초기 컴포넌트 설계 단계에서 컴포넌트 정제 기법

이종국 백종현
대우정보시스템
jklee690@disc.co.kr O baegjh@disc.co.kr

A Component Refinement Technique in Initial Component Design Stage

JongKook Lee JongHyun Baek
Daewoo Information Systems Co., Ltd

요 약

컴포넌트 기반 소프트웨어 공학은 재사용 가능한 컴포넌트를 조립하여 시스템을 개발하는 방법이다. 컴포넌트가 시스템 개발에서 효과를 발휘하기 위해서는 컴포넌트를 설계, 구현하기 위한 다양한 기법들이 제시되어야 한다. 컴포넌트 설계 기법은 아키텍처 설계, 컴포넌트 식별, 컴포넌트 정제, 컴포넌트 설계 상세화로 나눌 수 있다. 이 중에서 컴포넌트 정제는 컴포넌트의 특성을 가장 많이 반영하는 기법이며 어떤 기법을 사용하는가에 따라 컴포넌트 기반 시스템의 품질이 달라진다. 본 논문에서는 개발 생산성에 중점을 두고 컴포넌트를 정제하는 기법을 제시한다. 특별히 컴포넌트 사이의 관계를 최적화하는 기법을 제시한다.

1. 서론

컴포넌트 기반 소프트웨어 공학(CBSE)은 재사용 가능한 컴포넌트를 조립하여 시스템을 개발하는 방법이다. 전통적인 개발 방법에서는 개발자가 필요한 코드를 구현하여 시스템을 구축한다. 그러나 CBSE에서는 개발자는 필요한 컴포넌트를 수집하고 조립하여 시스템을 구축한다. 따라서 CBSE는 소프트웨어 개발 비용과 기간을 단축할 수 있는 새로운 패러다임으로 평가되고 있다[1].

컴포넌트가 시스템 개발에서 효과를 발휘하기 위해서는 컴포넌트를 설계, 구현하기 위한 다양한 기법들이 제시되어야 한다. 컴포넌트 구현 기법은 EJB, .NET 등 특정 컴포넌트 실행 플랫폼의 구현 기법을 이용하면 된다.

컴포넌트 설계 기법은 크게 네 가지로 나눌 수 있다.

첫째는 컴포넌트의 크기와 유형을 결정하기 위한 아키텍처 설계 기법이다.

둘째는 컴포넌트 식별 기법이다. 컴포넌트 식별에는 컴포넌트뿐 아니라 인터페이스를 도출하는 기법도 포함된다.

셋째는 식별된 컴포넌트를 성능과 재사용성 등을 고려하여 정제하는 단계이다.

넷째는 정제된 컴포넌트를 EJB, .NET 등의 플랫폼에 맞추어 변환하는 단계이다. 이 단계에서는 데이터베이스와의 관련성도 고려해야 한다.

아키텍처 설계에 대해서는 이미 많은 기법들이 제시되어 있다. 컴포넌트 식별 기법은 현재 많은 연구가 진행 중이다. 컴포넌트 상세 설계는 특정 플랫폼에 따라 기법이 달라지며 많은 연구결과가 나와 있다. 그러나 식별된 컴포넌트를 정제하여 최적의 컴포넌트를 도출하고 컴포넌트 사이의 관계를 정의하는 기법에 대한 연구는 드물다.

컴포넌트 정제 기법은 컴포넌트의 특성을 가장 많이 반영해야 하는 기법이며 컴포넌트 정제 기법에서 어떤 품질 요소를 반영하는지에 따라 시스템 개발에서 컴포넌트가 어떤 효과를 발휘할 것인지가 결정된다.

따라서 컴포넌트 정제 기법은 컴포넌트의 품질 요소에 따라 여러 기법들이 제시될 수 있다. 본 논문에서는 개발 생산성에 중점을 두고 컴포넌트를 정제하는 기법을 제시한다. 본 논문에서는 특별히 컴포넌트 사이의 관계를 최적화 하는데 중점을 두고 기법을 제시한다.

2. 컴포넌트 설계 원칙

이장에서는 컴포넌트 설계 원칙을 제시한다.

첫째, 컴포넌트를 식별 및 설계하기 전에 소프트웨어 아키텍처가 결정되어야 한다. 아키텍처는 컴포넌트의 유형을 결정하기 때문에 반드시 필요하다. 소프트웨어 아키텍처의 구성 요소 중에서도 architecture style이 컴포넌트의 유형을 결정한다. 만일 layered architecture style을 사용하여 시스템을 MVC로 분할한다면 컴포넌트는 UI, 컨트롤, 엔티티 타입으로 분리될 것이다.

둘째, 컴포넌트는 고정된 크기를 갖지 않는다. 컴포넌트는 한 시스템에서도 다양한 크기를 가질 수 있다. 컴포넌트는 컴포넌트 사이의 의존성이 단순할 정도로 커야 한다. 반대로 컴포넌트가 너무 크면 컴포넌트 개발이 어렵고 오류를 제거하기 어렵다.

셋째, 최적의 컴포넌트는 컴포넌트의 기능성을 표현하는 오퍼레이션과 관련 객체 사이에 cohesion이 있어야 하며 의존성이 단순해야 한다.

본 논문에서는 이상의 원칙에 따라 컴포넌트를 설계한다.

3. 컴포넌트 설계 절차

컴포넌트 설계 예제로 대학 학사 행정 시스템을 사용한다. 또한 학사 행정 시스템 아키텍처는 다음과 같이 결정되었다.

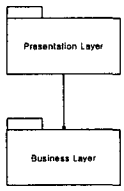

그림 1. 학사 행정 시스템 아키텍처

Business Layer는 컨트롤 클래스와 엔티티 클래스를 포함한다. 따라서 컴포넌트에도 컨트롤 클래스와 엔티티 클래스가 포함된다. UML Component의 지침은 컨트롤 타입의 컴포넌트와 엔티티 타입의 컴포넌트를 분리하는 것이지만 학사 행정 시스템에서는 한 컴포넌트가 두 가지 클래스를 포함하는 것으로 정의한다.

3.1 컴포넌트와 의존 관계 표현

학사 행정 시스템의 식별된 컴포넌트와 컴포넌트 사이의 관계는 다음과 같다.

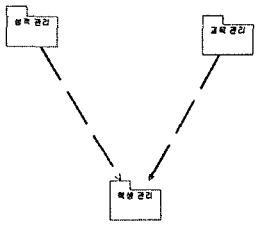

그림 2. 학사 행정 시스템의 식별된 컴포넌트들

3.2 critical use case를 컴포넌트에 할당

식별된 컴포넌트에 대하여 컴포넌트가 어떤 역할을 담당하는지 결정해야 한다. Use Case를 사용하여 컴포넌트의 역할을 결정한다. 그러나 너무 많은 Use Case를 사용하면 역할 결정에 혼란이 오기 때문에 조회용 Use Case를 제외하고 Critical Use Case를 선정한다. 선정된 Critical Use Case를 각 컴포넌트에 할당한다.

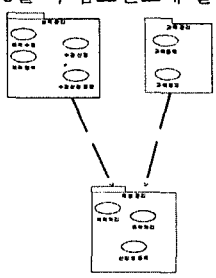

그림 3. 컴포넌트에 Use Case 할당

3.9 컴포넌트의 분할 및 통합

응집성을 고려하여 컴포넌트를 분할 및 통합한다. 컴포넌

3.4 use case 당 하나의 컨트롤 클래스 할당

Use Case Description을 사용하여 Use Case의 기능을 수행하는 컨트롤 클래스를 결정하고 이 컨트롤클래스의 기능을 오퍼레이션으로 표현한다. 컨트롤 클래스 사이의 호출 관계도 표현한다.

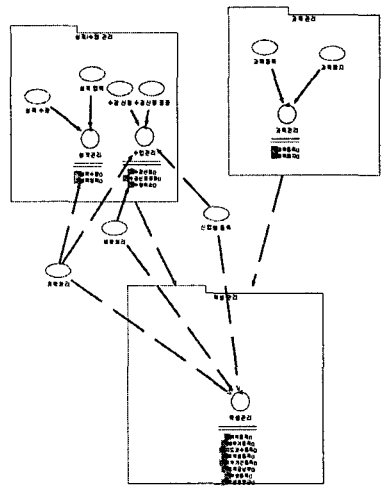

그림 5. 컴포넌트와 컨트롤 클래스의 관계

컴포넌트에 컨트롤 클래스를 할당한 후에는 Use Case는 제거한다.

3.6 컨트롤 클래스에서 사용하는 엔티티 클래스 도출, 컴포넌트에 할당

컨트롤 클래스에서 사용하는 엔티티 클래스를 도출하고 엔티티 클래스 사이의 관계를 설정한다.

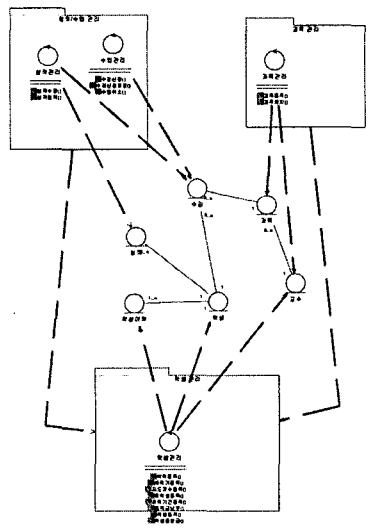

그림 6. 컴포넌트, 컨트롤 클래스, 엔티티 클래스의 관계

트 사이의 관계를 재설정할 경우 컨트롤 클래스가 새로 도출될 수도 있다. 이 단계에서 가장 중요한 결정 사항은 엔티티 클래스들 사이의 association을 컴포넌트 내부에서 캡슐화할 것인지 컴포넌트 사이의 호출 관계로 변형할 것인지이다. 학사 행정 시스템에서 학생 정보와 수강, 과목 정보를 연관하여 함께 보는 경우는 빈번하게 발생하지 않는다. 그러나 수강한 과목과 성적은 빈번한 join이 발생하기 때문에 같은 컴포넌트에 넣는 것이 좋다. 또한 과목 정보와 수강, 성적을 같이 보는 경우도 빈번하기 때문에 성적, 과목 컴포넌트를 통합하는 것이 좋다.

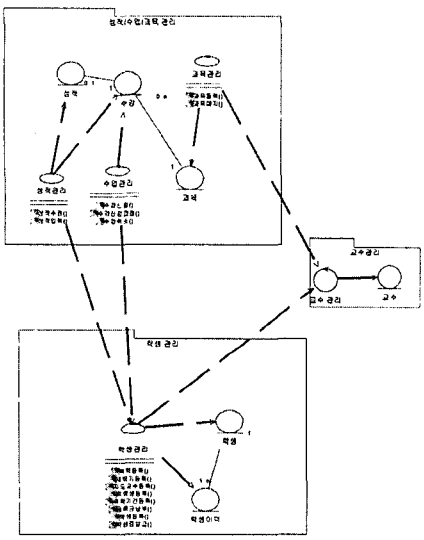

그림 4. 정제된 학사 행정 시스템 컴포넌트들

만일 특정 과목의 수업을 듣는 사람의 성적과 담당 교수를 모두 알고 싶은 경우에는 성적, 수업, 과목 컴포넌트에서 직접 교수 정보를 가져오는 것을 허용한다. 만일 특정 교수가 담당하는 과목을 알고 싶은 경우에는 주 조회가 과목이기 때문에 이 기능은 성적, 수업,과목 관리 컴포넌트에서 담당하도록 한다. 만일 모든 교수가 담당하는 과목과 성적을 알고 싶을 경우에는 두 가지 방법이 있다.
첫째, 설계된 컴포넌트를 모두 무시하고 별도의 조회 프로그램을 작성한다.
두 번째, 교수의 이름과 ID를 교수 컴포넌트에서 얻어온 후 성적, 과목 컴포넌트의 오퍼레이션을 호출하는 방법이다.

4. 결론

이상에서 컴포넌트를 정제하는 방법을 살펴보았다. 본 논문의 기법은 컨트롤 클래스와 엔티티 클래스를 사용하여 컴포넌트의 역할과 관리하는 정보를 결정하는 것이다. 가장 핵심적인 기법은 컴포넌트에 association을 할당할 것인지 인터페이스 호출로 대체할 것인지 결정하는 것이다. 이 기법은 데이터베이스 설계의 denormalization과 유사하다. 컴포넌트에 association을 할당하면 컴포넌트는 association을 구현하는 방법을 선택할 수 있고 조회시 성능을 향상시킬 수 있다. 그러나 association을 인터페이스 호출로 대체할 경우에는 성능이 떨어질 때 해결방법이 없다. 따라서 빈번하기 조회가 발생하는 클래스와 association은 반드시 컴포넌트 내부에 넣어야 한다.
이상의 기법은 컴포넌트 정제뿐 아니라 컴포넌트와 데이터베이스의 관계를 결정할 경우에도 유용한다.
본 논문에서는 컴포넌트와 컨트롤, 엔티티 클래스 사이의 다양한 상황에 따른 컴포넌트 분할, 통합에 대한 가이드는 제시하지 않았다. 향후 연구로 다양한 상황에서 사용할 수 있는 컴포넌트 설계 기법을 제시하겠다.

참고문헌

[1] Szyperski, C., Component Software-beyond Object-Oriented Programming, pp. 21, Addison Wesely, 2002.

[2] Cheesman, J. and Daniels, J., UML Components: A Simple Process for Specifying Component-Based Software, pp. 50, Addison-Wesley, 2000.

[3] D'Souza, D.F., and Wills, A.C., Objects, Components, and Frameworks with UML: The Catalysis Approach, pp. 91, Addison-Wesley, 1998.

[4] Allen, P., Frost, S., Component-Based Development for Enterprise Systems, Cambridge, 1998.

[5] Atkinson, C., Component-Based Product Line Engineering with UML, Addison Wesley, 2002.

[6] Allen, P., Realizing e-Business with Components, Addison-Wesley, 2000.

[7] Herzum, P., Sims, O., Business Component Factory A Comprehensive Overview of Component-Based Development for the Enterprise, Wiley & Sons, 2000.Building Reliable Component-Based Software Systems.