유즈케이스 기반의 컴포넌트 식별 방법

김태웅○ 김경민
{ twkim○, kmkim }@cs.inje.ac.kr
인제대학교 전자계산학과

An Approach to Component Identification based on Use-Case

Taewoong Kim○ Kyungmin Kim
Department of Computer Science, Inje University

요 약

컴포넌트 기반 개발 방법론이 확산됨에 따라 성공적인 컴포넌트 기반 프로젝트의 핵심 요소인 효과적인 컴포넌트 식별 방법에 관한 연구가 활발히 진행되고 있다. 이에 본 논문에서는 시스템이 사용자에게 제공하는 기능을 독립적으로 분류한 유즈케이스를 기반으로 하여 인터페이스를 식별하고, 식별된 인터페이스의 상호작용을 분석하여 컴포넌트를 식별하는 방법에 대해 제안한다. 이를 위하여 유즈케이스를 기반으로 외부 인터페이스를 식별하고, 시나리오를 이용하여 식별된 인터페이스 단위로 객체를 추출한다. 추출된 객체에서 공통 객체를 분석하여 내부 인터페이스와 컴포넌트를 식별하고 최종적으로 이러한 인터페이스의 상호작용과 의존성을 분석하여 컴포넌트를 식별 하고자 한다.

1. 서론

컴포넌트 기반의 소프트웨어 개발은 품질향상, 빠른 개발과 유지 보수의 효율성과 같은 소프트웨어 재사용에 의한 이익에서 얻을 수 있다. 컴포넌트란 한 단위로서 독립적으로 개발, 배포되어 질 수 있고, 요구되는 시스템 구성을 위해서 다른 컴포넌트와 연결되어질 수 있는 소프트웨어의 응집력 있는 패키지이고 일정한 기능을 수행할 수 있는 실체화 단위이어야 한다[1]. 이러한 정의가 현실화되기 위해서는 컴포넌트의 효율적인 식별, 응집성과 결합성이 좋은 독립적인 컴포넌트의 추출, 재사용을 위한 컴포넌트의 식별이 선행되어야 한다. 기존의 컴포넌트 기반 개발 방법론은 컴포넌트를 식별하기 위한 기본 지침들과 각 작업들 간의 관계를 명확히 제시하지 않고 추상적인 프로세스를 제시하고 있다. 또한 시스템 전체 도메인 중심의 컴포넌트 식별을 위한 절차와 방법을 제시하고 있어 실질적으로 적용하는데 어려움이 있다.

따라서 본 논문에서는 컴포넌트를 식별하기 위한 방법으로 시스템이 사용자에게 제공하는 기능을 독립적으로 분류한 유즈케이스를 기반으로 하여 인터페이스를 식별하고, 식별된 인터페이스의 의존성을 분석하여 컴포넌트를 식별하는 방법에 대해 제안하고자 한다. 또한 컴포넌트를 식별하기 위한 기본 지침을 제안하여 일관성을 유지 하도록 하였다.

2장에서는 기존의 컴포넌트 식별방법과 문제점에 대해서 기술하고, 3장에서는 본 논문에서 제안하는 컴포넌트 식별방법에 관하여 제시하고, 4장에서는 결론 및 향후 연구 과제를 기술한다.

2. 컴포넌트 식별 방법과 문제점

2.1 CBD96에서의 식별 방법

전체 도메인을 중심으로 타입 다이어그램과 비즈니스 타입 다이어그램을 추출하고 비즈니스 타입 다이어그램의 타입 중 핵심 타입을 중심으로 관련된 타입을 그룹화 하여 비즈니스 컴포넌트를 추출한다[2,3]. 한 개의 핵심 타입 다이어그램 당 하나의 인터페이스를 식별하며, 하나의 인터페이스에 한 개의 비즈니스 컴포넌트를 생각한다. 이러한 방법은 핵심 타입을 추출하기가 쉽지 않으며 분석가의 직관과 경험에 의존하는 문제점이 있다.

2.2 UML Components에서의 식별 방법

CBD96의 방법을 확장하여 Chessman과 Daniels가 제안한 방법론이다[4]. 이는 CBD96과 다르게 시스템 컴포넌트와 비즈니스 컴포넌트를 정확하게 분리하여 적용하고 있다. 유즈

케이스를 통하여 시스템 인터페이스와 오퍼레이션을 정의하고 비즈니스 타입 모델의 핵심 타입을 중심으로 비즈니스 인터페이스를 식별한다. 즉 한개의 비즈니스 타입 당 한 개의 인터페이스를 추가한다. 이것은 CBD96의 문제점을 그대로 가지며, 비즈니스 컴포넌트를 조합하여 시스템 컴포넌트를 완성하기 위해서는 다시 시스템 컴포넌트에 포함된 각 유즈케이스를 분석하여야 하는 번거로움이 있다.

2.3 UNIFACE에서의 컴포넌트 식별 방법

유즈케이스별로 비즈니스 기능 모델링을 통해서 기본적인 유즈케이스를 추출하고, 객체 모델링과 행위 모델링을 통해서 클래스 다이어그램을 완성한다. 클래스의 밀접한 관련을 통해 비즈니스 컴포넌트를 식별한다[5]. 또한 CBD96과 UML Component의 방법론의 문제점을 많이 보완하고 있다. 그러나 연관된 클래스의 그룹화에 대한 정확한 지침이 설정되어 있지 않고, 컴포넌트를 식별하는데 있어 유사한 기능을 어떻게 그룹화 할 것인지에 대한 명확한 지침이 설정되어 있지 않다.

2.4 RUP에서의 식별 방법

유즈케이스를 분석한 후 추출된 유즈케이스를 기반으로 정적 모델링과 동적 모델링을 통해서 시스템 아키텍처를 정의한 후 컴포넌트를 식별한다[6]. 이는 컴포넌트를 식별하는 방법만 제시하고 있고 제시한 방법을 사용해서 컴포넌트를 식별하기 위한 구체적인 지침과 절차를 제시 하지 않고 있으므로 분석가의 직관과 경험에 의존해야 하는 단점을 가지고 있다.

3. 유즈케이스 기반의 컴포넌트 식별 방법

이 장에서는 위에서 제시한 컴포넌트 식별 방법의 문제점을 보완하여 보다 효율적으로 컴포넌트를 식별하기 위한 외부 인터페이스 식별, 시나리오 기반 객체 추출, 공통 클래스의 추출, 내부 인터페이스 식별, 컴포넌트 식별에 대하여 제안한다.

3.1 외부 인터페이스 식별

요구분석 단계에서 생성된 유즈케이스는 사용자 관점에서 시스템이 사용자에게 제공하는 기능을 이미 독립적으로 분류하여 정의 한 것이다. 따라서 하나의 유즈케이스에 대해 서비스 가능한 하나의 외부 인터페이스를 식별한다. 외부 인터페이스는 컴포넌트 사용자나 조립자에게 보여질 수 있으며, 이 인터페이스를 통해 서비스를 제공받게 된다. 그림1은 물품 구매 시스템을 나타낸 유즈케이스 다이어그램이다.

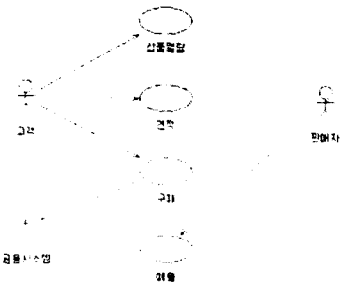

그림 1 유즈케이스 다이어그램

3.2 시나리오 기반 객체 추출

요구 사항 분석에서 유즈케이스와 함께 기술된 시나리오 즉, 하나의 유즈케이스 기능을 실행하기 위한 비즈니스 계층의 시나리오를 통해 객체를 추출하고 클래스의 관계를 타입 다이어그램으로 나타낸다. 따라서 하나의 유즈케이스에 대해 하나의 타입 다이어그램이 생성된다. 클래스가 가질 수 있는 스테레오 타입은 ≪state≫, ≪control≫,≪entity≫가 있으며 표1의 정의를 가진다. 그림2에서 그림5는 각 유즈케이스의 시나리오를 기반으로 추출된 클래스를 나타낸다.

스테레오타입	정 의
≪state≫	인터페이스의 상태를 가지는 상태 클래스
≪control≫	제어 클래스
≪entity≫	실체 클래스

표 1 스테레오 타입에 대한 정의

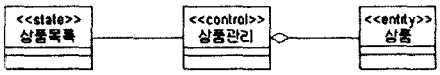

그림 2 상품열람 유즈케이스에 대한 객체 추출

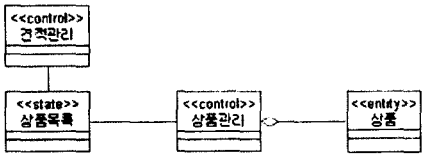

그림 3 견적 유즈케이스에 대한 객체 추출

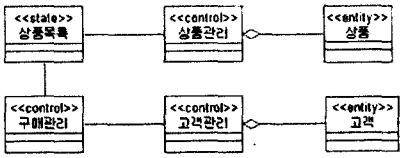

그림 4 구매 유즈케이스에 대한 객체 추출

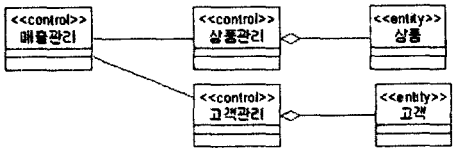

그림 5 매출 유즈케이스에 대한 객체 추출

3.3 공통 객체의 추출

3.2에서 추출된 타입 다이어그램에서 두개 이상의 타입 다이어그램에 사용되는 공통 상태 클래스와 공통 컨트롤 클래스를 추출한다. 추출된 각 클래스는 표2와 같다.

종 류	이 름	사용된 유즈케이스
상태 클래스	상품목록	상품열람, 견적, 구매
제어 클래스	상품관리	상품열람, 견적, 구매, 매출
제어 클래스	고객관리	구매, 매출

표 2 유즈케이스로부터 추출된 공통 클래스

3.4 내부 인터페이스 식별

3.3에서 추출한 공통 상태 클래스는 세 개의 외부인터페이스에서 사용하지만 컴포넌트의 외부에서는 존재를 알 필요가 없다. 따라서 상태 클래스를 묶어 하나의 내부 인터페이스로 식별하고 해당 인터페이스를 참조하는 오퍼레이션을 정의하여 외부로 서비스되는 외부 인터페이스와 구분한다. 공통으로 사용되는 제어 클래스는 엔티티 정보를 가지는 실체 클래스에 대한 연관을 가지고 모든 외부 인터페이스에 대해 공통 객체에 해당하므로 하나의 외부 인터페이스로 식별한다. 표 3은 식별된 인터페이스와 식별된 출처를 나타낸다.

인터페이스 종류	출 처
외부 인터페이스 1	상품열람 유즈케이스
외부 인터페이스 2	견적 유즈케이스
외부 인터페이스 3	구매 유즈케이스
외부 인터페이스 4	매출 유즈케이스
외부 인터페이스 5	상품관리 공통 제어 클래스 추출
외부 인터페이스 6	고객관리 공통 제어 클래스 추출
내부 인터페이스	상품목록 공통 상태 클래스 추출

표 3 식별된 인터페이스

3.5 컴포넌트 식별

컴포넌트는 하나 이상의 인터페이스로 구성되어 있으며 이러한 인터페이스는 사용자의 기능적인 관점으로는 서로 독립적임과 동시에 수행관점에서는 서로 의존성을 가지고 있다. 공통 상태 클래스로부터 식별된 내부 인터페이스와 연관이 있는 외부 인터페이스1,2,3을 묶어 컴포넌트1로 식별하고 나머지 하나의 인터페이스를 하나의 컴포넌트2로 식별한다. 또한 공통 제어 클래스로부터 식별된 외부 인터페이스5와 6은 유즈케이스로부터 식별된 4개의 외부 인터페이스와 연관되어 있으므로 각각을 별개의 컴포넌트를 두어 여기에 할당한다. 따라서 각각 하나의 외부 인터페이스를 가지는 컴포넌트 3과 4가 식별된다.

외부로 서비스 가능한 외부 인터페이스를 유출하여 다른 컴포넌트와 결합 가능하고 내부 인터페이스의 상호 작용과 의존성을 분석하여 컴포넌트로 식별함으로써 응집력 있는 소프트웨어 패키지가 된다. 그림 6은 최종적으로 식별된 4개의 컴포넌트를 나타낸다.

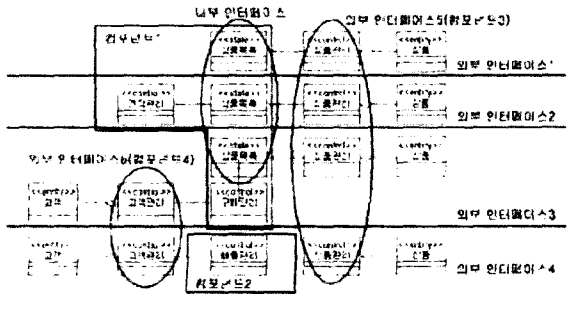

그림 6 식별된 4개의 컴포넌트

4. 결론 및 향후 과제

컴포넌트 기반 개발 방법론이 확산됨에 따라 성공적인 컴포넌트 기반 프로젝트의 핵심요소인 효과적인 컴포넌트 식별 방법에 관한 연구가 활발히 진행되고 있다. 그러나 기존의 컴포넌트 기반 개발 방법론은 컴포넌트를 식별하기 위한 기본 지침들과 각 작업들 간의 관계를 명확히 제시하지 않고 추상적인 프로세스를 제시하고 있어 실질적으로 적용하는데 어려움이 있었다. 또한 컴포넌트를 식별하기 위한 명확한 지침이 부족하기 때문에 분석가의 직관과 경험에 의해서 컴포넌트를 식별하여야 하는 단점이 있었다. 본 논문에서는 이러한 문제점을 해결하기 위하여 컴포넌트를 식별하는 방법으로 외부 인터페이스와 내부 인터페이스를 식별하고, 그들 간의 상호작용과 의존성을 파악하고, 인터페이스들을 조합하여 하나의 컴포넌트를 식별하는 구체적이고 실용적인 방법을 제시 하였다. 또한 컴포넌트를 식별하는 구체적인 지침을 제시하여 컴포넌트 식별의 일관성을 유지 하도록 하였다.

본 연구를 통하여 추상적인 지침과 명확하지 않은 작업들 간의 관계로 인하여 발생할 수 있는 컴포넌트 인터페이스 식별과정에서의 모호성을 감소시킬 수 있었다. 또한 식별된 인터페이스들의 상호작용 및 의존성을 분석하여 컴포넌트를 식별함으로써 응집력 있는 소프트웨어를 개발하고 일관성 있는 컴포넌트를 식별하는데 효과적임을 알 수 있었다.

향후 실제 다양한 도메인의 응용 시스템에서의 사례연구를 통해 효과적인 컴포넌트 식별에 대한 지침을 보완하고, 제안한 방법에 대한 실효성과 일관성을 검증하는 연구가 필요하다. 또한 컴포넌트를 식별하기 위한 각 작업들에서의 산출물을 표현하는 방법과 관리하는 컴포넌트 모델링에 관한 연구가 필요하다.

[참고문헌]

[1] Desmond Francis Dsouza, Alan Cameran wills, "Object, Component, and Frameworks with UML : Catalysis Approach" Addison Wesley, 1999

[2] Castek, "Exploring Comprehensive CBD Method", http://www.cbd-hq.com/PDFS/cbdhq_000615_cbde_for_ICSE.pdf

[3] Ivar Jacopson, "The Object Advantage", ACM Press, Addison-Wesley, 2000

[4] John Cheesman, John Daniels, "UML Components", Addison-Wesley, 2001

[5] Computerware corp., "UNIFACE Development Methodology:UNIFACE v.7.2" Compuware corp., 1998

[6] Ivar Jacobson, Grady Booch, James Rumbaugh, "The Unified Software Development Process", Addison-Wesley, 1999