〈 논문 〉

전문가시스템을 이용한 CAD 모델 수정 시스템

양정삼† · 한순흥* · 천상욱**

(2005 년 3 월 7 일 접수, 2005 년 12 월 15 일 심사완료)

A CAD Model Healing System with Rule-based Expert System

Jeongsam Yang, Soonhung Han and Sang-Uk Cheon

Key Words : CAD Model Healing(CAD 모델 수정), Design History(설계이력정보), Expert System (전문가시스템), Feature(특징형상), Parametric Data(파라메트릭정보)

Abstract

Digital CAD models are one of the most important assets the manufacturer holds. The trend toward concurrent engineering and outsourcing in the distributed development and manufacturing environment has elevated the importance of high quality CAD model and its efficient exchange. But designers have spent a great deal of their time repairing CAD model errors. Most of those poor quality models may be due to designer errors caused by poor or incorrect CAD data generation practices. In this paper, we propose a rule-based approach for healing CAD model errors. The proposed approach focuses on the design history data representation from a commercial CAD model, and the procedural method for building knowledge base to heal CAD model. Through the use of rule-based approach, a CAD model healing system can be implemented, and experiments are carried out on automobile part models.

1. 서 론

제품개발과 생산에 있어서 CAD 모델에 대한 의존도는 기업의 경쟁력 확보 차원에서 지속적으로 증가하고 있다. 과거에는 제품의 품질에 대한 개념이 생산공정에 맞춰졌지만, 최근에는 제품 설계단계의 CAD 모델 품질에 초점을 두고 있다. 설계과정에서 발생하는 ECOs(Engineering Change Orders)의 약 45%가 CAD 모델의 오류로 인해 발생하고 있고, 이를 해결하기 위해 설계자는 CAD 모델을 생성하기 위해 투입된 작업량의 20~70%을 모델 수정작업에 재 투입하고 있다.[1,2] 미국자동차 산업의 공급망(Supply chain)에서 발생하는 CAD 모델의 오류를 수정하기 위해서 연간 약 10 억불 이상의 추가 비용이 투입되고 있다.[3]

Fig. 1 은 완성차 제조업체의 CAD 시스템인 CATIA 에서 생성된 후, 생산을 위해 협력업체로 전달되어 IGES 로 변환된 Side mirror 형상이다. 변환과정에서 화살표로 표시된 부분에 Face 손실이 발생된 것을 볼 수 있다. 현업에서는 품질 문제를 해결하기 위해 많은 시간과 비용을 투입하고 있다. Fig. 1 의 Face 손실을 해결하기 위해서 현업의 설계자는 Base geometry 인 Surface 를 수정하고 새로운 Face 처리를 위해 약 1 시간의 작업 시간이 투입되었다. 품질 문제를 해결하거나 오류를 수정하는 책임은 후속공정 또는 협력업체의 설계자에게 전가되고 있다. 또한 수정된 CAD 모델은 최초 설계자에게 Feedback 되지 않아서 선행 공정과 후속공정 사이에 Data consistency 문제도 발생하고 있다.

오류 문제를 자동으로 해결하기 위해 진행된 기존 연구에서는 CAD 모델의 B-Rep(Boundary representation) 정보만을 가져와서 수학적인 해석을 통해 모델의 품질을 검증하고 오류를 수정하는 방

† 책임저자, 회원, 아주대학교 산업정보시스템공학부
E-mail : nurbs7@gmail.com
TEL : (031)219-2355
* 회원, 한국과학기술원 기계공학과 shhan@kaist.ac.kr
** 한국과학기술원 기계공학과 csu@icad.kaist.ac.kr

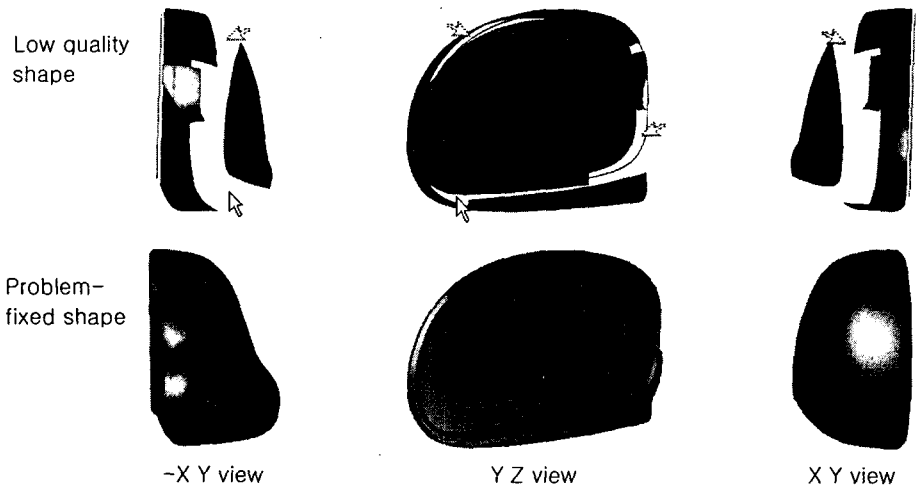


Fig. 1 Example of CAD model error, the face loss

법을 사용하였다.[4~11] 본 논문에서는 형상 모델링 과정에서 설계자의 설계의도(Design intent)가 포함되어 있는 3D CAD 모델로부터 설계이력(Design history) 데이터를 추출하여 설계지식(Design knowledge)으로 가공하고, 이를 CAD 모델의 오류를 수정하는데 활용하는 방법을 제시한다. 이를 위해서 사례연구를 통해 획득된 Rule 을 바탕으로 전문가시스템을 이용한 CAD 모델 수정 시스템을 개발하였다.

2. 관련연구

2.1 형상에 대한 오류 수정

CAD 모델의 오류 수정에 대한 연구는 (1) Exact B-Rep 형상을 이용하는 방법; (2) Faceted B-Rep 형상을 이용하는 방법; 그리고 (3) 경계곡선(Boundary curve)을 이용하는 방법으로 구분 된다.

(1) Exact B-Rep 형상을 이용하는 방법은 형상을 구성하는 위상요소와 기하요소를 표현하는 자료구조에 대한 수학적인 연산 과정을 통해 오류를 확인하고 이를 수정하였다. Hoffman 은 기하 공차에 의해 발생된 형상의 오류를 사용자에 의해 제어되는 콜백 메커니즘을 이용해 수정하였다.[4] Gu 는 Topological entity 를 Complementary model object tree 를 이용해 Topological entity 를 순차적으로 분류하여, 각각의 엔터티에 발생하는 오류를 수정하였다.[5]

(2) Exact B-Rep 을 다면체 B-Rep (Faceted B-Rep)의 Polyhedron 으로 근사화시켜 오류를 진단하고 수정하는 방법은 연산 속도가 빠른 장점을 가지고 있다. Barequet 는 복잡한 형상의 트림 곡면을 정렬되지 않은 다면체 형태의 작은 조각(Unordered lists of polygons or surfaces)으로 나눈 후, 순서에 따라 재 구성하는 해쉬 알고리즘 (Hashing algorithm)을 제안하였다.[7] 이를 바탕으로 다면체 사이에 발생하는 간격(Gap)을 인접 다면체에 Matching 하는 방법으로 오류를 수정하였다. 이 방법은 다면체의 면과 면 사이에 발생한 간격에 대해 수정(Stitching)이 가능하다.

(3) 경계곡선을 이용하는 방법은 곡면으로만 이루어진 형상에 대해서만 제한적으로 적용할 수 있다. Steinbrenner 는 차수가 다양한 곡면으로 이루어진 모델에 대해 인접 곡면 사이의 간격과 중첩을 진단하고 수정하는 연구를 하였다.[8] 이 방법은 경계곡선을 에지(Edge) 형태의 작은 곡선으로 나눈 뒤, 인접한 에지들 간의 관계를 비교하여 에지를 절단하고 합치는 과정 (Edge splitting and merging)을 반복 수행하여 간격과 중첩 같은 오류를 수정하였다. Volpin 은 G1 연속성을 갖는 곡면으로 수정하기 위해, 곡면의 면적과 곡률의 변화에 따라 곡면의 경계곡선을 구성하고 있는 곡선을 나누고, 이를 바탕으로 곡면을 재 구성하는 방법을 사용하였다.[9]

그밖에 Yang 은 STEP AP214 로 만들어진 형상을 AP214 스키마 엔터티(entity)에 대한 변경을 통해 오류를 수정하는 방법을 사용하였다.[10,11]

B-Rep 형상을 기반으로 오류를 수정하는 것은 2 가지 단점을 가지고 있다. 첫 번째는, B-Rep 형상의 위상요소(Topology)와 기하요소(Geometry)에 대한 변경을 통해 수정된 형상은, 설계자의 직관적인 의도가 반영되어야 하는 모델링 과정에서 설

계자의 의도와 다른 형상으로 왜곡되거나 형상 자체가 붕괴되는 위험이 있다.

두 번째는, 비록 오류가 정확히 수정되더라도 B-Rep 형상은 껍데기 형태(Dumb solid)이기 때문에, 수정된 결과가 Engineering 정보와 같이 생성된 Master CAD 시스템으로 반영되거나, 이전 설계부서로 Feedback 되기 어렵고, 기준모델(Master model)이 아닌 참조모델(Reference model)로만 사용이 가능하기 때문에 활용도가 떨어진다.

2.2 CAD 에서 전문가시스템의 활용

CAD 분야에서 전문가시스템의 적용은 구성설계(Configuration design), 설계정보의 해석 등과 같은 단위 연구 영역의 Stand-alone system 에서 벗어나 최근에는 제품개발의 전주기(Product life cycle)에 포괄적으로 활용되고 있다. 지식베이스의 추론 방식에 따라 (1) 지식베이스(Knowledge base); (2) 사례기반 추론 (Case-based reasoning); (3) 결정트리(Decision tree)로 나눌 수 있다.

(1) 지식베이스에 의한 전문가시스템은, 주어진 영역에 관한 인간 전문가의 지식을 알아내고 포착하려 시도한다. 지식베이스 내에서 전문가 지식을 표현하고, 질의에 대한 대답을 지식베이스의 자동 추론을 통해 얻는다. 인간 전문가가 참여하여 지식을 제공하는 것과 비슷한 결과를 얻을 수 있다.

(2) 사례기반 추론 기법은, 이미 알려져 있는 사례를 일련의 입력자료 형태로 데이터베이스에 보관한 후, 비슷한 사례에 대해 데이터베이스의 입력자료와 비교하여 결과를 추론한다. 새로운 상황이 발생하면 본인의 과거 경험 중 가장 비슷한 것과 비교하여 이를 이해하려는 인간의 성향과 동일하다.

(3) 결정트리는 "if-then" 문에 있는 복합 조건문과 비슷하다. 결정트리는 하나의 루트에서 시작해서, 일련의 입력 값을 근거로 각 노드의 조건(if)에 따라 최종 자식 노드에서 결정을 내린다.

본 연구에서는 명령어 패턴 분석(Commands pattern analysis)과 파라메트릭 데이터 분석(Parametric data analysis)을 기반으로 지식베이스를 구축하여 오류를 분석하고 수정하였다. 사례기반 추론 방법은 사례를 기반으로 주어진 사례에 대해 비슷한 결과를 찾는 방법으로, 명령어 패턴과 같은 정형화된 규칙에서는 좋은 결과를 얻기 힘들다. 결정트리 방법은 조건에 해당되는 입력 값이 사전에 결정되어 있지 않기 때문에, 명령어 패턴 분석과 파라메트릭 데이터 분석에 적당하지 않다.

3. 지식베이스의 구축

3.1 설계이력 정보의 추출

제품개발에 사용하고 있는 대부분의 CAD 시스템은, B-Rep 정보뿐만 아니라 설계이력을 포함하는 자료구조를 가지고 있고, 이를 특징형상 기반의 파라메트릭 CAD 시스템이라 부른다. Fig. 2 에서 보는 바와 같이 하나의 결과 형상 (Result solid)을 만들기 위해, 특징형상 명령어를 호출하게 되면 특징형상 템플릿(Skeleton of the feature)을 참조하여 설계자로부터 입력된 스펙 정보(Specification attributes)에 따라 결과 형상을 인스턴스 시킨다. 스펙정보에는 현재 활성화된 솔리드(Active solid), 이전 명령어에 대한 결과 형상(ResultIN solid), Alphanumerical input value 를 포함한다.

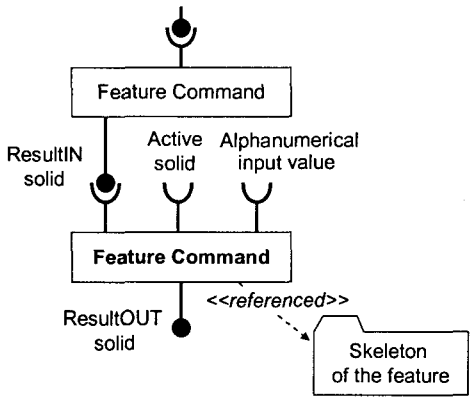
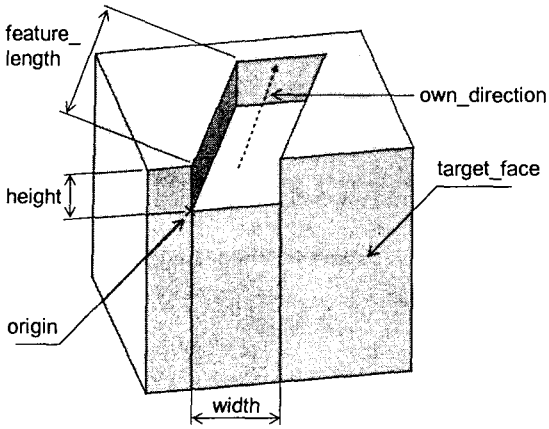


Fig. 2 Featurization in a feature-based parametric CAD system

Fig. 3 은 ResultOUT 된 Slot feature 로부터 설계이력 정보를 추출하여 매크로파라메트릭 방법론에 따라 EXPRESS schema 형태로 표현된 결과를 보여준다. Slot feature 를 인스턴스 하기 위해 필요한 파라메트릭 정보는 7 개로서, *result_object_name* 은 Slot 이 생성된 형상(ResultOUT solid)의 이름, *target_face* 는 Slot 이 위치하는 Active solid 의 face 이름, *own_direction* 는 Slot 의 진행방향, *height* 는 Slot 의 높이, *width* 는 Slot 의 폭, *length* 는 Slot 의 길이, *origin* 는 slot 의 위치 설정을 위한 기준점을 의미한다.

설계이력 정보는 많은 CAD 시스템에서 서로 다르게 표현되고 있다. 하나의 형상에 대한 설계이력 정보를 다수의 CAD 시스템에서 공유하고 설계지식으로 가공하기 위해서, 본 논문에서는 XSD(XML Schema Definition)를 이용해서 표준화된 설계이력을 정의하였다.

설계이력 스키마는 Fig. 4 에서 보는 바와 같이 형상을 생성하는 특징형상 명령어 그룹(FeatureCmds)과, 구속조건을 포함해서 참조 데이터를 생성하는 그룹(ReferenceElems)으로 구성된다.




```
ENTITY SOLID_Create_Feature_Slot;
    result_object_name : STRING;
    target_face        : STRING;
    own_direction      : direction;
    height             : positive_length_measure;
    width              : positive_length_measure;
    feature_length     : positive_length_measure;
    origin             : cartesian_point;
END_ENTITY;
```

Fig. 3 Macro-parametric expression of the feature command Slot

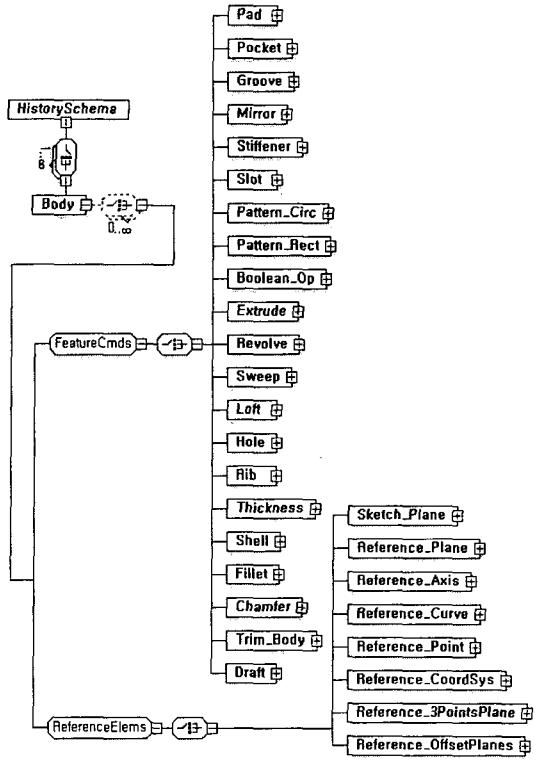
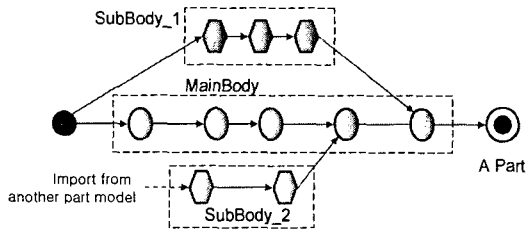


Fig. 4 Two groups of feature commands and reference elements for representing the design history[18]

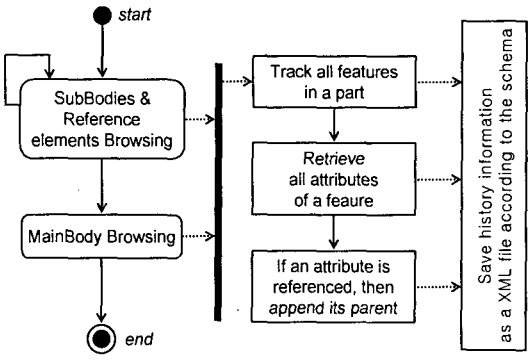
솔리드 형상을 생성하는 21 개의 특징형상 명령어들은 매크로파라메트릭 방법론[12~14]에서 제안된 모델링 명령어들을 바탕으로 정의되었다. 각각의 명령어는 특징형상을 인스턴스시키기 위한 Alphanumerical input value 형태의 파라메트릭 데이터를 포함한다. Plane, Axis, 좌표축 등과 같이 8 개의 참조 엘리먼트를 생성하는 명령어들로 이루어진 ReferenceElems 그룹은, 솔리드 형상을 만들기 위해 보조적으로 필요한 엘리먼트를 정의할 수 있다.

CAD 모델로부터 추출된 설계이력은 하나 이상의 Body 노드들로 구성되고, 각각의 Body 는 다수의 특징형상 명령어 또는 참조 엘리먼트를 포함한다.

Fig. 5(a)처럼 특징형상 기반의 CAD 시스템에서 모델링 하는 프로세스는, 목적하는 파트(Part)의 형상을 생성하기 위해 진행되는 MainBody 프로세스와, 불리언(Boolean) 작업과 같이 MainBody 에 종속되기 위해 생성되는 SubBody 생성 프로세스로 나눈다. SubBody 는 다른 설계자가 생성한 형상을 MainBody 로 가져오는 경우도 있다. 따라서 하나



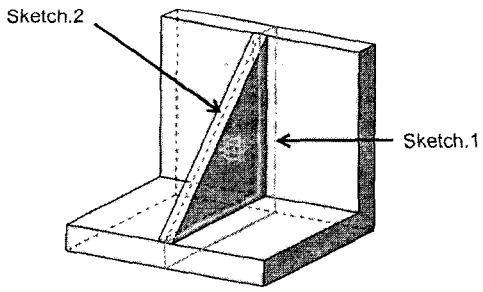
(a) Modeling process within a CAD system



(b) Design history extraction prcess

Fig. 5 Design history extraction from a CAD model

의 형상은 다수의 Part 노드로 이루어지고, 하나의 Part 노드는 순차적으로 기록된 명령어들로 이루어진 다수의 Body 노드들로 구성된다. CAD 모델로부터 설계이력을 추출하는 방법은, Fig. 5(b)에서 보는 바와 같이 일련의 리스트(List) 형태로 이루어진 설계이력에서, SubBody 들과 Reference 엘리먼트들에 대한 이력 정보를 먼저 추출하고, 마지막에 MainBody 의 설계이력을 추출한다. 이것은 마치 프로그래밍 언어에서 Sub-Function 을 Main-Function 의 앞 부분에 위치하는 것과 같은 방식이다. 모델의 오류 수정과 모델을 재 구성 할 경우에, MainBody 의 앞 부분에 정의된 SubBody 또는 Reference 엘리먼트를 먼저 읽는다. 불리언 작업과 같이 SubBody 를 매개변수 (Second operand) 형태로 이용하는 경우에, MainBody 에서 해당되는 SubBody 를 참조할 수 있다. 또한 설계 이력에서 오류가 발생된 위치를 추적할 경우에도 MainBody 에서 SubBody 로 접근이 용의하다.




```
<?xml version="1.0" encoding="UTF-8" ?>
- <HistorySchema
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xsi:schemaLocation="D:\HistorySchema.xsd">
  + <Body body_name="OpenBody">
  - <Body body_name="PartBody">
    - <Pad result_solid="Pad.1">
        <profile_curves profile_name="Sketch.1"
          parent_name="PartBody" />
        <first_limit_type>Dimension</first_limit_type>
        <first_length>71.000</first_length>
        <second_limit_type>Dimension</second_limit_type>
        <second_length>71.000</second_length>
        <reference_plane>xy plane</reference_plane>
      </Pad>
    - <Stiffener result_solid="Stiffener.1">
        <profile_sketch profile_name="Sketch.2"
          parent_name="OpenBody" />
        <flip>From_Side</flip>
        <start_condition>Dimension</start_condition>
        <start_depth>10.000</start_depth>
        <end_condition>Dimension</end_condition>
        <end_depth>0.000</end_depth>
      </Stiffener>
    </Body>
</HistorySchema>
```

Fig. 6 Example of an XML-formed design history file

Fig. 6 은 2 개의 특징형상 명령어를 사용해서 생성된 모델에 대해서 설계이력 스키마에 따라 XML 포맷으로 설계이력을 추출한 결과를 보여준다. 결과 형상의 이름이 Pad.1 인 Pad feature 는 Dimension 타입(Pad 의 Length 를 정의해서 형상을 생성)으로서 첫 번째 Length 와 두 번째 Length 를 71mm 로 정의하였다. 2D Profile 은 PartBody 아래에 위치한 Sketch.1 이 참조되었다. 이때 Reference plane 은 전역좌표 계의 XY plane 으로 선택되었다. Stiffener.1 은 OpenBody 아래에 있는 Sketch.2 를 Profile sketch 면으로 하여 측면에서 Stiffener 가 생성되도록 옵션을 정의하였다. 첫 번째 Offset 의 길이는 10mm 이고 두 번째 Offset 의 길이는 0mm 로 정의된 특징형상이다.

이와 같이 XML 형태로 추출된 설계이력 정보는 DOM(Document Object Model) Interface 를 이용해서 전문가시스템의 추론 과정과 오류 수정을 위한 설계이력 정보의 변경에 사용된다.

3.2 설계이력 정보의 해석

CAD 모델로부터 추출된 설계이력 정보를 통해 설계이력들간에 의존관계(Inter-dependency)를 해석하고 설계이력을 변경해서 오류를 수정할 수 있는 지식베이스를 구축한다. 의존관계를 해석하는 부분은 추출된 설계이력 리스트로부터 상위 명령어와 하위 명령어간의 종속관계에 대한 패턴을 해석하는 부분(Commands pattern analysis)과 각각의 명령어에 포함된 파라미터 또는 구속조건 등이 다른 명령어들과의 의존관계를 해석하는 부분(Parametric data analysis)으로 구분된다.

Commands pattern analysis 는 Fig. 7 에서 보는 바와 같이 6 개의 오류 항목에 대해 오류발생에 직접적으로 영향을 주는 13 개의 명령어와, 이들 13 개의 명령어에 종속되어 오류 발생에 영향을 주는 21 개의 명령들간의 관계에 대한 해석을 한다. 즉, 설계이력에서 특정 명령어를 수행한 ResultOUT solid 에서 오류가 발견되었고, 이 명령어가 수행되기 위해 종속적으로 연결되는 21 개의 명령어 집합과의 패턴 관계를 정의할 수 있다. 하나의 예로서 Fig. 7 에서 굵은 직선으로 표시된 Narrow Space 오류는 *Narrow Space: Pad(Pad.2)➔Draft(Draft.2)*의 명령어들의 관계에서 발생하였다. 즉, 오류가 발생한 부분은 Pad 명령어를 수행한 ResultOUT solid 에서 발견되었고, 이 Pad 명령어는 이전 명령어인 Draft 명령어의 ResultOUT solid 를 입력 받았다.

오류 항목과 명령들간의 패턴 관계는 현대기아자동차, DaimlerChrysler, 일본자동차공업협회(JAMA), ITI Solutions 에서 제공 받은 43 개의 오류

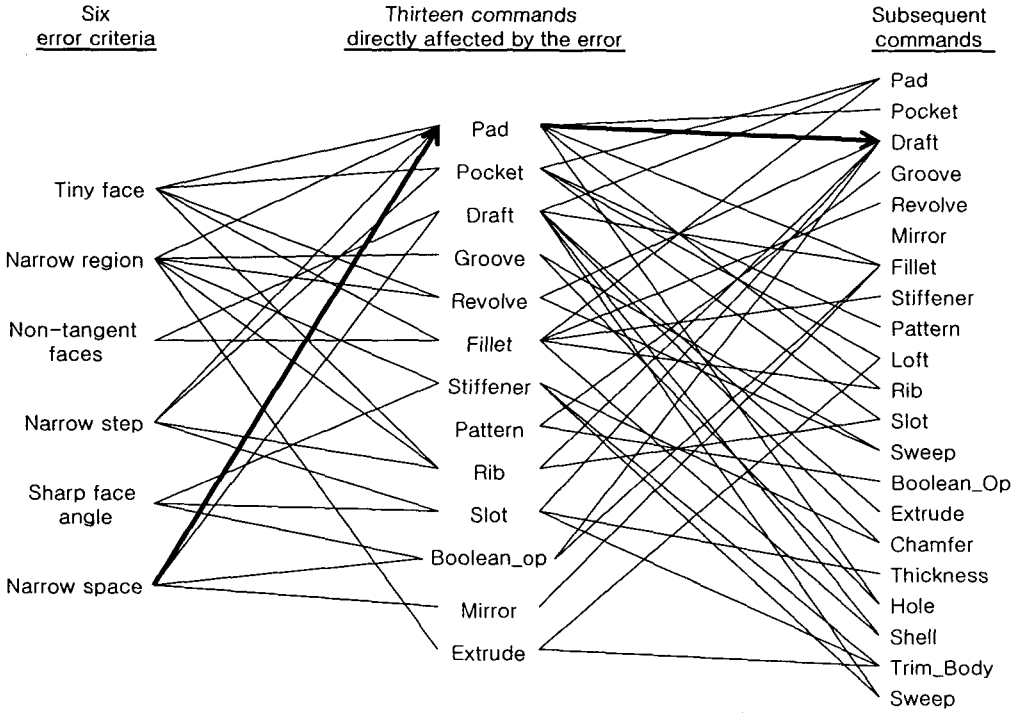


Fig. 7 Matrix for commands pattern analysis[18]

사례에 대한 연구를 통해 구성되어 지식베이스로 구축하였다. CAD 모델로부터 설계이력 정보가 전문가시스템으로 입력되면 지식베이스의 추론 과정을 통해 오류를 회피할 수 있는 새로운 설계이력을 생성할 수 있다.

특징형상 명령어 내부에 포함되어 있는 파라미터 또는 구속조건을 무시하고 Commands pattern analysis 를 통해 오류가 있는 명령어들의 순서만을 변경해서 모델을 재 구성할 경우, 전혀 다른 형상이 만들어지거나 붕괴되는 경우가 많다. Parametric data analysis 는 각각의 명령어에 포함된 파라미터 또는 구속조건 등이 다른 명령어들과의 의존관계를 이루고 있는지 여부를 해석한다. 즉, Feature_A 에 포함되어 있는 파라메트릭 데이터와 Feature_B 의 파라메트릭 데이터를 비교해서 서로 의존관계를 유지하고 있는 데이터를 해석한다. 해석된 결과는 CAD 모델로부터 추출된 XML 형식의 이력 파일 내에 저장되어 설계이력을 변경하여 오류를 수정 할 때 참조된다.

3.3 지식베이스의 구축

추출된 설계이력 정보와 설계이력들간에 의존관계는 본 논문에서 전문가시스템으로 사용하는 CLIP 의 지식베이스로 표현된다. 지식베이스는

```
(deftemplate pad
    (slot superbody (type SYMBOL)(default null))
    (slot name (type SYMBOL)(default null))
    (slot first_limit_type (type SYMBOL) (default null))
    (slot first_length (type SYMBOL)(default null))
    (slot second_limit_type (type SYMBOL)(default null))
    (slot second_length (type SYMBOL) (default null))
    (slot profile (type SYMBOL)(default null))
    (slot reference_plane (type SYMBOL)(default null))
    (slot next_command_type (type SYMBOL)(default null))
    (slot next_command (type SYMBOL)(default null)))
```

Fig. 8 A template for Pad feature command

전처리 모듈, 메인 모듈, 그리고 후처리 모듈로 구성된다. 전처리 모듈은 개별 특징형상 명령어의 표현을 위한 Rule, 설계이력 정보의 표현을 위한 Rule, 그리고 설계이력들간의 의존관계를 정의한 Rule 로 구성된다. 메인 모듈은 명령어 패턴을 인식하는 Rule, 설계이력 정보를 수정하는 Rule 로 구성된다. 후처리 모듈은 명령어 이력을 출력하는 Rule 로 구성된다.

전처리 모듈에서 특징형상 명령어는 설계이력 스키마에 근거하여 Template 으로 표현된다. Fig. 8 은 Pad 명령어를 표현하기 위한 Template 을 보여준다.

전문가시스템을 구성하는 Facts, Rules, 추론 엔진 중에서, CAD 모델로부터 추출된 설계이력은 Facts 의 집합으로 표현된다. Facts 자체는 서로 독립된 정보들을 표현한 것이고, 하나의 명령어가 하나의 Fact 가 되기 때문에 명령어의 순서 정보를 나타내기 위하여 Fact 에 관계 정보, 즉 순서 정보를 나타내는 데이터를 추가하였다.

```
(defrule tuple-pattern ""
        ?tuple <- (ordered-tuple ?a ?b)
        =>
        (retract ?tuple)
        (assert (pattern ?a ?b))

(defrule traverse-list ""
        ?old-list <- (list ?first ?second $?others)
        ?end-list <- (list $?aheads ?next-to-last ?last)
        =>
        (retract ?old-list)
        (retract ?end-list)
        (assert (ordred-tuple ?first ?second))
        (assert (list ?second $?others))
        (assert (tails ?next-to-last ?last))
        (assert (change-list yes)))

(defrule traverse-sub-list ""
        ?change-list <- (change-list yes)
        (list $?list)
        => (retract ?change-list))
```

Fig. 9 List traversal using CLIPS rules

```
(defrule search-narrow-step-commands-3 ""
        (declare (salience 1))
        (goal-is-to (exchange hole pocket))
        (problem ?holecmd ?pocketcmd)
        ?pocket <- (?cmd (superbody ?superbody)
        (name ?pocketcmd)

(reference_plane ?face1) (next_command ?cmd1))
        ?hole <- (hole (superbody ?superbody)
        (name ?holecmd)

(next_command ?cmd2))
        =>
        (assert (history (?pocket ?hole))))

(defrule modify-narrow-step-history-method-3 ""
        ?history <- (history (?pocket ?hole ?mirror))
        ?pocket <- (pocket (name ?name1)
        (next_command ?cmd1))
        ?hole <- (hole (name ?name2)
        (next_command ?cmd2))
        =>
        (retract ?history)
        (assert (reorder (?name1) (?name2)))
        (assert (output history)))
```

Fig. 10 One of design history healing rules

Fig. 9 는 명령어 이력에서 명령어 패턴을 찾기 위한 List 관련 Rule 의 일부로써, Fact 를 이용하여 명령어 List 를 나타내고, List 의 앞부분 n 개의 명령어를 추출한 뒤 List 의 가장 앞부분 명령어를 제거한 수정된 List 에서 다시 앞부분 명령어를 추출하는 기능을 정의하고 있다. 추출된 n 개의 명령어는 같은 순서를 가진 명령어가 명령어 이력에 존재하는지 여부를 탐색하는데 사용된다. 이때 메인 모듈에서의 명령어 패턴 인식은, Fig. 10 의 Rule 에서 정의하는 명령어 Tuple 들이 명령어 이력에 존재하는지 여부를 찾는 것이다. 명령어 Tuple 에 대응하는 명령어들이 명령어 이력 내에 존재하기 위한 제한 조건은, 명령어 이력에 있는 임의의 n 개의 명령어의 순서가 명령어의 Ordered tuple 의 순서와 같다. 임의의 n 개 명령어 사이에 다른 명령어가 존재할 수 있으나, 하나의 Ordered tuple 에 대응하는 명령어 이력 내의 명령어 집합과 다른 Ordered tuple 에 대응하는 설계이력 내의 명령어 집합 사이에 교집합이 없는 것이다. 명령어 패턴 인식을 통해 지식베이스에 있는 명령어 패턴과 동일한 패턴이, 입력된 명령어 이력에 존재하면 명령어 패턴에 대응하는 수정 Rule 이 Fire 된다. 각각의 수정 방법에 대한 설계이력 수정 Rule 들은, 문제가 되는 명령어를 찾는 Rule (Fig. 10 의 첫 번째 Rule)과, 그 순서를 바꾸는 것에 대한 Rule (Fig. 10 의 두 번째 Rule)로 구성된다.

후처리 모듈에서의 명령어 이력 출력 Rule 은 전문가시스템을 Call 하는 메인 프로그램에서 처리할 수 있는 형식으로 수정된 명령어 이력을 출력한다.

Fig. 11 은 앞에서 기술한 전문가시스템의 기능을 IDEF0 다이어그램으로 표현한 것이다.

4. 수정 시스템의 구현

4.1 시스템 구현

Table 1 은 CAD 모델 수정 시스템을 구현하기 위해 사용된 개발환경을 보여준다. 상용 CAD 시스템인 CATIA V5R11 을 기반으로, NASA Johnson Space Center 에서 개발된 전문가시스템인 CLIPS 6.0[15]을 이용하여 Q-Radier 를 개발하였다. "Tiny Face," "Narrow Region," "Non-Tangent Face," "Narrow Step," "Sharp Face Angle," "Narrow Space"의 6 개 항목에 대한 오류를 수정할 수 있다.

Q-Radier 는 4 개의 모듈로 구성된다. 설계이력을 추출하는 모듈 (Design history extraction module)은 CATIA V5 인터페이스 API 인 Component Application Architecture (CAA)를 이용하여 CAD 모델에서

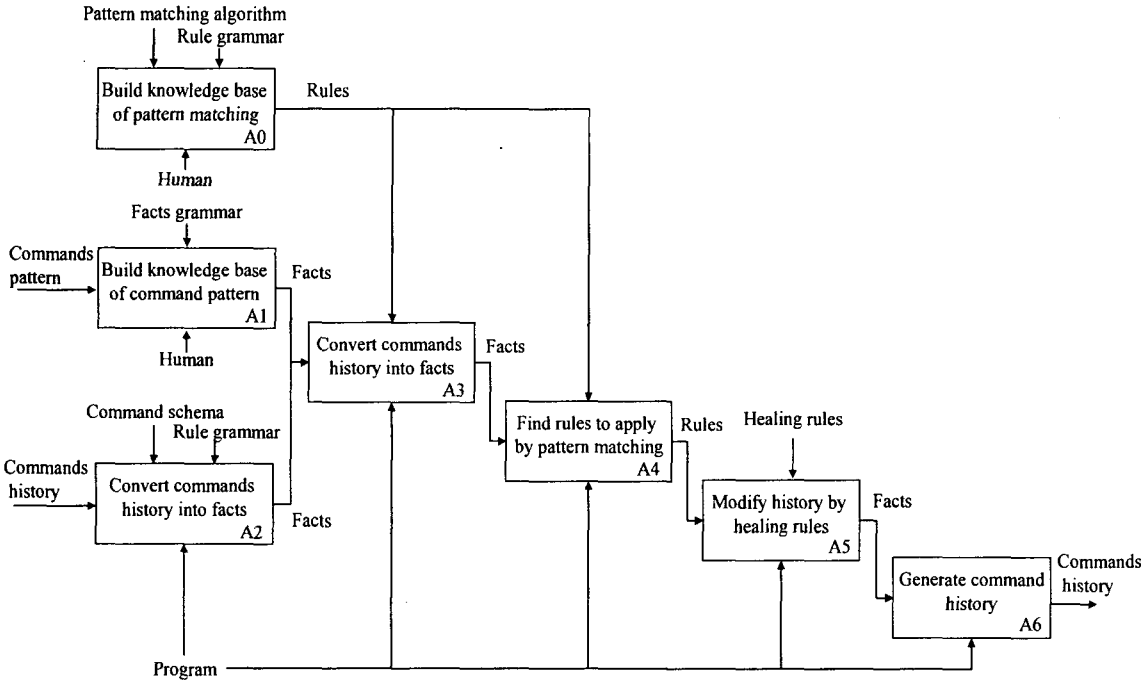


Fig. 11 Functional diagram of rule reasoning process (IDEF0 diagram)

Table 1 Programming environment

Tool	Title
CAD system	CATIA V5 R11 (Dassault Systemes)
Expert system	CLIPS 6.0 (NASA)
CAD Interface	Component Application Architecture (Dassault Systemes)
Programming Language	C++, MS-Win32 APIs (Microsoft)
XML Interface	Document Object Model (W3C)

설계이력 정보를 추출하여, 3.1 절에서 소개된 설계이력 스키마에 따라 XML 구조의 파일을 생성한다.

오류의 위치를 찾는 모듈 (Geometry checking module)은 CAA 를 이용해서 CATIA 모델의 B-Rep 정보를 읽어온다. 이를 바탕으로 6 개 오류 항목에 대한 사용자의 기준값을 바탕으로 수학적 연산을 통해 설계이력 정보에서 오류가 발생한 위치를 찾는다.

CLIPS 6.0 의 Open source code 를 일련의 리스트 (list) 형태의 설계이력 정보를 Rule 로 정의하고, 추론(Backward reasoning) 기능을 수행할 수 있도록

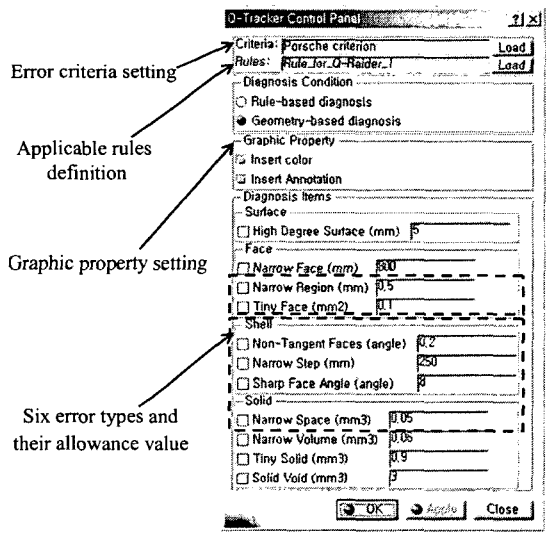


Fig. 12 A control panel of the Q-Raider

수정하여 CATIA 내부 모듈로 포함하였다. 구축된 CLIPS 의 지식베이스는 48 개의 명령어 패턴 매칭 Rule, 39 개의 수정 Rule, 그리고 명령어들간에 의존관계를 해석하는 37 개의 Rule 로 구성되었다.

수정 Rule 의 결과에 따라 설계이력 정보를 재구성하는 모듈 (History reconstruction module)은 변경된 설계이력이 CATIA 환경에서 Re-building 이 가능한지 여부를 확인하고, 변경된 설계이력에 대

한 구속조건을 재 정의한다.

Fig. 12 는 CATIA V5 시스템 내부에 Plug-in 되는 Q-Raider 의 Control panel 을 보여준다. 사용자는 Control panel 을 이용해서, 6 개의 검증 항목에 대한 항목별 기준값의 설정, 오류 수정에 적용하는 Rule 의 설정, 오류가 확인된 엘리먼트에 대한 그래픽 설정, 그리고 6 개 오류 검증 항목에 대한 허용치를 지정할 수 있다.

사용자 인터페이스를 위해 MS-Win32 API 를 이용하였고, XML 스키마와 설계이력 파일에 접근을 위해 DOM 인터페이스를 사용하였다.

4.2 오류의 수정 프로세스

본 논문에서는 오류를 수정하기 위해 설계이력 정보를 이용하고, CAD 모델에 있는 오류의 위치를 파악하기 위해 기존에 연구된 방법과 마찬가지로 B-Rep 정보에 대한 위상 구조와 기하구조의 수학적인 연산 방법을 이용하였다. Rossignac Complex Cell[16] 자료구조를 가지고 있는 CATIA V5 모델을 대상으로 6 개의 오류항목에 대한 검증을 한다. 이들 6 개의 오류 항목에 대한 용어 정의는 SASIG Product Data Quality Documentation V1.0[17]을 참고 하였다.

Fig. 13 은 오류를 수정하는 과정을 보여 준다. (1) Q-Raider 가 구동되면, CATIA 시스템에서 활성화된 형상 모델로부터 자료구조(Data structure)를 초기화하여 위상요소와 기하요소에 대한 데이터와 그래픽 정보를 가져온다. (2) 6 개의 오류 항목에 대한 기준값을 바탕으로 기하요소와 위상요소에 대한 수학적인 연산과정을 통해 오류를 검증한다. (3) 오류가 발생한 위치에 대해 Color 와 Arrow 를 표시하고, 오류 내용에 대해 Annotation 으로 표기한다. 마지막으로 (4) 전문가시스템에 구축된 Rule 을 바탕으로 추론과정을 통해 설계이력을 재 구성하여 오류를 수정한다.

전문가시스템 내부에서 오류를 수정하는 과정은 6 개의 단계로 진행된다. (a) 형상정보에 대한 연산과정을 통해 확인된 오류 결과를 바탕으로 CAD 모델로부터 설계이력 정보를 추출한다. (b) 설계이력 정보에서 오류가 발생한 위치를 찾는다. (c) 전문가시스템의 Rule 을 바탕으로 추론 과정을 통해 오류를 회피할 수 있는 이력 정보를 제시한다. (d) 제시된 이력 정보를 기반으로 새로운 설계이력을

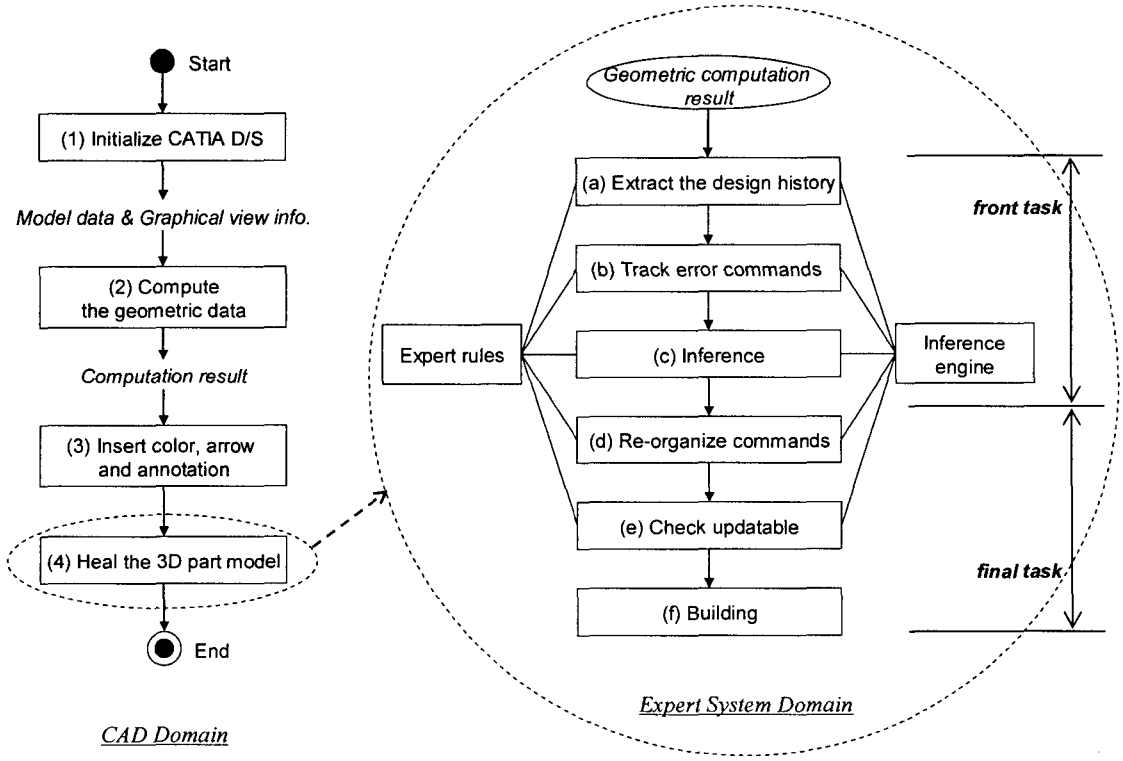


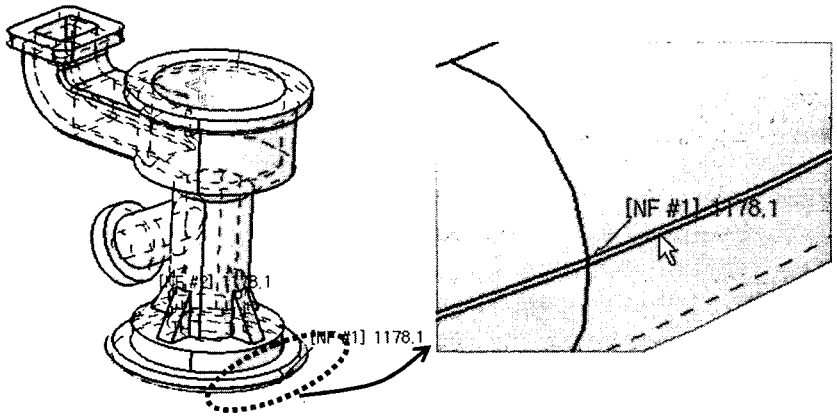
Fig. 13 Healing process

재 구성한다. (e)수정된 설계이력에 대해 CAD 시스템이 내부적으로 Building 이 가능한지 확인한다. (f)마지막으로 변경된 설계이력에 대해 새로운 구속조건을 정의하고 최종 형상을 생성한다.

(a)에서 (d)까지는 지식베이스의 Rule 에 따라 설계이력 정보 사이에 발생된 의존관계에 대한 해석과 각각의 명령어 패턴 분석, 그리고 파라메트릭 데이터 분석을 위한 해석 프로세스가 진행된다.

4.3 실험 결과

첫 번째 실험으로서, Fig. 14 는 설계이력을 변경하여 CAD 모델의 오류를 수정한 예를 보여 준다. 설계자는 아래쪽 Base 형상 부분을 만들기 위해서 2D Circle 의 Profile (Sketch.11)를 이용하여 Pad feature (Pad.7)를 생성한 후에, Pad.7 에 포함되어 있는 두 개의 Edge (Edge.14, Edge.15)를 이용하여 Edge fillet feature (EdgeFillet.5)를 생성하였다.




```
+ <Body body_name="Body.2">
- <Body body_name="PartBody">
  + <Shaft>
  + <EdgeFillet>
  + <Stiffener>
  + <CircPattern>
  + <Pad>
  + <Pad>
  + <Rib>
  + <EdgeFillet>
  + <Pad>
  + <EdgeFillet>
  + <Remove>
  - <Pad>
      <name>Pad.7</name>
      <first_limit_type>0</first_limit_type>
      <first_length>10.000000</first_length>
      <second_limit_type>0</second_limit_type>
      <second_length>0.000000</second_length>
      <profile_name name="Sketch.11" super_type="PartBody"
      <reference_plane>Face.1</reference_plane>
    </Pad>
  - <EdgeFillet>
      <name>EdgeFillet.5</name>
      <fillet_type>0.000000</fillet_type>
      <radius>4.900000</radius>
      <propagation_type>0</propagation_type>
      <target_edges>Edge.14</target_edges>
      <target_edges>Edge.15</target_edges>
    </EdgeFillet>
  </Body>
+ <Body body_name="Open_body.1">
</HistorySchema>
```

```
+ <Body body_name="Body.2">
- <Body body_name="PartBody">
  + <Shaft>
  + <EdgeFillet>
  + <Stiffener>
  + <CircPattern>
  + <Pad>
  + <Pad>
  + <Rib>
  + <EdgeFillet>
  + <Pad>
  + <EdgeFillet>
  + <Remove>
  - <Shaft>
      <name>Shaft.3</name>
      <first_limit_angle>360.000000</first_limit_angle>
      <second_limit_angle>0.000000</second_limit_angle>
      <profile>Sketch.12</profile>
      <axis />
    </Shaft>
  </Body>
+ <Body body_name="Open_body.1">
</HistorySchema>
```

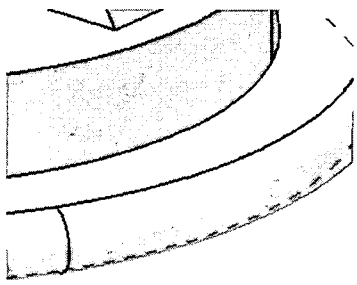


Fig. 14 Case study 1: healing of the narrow faces

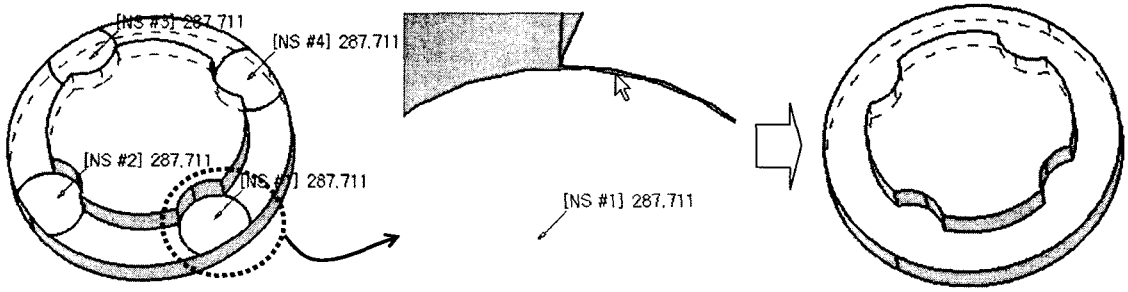
Geometric data 에 대한 오류 검증 과정을 통해 외곽 경계곡선 (Outer loop) 길이와 내부 경계곡선 (Inner loop)의 길이의 비가 1178 인 Narrow face 가 발견되었다. Narrow face 는 모델링 과정에서 Project 이 불가능하여 Offsetting, Drafting, Filleting 이 어렵다. 또한 메쉬 생성이 어려워 CAE 해석 계산이 길어지고, 데이터 변환 시 Face 가 손실될 가능성이 높다.

Fig. 14 의 오른쪽 XML 파일은 전문가시스템을 이용해서 오류가 수정된 설계이력정보를 보여준다. 아래쪽 Base 형상을 만들기 위해서 사용한 명령어들 (Pad.7, EdgeFillet.5)을 Base 형상의 단면과 같은 Profile(Sketch.12)를 자동 생성한 후에, Profile 을 회전시킨 Shaft feature 로 변경하여 Shaft.3 을 생성하였다. 회전체 대칭형 형상에 Narrow face 가 발생하는 경우에 대해 새로운 Profile 을 생성해서 오류를 수정하는 Rule 을 이용하였다. Fig. 14 의 하단 그림은 수정된 설계이력을 CATIA 에서 Building 한 형상을 보여준다.

두 번째 실험으로서, Fig. 15 는 Narrow step 이 발견된 형상을 보여 준다. 설계자는 위쪽 원판과 아래쪽 원판에 의해 솔리드를 생성한 후, Circular Pattern 으로 생성된 4 개의 실린더에 대해 Removal Boolean 을 수행하였다. 이때 Boolean 명령어가 적용된 아래쪽 원판에서 0.09mm 깊이의 Narrow step 이 발생하였다. Step 이 발생한 부분의 Face(화살표 표시)에서 연속된 Edge 들 사이의 길이의 비율이 287 이다. Narrow step 의 오류가 발생한 Feature 명령어의 위치를 확인하고, 오류에 영향을 주는 명령어들간의 의존 관계를 전문가시스템을 통해 추론하여, 설계이력정보에서 명령어 순서가 변경된 설계이력을 생성하였다. 즉, 오류를 수정하기 위해 Pad.2 와 Remove.1 의 순서를 Pad.4 와 Removal.1 으로 변경해서 오류를 수정하였다.

5. 결 론

양질의 디지털 데이터는 기업의 중요한 자산이 되고 있다. 그러나 CAD 모델의 오류와 같은 품질 문제로 인해 많은 인력과 시간 그리고 비용이 투입되는 뿐만 아니라, 치명적인 문제로 인해 생성된 데이터를 재 사용하지 못하는 경우도 발생하고 있다.




```
- <Body body_name="Body.2">
  + <Pad>
  + <CircPattern>
  </Body>
- <Body body_name="PartBody">
  + <Pad>
  - <Pad>
      <name>Pad.2</name>
      <first_limit_type>0</first_limit_type>
      <first_length>10.000000</first_length>
      <second_limit_type>0</second_limit_type>
      <second_length>0.000000</second_length>
      <profile_name name="Sketch.2" super_type="PartBody" />
      <reference_plane>xy plane</reference_plane>
    </Pad>
  - <Remove>
      <name>Remove.1</name>
      <second_operand>Body.2</second_operand>
    </Remove>
  </Body>
</HistorySchema>
```

```
- <Body body_name="Body.2">
  + <Pad>
  + <CircPattern>
  </Body>
- <Body body_name="PartBody">
  + <Pad>
  - <Remove>
      <name>Remove.1</name>
      <second_operand>Body.2</second_operand>
    </Remove>
  - <Pad>
      <name>Pad.4</name>
      <first_limit_type>0</first_limit_type>
      <first_length>-10.000000</first_length>
      <second_limit_type>0</second_limit_type>
      <second_length>0.000000</second_length>
      <profile_name name="Sketch.2" super_type="PartBody" />
      <reference_plane>xy plane</reference_plane>
    </Pad>
  </Body>
</HistorySchema>
```

Fig. 15 Case study 2: healing of the narrow steps

CAD 모델의 수정에 대한 기존 연구는 대부분 B-Rep 형상을 대상으로 하고 있다. Engineering data 가 제외된 B-Rep 형상은 수정 후에 선행공정이나 후속공정에 수정된 결과가 반영되기 어렵다.

본 논문에서는 이를 해결하는 방법으로, 설계의도(Design intent)가 반영된 설계이력 정보를 Rule 로 구축된 전문가시스템을 활용해서, CAD 모델을 수정하는 시스템을 구현하였다.

전문가시스템의 지식베이스는 오류가 포함된 명령어 이력을 수정하여 오류가 없는 명령어 이력을 생성하는 Rule 의 집합이다. C++과 같은 범용 프로그래밍 언어를 이용해서 하드코딩을 통해 이러한 Rule 을 구현 할 수 있지만, 전문가시스템의 추론 엔진에 해당하는 알고리즘의 구현이 필요하고, Rule 을 수정 및 추가할 때마다 시스템을 재 컴파일 해야 하는 단점이 있다. 전문가시스템을 이용하면, 일단 지식베이스의 골격을 구성한 후에는 Rule 의 수정 및 추가가 편리하기 때문에, 설계 이력에서 나타나는 다양한 오류 상황에 유연하게 대처할 수 있는 장점이 있다.

향후 시스템을 확장하기 위해서는 오류 항목의 확장과 신뢰성 있는 오류 수정을 위해서 추가적인 지식베이스 구축이 필요하다.

후 기

이 논문은 2005 년도 한국학술진흥재단의 지원에 의하여 연구되었음. (KRF-2005-M01-10306).

참고문헌

(1) Walker, D., 2001, "Introduction of TOPGUN XI," *Proceedings of 2001 COE Conference*, Anaheim, CA, April 4.

(2) Izurieta, C.E., 2000, "CAD Model Interoperability Solutions for Rapid Prototyping," *Rapid Prototyping*, Vol. 6, No. 2, pp. 1~4.

(3) Tassey, G., 1999, Interoperability Cost Analysis of the U.S. Automotive Supply Chain: Final Report, RTI Project Number 7007-03, Research Triangle Institute.

(4) Hoffman, C.M. and Robert J.A., 1998, "CAD and the Product Master Model," *Computer-Aided Design*, Vol. 30, No. 11, pp. 905~918.

(5) Gu, H., Chase, T.R., Cheney, D.C., Bailey, T. and Johnson, D., 2001, "Identifying, Correcting, and Avoiding Errors in Computer-aided Design Models Which Affect Interoperability," *Journal of Computing and Information Science in Engineering*, Vol. 1, pp. 156~166.

(6) Deshpande, V., Fornasier, L., Gerteisen, E.A., Hilbrink, N., Mezentsev, A., Merazzi, S. and Wöhler, T., 2000, "Virtual Engineering of Multi-Disciplinary Applications and the Significance of Seamless Accessibility of Geometry Data," *Future Generation Computer Systems*, Vol. 16, pp. 435~444.

(7) Barequet, G., 1997, "Using Geometric Hashing to Repair CAD Objects", *IEEE Computational Science & Engineering*, pp. 22~28.

(8) Steinbrenner, J.P., Wynman, N.J. and Chawner, J.R., 2001, "Procedural CAD Model Edge Tolerance Negotiation for Surface Meshing," *Engineering with Computers*, Vol. 17, pp. 315~325.

(9) Volpin, O., Sheffer, A., Bercovier, M. and Joskowicz, L., 1998, "Mesh Simplification with Smooth Surface Reconstruction," *Computer-Aided Design*, Vol. 30, No. 11, pp. 875~882.

(10) Yang, J. and Han, S. 2002, "Healing of STEP AP214 Automotive CAD Data," *Transactions of the Society of CAD/CAM Engineers*, Vol. 7, No. 3, pp. 170~176. (in Korean)

(11) Yang, J., Han, S.H. and Park, S.H., 2004, "A Method for Verification of CAD Model Errors," *Journal of Engineering Design*, Vol. 16, No. 2, 2005.

(12) Choi, K.H., Mun, D.H. and Han, S., 2002, "Exchange of CAD Part Models Based on the Macro-Parametric Approach", *International Journal of CAD/CAM (www.ijcc.org)*, Vol. 2, No. 2, pp. 23~31.

(13) Mun, D., Han, S. and Oh, Y.C., 2003, "A Set of Standard Modeling Commands for the History-Based Parametric Approach," *Computer Aided Design*, Vol. 35, pp. 1171-1179.

(14) Yang, J., Han, S., Kim, B.-C. and Park, C.-C., 2003, "A Macro Parametric Data Representation for CAD Model Exchange using XML," *Transactions of the Korean Society of Mechanical Engineers*, Vol. 12, No. 12, pp. 2061~2071. (in Korean)

(15) CLIPS: A Tool for Building Expert Systems, http://www.ghg.net/clips/CLIPS.html

(16) Rossignac, J. and O'Connor, M., 1989, SGC: A Dimension Independent Model for Point Sets with Internal Structures and Incomplete Boundaries," Geometric Modeling for Product Engineering. Eds M. Wosny, J. Turner, K. Preiss, North Holland, pp. 145~180.

(17) SASIG PDQ Documentations V1.0, ISO TC184-SC4: SC4 N-DOCS, http://www.tc184-sc4.org/SC4_Open/SC4_and_Working_Groups/SC4_N-DOCS/1250-1499/maindisp.cfm?bk=7480.

(18) Yang, J. and Han, S., "Healing of CAD Model Errors Using Design History", *Transactions of the Society of CAD/CAM Engineers*, 2005. (Submitted)