복합제품의 진화환경에서 제품추적 시스템

진 민 김종명[0]
경남대학교 컴퓨터공학과
mrpoon@hawk.com.kyungnam.ac.kr

Product Instance Trace System in Composite Product Evolution

Min Jin, Jong-Myoung Kim
Dept. of Computer Science & Engineering, Kyungnam University

요 약

본 논문은 복합제품의 진화환경에서 버전과 콘피규레이션 관리기법을 사용하여 제품의 진화를 관리하면서 복합제품의 구조정보뿐만 아니라 진화하는 버전과 콘피규레이션에서 다양하게 생성된 개별제품에 대한 접근을 지원하는 복합제품추적시스템을 설계하고 구현하였다.

1. 서 론

기술의 발전속도가 빠르고 시장의 욕구가 신속하게 변함에 따라 다양한 복합제품들이 제조 및 판매되고 있으며 제품의 생성주기는 점점 짧아지고 있다. 복합제품들은 단순히 몇 개의 부품으로 구성되어 지는 것이 아니라 복잡하고 다양한 부품들로 구성되어지고, 다시 부품들은 계속적으로 다른 부품들과 조합하여 구성되어진다. 이러한 부품들은 성능의 개선, 사용자의 새로운 요구, 기술 등의 이유로 지속적으로 진화한다. 부품의 진화는 복합제품을 구성하는 다른 부품에게도 영향을 끼쳐 관련 부품의 진화를 촉진시킬 수 있다. 따라서 지속적으로 진화하는 부품을 효율적으로 관리할 필요가 있다. 본 논문에서는 제품진화환경에서 버전과 콘피규레이션을 관리하고 각 버전과 콘피규레이션에서 생성된 제품에 대한 구조 및 개별제품정보를 추적할 수 있는 시스템을 설계하고 구현한다.

2. 버전과 콘피규레이션

2.1 버전의 생성

제품은 클래스로 표현되고 복합제품은 복합객체로 표현할 수 있으며, 이들의 변경은 스키마 변경으로 보고 제품의 진화를 해당 클래스의 버전으로 표현한다. 그러므로 버전은 클래스로부터 유도되거나 다른 버전으로부터 유도되어 만들어진다. 그런데 제품에 대한 버전 유도는 객체지향개념에서의 상위클래스-하위클래스의 유도와는 그 성격이 다르다. 상위클래스-하위클래스의 유도는 하위클래스에서 다시 정의되는 속성을 제외한 모든 속성을 상속받지만, 제품의 버전 유도는 시간의 흐름에 따른 변경의 기록이므로 부모버전의 속성에서는 선택적으로 자식에게 상속된다. 제품의 진화는 한 부품의 변경을 요구할 수 있고, 바뀐 부품은 다시 연쇄적으로 다른 부품의 변경을 요구하게 된다. 이를 버전 전파라 부르며, 버전 전파가 과다해지면 버전 범람이 발생하게 되어 수많은 버전이 발생해 관리하기 어려워지는 문제점이 발생한다. 이를 해결하기 위하여, 부품의 속성 중 다른 부품의 변경을 요구하는 중요 속성을 인터페이스 속성으로 분류하고 인터페이스 속성의 변경만을 복합객체에 대하여 전파하도록 한다[1,2,3,4].

2.2 콘피규레이션의 생성

클래스와 버전, 콘피규레이션과 인스턴스와의 관계를 나타내면 <그림 1>과 같다. 버전은 직접적인 인스턴스를 가질 수 없다는 점에서 추상적 개념이며 콘피규레이션은 직접 개별 제품을 나타내는 인스턴스를 가질 수 있다는 점에서 구체적 개념이다. 버전에는 IS-A계층 구조가 존재하지만 콘피규레이션에는 IS-A계층 구조가 없고, 콘피규레이션이 복합제품구성품의 구체적인 명세이기 때문에 선택 부품의 의미를 가진 복합 참조가 없다. 콘피규레이션은 구성 부품의 콘피규레이션이 먼저 생성되어야만 생성될 수 있으며 콘피규레이션 공간에서 전파는 허용되지 않는다[1,5].

콘피규레이션은 클래스나 버전 또는 기존 콘피규레이션으로부터 생성되며 콘피규레이션은 주어진 복합제품의 완전한 정보를 갖는다.

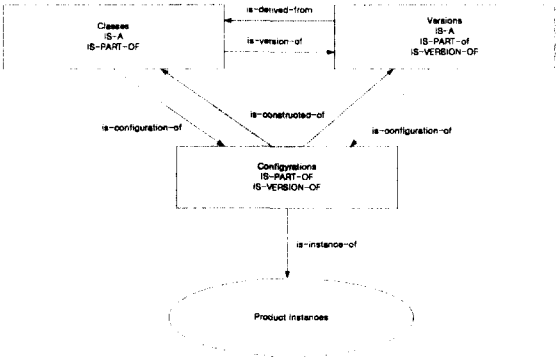

그림 1. 버전, 콘피규레이션, 인스턴스의 관계

※ 이 연구는 1999년도 경남대학교 학술연구조성비 지원에 의하여 수행되었음

2.3 확장된 SQL

기존의 SQL에 VERSION 조건절과 CONFIGURATION 조건절을 추가하여 질의를 지원하고 WHERE 조건절에 의미정보를 명시함으로써 속성의 의미정보를 이용한 질의를 <그림 2>와 같은 형식으로 지원한다.

```
SELECT attribute[, attribute]*
FROM class-name
[VERSION {all | version all | version[, version]*}]
[CONFIGURATION {all | Configuration all | Configuration [, Configuration ]*}
WHERE attribute[semantic-information] op value | all

attribute ∈ attributes of the corresponding class
version ∈ version numbers of the corresponding class
Configuration ∈ Configuration numbers of the corresponding class
configuration-op : comparison operator of Configuration
op : comparison operator
value ∈ literal values or attribute specification
```

그림 2. 확장 SQL문의 정의

- 모든 버전의 모든 콘피규레이션에 대한 질의
  select attribute[, attribute]*
  from class-name
  version all
  configuration all
  where attribute(semantic-information) op value
- 특정 버전의 모든 콘피규레이션에 관한 질의
  select attribute[, attribute]*
  from class-name
  version version[, version]*
  configuration all
  where attribute(semantic-information) op value

3. 복합제품추적 시스템

복합제품 추적시스템은 진화하는 구성부품의 결합가능성을 확인하고 제품의 콘피규레이션을 생성하며, 버전의 생성 및 관리 등을 위한 그래픽 사용자 인터페이스를 제공한다.

<그림 3>의 Class의 메뉴를 살펴보면 클래스의 정의하는 Class Definition과 클래스의 정보를 보는 Class Information, 클래스의 계층구조를 볼 수 있는 Class Composition Hierarchy를 제공한다.

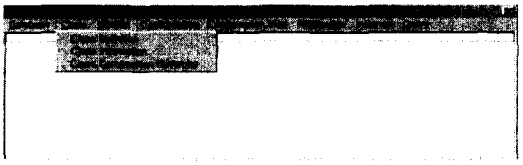

그림 3. 클래스 메뉴

<그림 4>의 Version메뉴를 살펴보면 버전의 유도할 수 있는 Version Dervation과, 버전의 유도 계층 관계를 나타내는 Version Derivation Hierarchy, 버전의 계층구조를 나타내는 Version Composition Hierarchy, 버전의 관리를 위한 Version Manager는 버전의 정보와 버전의 의미정보를 관리한다.

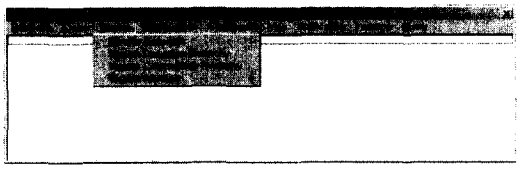

그림 4. 버전의 메뉴

<그림 5>의 Configuration 메뉴를 살펴보면 콘피규레이션을 생성할수 있는 Configuration Construction과 이미 만들어진 콘피규레이션으로부터 새로운 콘피규레이션을 유도할수 있는 Configuration Derivation, 그리고 콘피규레이션의 유도 과정을 나타내는 Configuration Derivation Hierarchy, 콘피규레이션의 구성계층 구조를 나타내는 Configuration Composition Hierarchy, 콘피규레이션의 여러 정보를 표현하는 Configuration Information을 가진다.

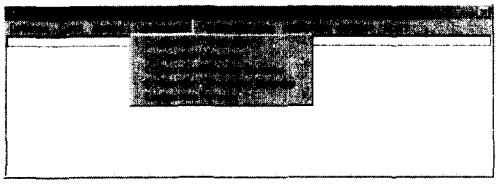

그림 5. 콘피규레이션의 메뉴

Configuration Map 메뉴는 콘피규레이션 맵을 생성할수 있는 Configuration Map Creation, 콘피규레이션의 수정을 위한 Configuration Map Adjustment, 이미 생성된 콘피규레이션 맵을 볼 수 있는 Configuration Map Browsing을 가진며, Notification 메뉴는 통지에 관한 정보를 메뉴들이다. 버전이 인터페이스 변경에 의해 생성되었을 때 상위 버전으로 전파하기 위해 발생하는 버전 전파 통지 기능은 Version Propagation Notification, 콘피규레이션 맵 생성 통지기능은 Configuration Map Creation Notification, 콘피규레이션 맵 수정 통지기능은 Configuration Map Adjustment Notification, 설계자에 의해 이미 인지된 통지에 관한 정보를 보여주는 Notice Browsing을 가진다.

Instance메뉴는 인스턴스를 탐색하고 보여주는 Instance Explorer와 인스턴스의 계층구조를 나타내는 Instance Hierarchy를 가지고, 마지막으로 Query메뉴는 위에서 설명한 확장 SQL문을 보다 쉽게 사용하기 위한 인터페이스이다.

3.1 클래스 선언

<그림 6>은 클래스 선언을 위한 화면이며 클래스를 선언하기 위해 필요한 사항은 아래와 같다.

첫째, Super 클래스의 지정

둘째, 속성의 추가(Add Attribute)와 선택된 속성의 삭제 (Delete Attribute)
셋째, 특정 속성의 인터페이스 속성 부여와 해제 및 복합속성 부여와 해제
넷째, Assembling 속성 추가와 Correspondence 속성 추가 등의 기능들을 가진다. 새로 만들어진 클래스는 버전#1을 동시에 생성한다.
다섯째, 슈퍼클래스 검색 및 상속

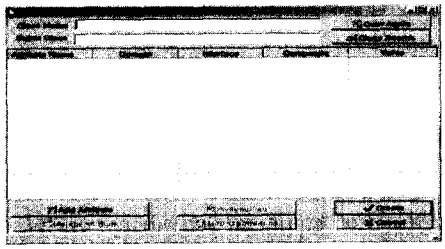

그림 6. Class 정의 화면

3.2 콘피규레이션 유도

생성된 콘피규레이션에서 새로운 콘피규레이션을 유도할 수 있으며 <그림 7>은 콘피규레이션 유도 화면이다. 콘피규레이션의 유도과정은 버전이 유도되는 과정에서 조건적으로 발생되는 버전 전파를 콘피규레이션 유도과정에서는 발생되지 않는다. 콘피규레이션 유도과정에서 콘피규레이션의 유효성 검사를 하지 않도록 부속부품의 버전은 선택할 수 없고, 이미 유효하다고 판정된 버전의 콘피규레이션만을 선택할 수 있도록 제한을 한다.

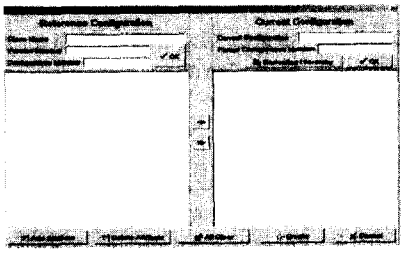

그림 7. 콘피규레이션 유도화면

3.3 통지

인터페이스 변경에 의해 새로운 버전이 유도되었다는 것을 사용자에게 알리기 위해 버전 전파 통지를 제공한다. 사용자는 통지를 인지하고 전파에 의해 유도되는 새로운 버전을 만들 수 있다. 버전 전파 통지 목록 중에 한 항목을 인지하면 인지한 시간을 저장한다.

3.5 질의

<그림 8>의 질의 인터페이스는 위에서 설명한 SQL의 확장 질의가 가능하도록 하였다.

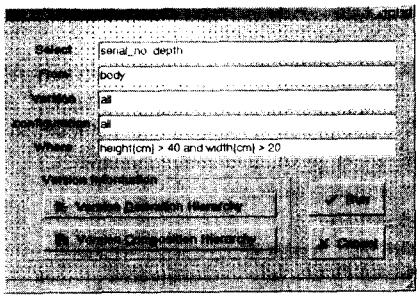

그림 8. 질의 인터페이스

Version Derivation Hierarchy는 버전의 계층구조를 볼 수 있도록 트리 구조를 나타내며 Version Composition Hierarchy는 복합제품이 어떻게 구성되어 있는지를 나타낸다.

4. 결론

오늘날 생산되는 대부분의 제품은 복잡하고 진화하는 특징을 가진다. 이러한 진화하는 제품을 효율적으로 관리하기 위해서는 제품의 버전관리와 콘피규레이션 관리가 필요하다. 본 연구에서는 버전과 콘피규레이션 관리 기법을 이용하여 복합제품의 진화를 관리하고 진화과정에서 각 버전과 콘피규레이션에서 생성된 개별 제품의 구조 및 개별제품정보를 추적하는 제품추적 시스템을 설계하고 구현하였다.

본 시스템은 실제 응용에서 구체적인 제품에 대한 적용을 통하여 그 유용성을 검정하면서 추가적으로 보완되어야 할 것이며 유효한 콘피규레이션 생성을 위한 효과적인 방법에 대한 연구도 필요하다.

참고문헌

[1] Min Jin and Hye-Sun Ryu, "Configuration Management System for Supporting Product Evolution", International Conference on Mechatronic Technology, 1999

[2] S. R. Monk and I. Sommerville, "A Model for Versioning of Classes in Object-Oriented Databases", In *Proceedings of the 10th British National Conference on Databases*, pp. 42-58, July 1992

[3] S. R. Monk and I. Sommerville, "Schema Evolution in OODBs Using Class Versioning", *SIGMOD Record*, Vol. 22, No. 3, September 1993

[4] W. Kim and H. T. Chou, "Versions of Schema for Object-Oriented Databases", In *Proceedings of the 14th VLDB Conference*, pp. 148-159, 1988

[5] G. Hedin, L. Ohlsson, J. McKenna,"Product Configuration Using Object-Oriented Grammars ", In Proceedings of the 8th International Symposium on System Configuration Management (SCM-8), 1998