing, whereas the complexity of products is increasing. Furthermore, as the globalization of companies continues, the development of products surmounts the limitations of geographical boundaries and is enhanced by collaborative design in a distributed environment where there are frequent exchanges of product data for common parts and components.

When developing products, designers and engineers often encounter poor quality product data, and they waste considerable time and money overcoming this problem. One company, for instance, reported that it spent as much as 50 percent of its time repairing product models in downstream functions, such as rapid prototyping, finite element analysis, and CNC programming. The company had to deal with poor quality CAD data that were not properly modeled for use in these downstream functions. In another example, an original equipment manufacturer estimated that, within the company and between the company and its suppliers, it had as many as 453,000 exchanges of product data each year, and that it spent an average of 4.9 hours solving each poor-quality problem. As for the cost of imperfect interoperability, the members of the US automotive supply chain reportedly spend at least $1 billion a year [1]. Similarly, in the Japanese automotive industry, which has as many as 250,000 product data exchanges a year, the cost of these exchanges is approximately $68 million a year, and 1.5 hours of lead time are lost whenever one item of poor quality or unusable data needs to be repaired or replaced [2].
Previous studies on poor-quality CAD models have focused on the boundary representation (B-Rep) of 3-D shapes [3-8]. They corrected errors by mathematically analyzing the topological and geometric elements of the B-Rep model. Most commercial applications use the B-Rep data to analyze and correct CAD model errors. However, although the B-Rep approach is effective for locating errors with the maximum or minimum tolerance as defined by the designer, an unstable topological structure can unintentionally distort a repaired B-Rep shape or cause the shape to collapse. As a result, only limited types of errors can be corrected, and automation of the correction process is difficult. Designers are therefore reluctant to correct errors with commercial applications that are based on the B-Rep approach.

To repair errors, we now propose a method that can reconstruct the design history of a CAD model. The design history refers to the chronological order in which a designer created the various features of a 3-D shape. In a CAD system, there are different ways of defining a shape. Thus, the design includes the geometry-controlling parameters, the geometric design features, the feature information, the design history tree, the parameterization data, and the constraints. We therefore analyzed several 3-D parametric models from automotive companies to define the relationship between the various feature commands, as well as the relationship between the parametric data of each feature. We also defined a design history schema in order to structure the design history information extracted from a 3-D model created in a commercial CAD system. We then used the schema to repair the CAD model through the rule reasoning of the design history. Finally, to verify the proposed method, we developed a CAD model correction system called Q-Raider, which can repair the following six types of CAD
model errors: tiny faces, narrow regions, non-tangent faces, narrow steps, sharp face angles, and narrow spaces.
2. Related works

2.1 Error corrections based on B-Rep

Previous studies on correcting CAD model errors can be classified into an exact B-Rep approach, a faceted B-Rep approach, and a boundary curve–based approach. In the exact B-Rep approach, CAD model errors are checked and corrected through a mathematical computation of a data structure that represents the topological and geometric elements of the B-Rep model. Hoffman et al. [3] proposed the architecture for a master model, and their architecture combines with downstream application processes to produce different views of features. To correct the problem of geometric tolerance between different downstream applications, the architecture of Hoffman et al. uses a call-back mechanism that the user can control. Gu et al. [4] sequentially sorted topological entities by using a complementary model object tree, and they used the tree to correct topological errors.

The faceted B-Rep approach, which approximates the exact B-Rep model in terms of a polyhedron, corrects errors faster than the exact B-Rep approach. Barequet et al. [5] proposed a geometric hashing algorithm that sequentially reconfigures polygons after dividing the trimmed surface of a complex 3-D shape into unordered lists of polygons. The hashing algorithm corrects errors by stitching the small gaps between the polygons.

The boundary curve–based approach can only be used for 3-D shapes made of surfaces. Steinbrenner et al. [6] checked and corrected the gaps or overlaps between adjacent curves in a 3-D shape that consisted of curved surfaces with various degrees. This method checks and corrects the gaps or overlaps through an edge-splitting and merging process after the boundary curve has been divided into small edge curves.
correct the $G^1$ discontinuity on surfaces (that is, the non-tangent angle between adjacent surface patches), Volpin et al. [7] simplified the original free-form surface model: they first divided the regions on the basis of curvature variation; next, they generated a boundary-conforming finite element quadrilateral mesh of the regions; and, finally, they fitted a smooth surface over the quadrilateral mesh.

Most studies discuss the B-Rep shape of CAD models and a B-Rep correction can be applied to a limited number of error types such as gaps and overlaps. Moreover, there is inevitably a loss of data when the shape is being simplified by the B-Rep approach.

### 2.2 Persistent naming problems

Although there is no previous study on checking and repairing CAD models on the basis of the design history, considerable attention has been given to research on the persistent naming problems associated with the exchange and modification of parameterized feature-based CAD models [8-16].

The literature reveals two approaches to persistent naming problems: topological and geometric. With respect to topological naming, Kripac [8] proposed a topological ID system in which IDs are assigned to topological entities (such as faces, edges, and vertices) in solid models. When the design history is modified and then automatically reevaluated to produce a new solid model, the Kripac system uses two searching algorithms (namely a forward algorithm and a backward algorithm) to ensure that the IDs of the topological entities in the previous model are mapped to the IDs of the corresponding topological entities in the new model. However, Kripac’s algorithm is difficult to implement and he failed to address details such as the mechanism of naming faces.
After proposing a topological naming method based on faces, Wu et al. [9] presented the concept of parametric space information to solve the topological ambiguity that occurs when naming, recording and retrieving topological entities; however, they ignored name matching and the ambiguity that arises from face merging.

Mun et al. [10] proposed a disambiguation method based on object space information and secondary names. Instead of using parametric space to compare topological entities with the same name, Mun et al. used object space information, which is similar to Wu’s parametric space information. Basically, they used the object space of a feature’s 3-D Cartesian coordinates to record and retrieve topological entities.

Capoyleas et al. [11] proposed a topological naming method that exploits feature-specific information such as the profile and path of extrusions. To solve the ambiguity problem, Capoyleas et al. used information on either the local orientation of adjacent topological entities, such as the edge or the face, or on the feature orientation, such as the extrusion path or rotational axis.

Vergeest et al. [25] endeavored to anticipate the feasibility of interoperability in an approach that systematically analyzes and models the requirements of a shared infrastructure. An inherent incompatibility between different CAX models exists in CAD translations due to system-specific modeling functionalities. For example, the hole creation command of Pro/E cannot be directly translated to CATIA because Pro/E provides more diverse ways to generate hole features than CATIA. Such incompatible modeling commands can be mapped through 1:N or N:1 mappings.

The geometric naming method identifies selected entities by comparing geometric coordinates. While this method raises no problems of ambiguity, it does have problems with matching names. To resolve the naming problem that occurs whenever CAD
models are exchanged, Ranger [12], who is a member of the STEP parametrics group, proposed a method of transferring the explicit geometry of a referenced topology. The commercial CAD system SolidWorks, on the other hand, persistently names topological entities by using the entity types and 3-D coordinates of the referenced entities [13].

In another approach, Rapporport [14] focused on resolving ambiguity and suggested several solutions such as special entity geometry, carrier signs, adjacency, and local ordering. He proposed a generic geometric complex, which is a modeling scheme for families of geometric objects; he also proposed an invariant naming method of the generic geometric complex, which is a more general and flexible method than other methods in the literature. Moreover, Rapporport [15] proposed a new type of architecture for exchanging parameterized feature-based CAD models: namely universal product representation, which supports many of the data levels used by current CAD systems.
3. CAD model correction based on design history

3.1 Classification and checking of CAD model errors

CAD model errors fall into three groups: errors that come from the numerical inaccuracy of a shape; errors that occur in downstream functions as a result of design models; and errors that may be due to the poor design skills or mistakes of designers.

With the first group of errors, we can mathematically compute the topological and geometric elements to check the numerical inaccuracies, such as the $C^0/G^1/G^2$ discontinuity, the small and void faces, and the self-intersections. Using a computation based on the allowance values set by the designer, we can then correct the inaccuracies by modifying the topological and geometric elements to conform with the data structure of the CAD system in use. These types of errors, which are caused by mismatches in the numerical inaccuracies of the heterogeneous kernel while data is being exchanged between CAD systems, also occur because of technical problems in the CAD system itself.

With the second group of errors, a number of problems arise when the design data generated for one purpose are shared with other members of a design team. Those problems are due to the different software and hardware systems that are used throughout the design and manufacturing process. Furthermore, because each team has its own proprietary data representation, the teams translate and store the CAD data in incompatible formats, thereby complicating the exchange of data. The resulting data may contain errors or be incomplete in a way that makes them unusable for downstream applications. For example, a CAD model created with seven-degree polynomial surfaces for the purpose of generating a stylish shape in an upstream application may
experience loss in a downstream application that allows for the representation of only five-degree polynomial surfaces.

In the third group of errors that we considered, the problems may be caused by designers as a result of poor or incorrect practices of generating CAD data. A third to a half of all quality problems stem from either poor design skills or the inexperience of designers [1,17].

We conducted a six-month case study to better understand the nature of CAD model errors and to analyze the frequency and lead times of errors. The study was based on the intensive error checking of 32 body-in-white models and 11 powertrain models, all of which were provided by two automotive companies [18]. First, we used CADI/Q of TranscenData for the error checking, and then we applied 64 error types, which were related to either the geometric quality referred to in the SASIG Product Data Quality Guidelines V2.0 [19] or to the in-house tolerance of the automotive company.

Table 1 shows that the tiny faces cause 29 percent of errors and that the other errors are caused by narrow steps (17 percent), narrow spaces (16 percent), and narrow regions (11 percent). Besides those four error types, we included non-tangent faces (5 percent) and sharp face angles (3 percent), both of which may significantly increase the lead times if transferred to downstream stages. Table 1 also shows the definition, the measurement method, the tolerance value and the frequency of the six error types. To determine the root causes from the viewpoint of the design history, we analyzed the errors with help of a design expert.
Table 1. Classification and frequency of the six error types.

<table>
<thead>
<tr>
<th>Error types</th>
<th>Definition</th>
<th>Measurement</th>
<th>Tolerance</th>
<th>Frequency of occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiny faces</td>
<td>The overall extent of each face is too small</td>
<td>The diagonal length of the bounding box of a face</td>
<td>Extent &lt; 0.05 mm</td>
<td>29 %</td>
</tr>
<tr>
<td>Narrow regions</td>
<td>A narrow portion of a face</td>
<td>The maximum width of the narrow portion of a face</td>
<td>Max. width &lt; 0.02 mm</td>
<td>11 %</td>
</tr>
<tr>
<td>Non-tangent faces</td>
<td>A non-tangent angle between adjacent faces (a $G^1$ discontinuity)</td>
<td>The maximum angle between surface normals evaluated at pairs of the nearest points on each face along a common edge ($C^0$ discontinuity)</td>
<td>Angle &gt; 1.0 degree</td>
<td>5 %</td>
</tr>
<tr>
<td>Narrow steps</td>
<td>A narrow portion of a face that forms a sharp step between adjacent faces</td>
<td>The maximum width of the narrow portion of a face</td>
<td>Max. width &lt; 0.02 mm</td>
<td>17 %</td>
</tr>
<tr>
<td>Sharp face angles</td>
<td>A large angle between adjacent faces</td>
<td>The maximum angle between pairs of face normal vectors along a common boundary</td>
<td>Angle &lt; 1.0 degree</td>
<td>3 %</td>
</tr>
<tr>
<td>Narrow spaces</td>
<td>A narrow portion of the space surrounding a solid</td>
<td>The maximum width of the narrow space</td>
<td>Max. width &lt; 0.01 mm</td>
<td>16 %</td>
</tr>
</tbody>
</table>

We used the B-Rep data to check the CAD model errors. A CAD model error is considered to exist whenever the result of an evaluation of the topological and
geometric elements of a CAD model is greater than a user-defined threshold. For example, if a design team requires the maximum gap for a $C^0$ continuity of a curve to be 0.01 mm but the gap between two adjacent edges is more than 0.01 mm, the curve has a $C^0$ discontinuity error. Furthermore, if the surface’s minimum radius of curvature is smaller than the allowed value, the surface is considered a $G^1$ discontinuity, even if the curve was deliberately created by the designer for style. Thus, we used the B-Rep data to find the errors in the parametric model, and, to correct the errors, we used the design history and the reasoning of an expert system.

3.2 Extraction of the design history

Most commercial CAD systems used in product development have data structures that include the design history as well as the B-Rep data. Those systems are called feature-based parametric CAD systems, and the CAD models created in those systems are called parametric models.

Fig. 1. Representation of the B-Rep data and the design history.
Fig. 1 illustrates the feature representation structure of a parametric model. The parametric model contains the feature command history, which defines each of the geometric features; it also shows the B-Rep shape produced by the Pad and Fillet feature commands. Thus, we can generate the B-Rep shape for Pad by using a single Pad feature command and two feature parameters, namely the 2-D profile (Sketch) and the height ($H$). We can also generate another B-Rep shape, the 2-D Geometry for Fillet, by selecting two lines from the previous B-Rep resultant shape (B-Rep for Pad) and the radius of the Fillet. In short, we can find CAD model errors by using the B-Rep resultant shape to mathematically compute the topological and geometric elements, and we can correct the errors by modifying the design history.

The design history refers to the chronological order in which features were created and to the constraints of overcoming limitations. Feature-based parametric CAD systems have a design history (or a so-called parametric specification) comprised of modeling functions, and each of the modeling functions is attached through its parameters to topological entities defined in a previous resultant shape. The first step in repairing CAD model errors is to extract the design history to see how the model has been created. This information includes the history of the feature commands, the parameterization data, and the constraints of the parametric model. If the information is to be shared with other CAD systems, the extracted design history needs to be structurally defined. Thus, we used the XML schema definition to represent the design history of a 3-D parametric model.
Fig. 2. Design history schema with the feature and reference command groups.

The design history schema of Fig. 2, which we formatted with Altova’s commercial XML editor XMLSpy, is represented as an XML schema. The schema consists of two groups: the feature command group, \textit{FeatureCmds}, in which each of the geometric...
features is defined; and the reference command group, \textit{ReferenceElems}, in which the reference elements such as constraints or reference planes are represented. In \textit{FeatureCmds}, 20 feature commands are defined on the basis of the core commands of the macro-parametric methodology [20-22], and each command includes the parameters for feature instantiation. In \textit{ReferenceElems}, eight reference elements are used to define the reference data, such as the reference plane, the axis, and the local coordinates; and the reference data are used to generate a 3-D solid. As shown in Fig. 2, the root node in the design history schema, \textit{HistorySchema}, must have one or more body nodes, and each body node consists of either multiple elements or a single element from \textit{FeatureCmds} and \textit{ReferenceElems}.

Fig. 3 shows how to extract and structuralize the design history from a 3-D CAD
model. After using the CAD system APIs, CATIA CAA of Dassault Systemes, to read a 3-D parametric feature model, the design history extractor makes the design history data that can be used to repair CAD model errors. The design history data are generated according to the design history schema. If the repaired design history is transferred to any other downstream application, a 3-D model can be generated without errors in these systems.

Fig. 4. XML schema definition of the Pad feature command.
Fig. 4 shows an XML schema definition of the Pad feature with the following seven parameters:

- **profile_curves**: shows the profile of the extruded feature. If more than two profile curves are given, the first specifies the outer profile and the others specify the inner profile.
- **first_limit_type**: specifies the end condition of the protrusion along the normal direction of the plane on which the profile curves are defined.
- **second_limit_type**: specifies the end condition of the protrusion along the opposite normal direction of the plane on which the profile curves are defined.
- **first_length** and **second_length**: specifies the lengths of the two given protrusive directions.
- **reference_plane**: shows the basis plane where the pad is generated.
- **own_direction**: shows the direction of the protrusion.

In the case of the **profile_curves** element, when a parameter of a feature command refers to the parameter of another feature in the design history, the string-typed identification name is used as a value of the parameter. The name of the referred element’s parent element, which is illustrated by the arrow in Fig. 4, is defined together with the referred element to avoid the ambiguity of duplicate strings in the design history file. By using a character string to refer to an external element, we gain access to a specific, externally referred element when analyzing the interdependency of the feature commands in the design history or when reconstructing the design history for error correction.
Fig. 5. Example of an XML-formatted design history file that was extracted from a 3-D parametric model. The model was created with the feature-based parametric CAD system CATIA V5. The name of the solid that resulted from the Pad feature is Pad.1, and we declared the first and second types of limitations to be dimensional. The first protrusion is 30.0 mm long, and the second protrusion is 5.0 mm long. The 2-D profile refers to Sketch.1, which is located under the PartBody element, and we selected the X-Y plane of the global coordinate system as a reference plane. The resultant solid Fillet.1, which is instantiated by the Fillet feature with a radius of 2.0 mm at Edge.5, maintains tangent continuity. We selected Edge.5 from the previous resultant solid Pad.1.

After extracting the design history file in XML format from a 3-D parametric model,
we used the file for the reasoning of the knowledge base and for reconstructing the design history through the interface of the document object model (DOM).

### 3.3 Solving the persistent naming problem and the name matching problem

When repairing CAD model errors by modifying the design history, we encountered a persistent naming problem for topological entities, as well as a name matching problem, especially when reevaluating a model after the modification.

To generate each feature in the design history, we referenced the topological entities of an existing B-rep model, enabling the resultant B-rep model to be referenced later by other features. This process led to the problem of assigning names to topological entities, and, to resolve the problem, we used a face-based topological naming method that is similar to the method of Wu et al. [9]. Accordingly, as shown in Fig. 6, we named the topological entities (faces, edges and vertices) of the B-Rep model in a topological naming history. We also named the following six types of transitions:

- **creation**: a new entity that appears in the resultant body
- **modification**: an entity which is used but does not appear in the resultant body, and which is replaced by a new entity
- **deletion**: an entity that disappears in the resultant body
- **absorption**: a kind of modification in which several entities are merged into the resultant body
- **subdivision**: a kind of modification in which an entity is cut from the resultant body
- **preservation**: no modification.
In Fig. 6, the label F2.6, which is one of faces that was generated from the *Fillet* command, is a new face. This label was referred to the labels F1.1 and F1.5 of the previous resultant model, which was generated from the previous feature command *Pad*. The F2.2 label was modified from the F1.1 label of Pad.1, and the F2.3 label is the same face entity as the F1.3 label of Pad.1. In addition, the F3.1 label, which was generated from the *Pocket* command, was made by the lateral curve C1; the label was then deleted after the *Pocket* command had been generated.
Fig. 6. Persistent naming method based on faces.

A face entity
A new created face entity
A reference line for entity creation or modification
A reference line to the previous entity

[F1.1(Pad.1)] – Modification → F2.1
...  
[F1.3(Pad.1)] – Preservation → F2.3
...  
[F1.1(Pad.1), F1.5 (Pad.1)] – Creation → F2.6
...  
[C1 (lateral)] – Creation → F3.1
...  
[C1, C2, C3, C4 (Top)] – Creation → F3.6
...  
[F2.1(Fillet.1)] – Modification → F4.1
[F3.2(Pocket.1)] – Creation → F4.2
...  
[F2.4(Fillet.1)] – Preservation → F4.8
...  
[F2.7(Fillet.1)] – Modification → F4.12
...  
[F3.1] – Deletion
There are two solutions for the name matching problem: a global matching method and a local matching method [16]. To find the topological entity of a new model that corresponds to a selected topological entity of an old model (1:N comparison), the local matching method compares all the topological entities of the new model with the selected topological entity of the old model. In contrast, the global matching method saves the history of the topological evolution of both the old and new models and compares the evolution history of the old model with that of the new model (N:N comparison). To identify a topological entity in the previous model and to find the same entity in the repaired model, we used the global matching method. The topological naming history, including the topological names, the transitions and the relation between the relevant entities, was applied for the naming matching.

3.4 Interpretation of the design history

To correct CAD model errors with the aid of the extracted design history, we first analyzed the interdependencies of the feature commands and then defined the relations between the corresponding feature command patterns. For the interdependency analysis, we analyzed the feature command patterns and the parametric data. Next, we interpreted the matching relations in the patterns of the upstream and downstream commands of the extracted design history, and we examined how the parameters and constraints of each feature command depended on the parameters and constraints of the other feature commands.
Fig. 7. Relation between the command patterns and the types of CAD model errors.

Fig. 8. The command patterns that occur in narrow spaces.
Fig. 7 shows the feature command patterns for the root causes of the errors identified in the case study that we referred to in section 3.1. With the assistance of a design expert, we conducted the case study for six months to find the root causes of the errors in terms of the design history. From the patterns, we interpreted the relationship between six types of errors; 13 features of the root causes, which directly affect these types of errors; and 20 subordinate features, which are coupled with those features of the root causes.

First, we mathematically computed the topological and geometric elements to check the location of the errors in the resultant solid shape that is instantiated by a feature command in the design history. We then defined the relation between the command patterns of the feature commands and the 20 commands subordinately connected to the feature commands.

Fig. 8 shows more details of the command patterns that caused the narrow spaces of Fig. 7. There are 16 command patterns that cause narrow spaces. The features of the root cause that directly affect the narrow spaces are \textit{Pad}, \textit{Boolean\_op}, \textit{Draft} and \textit{Mirror}, and, as shown in Fig. 8, the subordinate features that are coupled with the features of the root cause are linked with each corresponding feature of the root cause. The 16 command patterns in the narrow spaces are represented as 16 rules of the knowledge base. For example, to match the feature command patterns, we represented the narrow space, which is shown in Fig. 9 by an arrow, in the form of “Narrow Space: Pad (Pad.2) $\rightarrow$ Draft (Draft.1).” In the XML file of the design history, the narrow space occurs in the resultant solid \textit{Pad.2}, which directly influences the narrow space; moreover, \textit{Pad.2} is linked to \textit{Draft.1}, and \textit{Draft.1} is subordinately connected to \textit{Pad.2}. 

25
Fig. 9. Example of the narrow space in the design history.
If we reconstruct the design history solely by modifying the sequence of the feature commands, that is, without considering the parameters or constraints of the features in the design history, the shape occasionally becomes distorted or collapses. By analyzing the parametric data, we can check if the parameters or constraints in each feature command depend on any of the parameters or constraints of the other feature commands.

Fig. 10 shows how we analyzed the parametric data. The parametric data of Feature_A are compared with those of Feature_B, and, if one parameter or constraint of Feature_A correlates with one or more parameters of Feature_B, the parameters or constraints that are linked to the resolution are transferred to the expert system.
3.5 **Building the knowledge base**

The knowledge base is a set of rules and facts that can generate a new design history without errors. The design history extracted from a CAD model, as well as the interdependency of the feature commands, is represented as a rule and a fact in the knowledge base of an expert system. The knowledge base has a pre-processing module, a main module and a post-processing module.

Fig. 11 shows the functional configuration and rule reasoning process as an IDEF0 diagram. The input data needed to start a repairing process must contain the design history extracted from a CAD model, the feature command pattern verified during the pre-processing, a root cause feature and a subordinate feature. The pre-processing module has rule templates for expressing each feature command, rules for expressing the design history and rules for defining the interdependency of the features. Moreover, the extracted design history is represented as a set of facts: that is, facts for each feature command and facts for the sequential relation between the feature commands.

The main module has rules for detecting the command patterns and rules for correcting the design history. By analyzing the command patterns, the main module determines whether the command tuples defined in the rule exist in a list of features; in addition, if a pattern that is identical to the command patterns in the knowledge base exists inside the input command history, the main module fires a repairing rule that corresponds to the command patterns. The repairing rules are composed of rules that find the error candidate command and rules that change the order.

The output rules in the post-processing module create a new design history that generates a CAD model in the CAD system.
For the expert system, we used CLIPS V6.0, which was developed by the NASA Johnson Space Center [23]. The knowledge base of the expert system has 92 rules for matching feature command patterns, 37 rules for checking the dependence of the parametric data, and 72 rules for reconstructing the design history. When the expert system processes the design history extracted from a parametric model, it uses the reasoning process of the knowledge base to reconstruct a new file of the design history.
Fig. 11. Functional diagram of the rule reasoning process (IDEF0 diagram).
3.6 Correcting CAD model errors

With the aid of B-Rep data, we can check CAD model errors by computing the topological and geometric elements, and we can use the design history to correct the errors. The CAD model correction system that we developed can check and repair a CATIA V5 model that has the data structure of a complex cell of Rossignac [24].

When correcting a CAD model error, the first step is to locate the error in the feature commands. As shown in Fig. 12, the process of checking the six types of errors is conducted on the current active solid shape, *MainBody*. Whenever one or more errors is found in *MainBody*, the process sequentially checks for errors in the resultant solid shapes instantiated by the feature commands until the process locates the errors.

![Fig. 12. Process for checking the location of errors in feature commands.](image-url)
Fig. 13. Correction process.
Fig. 13 shows the error correction process of the implemented design history–based correction method. The main correction program is invoked in the CATIA V5 CAD system, which is linked to the CLIPS expert system. Once started, the program retrieves the design history data from a parametric feature model activated in the CAD system, and analyzes and identifies six types of errors within the model.

After checking the locations of errors in the feature commands of the model, as shown in Fig. 12, the result and the design history information extracted from the model are transferred to the knowledge base. In accordance with the rules of the knowledge base, three analyses, as mentioned in section 3.4 and 3.5, are executed in the expert system: an interdependency analysis of the feature commands, an analysis of the feature command patterns, and an analysis of the parametric data. Through the reasoning of the expert system, the design history is reconstructed in the CAD system and, thus, the new constraints and parameters are defined for the new design history. Finally, after a reevaluation of whether the CAD system can internally rebuild the new design history, the repaired solid shape is generated. If the new design history that the expert system presents cannot be built in the CAD system, the correction process goes back to the expert system until the CAD system accepts the new design history.
4. Implementation and experiment

To verify that the design history–based approach can correct CAD model errors, we used the CATIA V5 environment to develop the CAD model correction system Q-Raider. Q-Raider uses CLIPS V6.0 as the expert system shell. Furthermore, to integrate CLIPS with other applications, such as CATIA V5, we revised the CLIPS source codes. These open architecture codes represent the sequential history of the feature commands and provide an analysis of the design history rules.

Fig. 14. The structure of the CAD model correction system.
As shown in Fig. 14, the CAD model correction system has five modules: a design history extraction module, a geometry checking module, a knowledge reasoning module, a topological naming and name matching module, and a history reconstruction module. The design history extraction module extracts the design history from a CAD model with the aid of component application architecture, which is a CATIA interface library. It then creates an XML design history file according to the design history schema described in section 3.2.

After reading the B-Rep data of the CAD model, the geometry checking module locates the errors in the design history by mathematically computing the B-Rep data in terms of the tolerance value for the type of error.

The knowledge reasoning module then executes backward chain reasoning with the rules stored in the knowledge base.

The topological naming and name matching module, which is used for assigning names to topological entities and for matching entities of the previous model with entities of the repaired model, is connected to the design history extraction module and the history reconstruction module.

The history reconstruction module rebuilds the design history in accordance with the results of the knowledge reasoning. It then checks whether the modified design history can be rebuilt in CATIA V5, and, finally, it redefines the constraints of the information on the design history.

The CAD model correction system is linked to three external interface modules: the Dassault Systemes component application architecture, which is used for accessing the data structure of CATIA models; Microsoft Win32 APIs, for the graphical user interface;
and a document object model interface module, for accessing both the XML schema and the design history file.

Fig. 15 shows a graphic of Q-Raider’s main window and control panel. The control panel enables the user to configure the tolerance values of the six types of errors, the applicable rules, and the graphical setting for the error elements. Aside from being able to check these six types of errors, Q-Raider has recently acquired five additional functions that enable it to check high degree surfaces, narrow faces, narrow volumes, tiny solids, and void solids.

Fig. 15. Q-Raider’s main window and control panel.
The CAD model correction system can be applied to tiny faces, narrow regions, non-tangent faces, narrow steps, sharp face angles, and narrow spaces. Fig. 16 shows a crankshaft model with problems related to a tiny face and narrow step. The arrows in Fig. 16 indicate that the model has a tiny face with an area of 0.094 mm$^2$ and narrow steps in which the two adjacent edges have a ratio of 484:1 and a width of 0.09 mm. The design history extracted from this model comprises one $MainBody$, five $SubBodys$, and several reference elements.

While modeling the left part of the crankshaft, the designer revolved the profile of the crank 360 degrees around the axis of the shaft and then removed the swept volume (refer to part B in the design history of Fig. 16). Although the lower end of the crank was created in the proper shape, the upper end of the crank caused a tiny face near the trajectory of the profile. This tiny face was caused by the difference between the axis of the crank and the axis of the shaft. To correct the tiny face, the CAD model correction system finds which feature command ($Shaft.3$ in $Body.4$) in the design history caused the tiny face, and then creates a modified design history. After reasoning the knowledge base, the system then inserts $Shaft.7$ into the $Body.4$ group, and $Curve.12$ into the $ReferenceElems$ group (refer to part b in the design history).

The clutch disk on the right of the crankshaft is formed by two $Pad$ feature commands ($Pad.1$ and $Pad.2$) and a Boolean-type $Cut$ feature command ($Remove.3$) with $Body.1$. The $Body.1$ command is formed by the circular pattern feature command ($CircPattern.1$) with $Pad.3$ (refer to part C in the design history). The narrow step, which is indicated by the arrow, occurred in the lower face of the larger disk. In the geometric checking process, the location of the narrow step in $Remove.3$ was corrected by altering $Pad.2$ and $Remove.3$ (refer to part c in the design history).
Fig. 16. Example of a correction to a crankshaft model.
5. Conclusion

In commercial applications, correcting CAD model errors with B-Rep data has the following three drawbacks: First, when the B-Rep shape is repaired by modifying the topological and geometric elements without considering the design intent, the modeler may unintentionally distort the contextual meaning or cause the geometric structure to collapse. Second, the computations of correcting CAD model errors are prolonged because the B-Rep model can represent a wide class of objects; in addition, the data structure of the B-Rep model is complex, and the computations require a large memory. Third, even when a B-Rep shape is properly repaired, there is a problem of reusing the shape or transferring it to the upstream design team that created the CAD model. This problem arises because the B-Rep model is a dumb solid—that is, it has no parameter or constraint information such as the engineering data on how it is created. Unlike a parametric feature model, the B-Rep model can be used only as a reference model—not as a master model.

To repair CAD model errors, we propose an approach based on the design history. We also describe four parts for repairing CAD models. The first part defines the design history schema that represents the design history extracted from a CAD model. We can apply the schema not only to CATIA V5 but to other CAD systems. The second part introduces a topological naming history for assigning names to topological entities and for matching entities of the previous model and entities of the repaired model. The third part analyzes the interdependency and parametric data of feature commands. The fourth part describes the necessary processes for reconstructing the design history. From this final process, we developed a CAD model correction system to verify the design.
history–based approach.

For future research, we hope to extend the CAD model correction system by accumulating the corrective knowledge from more case studies. Creating a CAD model is a process, and, to construct the appropriate contextual meaning for the designer’s experience and intuitive judgment, we need to help the designer understand the functionality of the CAD system. Consequently, the design intent must be reflected in the CAD model during the modeling process. We therefore hope to extend the rules of the knowledge base with heuristic design intent and contextual meaning.
Acknowledgements

This work was supported by the Korea Research Foundation Grant (KRF-2005-M1-10306).
References


[10] Mun DH, Han SH. An approach to persistent naming and naming mapping based on OSI and IGM for parametric CAD model exchanges. Proceedings of the fifth Japan-Korea CAD/CAM Workshop on Digital Engineering Workshop (DEWS),


