한국 CAD/CAM 학회 논문집
제8권 제2호 2003년 6월 pp. 90-100

학술논문

일관된 제품자료관점을 지원하는 설계변경 전달에 관한 연구

도 남 철*

Propagation of Engineering Changes for Supporting Consistent Product Data View

Do, N. C.*

ABSTRACT

Engineering change (EC) objects are the data structure and related operations that can support applications for EC procedures or processes. Their functionalities include controlling management data, specifying related product structure, and archiving a history of product structure changes for EC management. In this paper we introduce a systematic approach to support the propagation of changes between different product structure views using the history of structure changes in EC objects. The change propagations supported by EC objects enable designers to maintain the consistency of multiple product structure views for engineering, manufacturing or even customer support applications. This paper also includes EC examples and experimental implementations for the proposed EC objects.

Key words : Engineering changes, Product data views, Change propagations

1. 서 론

정보화 시대인 현대의 기업은 경쟁력 있는 제품을 개발, 생산, 그리고 서비스하기 위하여 제품 일생주기 동안에 일관된 제품정보를 유지 하여야 한다. 설계변경(Engineering changes)은 전체 일생주기 중에 제품정보의 변화를 일으키는 주된 요인이며, 동시에 변화에 관계된 많은 정보의 생성 및 처리를 주도한다. 그러므로 경쟁력 있는 제품을 생산하기 위하여는 항시 일관된 제품정보를 유지 시키는 설계변경 관리 체계가 필요하다. 특히 기업간 협동작업이 전략적 요소가 되고 있는 최근에는 다양한 응용분야의 일관된 제품정보 공유를 위한 설계변경 관리의 중요성이 더욱 커지고 있다.

하지만 현재 설계변경에 관한 연구는 비교적 적을 뿐만 아니라[1] 설계변경 관리를 지원하는 상업용 정보 시스템들도 제품구조(Product structure), 제품형상(Product configuration) 그리고 제품자료관점(Product data view)과 통합된 설계변경을 지원하지 못하고 있다. 특히 기업에서는 이질적인 정보 및 관리 시스템을 이용하여 설계변경을 다루기 때문에 효과적이고 통합된 관리에 어려움을 겪고 있다.

본 논문에서는 제품구조와 통합된 설계변경 객체(Engineering change objects)를 이용하여 복수의 제품자료관점의 일관성을 자동으로 유지할 수 있는 체계를 제안한다. 이 체계에서 설계변경 객체는 제품구조, 제품형상 그리고 제품자료관점과 긴밀히 통합되어 설계변경 이력 및 설계변경 단위의 구현이 가능하며 제품자료관점은 기본제품자료(Base product data)로부터 자료 중복 없이 특정 응용분야를 위한 새로운 자료 구조를 유도한다.

본 논문에서 사용하는 모델은 설계변경 객체와 제품자료관점 그리고 그들 사이의 관계에 중점을 두고 있다. 제안된 설계변경 객체는 제품형상 및 제품구조 등 다른 제품자료 모델(Product data model)과도 일관성 있게 통합되어 있으나 제품형상, 제품구조는 본 논문의 관심대상이 아니며, 이는 저자의 이전 연구[2]에 소개되어 있다. 본 논문에서는 설계변경 확산을 위하여 확장된 설계변경 객체의 연산과 이를 지원하는 제품자료관점 모델의 상호관계가 관심 영역이 된다.

본 논문의 2장은 관련연구와 이에 대한 본 연구와의 차이를 서술한다. 3장에서는 통합제품자료 모델 중

*정회원, 경상대학교 공과대학 산업시스템공학부
- 논문투고일: 2002. 11. 14
- 심사완료일: 2003. 04. 18

관심 대상인 제품자료관점과 설계변경 자료모델에 관하여 알아본다. 4장에서는 기본제품자료에서 제품자료관점으로의 설계 변경 전달 Procedure를 제안한다. 5장에서 제안된 Procedure의 상업용 데이터베이스 관리시스템을 이용한 구현을 설명하고, 6장에서는 논점을 재정리하며 결론을 도출한다.

2. 관련 연구

2.1 설계변경에 관한 연구

설계변경의 관련연구 분야 중 하나로 설계이력(Design history)에 관한 연구를 들 수 있다. 설계이력에 관한 연구는 설계과정을 추적하여 이에 관련된 지식을 추출하고 이를 재 사용하는데 목적을 두고 있다[3]. 그러므로 설계하는 과정의 저장 뿐만 아니라 이 기록에서 지식을 추출하기 위한 도구 및 기제들을 고안하여 사용하고 있다. 이에 반하여 설계변경은 주로 설계변경 결과를 기록하고 이를 관련된 분야에 효율적으로 전달하기 위한 부분에 관심을 가지고 있다. 그러므로 설계이력에 관한 연구 중에 설계과정을 기록하는 부분은 설계변경의 이력기록과 밀접한 관계가 있다. 본 연구에서는 설계이력 연구들의 이력 기록 부분을 관련 연구에 포함시켜 비교해 보고자 한다.

설계변경에 관한 연구는 제품구조와 통합되어 있는가 여부에 따라 크게 두 가지로 나눌 수 있다. 하나는 제품구조와 관계없이 독립된 문서관리나 설계변경 관리 시스템을 이용하여 설계변경을 관리하는 방법이고 다른 하나는 제품구조 변경이 설계변경 이력으로 기록되어 제품구조상에 설계변경 정보를 통합하는 접근 방법이다.

2.1.1 제품구조를 고려하지 않은 설계변경

일반적으로 제품구조를 고려하지 않는 설계변경 관리방법은 설계변경 관리와 설계변경으로 인한 제품구조 변화를 분리하여 관리하게 된다. 이러한 분리는 설계자가 설계변경 프로세스 중에 가장 중요하며 자주 발생하는 제품구조 및 관련 정보의 분석 및 검토를 어렵게 하며 이질적이고 복잡한 정보시스템 환경에서 다양한 정보를 가공하고 수집해야 하는 부담을 주게 된다[4]. 그러므로 이 접근 방법은 설계변경 처리 과정에 많은 시간을 소요하도록 할 뿐만 아니라 많은 오류를 발생시킬 수 있는 단점이 있다. 이러한 제품구조와 분리된 연구로는 [1,5,6,7,8,9]의 연구가 있으며 Huang의 연구[1]에서는 추후 연구로 제품구조와 통합된 설계변경 관리 시스템을 제안하고 있다.

2.1.2 제품구조와 통합된 설계변경

Shah[3]는 Hyper text 등으로 제공되는 설계변경 시스템의 단점을 지적하며 제품자료 호환 표준인 ISO STEP[10]을 기초로 설계 Process 지원 Entity를 포함한 설계이력 관리 시스템을 제안하였다. 이 연구는 데이터베이스를 이용한 질의를 포함한 효과적인 설계변경 정보관리 시스템을 제안하였으나 설계변경 이력을 통한 일관성 유지 문제는 다루고 있지 않다.

상용제품으로 Enovia VPM[11]은 제품구조와 밀접히 연관된 Action이라는 객체를 지원한다. 특히 Action은 제품구조의 변경을 기록하고 이를 사용자가 일람할 수 있도록 하는 기능을 가지고 있다. 하지만 Enovia VPM은 제품자료관점 기능을 지원하지 않으며, 해당 설계변경이력을 단지 기록만할 뿐 이를 재사용하는 기능을 제공하지 않는다.

STEP PDM Schema[12]는 ISO STEP의 제품자료 관리를 위한 Entity를 재 구성하여 만들어진 제품정보 모델이다. STEP PDM Schema는 'versioned_action_requirement', 'action_method' 그리고 'directed_action' entity로 구성된 설계변경요구(Engineering change request) 및 설계변경명령(Engineering change order) 구현에 필요한 자료구조를 지원하고 있으며 'product_definition_formation'과 'product_definition_relationship' entity로 표현되는 부품이나 부품구조를 설계변경 Entity에 Associate시키고 있다. 하지만 제품구조 변경 이력을 기록할 구조가 명시적으로 표현되지 않고 있으며, 본 논문에서 제안하고 있는 제품변경 이력을 이용한 제품자료관점의 일관성 유지 문제는 아직 지원하지 못하고 있다.

Peng[13]은 ISO STEP을 기반으로 자체적으로 고안한 'engineering_change_notice', 'EC_distribution' 그리고 'EC_dep' entity를 추가하여 설계변경과 제품구조를 포함한 7가지 모델을 통합한 제품자료 모델을 제시하였다. 이 연구에서 제안한 모델 중 설계변경 모델에서는 설계변경 이력에 대한 언급이 없이 설계변경 관리에 연구의 중심을 두었다. 그러므로 설계변경의 제품자료관점에 관한 전달도 연구에서 다루고 있지 못하다.

2.2 제품자료관점(Product data view)에 관한 연구

DH Brown[14]은 일반적인 기업체에서 제품전반에 걸친 빠른 의사결정(Total-product decisions)을 지원하는 효과적인 제품정보 관리를 위하여 서로 다른 12개의 제품자료관점이 필요함을 보고하고 있다. 또한 이 제품자료관점은 상호간 일관성 유지가 매우 중요한

것으로 보고하고 있다. 그러나 제품자료관점은 일반적인 데이터베이스 관점(Database view)과는 달리 기초 자료구조(Base database)의 변경이 각 자료관점으로 파급되는 데는 복잡한 과정이 필요하며 특성상 다른 제품자료 표현 영역과 통합된 자료 모델의 일부로 표현되고 있다. 통합된 제품자료모델에 관한 연구 중에 제품자료관점과 설계변경 관리를 표현하고 있는 연구는 다음과 같다.

PDM Schema[12]는 설계변경뿐만 아니라 STEP의 Generic resource에 표현된 'product_definition' entity를 사용하여 제품자료관점을 표현할 수 있다. 이는 Action entity를 중심으로 하는 설계변경 Entity와 연동될 수 있으나 기본제품자료를 표현하는 'product' 및 'product_definition_formation' entity와 제품자료관점('product_defintion') entity 간의 설계변경 전달에 관한 내용을 포함하고 있지 못하다. Peng[13]의 연구에서도 PDM Schema[12]와 동일한 STEP Schema를 기본으로 이용하여 설계변경관리에 관련된 확장된 Entity를 제안하였으므로 역시 설계변경이력과 설계변경 전달을 표현하지 못하는 문제를 가지고 있다.

Kate[15]은 'Component Hierarchy', 'Version History' 그리고 'Equivalencies'와 이들간의 관계로 구성된 제품자료모델을 제안하였다. 각각은 본 논문에서 제안하고 있는 제품사양을 포함한 제품구조, 설계변경이력 그리고 제품자료관점과 대응될 수 있다. 그의 연구에서는 'Configuration'이 'Version History'와 긴밀히 통합되어 있으나 본 논문에서 제안하고 있는 'Version history'를 이용한 'Equivalencies'로의 설계변경 전달은 표현되지 않았다.

Hamer[16]는 'Version', 'View', 'Hierarchy', 'Status' 그리고 'Variant' 등 5가지 Dimension으로 구성된 제품설계자료 모델을 제안하고 이들 중에 2개의 Dimension을 조합한 모델(예로 'Hierarch' & 'View' 혹은 'Version' & 'View')도 제안하였다. 하지만 해당 연구에서는 조합된 모델에서 생성될 수 있는 모델의 종류와 이때 현실적으로 생길 수 있는 문제점을 나열하였고 이를 해결할 수 있는 구체적 모델을 제시하지 않고 있다. 특히 3차원 이상의 문제에 관하여 가능성을 언급하였지만 실제 모델을 제시하지는 못하고 있다. 이에 반하여 본인의 연구는 제품형상, 조립구조, 설계변경 그리고 제품자료구조로 이루어진 구체적인 4차원적인 통합모델을 기반으로 설계변경 및 설계변경의 제품자료관점으로 전달을 제안하고 있다.

위의 관련 연구를 종합하면 기존의 설계변경에 관한 연구는 제품이력과 설계변경 단위 등에 관한 제품구조 관점의 연구가 부족하며, 기존의 제품자료관점에 관한 연구는 전체 제품자료모델에 통합에 관한 문제에 치중하여 어떻게 기본제품자료의 변화를 제품자료 관점에 전달 시켜 전체 제품정보의 일관성을 유지할 것인가에 관한 연구가 미흡하다. 그러므로 본 연구에서는 설계변경과 제품자료관점을 통합하여 각각의 단점을 보안하는 Framework을 제공한다. 이 Framework에서는 제품구조 및 제품자료관점과 통합된 설계변경 객체의 설계변경 이력 정보를 바탕으로 기본제품자료의 설계변경을 다양한 제품자료관점으로 효율적으로 파급 시킬 수 있는 체계적인 방법을 제공한다.

3. 일관된 제품구조를 위한 모델

저자는 [2]에서 제품 정보 일관성 유지를 위한 제품구조 체계를 제안하였다. 이 체계는 제품형상, 조립구조(Assembly structure), 제품자료관점 그리고 설계변경 Schema로 구성되어 있다[2]. 이 4개의 Schema는 각기 관련된 *Entity*의 집합으로 구성되어 있으며 Schema 상호간에도 긴밀히 통합되어 있다. 본 장에서는 이 Schema 중에 관심 사항인 제품자료관점과 설계변경 Schema와 이들 사이의 관계에 관하여 살펴본다.

3.1 Product Data View Schema

제품자료관점 Schema는 제품의 개발, 생산 그리고 사용 주기동안 하나의 기본제품자료에 대하여 자료의 중복 없이 다양한 응용시스템을 지원하기 위한 서로 다른 자료 관점(Data view)을 제공하는 Entity 집합을 뜻한다. 각 응용 시스템은 기본제품자료로부터 자신의 응용 목적에 따라 기본 자료를 직접히 여과하거나 변형 시켜야 한다. 이때 중요한 제약조건은 기본제품자료와 다양한 자료 관점 사이의 일관성 유지이다. 특히 기본제품자료에서 설계변경과 같은 동적인 변경을 어떻게 각 응용 시스템에 전달(*Change propagation*) 시키는지 여부가 일관성 유지의 중요한 문제 중에 하나이다.

제품자료관점의 예로 기술자료 출판중의 하나인 (Service) parts catalog를 들 수 있다. Parts catalog는 제품설계 자료를 기본제품자료로 하여 고객이나 서비스 요원이 쉽게 운용중인 제품을 서비스하고 필요한 부품을 공급할 수 있도록 서비스 부품만 Filtering 하고 제품구조를 Catalog의 순서에 맞게 변형 시킨 제품자료관점의 응용 분야 중에 하나이다. 이때 이 응용 분야를 위하여 기본제품자료로부터 새로

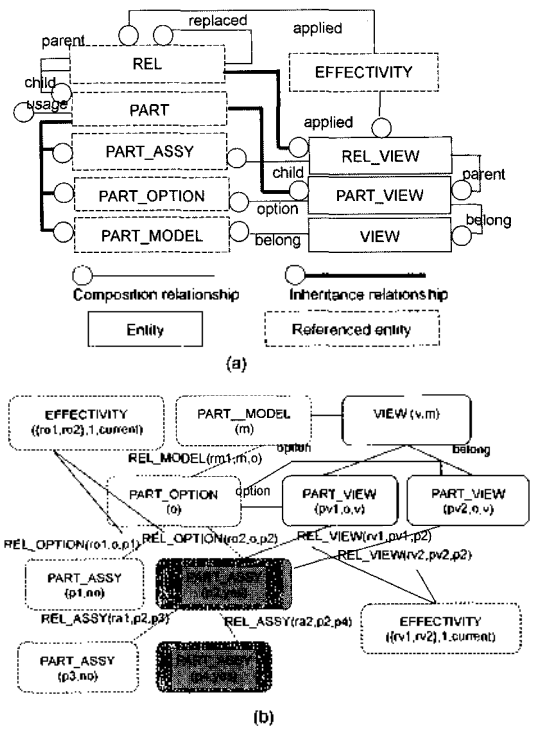


Fig. 1. 제품자료관점 Schema(a) 및 예(b).

운 복사본을 만들지 않고 단지 새로운 관점을 만들면 추후 기본제품자료의 설계 변경 등으로 Catalog의 개정판 출판 시 신속하고 중복되지 않는 자료관리에 의한 일관성 유지가 가능하게 된다.

Fig. 1(a)는 제품자료관점 Schema의 Entity와 상호관계를 보여 주고 있다. 'VIEW' entity는 제품의 모델 군을 표시하는 'PART_MODEL' entity의 한 관점을 표시한다. 예로 특정 모델의 Service parts catalog를 대표하는 Entity가 올 수 있다. 'PART_VIEW' entity는 Generic entity인 'PART' entity로부터 상속받는 Entity로써(굵은 선으로 표시) 'VIEW' entity를 구성하는 요소로 제품형상의 구성 요소인 'PART_OPTION' entity와 대응된다. 'REL_VIEW' entity는 일종의 Association entity로써 'PART_VIEW'와 기본제품자료를 구성하는 'PART_ASSY'와 연관을 가지고 있다. 아울러 'EFFECTIVITY' entity를 'REL_VIEW' entity에 정의함으로써 'PART_VIEW' instance가 조건에 따라 다른 제품구조를 표현할 수 있는 Variant구조를 가지게 한다. 이외에 중요한 역할을 하는 Attribute로써 'PART' Entity의 'usage' attribute를 들 수 있다. 이 Attribute는 해당 부품이 특정 제품자료관점에서 사용되는지 여부를 표시한다. 즉, 이 Attribute는 기본제품자료로부터 제품자료관점에 필요한 부품만 Filtering 하는데 사용하게 된다.

위의 Entity들로 제품자료관점을 형성할 때는 'PART' 객체를 기본 자료로 삼아 Mapping과 Filtering의 두 가지 기제를 사용한다. Mapping이란 상위 부품을 공유함으로써 기본제품자료에 있는 제품구조를 공유할 때 사용되는 기제이고 Filtering은 위의 설명과 같이 기본제품자료에 속한 부품 중 해당 제품자료관점에 사용되는 부품만 선별하는 것이다. Fig. 1(b)는 위의 제품자료관점 Schema를 이용하여 어떻게 기본제품자료로부터 특정 제품자료관점을 도출하는 지를 보여주고 있다.

Fig. 1(b)는 Fig. 1(a)의 Schema의 Instance를 다음의 Predicate 형식으로 표시한 것이다. Predicate에 쓰인 Attribute 중 'ID'는 객체 식별자(Object identifier)를 뜻한다. 그리고 이해를 돕기 위하여 'REL' entity 및 특정 Entity간의 관계를 실선으로 표시하였다.

PART_MODEL(ID), PART_OPTION(ID),
PART_ ASSY(ID,usage),
REL_MODEL(ID,parent,child),
REL_OPTION(ID, parent,child),
REL_ASSY(ID,parent,child),
VIEW(ID,belong), PART_VIEW(ID,option,view),
REL_VIEW(ID,parent, child),
EFFECTIVITY({applied},start,end)

'PART_MODEL', 'PART_OPTION' 그리고 'PART_ASSY'로 구성된 기본제품자료는 전형적인 Option BOM[17] 형태로 제품구조를 표현하고 있다. 제일 상위에는 제품의 개념적 모델을 뜻하는 'PART_MODEL(m)'이 있고, 이는 'REL_MODEL(rm1,m,o)'을 이용하여 제품형상을 구성하는 'PART_OPTION(o)'와 연결 되어 있다. 'PART_OPTION(o)'는 실제적 제품구조를 표현하고 있는 'PART_ASSY(p1,yes)', 'PART_ASSY(p2,yes)'와 'REL_OPTION(ro1,o,p1)', 'REL_OPTION(ro2,o,p2)'으로 연결되어 있다.

'VIEW(v,m)'는 'belong'관계로 연결된 모델'PART_MODEL(m)'의 한 관점을 대표하는 객체이다. 예로 Parts catalog 자료 관점이 될 수 있다. 이 객체는 제품자료관점을 구성하는 객체인 'PART_VIEW(pv1,o,v)', 'PART_VIEW(pv2,o,v)'로 구성되어 있으며 이는 'PART_OPTION(o)'를 'option' attribute로 연결함으로써 제품형상에 따라 필요한 제품자료관점 정보를 연결할 수 있다. 'PART_VIEW(pv1,o,v)'와 'PART_VIEW(pv2,o,v)'는 'REL_VIEW' entity를 이용하여

'PART_ASSY(p1,yes)'를 'PART_OPTION(o)'와 공유하고 있으며, Parts catalog관점을 구성하는 Chapter 등이 예로 될 수 있다. 그리고 'PART_OPTION'과 연관관계('option' attribute를 통한)는 추후 어떤 제품 사양에 대한 Chapter인지를 구분하는데 사용된다. 이와 같이 상위 부품('PART_ASSY(p1,yes)')을 공유하므로써 하위 제품구조('PART_ASSY(p4,yes)'를 포함한 하위 제품구조)를 모두 공유하기 위하여 Mapping 기제를 사용하고 있다. 반면에 'PART_ASSY'의 Attribute중에 'usage' attribute는 각각의 'PART_ASSY' 객체가 해당 제품자료관점에 속하는지 여부를 표시한다. Fig. 1(b)에서 'PART_ASSY(p3,no)'가 'PART_ASSY(p1,yes)'의 하위 부품으로 Mapping 기제상으로는 제품자료관점에 참여하는 부품처럼 포함되었으나 'usage' attribute가 'no'이므로 해당 제품자료관점에 포함되지 않는다. 이를 표현하기 위하여 Filtering 기제를 사용하였다.

제안된 제품자료모델은 'PART_OPTION' entity를 Variant로 채용하고 있다. Variant란 특정 조건(Effectivity)에 의하여 자신의 제품의 구조를 변화시킬 수 있는 제품을 뜻한다. Effectivity에는 다양한 종류가 있을 수 있으나 본 논문에서는 서술을 단순화시키기 위하여 호기수 Effectivity(Serial numbered effectivity)만을 표현한다. Fig. 1(a)에서 Effectivity는 'REL_OPTION' 객체에 적용된 것을 볼 수 있으며, 'PART_VIEW'도 Variant 구조를 가지고 있으므로 각 'PART_OPTION'에 대응하는 'PART_VIEW'와 'PART_ASSY'를 연결하는 'REL_VIEW'에도 Effectivity가 적용된 것을 볼 수 있다. 특히 Fig. 1(b)의 예에서 보는 바와 같이 'PART_ASSY' 객체를 'PART_OPTION' 객체와 공유하는 (해당 객체와 'option' attribute로 연결된) 'PART_VIEW'객체는 해당 'REL_OPTION' 객체에 정의된 Effectivity를 공유하고 있다(Fig. 1(b)의 1-current Effectivity).

3.2 Engineering Change Schema

설계변경 Schema는 설계변경 객체를 표현하기 위한 Entity의 집합이며, 설계변경 객체란 제품의 설계변경을 기록, 관리하며 설계변경 관련 정보를 연결하는 역할을 하는 자료구조 및 관련 연산을 뜻한다. 설계변경 객체의 기록 대상은 궁극적으로 제품형상 하에 정의된 제품구조에 영향을 미치는 제품정보의 변화이다. 즉 변화의 대상은 'REL' entity의 하위 Entity('REL_MODEL', 'REL_OPTION', 'REL_ASSY' 그리고 'REL_VIEW' entity) 들의 변동이다.

설계변경 대상이 되는 'REL' entity에는 추가(Add), 삭제(Cut) 그리고 치환(Replace)의 3가지 연산이 가능하다. 추가 연산은 대상 'PART' 객체에 특정 'PART'를 하위부품으로 추가하는 것이며, 자료구조상으로는 두 'PART' 객체를 연결하는 'REL' instance가 1개 생성되는 결과를 가져온다. 삭제 연산은 대상 'PART'의 부품인 특정 'PART' instance를 부품관계에서 삭제하는 것이다. 자료구조 관점에서는 두 'PART'를 연결하는 'REL' instance가 삭제된 것을 뜻한다. 마지막 연산은 대상 부품을 Parameter로 정해진 'PART'로 치환 시키는 연산이다. 치환은 삭제와 추가 연산을 연속 적용한 것으로 볼 수 있으나 변경 이전과 변경 이후의 부품사이에 특정한 호환성(상호, 순 호환, 역 호환성 등) 이 존재할 경우를 표시하는 점이 다르다.

설계변경 객체는 관리정보, 제품구조 연관 그리고 제품변경이력의 3가지 구성 요소로 이루어져 있다. 관리정보란 설계변경번호, 종류, 시작 및 종료 시간 등의 관리적 필요에 의하여 만들어졌다. 제품구조 연관은 설계변경에 참여하는 제품구조를 설계변경 객체와 연결하여 추후 쉽게 설계변경 정보를 검색하거나 검토할 수 있도록 하는 부분이다. 마지막 설계변경 이력은 해당 설계변경이 관리하는 제품구조의 변경이력을 기록하는 부분이다. 이를 표현하는 설계변경 Schema에 속한 Entity의 구조가 Fig. 2에 나타나 있다.

설계변경 Schema의 주요 Entity는 'ENG_CHG' entity이며 이 Entity에 속한 다양한 Attribute를 이용하여 설계변경번호, 설계변경 이유 등을 관리한다. 또한 해당 단위 설계변경이 일어나는 Root 부품을 연결한 'root' attribute는 설계변경의 범위 및 단위를 설정한다. 'ENG_CHG' 객체는 재귀적인 포함관계를 가지는데 이는 'sub_ec' attribute로 구현되어 있다. 이는 몇 개의 설계변경 단위를 조합하여 전체 설계변경 단위를 중복 없이 표현하는 역할을 한다. 설계변경이력을 대표하는 것이 'EC_HISTORY' entity이며 이 Entity는 적용대상 객체인 'REL' 객체와 적용연산을

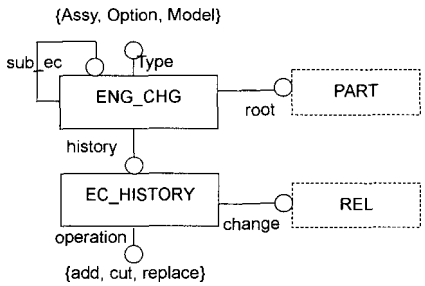


Fig. 2. 설계변경 Entity의 구조.

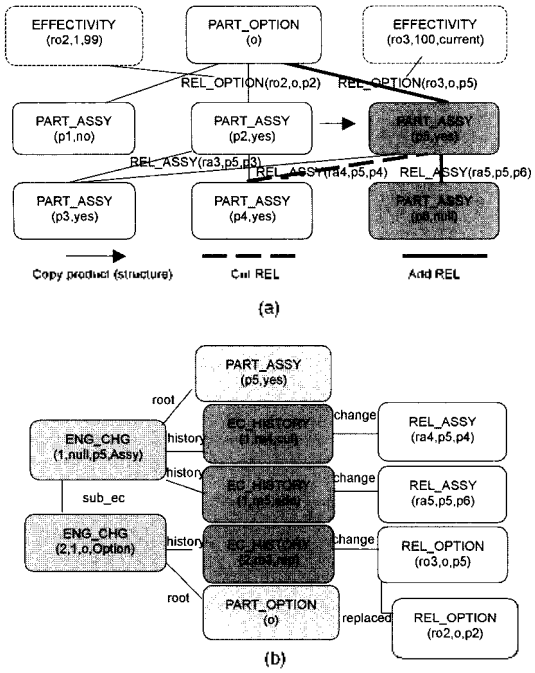


Fig. 3. 설계변경에 의한 구조변경(a) 및 기록 예(b).

연결하는 'change'와 'operation' attribute을 포함하고 있다.

Fig. 3(a)는 Fig. 1(b)의 Instance model에 대한 설계변경 적용 예를 나타내고 있으며 Fig. 3(b)는 Fig. 3(a)의 설계변경을 표현하는 설계변경 Schema의 Instance model을 보여주고 있다. 이를 표현하는데 Fig. 2의 Entity를 표현하는 다음의 Predicate와 Fig. 1(b)에서 사용한 Predicate를 사용하였다.

ENG_CHG(ID, sub_ec, root, type),
EC_HISTORY(ec_id, change, operation)

설계변경 단위(Unit of engineering changes)란 하나의 설계변경 객체에 어떤 범위의 설계변경 연산(추가, 삭제 그리고 치환)과 이로 인한 구조변경을 관리할 것인가를 뜻한다. 이 단위는 생산에 설계변경을 적용하는 기준으로 사용된다. 제안된 모델은 Assy type 설계변경 객체인 경우 단일 목적에 의하여 Option에 연결되는 한 'PART_ASSY' 객체(해당 설계변경 객체의 'root' attribute)하의 모든 변경을 설계변경단위로 처리한다. Option이나 Model Type 설계변경 인 경우에는 단일목적에 의하여 하나의 'PART_MODEL'이나 'PART_OPTION' 객체(해당 설계변경 객체의 'root' attribute)에 일어난 변경을 설계변경 단위로 처리한다.

설계변경 단위는 Root 부품을 중심으로 하는 제품 구조와 밀접하게 관계 되어 있는데 이는 설계변경 객체의 재 사용성을 높이도록 설계되었기 때문이다. n개의 'PART_OPTION' 객체와 연결된 1개의 'PART_ASSY' 객체에 대한 설계변경은 해당 'PART_ASSY'를 연결하고 있는 n개의 'PART_OPTION' 객체의 설계변경을 유발시킨다. 이때 설계변경 단위는 해당 'PART_ASSY' 객체에 대한 설계변경 객체 1개 그리고 이를 포함하는 모든 'PART_OPTION' 객체에 대한 설계변경 객체 n개로 구성되며, Option type 설계변경 객체는 Assy type 설계변경 객체를 'sub_ec' attribute로 연결하고 있다. 그러므로 각 Option type 설계변경 객체는 자신의 설계변경의 원인이 되는 'PART_ASSY' 객체의 설계변경 정보를 중복 없이 공유하게 된다.

3.3 제품자료관점과 설계변경 객체의 연관성

다수의 제품자료관점이 정의된 기본제품자료에 대한 설계변경은 연관된 제품자료관점에 대한 변경전달을 필요로 한다. 즉 변경된 기본제품자료와 일관된 관계를 가능하면 변화 발생시점에서 빠른 시간 안에 다시 유지 시켜야 한다. 이러한 변경전달은 기업 내부뿐만 아니라 기업간의 동시 제품개발 등에서는 전체 개발 과제에 걸친 전략적인 문제가 될 수 있다. 다음 장에서는 본 장에서 설명한 제품자료관점과 설계변경 객체를 이용하여 어떻게 기본제품자료에 대한 설계변경을 관련된 제품자료관점에 전달 시킬 수 있는지를 설명한다. 보다 쉬운 이해를 위하여 예를 함께 사용하여 설명한다.

4. 설계변경 이력과 전달

4.1 설계변경 전달 Procedure

기본제품자료에서 설계변경이 일어난 후 이를 관련된 제품자료관점으로 전달 시키는 과정은 다음과 같은 4단계의 작업이 필요하다.

STEP 1 - 기본제품자료의 설계변경 대상 객체 확인
STEP 2 - 변경전달 대상 제품자료관점 객체 검색
STEP 3 - 설계변경 연산에 따른 변경전달 계산
STEP 4 - 변경전달 적용 및 의사결정 지원

4.2 STEP 1 - 설계변경 대상 확인

이 단계는 전달 대상이 되는 Assy type 설계변경 객체와 이 객체를 'sub_ec' 관계로 연결하는 Option type 설계변경 객체('ENG_CHG' instance)의 설계변경이력 자료('EC_HISTORY' instance)를 검토하여

설계변경 전달 대상이 되는 'PART_ASSY' instance 객체를 찾는 과정이다. 대상 설계변경 단위의 예로 Fig. 3(b)에 표시된 Option type과 Assy type 설계변경객체('ENG_CHG(1,null,p5,Assy)'와 'ENG_CHG (2,1,o,Option)')와 이에 연결된 'EC_HISTORY' instance가 될 수 있다. 제안된 모델에서는 3가지 Case가 발생할 수 있다. 해당 Case를 표시하는데 'eng_chg'는 검토 대상인 설계변경 객체를 뜻하고 참가하는 Entity 표현은 제안된 제품자료관점과 설계변경 객체 Entity들(Fig. 1(a) 및 Fig. 2 참조)을 기반으로 하였으며 객체간의 Class composition 관계는 Dot notation으로 표현하였다.

CASE 1:

```
INPUT ENG_CHG eng_chg /* The type of 'eng_chg' object is the ENG_CHG entity */
FOR( ec_history is an eng_chg.history) /* ec_history is an element of eng_chg.history */

IF(ec_history.operation == 'cut' AND ec_history.change.child.usage == 'yes')
PART_ASSY part_assy = ec_history.change.child;

IF(ec_history.operation == 'replace' AND ec_history.change.replaced.child.usage == 'yes')
PART_ASSY part_assy = ec_history.change.replaced.child;
```

CASE 2:

```
INPUT ENG_CHG eng_chg
FOR (ec_history is an eng_chg.history)
IF (ec_history.operation == 'add')
PART_ASSY part_assy = ec_history.change.child;
```

CASE 3:

CASE 1,2 이외의 경우

Case 1에서는 설계변경 'eng_chg'에 참여한 'PART_ASSY' instance가 제품자료관점에 사용되었고 (usage='yes') 이 부품을 제품구조에서 삭제하거나 치환 시킨 경우이다. 이 경우 해당 부품이 제품자료관점에 쓰일 가능성이 있으며 STEP 2를 거쳐 해당 부품이 쓰이는 제품자료관점의 'PART_VIEW' instance를 찾아 해당 설계변경을 전달 시켜야 할 필요가 있다. 그러므로 다음 단계를 위하여 'part_assy' 객체에 현재 검사 중인 'PART_ASSY' instance를 저장한다. 예로써 Fig. 3(a)의 예제에서 'PART_ASSY(p4,yes)'가 제품구조에서 삭제될 경우에 Case 1이 적용될 수 있다.

Case 2에서는 설계변경 'eng_chg'에서 새로운 'PART'가 추가된 경우이다. 이 경우 추가된 부품은 아직 해당 관점에서 사용할지 여부가 결정되지 않았으므로 usage='null'이 된다. 이는 Step 4에서 해당 제품관점의 부품을 결정하는 Part planner의 의사결정에 의하여 처리된다. Fig. 3(a)의 예제에서 'PART_ASSY(p6,null)'의 추가는 새로운 부품을 제품구조에 추가하는 것이므로 Case 2에 해당한다.

이외의 모든 설계변경의 경우(Case 3)는 usage='no'인 'PART'가 삭제되거나 치환 된 경우이며 이는 제품자료관점과 관계가 없기 때문에 고려되지 않는다.

4.3 STEP 2 - 변경전달 대상 검색

이 단계는 위의 Case 1의 경우에 해당하는 기본제품자료('part_assy' instance)와 설계변경 객체('eng_chg' 및 'ec_history' instance)에서 자동으로 설계변경 전달대상인 'PART_VIEW' instance 객체를 찾는 과정이다. 대상 'PART_VIEW' instance 객체를 찾기 위하여서는 다음의 두 가지 단계를 거쳐야 한다.

STEP 2-1: 변경된 부품('part_assy' instance)이 'PART_VIEW' instance객체와 연결되어 있는지를 확인하고 연결된 객체를 찾는다.

STEP 2-2: 찾은 'PART_VIEW' instance 객체가 변경된 부품이 속한 'PART_OPTION' 객체와 연관된 객체인지 확인한다.

2-1 단계를 진행하기 위하여 Case 1을 만족하는 'PART_ASSY' 객체 'part_assy'에 대하여 Backward method[18,19]를 적용하면 해당 'part_assy' 객체를 지칭하고 있는 'PART_VIEW' instance 객체('part_view' 객체)가 Return된다. 이때 Backward method는 모든 Entity가 상속 받는 Generic class인 'Object' class에 속하는 Method로써 Message passing 대상인 객체를 'attr_name' attribute 값으로 가진('att_name' attribute를 통하여 Class composition 관계를 가진) 'att_domain'에 속한 객체의 Object identifier를 되돌려 주며 다음과 같이 표시한다.

```
Object Backward(att_domain, attribute_name)
```

Backward method를 사용하여 'PART_VIEW' instance를 찾는 과정은 다음과 같이 표시할 수 있다.

```
PART_VIEW part_view = part_assy.Backward(REL_VIEW, child).parent;
```

위의 식은 'part_assy' 객체와 'REL_VIEW'로 연

결된 'PART_VIEW' 객체('part_view' 객체)를 찾아 준다. 예로써 STEP 1에서 Case 1을 만족하는 Fig. 3의 'PART_ASSY(p2,yes)'에 Backward Method를 적용하면 적용된 객체를 'PART_VIEW(pv1,o,v)'와 'PART_VIEW(pv2,o,v)'로 연결하는 'REL_VIEW' instance 객체가 Return 되고(part_assy.Backward (REL_VIEW, child))이 객체의 'parent' attribute를 통하여 'PART_VIEW' instance객체를 지칭할 수 있다.

PART_ASSY(p2,yes).Backward(REL_VIEW,child) = {REL_VIEW(rv1,pv1,p2),REL_VIEW(rv2,pv2,p2)}

하지만 'PART_ASSY(p3,yes)'인 경우는 제품구조에서 삭제되고 usage='yes'이지만 상위부품의 Mapping에 의하여 제품자료관점과 관계를 가지고 있으므로 Backward method를 적용했을 경우 Return 되는 'REL_VIEW' 객체가 없다.

PART_ASSY(p3,yes).Backward(REL_VIEW,child) = { }

Fig. 4는 Fig. 3의 예제에 STEP 2-1이 적용된 결과를 보여 준다. STEP 1에서 선택된 'PART_ASSY (p2,yes)'로부터 연결된 PART_VIEW 객체를 찾기 위하여 'REL_VIEW' 객체를 이용하여 'PART_VIEW (pv1)'과 PART_VIEW(pv2)를 검색한다(Fig. 4의 화살표 참조).

2-2 단계는 2-1단계에서 계산한 'part_view' 객체에 대하여 다음의 조건을 확인하여 값이 참일 경우에만 STEP 3단계로 진행되게 된다. 다음 조건에서 'eng_chg'는 현재 검토중인 설계변경 객체, 'part_view'는 2-1 단계를 통과한 'PART_VIEW' 객체를 뜻한다.

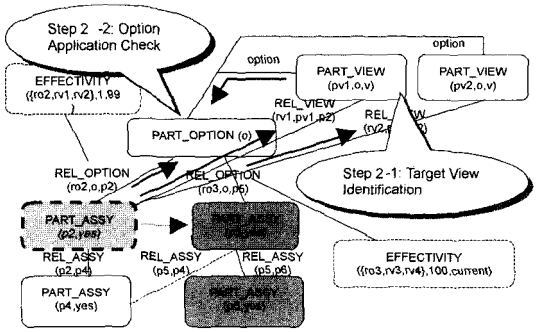


Fig. 4. 설계변경전달 2-1과 2-2단계.

```
IF ((eng_chg.type='Assy' AND eng_chg.Backward
  (ENG_CHG, sub_ec).root == part_view.option)
OR (eng_chg.type='Option' AND eng_chg.root ==
  part_view.option))
            GO ON;
ELSE
            BREAK;
```

위의 조건은 현재 검사중인 설계변경객체 'eng_chg'가 Assy type 설계변경 객체일 경우 이를 'sub_ec' attribute 값으로 가지는 Option type 설계변경 객체의 'root' attribute인 'PART_OPTION' instance와 설계변경에 참여한 'part_assy' instance와 연결된 'part_view' 객체의 'option' attribute값이 같은지를 검사하는 것이다. 설계변경 객체가 Option type이면 'root' attribute 값을 비교하면 된다. 이 조건의 의미는 검사중인 설계변경의 제품형상관점의 적용대상('OPTION' entity)과 현재 'PART_ASSY' instance를 공유하고 있는 'PART_VIEW' instance의 적용대상이 같은지를 검사한다.

Fig. 4에서 Fig. 3의 예제에 STEP 2-2가 적용된 내용을 보여 준다. STEP 2-1에 의하여 선택된 'PART_VIEW(pv1)'이 STEP 1에서 선택된 'PART_ASSY (p2)'와 'PART_OPTION(o)'를 공유하는 지를 확인하기 위하여 Fig. 4의 화살표와 같이 상호 Check하게 된다.

4.4 STEP 3 - 설계변경 전달 계산

STEP 2에서 해당 'PART'에 대응하는 'PART_VIEW' 객체가 있는 경우에 변경전달을 계산하게 된다. 계산과정은

STEP 3-1 변경전달로 야기된 Effectivity 변경 및 복사

STEP 3-2 변경전달을 위하여 'REL_VIEW' instance 생성

이며 각 단계의 계산을 위하여 다음과 같은 조건이 적용될 수 있다.

STEP 3-1

IF ((eng_chg.operation == 'cut' OR eng_chg.operation == 'replace') AND (eng_chg.type =='Option'))

(part_assy.Backward(REL_VIEW,child)).Backward (EFFECTIVITY, applied) = (part_assy.Backward (REL_OPTION, child)).Backward(EFFECTIVITY, applied);

STEP 3-2

```
IF (eng_chg.operation == 'replace')
  {
  REL_VIEW new_rel_view = NEW REL_VIEW
     (New_ID(), part_view, ec_history.change.child);
  IF (eng_chg.type=='Option')
  new_rel_view.Backward(EFFECTIVITY,  applied)
     =(ec_history.change.child.Backward(REL_
     OPTION,  child)).Backward(EFFECTIVITY,
     applied);

IF (eng_chg.type == 'Assy')
new_rel_view.Backward(EFFECTIVITY, applied) =
  (eng_chg.Backward(ENG_CHG,sub_ec).ec_history.
  change)
     .Backward(EFFECTIVITY,applied);
     }
```

STEP 3-1의 식은 'part_view'와 'part_assy'객체를 연결하는 'REL_VIEW' instance 객체에 설계변경의 삭제 혹은 치환 연산으로 인해 적용된 Effectivity 정보를 추가하는 식이다. 추가될 Effectivity는 'part_assy'와 'PART_OPTION' instance 객체를 연결하는 'REL_OPTION' instance상에 정의된 Effectivity를 복사하게 된다. 해당 Effectivity 객체를 찾기 위하여 Effectivity가 정의된 'REL_OPTION'을 'part_assy' 객체에 Backward method를 적용하여 찾고 ('part_assy.Backward(REL_OPTION, child)') 이에 적용된 Effectivity 값을 찾기 위해 다시 Backward method ('Backward(EFFECTIVITY, applied)')를 적용한다. 복사 대상이 되는 'REL_VIEW' 객체 상의 Effectivity를 표시하기 위하여 같은 방식으로 Backward method를 사용한다.

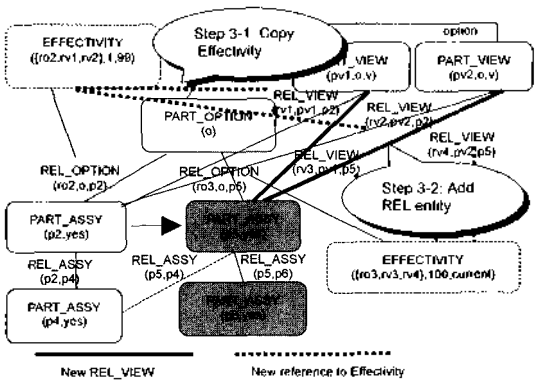


Fig. 5. 설계변경전달 3-1과 3-2 단계.

Fig. 5는 Fig. 3의 설계변경에 대하여 위의 계산에 의하여 변경 전달이 계산되어 적용되었을 경우의 제품구조를 보여 주고 있다. 이 그림에서 STEP 3-1의 예로 Fig. 5에서 'REL_OPTION(ro2,o,p2)'에 정의된 'EFFECTIVITY(ro2,1,99)'가 'REL_VIEW(rv1,pv1, p2)'과 'REL_VIEW(rv2,pv2,p2)'에 복사된 것을 볼 수 있다.

STEP 3-2를 위한 식은 치환 연산의 경우 기존의 부품을 치환하는 새로운 부품(ec_history.change.child)과 'PART_VIEW'를 연결하는 'REL_VIEW' instance를 생성하고(NEW REL_VIEW(New_ID(), part_view, ec_history.change.child)), 이에 적용될 새로운 Effectivity 객체를 기본제품정보로부터 복사하는 역할을 한다. 예로 Fig. 5에서 'PART_ASSY(p2,yes)'를 치환한 'PART_ASSY(p5,yes)' 객체에 대하여 'PART_VIEW (pv1,o,v)'와 'PART_VIEW(pv2,o,v)'를 연결하는 'REL_VIEW(rv3,pv1,p5)'과 'REL_VIEW(rv4,pv2,p5)'를 생성하고, 각 객체에 'PART(p5,yes)'와 'PART_OPTION(o)'를 연결하는 'REL_OPTION(ro3,o,p5)'에 정의된 'EFFECTIVITY(ro3,100,current)'를 복사한다. Fig. 5는 결과적으로 지금까지 설명된 STEP 1에서 STEP 3까지의 전 과정을 거쳐 설계변경 전달이 완료된 제품구조를 보여 주고 있다.

4.5 STEP 4 - 전달적용에 대한 의사결정

STEP 1과 STEP 3에서 생성된 자료를 적용되기 전에 Part planner의 의사결정을 거쳐야 한다. Plat planner에게는 한 단위 설계변경 당 다음과 같은 내용의 의사결정 요청이 전달된다.

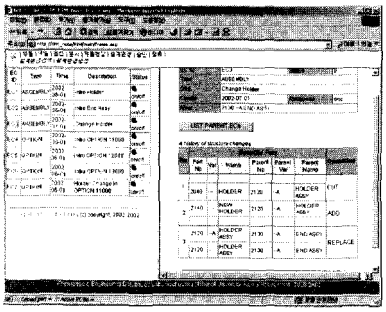
1) STEP 1, CASE 2에서 추가된 부품
2) STEP 3의 치환 연산에서 추가된 부품 확인

결과적으로 Part planner는 제품설계 부서에서의 설계변경이 일어나면 이에 대한 설계변경 통지를 받고 이 설계변경 통지를 본 논문에서 제안한 Procedure를 통하여 계산한 후 마지막 단계에서 나온 결과에 대한 최종 의사결정을 하고 이를 실행하면 해당관점으로의 설계변경 전달을 완료할 수 있다. 특히 제안된 단계들은 전산시스템을 통하여 대부분 구현가능 하도록 정의되어 있으므로 많은 부분이 데이터베이스 관리시스템과 응용프로그램을 통하여 구현할 수 있다. 다음 장에서는 앞에서 제안한 제품구조 및 설계변경 전달 Procedure를 구현한 정보시스템과 그 이용방법을 소개한다.

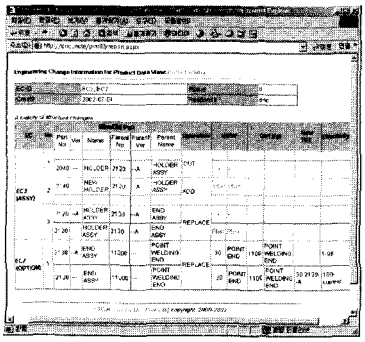
5. 구현 예

현재 제안된 모델은 관계형 데이터 모델로 변화시켜 상용 관계형 데이터베이스 관리시스템을 이용하여 구현되어 있으며, 이를 기반으로 Web 기반의 제품정보관리 시스템 시제품을 개발하고 있다. 이 시스템은 일반적인 제품정보관리 시스템의 기능인 제품구조, 설계문서, 제품형상 그리고 설계변경 관리 기능을 가지고 있다. Fig. 6(a)는 제품정보관리 시스템에서 제공하는 설계변경객체 보기 화면을 보여 주고 있다. 이 화면은 설계변경공지(Engineering Change Notification) 용으로 사용되는 화면으로 설계변경 객체에서 저장되고 있는 설계변경 관리, 설계변경 관련 제품구조, 설계변경 이력(Fig. 6(a)의 우측화면 하단)에 관한 정보를 보여 주고 있다. 이는 설계자들이 설계변경 활동이 끝나고 그 결과를 생산, 조달 등의 관련 부서에 공지하기 위하여 사용한다.

Fig. 6(b)는 Fig. 6(a)의 설계변경에 대하여 특정 제품자료관점에 추가 될 제품구조와 Effectivity를 보여주는 보고서이다. 예로 특정 제품자료관점은 설계에서 작성한 제품구조상에 정의된 Part Catalog를 위한 제품 구조일 수 있다. 이 보고서는 설계에서 실행한 설계변경(Fig. 6(a))에 대하여 제안한 설계변경 및 제품자료관점 Schema와 변경전달 Procedure에 의하여 시스템에서 자동 생성된다. 해당 Parts catalog를 편집하는 Part planner에게 통지된 이 보고서는 삭제되거나 치환된 부품에 대한 자동 제품구조생성 부분과 새로이 추가되어 Part planning이 필요한 부품 목록을 보여 주고 있다(Fig. 6(b)의 하단). Part planner는 이 보고서를 토대로 시스템에서 제공하는 기능을 적용할지 여부와 새로이 추가된 부품을 Parts catalog에 적용여부에 대한 의사결정을 내리면 된다.



(a)



(b)

Fig. 6. 설계변경 객체 (a) 및 설계변경 전달(b).

6. 결 론

기존의 설계변경에 관한 연구는 제품이력과 설계변경 관리에 관한 제품구조 관점의 연구가 부족하며, 기존의 제품자료관점에 관한 연구는 관련된 모델들의 통합에 관한 문제에 치중하여 어떻게 기본제품자료의 변화를 제품자료관점에 전달 시켜 제품정보의 일관성을 유지할 것인가에 관한 연구가 미흡하다. 그러므로 본 연구에서는 두 가지 연구방향의 단점을 보안하여 설계변경에서 관리되는 제품구조 중심 설계이력을 기반으로 변경사항을 다양한 응용 제품자료관점에 효과적으로 전달할 수 있는 체계를 제시하였다.

연구의 결과로 첫째, Filtering과 Mapping 기제를 사용하여 자료의 중복 없이 효과적으로 기본제품자료와 제품자료관점을 통합하였으며, 둘째, 이를 바탕으로 기본제품자료에서 일어난 변경을 설계변경 이력을 이용하여 각 제품자료관점에 전달 시키는 Procedure를 제안하였다. 제안된 Procedure는 통합된 제품자료 모델과 Backward propagation을 이용하여 제품관점의 변경 대상을 확인하고 변경전달을 효과적으로 계산하는 Rule과 연산을 제공한다. 마지막으로 제안된 제품정보모델과 Procedure는 전산시스템에 적용할 수 있도록 정의되었으며 이를 바탕으로 설계변경 전달을 자동화하고 Part planner 의사결정을 지원할 수 있는 설계변경 전달시스템을 구현하였다.

제안된 설계변경 전달은 본 연구에서 제안한 제품구조 상에서만 가능하도록 작성되어 있으므로 일반적인 제품구조에 적용이 불가능한 문제점이 있다. 설계변경 전달을 보다 보편적 방법으로 만들기 위하여 일반적인 제품구조에 대하여도 적용 가능하도록 확장할 필요가 있다. 이 확장의 한 방법으로 ISO STEP 표준[10]으로 표현된 제품구조에 설계변경 전달을 적용할 수 있도록 설계변경 전달을 ISO STEP 표준으로 표현하는 방법에 관한 추후 연구가 필요하다.

참고문헌

1. Huang, G. Q., Yee, W. Y. and Mak, K. L., "Development of a Web-based System for Engineering Change Management," *Robotics and Computer-Integrated Manufacturing*, Vol. 17, No. 3, pp. 255-267, 2001.
2. Do, N. C., Kim, H., Kim, H. S., Lee, J. Y. and Lee, J. H., "Web-based Product Data Management and Parts Catalog Publication System for Collaborative Product Development," *Proc. of IIWAS2001*, pp. 379-387, 2001.
3. Shah, J. J. and *et al.*, "Database Infrastructure for Supporting Engineering Design Histories," *Computer-aided Design*, Vol. 28, No. 5, pp. 347-360, 1996.
4. Agilesoft, "Why Document Management Systems Don't Work for the ECO Process," *White Paper*, Agilesoft, 1999.
5. Brown, D. C., "Using Design History Systems for Technology Transfer," *Proc. of MIT-JSME Workshop on Cooperative Product Development*, pp. 545-559, 1989.
6. Hardwick, M., Spooner, D., Downie, B., Ferris, M., Jiang, Z. and DeWeese, T., "A STEP Entity Control System for Concurrent Engineering," *Proc. of The First International Conference on Concurrent Engineering: Research and Applications*, 1994.
7. Hardwick, M., Downie, B., Kutcher, M. and Spooner, D., "Concurrent Engineering with Delta Files," *IEEE Computer Graphics and Applications*, Vol. 15., No. 1, pp. 62-68, 1995.
8. Baya, V., *Information Handling Behavior of Designers During Conceptual Design: Three Experiments*, Doctoral thesis, Department of Mechanical Engineering, Stanford University, 1996.
9. Gate, R., "Production Implementation of STEP for the C-17 Program," *Proc. of The International Conference on Concurrent Enterprise*, 1996.
10. ISO 10303-1, *Part 1: Overview and Fundamental Principle*, National Institute of Standards and Technology, 1992.
11. Enovia Inc., *Enovia Virtual Product Model User's Guide* Version 1, Enovia, 1998.
12. PDM Implementer Forum, *Usage Guide for the STEP PDM Schema V1.2 Release 4.3*, 2002.
13. Peng, T.-K. and Trappey, A. J. C., "A Step Toward Step-compatible Engineering Data Management: the Data Models of Product Structure and Engineering Changes," *Robotics and Computer-Integrated Manufacturing*, Vol. 14, No. 2, pp. 89-109, 1998.
14. Browe, D. H. Inc., "Fundamentals of Shred Product Structure," *PD Report*, 2001.
15. Katz, R. H., "Toward a Unified Framework for Version Modeling in Engineering Databases," *ACM Computing Surveys*, Vol. 22, No. 4, pp. 375-408, 1990.
16. Hamer, P. V. D. and Lepoeter, K., "Managing Design Data: Five Dimensions of CAD Frameworks, Configuration Management, and Product Data Management," *Proc. of the IEEE*, Vol. 84, No. 1, pp. 42-56, 1996.
17. Satory, L. G., *Manufacturing Information Systems*, Addison-Wesley, 1988.
18. Do, N. C., *Backward Propagation of User-Defined Integrity Constraints in Active Object-Oriented Databases*, Doctoral Thesis, Department of Industrial Engineering, Pohang University of Science and Technology, 1996.
19. Do, N. C., Bae, S. M. and Choi, I. J., "Constraint Maintenance in Engineering Design System: An Active Object-Oriented Approach," *Computers Ind. Engng*, Vol. 33, Nos. 3-4, pp. 643-647, 1997.

도 남 철

1991년 포항공과대학교 산업공학과 학사
1993년 포항공과대학교 산업공학과 석사
1996년 포항공과대학교 산업공학과 박사
1996년 삼성중공업 중앙연구소 선임연구원
1998년 복보건설기계 코리아 CAD/PDM 팀 과장
2001년 한국전자통신연구원 컴퓨터소프트웨어연구소 동사공학팀 선임연구원
2002년~현재 경상대학교 산업시스템공학부 조교수
관심분야: 제품자료모델, 제품성보제약조건, 제품자료통합, 제품개발지식표현