클래스들 간의 정적·동적 관계에 의한 2단계 컴포넌트 식별방법

(A Two-Phase Component Identification Method using Static and Dynamic Relationship between Classes)

최 미 숙 † 조 은 숙 †† 박 재 년 ††† 하 종 성 ††††
(Mi-Sook Choi) (Eun-Sook Cho) (Jai-Nyun Park) (Jong-Sung Ha)

요 약 컴포넌트 개발 프로세스에서 재사용 가능한 독립적인 컴포넌트의 식별은 가장 중요한 작업이면서 어려운 작업이다. 따라서 현존하는 컴포넌트 개발 방법론들에서는 컴포넌트 식별을 위해서 체계적이고 명확한 기준이 제시되지 않아 대다수 개발자의 직관과 경험에 의존하고 있다. 그 결과 평이한 개발자에 의해서 소프트웨어의 재사용 단위인 컴포넌트를 식별하기가 쉽지 않다. 따라서 본 논문에서는 컴포넌트를 용이하게 식별할 수 있도록 유스케이스 다이어그램, 클래스 다이어그램 그리고 시퀀스 다이어그램 등 도메인 모델을 기반으로 컴포넌트를 식별하는 기준과 방법을 제시한다. 본 논문에서는 2단계 즉, 시스템 컴포넌트 식별과 비즈니스 컴포넌트 식별을 통하여 컴포넌트를 식별하는 방법을 제시한다. 특히, 제안된 기법에서는 컴포넌트 식별에 있어서 구조적 특성 뿐만 아니라 메소드 호출 유형과 방향에 따른 의존성 특성을 함께 고려하고 있다. 이러한 제안된 기법의 실용성을 검증하기 위해 사례 연구와 기존 식별 방법과의 비교 분석 및 평가를 제시한다.

키워드 : 시스템 컴포넌트, 비즈니스 컴포넌트, 컴포넌트 식별, 컴포넌트 개발 방법론

Abstract It is difficult to identify reusable and independent components in component-based development(CBD) process. Therefore existing methodologies have dealt the problem of component identification based on only developer's intuition and heuristics. As a result, it is difficult to identify the business components by common developers. Therefore, in this paper, we propose a new baseline and technique to identify the business components based on domain model such as use case diagrams, class diagrams, and sequence diagrams. Proposed method identifies components through two phases; system component identification and business component identification. Especially, we consider structural characteristics as well as dependency characteristics according to methods call types and directions in identifying components. We also present a case study and comparative analysis and assessment to prove the practical use of our technique.Abstract

Key words : System Component, Business Component, Component Identification, Component Development Method

1. 서 론

최근 정보통신의 발달에 따라 소프트웨어 개발 환경이 빠르고 복잡하게 변화 되면서 제한된 시간에 저렴한 비용으로 원하는 시스템을 개발할 수 있는 소프트웨어 개발 방법론인 컴포넌트 기반 개발 방법론이 1990년대 말부터 나타나기 시작했다. 이 방법론은 독립적인 기능의 조각인 컴포넌트를 조립함으로써 시스템을 개발하는 방법으로 사용자의 변화하는 요구 사항을 보다 쉽게 충족시켜 고품질의 시스템을 효율적으로 개발할 수 있는 기술이다. 컴포넌트란 소프트웨어의 한 단위로써 독립적으로 개발, 배포되어질 수 있고 요구되는 시스템 구성을 위해서 원시 코드의 수정 없이 다른 컴포넌트와 연결되어질 수 있는 소프트웨어의 응집력 있는 패키지이고 일

· 본 논문은 2004년도 숙명여대 교내연구비에 의해 지원되었음
† 정 회 원 : 우석대학교 컴퓨터공학과 교수
khc67 kr@hanmail.net
†† 종신회원 : 동덕여자대학교 컴퓨터정보과학부 교수
escho@dongduk.ac.kr
††† 종신회원 : 숙명여자대학교 정보과학부 교수
jnpark@sookmyung.ac.kr
†††† 종신회원 : 우석대학교 컴퓨터공학과 교수
jsha@core.woosuk.ac.kr
논문접수 : 2004년 9월 16일
심사완료 : 2004년 10월 21일

정한 기능을 수행할 수 있는 실체화의 단위이어야 한다[1]. 따라서 컴포넌트 기반의 시스템 구축을 위하여 가장 중요한 점은 기능적으로 재사용이 가능하고 유지보수 단계의 효율적인 시스템 관리를 위해서 상호 의존성이 적은 독립적인 컴포넌트를 잘 식별하는 것이다. 그러나 컴포넌트를 식별하기 위한 기존 방법론들[2-5]은 재사용 가능한 독립적인 컴포넌트를 식별하기 위하여 개발자의 직관과 경험에 의하여 컴포넌트를 식별하는 방법을 제시하고 있으므로 좀 더 명확한 기준과 방법이 필요하다. 또한 컴포넌트 아키텍쳐는 서브 시스템의 단위가 되는 시스템 컴포넌트와 시스템 컴포넌트의 기능을 실행하기 위한 비즈니스 컴포넌트의 조합으로 구성되는데 기존의 컴포넌트 방법론들은 대 다수 비즈니스 컴포넌트 식별 방법만[2-8]을 제시하고 있다.

따라서 본 논문에서는 기존 방법들의 문제점을 보완 확장하여 컴포넌트를 용이하게 식별할 수 있도록 유스케이스 다이어그램, 클래스 다이어그램 그리고 시퀀스 다이어그램 등 도메인 모델을 기반으로 컴포넌트를 식별하는 기준과 방법을 제시한다. 본 논문에서 제시하는 컴포넌트 식별 방법은 먼저 서브 시스템의 단위가 되는 시스템 컴포넌트를 식별한 후 비즈니스 컴포넌트를 식별한다. 서브시스템이 포함하는 클래스들은 전체 시스템이 포함하는 클래스들의 수보다 적기 때문에 더욱 효율적으로 컴포넌트를 식별할 수 있다. 또한 클래스 간의 구조적 특성, 클래스 간의 메소드 호출 유형과 메소드 호출 방향에 의한 클래스 간의 정적 그리고 동적인 관계에 의한 종속적 특성과 의존의 강도를 부여하여 효율적으로 컴포넌트를 식별할 수 있는 기준과 방법을 제안한다. 그리고 전자상거래 도메인인 인터넷 쇼핑몰을 통한 사례를 제시하고 기존의 컴포넌트 식별 방법과의 비교 분석을 통해서 컴포넌트가 효율적으로 식별됨을 검증한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 컴포넌트 식별 방법과 문제점을 기술한다. 3장에서는 본 논문에서 제안한 컴포넌트 식별 방법과 절차를 제시한다. 4장에서는 전자상거래 시스템 사례연구와 비교평가를 통해서 컴포넌트가 효율적으로 식별됨을 검증한다. 5장에서는 결론 및 향후 연구과제를 제시한다.

2. 관련연구

컴포넌트 기반의 시스템은 컴포넌트를 두 가지 단위로 구분할 수 있다. 유사한 기능을 그룹화하여 기능적 재사용이 가능하도록 컴포넌트화 한 시스템 컴포넌트와 시스템 컴포넌트의 기능을 수행하기 위하여 구성되는 비즈니스 컴포넌트가 존재한다[4]. 시스템 컴포넌트는 기능적 재사용의 단위이면서 서브시스템의 단위이다. 시스템 컴포넌트는 의미적으로 유사한 기능을 수행할 수 있는 유스케이스를 중심으로 식별됨으로 각각의 유스케이스는 시스템 컴포넌트가 수행해야 할 시스템 인터페이스가 된다. 시스템 컴포넌트가 포함하는 유스케이스의 기능을 실행하기 위하여 서로 밀접하게 관련된 클래스들을 그룹화하여 식별된 비즈니스 컴포넌트들의 상호작용에 의하여 실행된다. 즉, 시스템 컴포넌트의 인터페이스를 실행하기 위하여 독립적인 부품의 단위가 되는 컴포넌트가 비즈니스 컴포넌트이다.

Computer Associates사의 Cool Joe가 기반으로 하는 CBD96[3] 방법은 핵심 타입에 의해서 밀접하게 관련된 객체들을 그룹화하여 컴포넌트를 식별하는 비즈니스 컴포넌트 식별 방법이 존재하지만 시스템 서비스 측면의 기능을 재사용 할 수 있는 시스템 컴포넌트에 대한 개념이 없고 전체 도메인을 중심으로 비즈니스 컴포넌트를 식별하고 개발자의 직관과 경험에 의하여 비즈니스 컴포넌트를 식별해야 하는 어려움이 존재한다.

CBD96[3]의 컴포넌트 개발 방법을 확장하여 Cheesman과 Daniels가 제안한 Advisor방법[4]은 시스템 서비스 측면의 기능적 재사용이 가능하도록 시스템 컴포넌트의 정의가 명확히 명시되어 있지만 시스템 컴포넌트와 비즈니스 컴포넌트가 명확히 분리되어서 식별되기 때문에 비즈니스 컴포넌트를 조합하여 시스템 컴포넌트를 완성하기 위해서는 다시 시스템 컴포넌트의 인터페이스인 각 유스케이스를 분석하여 어떠한 타입, 즉 어떠한 클래스가 어느 시스템 컴포넌트에 포함될 것인지를 결정해야 한다. 또한 그 클래스는 결정된 시스템 컴포넌트의 어느 비즈니스 컴포넌트에 포함되는지를 다시 한 번 고려하여야 한다. 또한 비즈니스 컴포넌트를 식별함에도 핵심 타입에 의해서 밀접하게 관련된 객체들을 그룹화하여 컴포넌트를 식별하므로 개발자의 직관과 경험을 필요로 한다.

Rational사의 RUP[5]는 Booch의 방법론, Rumbaugh의 방법론, Jacobson의 방법론을 통합한 방법론이다. RUP 방법에서의 컴포넌트 식별은 컴포넌트 식별단계에서 컴포넌트를 식별하기 위한 방법만을 제시하고 있지 컴포넌트를 식별하기 위한 구체적인 지침과 절차를 제시하고 있지 않으므로 개발자의 직관과 경험에 의하여 시스템 컴포넌트와 비즈니스 컴포넌트를 식별할 수 밖에 없다.

ETRI가 제안한 마르미III[6]의 컴포넌트 식별 방법은 대표적으로 객체모형을 통한 컴포넌트 식별방법과 UDA(Usecase Data Access) 테이블을 통한 컴포넌트 식별방법이 존재한다. 객체모형을 통한 컴포넌트 식별방법은 엔티티 유형의 클래스를 중심으로 핵심 클래스를 식별한 후 핵심클래스와 관련된 주변 클래스들을 묶어 컴포넌트 후보를 도출한다. 이 방법은 기존의 방법론인

CBD96 방법과 유사하다. UDA(Usecase Data Access) 테이블을 통한 컴포넌트 식별방법은 유스케이스가 참조하는 클래스의 메소드가 C(생성) D(삭제) W(수정) R(참조)의 관계에 대하여 가중치를 부여하고 친화성 분석에 의하여 컴포넌트를 식별한다. 그러나 마르미 III는 유스케이스와 클래스 간의 관계만을 분석하여 컴포넌트를 식별하기 때문에 클래스 간의 메소드 호출 유형이나 방향성을 고려하지 않고 있으므로 정확하게 컴포넌트를 식별할 수 없다. 또한 임의의 유스케이스가 기능을 실행하기 위해서 클래스를 참조만 한다면 클래스와 유스케이스가 그룹화 될 조건이 제시되지 않으므로 참조관계만을 가진 유스케이스는 어느 후보 컴포넌트로 들어가도 상관없다고 제시하고 있다. 그러나 시스템 컴포넌트를 식별한 후 비즈니스 컴포넌트를 식별한다면 이러한 문제는 해결되는데 마르미 III 역시 시스템 컴포넌트의 개념이 정의되지 않고 전체 도메인에서 컴포넌트를 식별한다.

[7]이 제안한 컨셉 분석에 의한 컴포넌트 식별(Component Identification via Concept Analysis)방법은 클래스 간의 메시지 호출 수 만을 고려하여 그룹화하여 비즈니스 컴포넌트를 식별한다.

[8]의 방법 역시 클래스 간의 메소드 호출 유형이나 방향성을 고려하지 않고 있으므로 정확하게 컴포넌트를 식별할 수 없다.

3. 클래스들 간의 정적·동적 관계 기반의 2단계 컴포넌트 식별방법

본 장에서는 객체기반 도메인 모델에서의 클래스들 간의 의존관계의 특성을 정적인 측면과 동적인 측면을 기준으로 분류하고 그러한 특성을 적용하여 시스템 컴포넌트를 기반으로 비즈니스 컴포넌트를 식별하는 2단계 컴포넌트 식별 방법에 대한 기준, 방법 그리고 절차를 제시한다.

3.1 객체기반 도메인 모델에서의 클래스들 간의 정적, 동적 관계의 특성 분류

클래스 간에 종속성이 존재한다는 것은, 클래스 C2와 C1에 대하여 <C2, C1> ∈ R의 관계가 존재한다면 클래스 C2에 대한 객체가 클래스 C1의 객체에게 메시지를 보냄으로 이루어지고 C2 객체의 메시지 호출에 의하여 C1의 객체가 영향을 받으므로 C1 객체는 C2 객체에 의존 또는 종속되어 있다고 정의할 수 있다[10,11]. 본 절은 이러한 특성을 클래스 간의 정적인 측면의 구조적 관계, 동적인 측면의 메시지 호출의 유형 그리고 메시지 호출의 방향에 따라 분리하여 제시한다.

3.1.1 클래스 간의 구조적 특성에 의한 정적관계의 종속성

클래스들의 구조적 관계는 포함관계, 상속관계 그리고 연관관계가 존재한다. 클래스들의 관계가 포함관계라면, 즉, A 클래스가 B 클래스와 C 클래스를 포함하고 있다면 A 클래스의 수정은 B클래스와 C 클래스에 영향이 그대로 전파되고 또한 A 클래스에 대한 객체의 연산은 B 클래스에 대한 객체와 C 클래스에 대한 객체들에 그대로 전파된다. 예를 들어 A 클래스에 대한 객체가 생성, 삭제, 수정 그리고 참조 되어진다면 동시에 B 클래스와 C 클래스에 대한 객체에 똑같은 연산이 전파되어진다. 클래스 간의 상속관계도 유사한 구조적 특성을 가지고 있다. 그러므로 포함관계와 상속관계의 클래스들은 상위 클래스의 수정이나 상위 객체의 수정은 하위 클래스나 객체들에 직접적으로 강한 영향을 전파함으로 수직적으로 주종관계를 이루며 강하게 결합되어있고 종속되어있다고 정의할 수 있다. 따라서 보다 독립적이어야 하고 컴포넌트 간의 종속성이 최소한이어서 서로 영향을 적게 받아야 하는 컴포넌트의 특성상 이러한 관계의 클래스들은 분리되어질 수 없고 반드시 하나의 컴포넌트 안에 포함되어져야 한다. 클래스들의 관계가 연관관계라면, 즉, A클래스와 B 클래스 사이에 연관관계가 존재한다면 A 클래스와 B 클래스 사이에 메시지 호출에 의한 수평적 관계이다. 즉, 클래스 A와 B에 대하여 <A, B> ∈ R의 연관관계가 존재한다면 클래스 A와 B의 연관관계는 클래스 A에 대한 객체가 클래스 B의 객체를 생성, 삭제, 수정 그리고 조회하는 메시지를 보내거나 생성, 삭제, 수정 그리고 조회 등을 포함한 메시지 호출에 의해서 이루어진다. 연관관계는 클래스들 간의 관계가 포함관계나 상속관계와 같이 클래스의 영향이 다른 클래스에 그대로 전파되는 것이 아니라 메시지 호출 유형에 따라서 클래스와 객체에 대한 수정 영향의 강도 즉 의존의 강도가 달라진다. 즉, 메소드 호출 유형에 따라서 전혀 영향을 받지 않는 클래스 간의 관계도 존재 한다는 것이다. 따라서 클래스의 구조적 특성에 의한 클래스 간의 종속의 강도는 포함관계 > 상속관계 > 연관관계 순이고 컴포넌트의 특성상 포함관계나 상속관계의 클래스들은 분리되어질 수 없으므로 하나의 컴포넌트 안에 포함되어져야 하지만 연관관계에 있는 클래스들은 메소드의 호출 유형에 따라서 종속의 관계가 달라지기 때문에 연관관계의 클래스들은 분리되어져 각각 다른 컴포넌트로 포함되어질 수도 있고 아니면 하나의 컴포넌트 안에 모두가 포함되어질 수도 있다.

3.1.2 클래스 간의 메시지 호출 유형에 따른 동적관계의 종속성

소프트웨어 요소인 클래스 간의 의존은 메소드 호출에 의해서 이루어 진다. 메소드 호출 유형에 의해서, 클래스 간의 의존의 강도를 평가하기 위해서 본 논문에서

는 메소드 호출 유형을 다음과 같이 분류한다. 즉 <A, B> ∈ R의 관계에서 메시지의 유형은 다음과 같다.

① A 객체가 B 객체를 생성하는 메소드를 호출할 경우
② A 객체가 B 객체를 삭제하는 메소드를 호출할 경우
③ A 객체가 B 객체를 수정하는 메소드를 호출할 경우
④ A 객체가 B 객체를 참조하는 메소드를 호출할 경우
⑤ A 객체가 B 객체에게 ①,②,③,④의 경우 등을 포함하는 메소드를 호출할 경우

①과 ②의 경우는 A 객체에 의해서 B 객체의 구조가 변경되므로 B 객체를 참조하여 기능을 수행하는 다른 객체들은 전체적으로 구조적 측면에서 영향을 받는다. ③의 경우는 ①과 ②의 경우처럼 구조가 변경되는 것이 아니라 객체의 값이 변경되므로 B 객체를 참조하여 기능을 수행하는 다른 객체들은 기능을 수행할 수 없는 것이 아니라 기능은 수행하되 값의 변경만 이루어진다. ④의 경우는 ①과 ②의 경우처럼 구조가 변경되는 것도 아니고 ③의 경우처럼 값의 변경이 이루어지는 것도 아니므로 B 객체를 참조하여 기능을 수행하는 다른 객체들은 전혀 영향을 받지 않는다. 따라서 ④의 경우 객체 A가 a 컴포넌트에 그리고 객체 B가 b컴포넌트에 포함되어 a 컴포넌트의 객체 A가 b컴포넌트의 객체 B를 참조하는 메시지를 보낸다면 ①과 ②의 경우처럼 구조가 변경되는 것도 아니고 ③의 경우처럼 값의 변경이 이루어지는 것이 아니므로 b 컴포넌트 내의 B 객체를 참조하여 기능을 수행하는 다른 객체들은 전혀 영향을 받지 않으므로 두 컴포넌트 간의 종속관계는 ①, ② 그리고 ③의 경우 보다 가장 약한 의존관계를 가지고 있다고 정의할 수 있다. ⑤의 경우는 ①,②,③,④ 중에 어떠한 메소드를 포함하는 메시지를 호출할 경우에 따라서 달라진다.

3.1.3 메시지 방향에 따른 클래스 간의 동적관계의 종속성

객체 <A, B> ∈ R의 관계에서 A 객체가 B 객체에게 B객체를 생성하는 메시지를 보내는 경우에는 A 객체에 의해서 B 객체가 종속되는 관계이고 B 객체의 생성은 B 객체를 조작하는 다른 객체들에게 구조적으로 영향을 미친다. 그러나 A 객체가 B 객체에게 B 객체의 자료를 참조하는 메시지를 보낸다면 A 객체는 B 객체의 참조라는 메시지에 의해서 종속되는 관계이나 A 객체는 단지 B 객체의 데이타를 단지 참조만하기 때문에 B 객체는 전혀 영향을 받지 않는다. 따라서 객체들 간의 메시지 호출 방향에 의해서 종속되는 관계가 달라지므로 반드시 메시지 호출의 방향을 고려하여야 한다.

① A→B : A객체가 B 객체에게 메소드를 호출할 경우
② A↔B : A객체와 B 객체가 서로 메소드를 호출할 경우

①의 경우A 객체가 B 객체에게 어떠한 유형의 메소드를 호출하느냐에 따라 의존의 강도가 달라진다. ②의 경우 기존에는 무조건 양방향의 메소드 호출 관계가 존재한다면 강한 종속관계를 가지고 있다라고 정의[8] 하였으나 본 논문에서는 양 방향의 메소드 호출관계가 존재하더라도 메소드 호출 유형에 따라서 의존의 강도가 달라진다.

3.2 컴포넌트 식별 기준

3.1절에서 제시한 클래스들 간의 의존관계의 특성에 의하여 컴포넌트 식별 기준을 다음과 같이 정의한다.

정의 1. 클래스 간의 관계가 포함관계나 상속관계라면 그들 사이에 강한 종속관계가 존재하므로 그러한 관계의 클래스들은 하나의 컴포넌트 안에 포함되어져야 한다.

정의 2. 클래스 간의 관계가 연관관계이면서 메시지 호출 유형이 생성이나 삭제관계라면 그들 사이는 객체의 구조를 바꾸는 강한 종속관계가 존재하므로 하나의 컴포넌트 안에 포함되어져야 한다.

정의 3. 클래스 간의 관계가 연관관계이면서 메시지 호출 유형이 참조관계라면 호출을 받는 객체는 전혀 영향을 받지 않으므로 그들 각각이 다른 컴포넌트로 포함된다 하더라도 각각의 컴포넌트는 독립성을 만족한다.

정의 4. 클래스 간의 관계가 연관관계이면서 메시지 호출 유형이 수정관계라면 호출을 받는 객체의 구조를 바꾸는 것이 아니라 단지 객체의 값만 변경되므로 그들 각각은 하나의 컴포넌트 안에 포함되어 질 수도 있고 분리되어 질 수도 있다.

3.3 시스템 컴포넌트의 식별 방법

기존의 컴포넌트 식별 방법은 시스템 컴포넌트를 추출하기 위한 구체적인 지침이 없고 의미적으로 유사한 유스케이스만을 그룹화 해서 시스템 컴포넌트를 추출한다. 그러나 Bunge의 유사성(Similarity) 정의[12]에 의하여 두 개의 메소드가 얼마나 유사한 기능을 가지고 있는가의 척도는 두 메소드가 공통으로 참조하는 인스턴스 변수의 집합에 의하여 결정된다. 즉, 기능의 유사성은 각각의 유스케이스가 공통으로 참조하는 객체들의 집합에 의하여 결정된다. 그러므로 본 논문에서는 업무를 잘 모르는 개발자라 하더라도 시스템 컴포넌트를 효율적으로 추출하기 위하여 유스케이스만이 아니라Actor 그리고 유스케이스에 포함된 객체도 고려하여 시스템 컴포넌트를 추출한다. 또한 시스템 컴포넌트 기반의 비즈니스 컴포넌트 식별이 가능하도록 하기 위하여 3.1절과 3.2 절에서 제시한 클래스들 간의 의존관계의 특성을 적용하여 한 시스템 컴포넌트의 클래스가 다른 시스템 컴포넌트에 중복되어 포함되지 않도록 공유 클래스를

표 1 공통 클래스 배치를 결정하기 위한 적용 규칙

규칙	공통 클래스와 시스템 컴포넌트 1과의 관계	결합도	공통 클래스와 시스템 컴포넌트 2와의 관계	공통 클래스의 배치
규칙1	Composition	>	Association	컴포넌트1
규칙2	Aggregation	>	Association	컴포넌트1
규칙3	Inheritance	>	Association	컴포넌트1
규칙4	Create	>	Delete, Write, Read	컴포넌트1
규칙5	Composition, Aggregation, Inheritance	∪	Composition, Aggregation, Inheritance	컴포넌트1 + 컴포넌트2

표 2 식별된 시스템 컴포넌트와 포함된 공통 클래스 액터

액터 (Actor)	유스케이스 (UseCase)	Classes										
		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Actor1	Usecase1	√		√								
	Usecase2	√		√								
Actor2	Usecase3		√	√	√							
	Usecase4		√	√	√							
	Usecase5			√	√							
	Usecase6				√	√	√			√	√	
Actor3	Usecase7				√	√	√					
	Usecase8				√	√	√			√	√	√

재배치한다. 한 시스템 컴포넌트가 포함하는 클래스들이 다른 시스템 컴포넌트의 클래스들과 중복되지 않으므로 추출된 시스템 컴포넌트를 기반으로 비즈니스 컴포넌트 식별이 가능하다. 전체 시스템의 부분 집합인 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 추출하므로 시스템을 이해하기가 더욱 쉽고 비즈니스 컴포넌트를 식별하는 것이 더욱 효율적이다. 다음 표 1은 시스템 컴포넌트가 포함하는 공통 클래스를 배치하기 위하여 3.1절에서 제시한 클래스들 간의 의존관계의 특성을 적용한 적용규칙을 정의한다.

다음은 요구사항 분석 단계와 분석 단계를 기반으로 본 연구에서 제안한 시스템 컴포넌트 식별 방법을 적용한 절차를 제시한다.

절차 1. 요구사항을 분석하여 유스케이스를 추출하고 요구사항 분석 모델을 완성한다.

절차 2. 각 유스케이스의 시나리오, 즉 이벤트의 흐름에 따라 RUP의 분석 프로세스인 유스케이스 실체화 과정을 통해서 객체를 도출한다.

절차 3. 실체화 과정에 의해서 도출된 객체를 중심으로 시퀀스 다이어그램과 클래스 다이어그램을 완성한다.

절차 4. 추출된 유스케이스 중 시스템 서비스 관점에서 유사한 기능을 가진 유스케이스와 동일한 객체를 참조하는 정도를 고려하여 유스케이스를 그룹화하여 시스템 컴포넌트를 식별한다.

절차 4-1. 도출된 유스케이스 중 포함 관계의 유스케이스는 하나의 유스케이스로 그룹화한다.

절차 4-2. Actor별 유스케이스와 이 유스케이스에 참여하는 엔티티 객체와의 관계 테이블을 만든다.

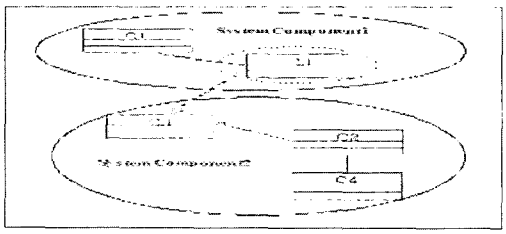


그림 1 식별된 시스템 컴포넌트 사이에 존재하는 공통 클래스의 배치

절차 4-3. Actor별로 유스케이스와 객체와의 관계 테이블을 보고 의미적으로 유사성이 높은 기능과 유스케이스와 유스케이스에 참여하는 객체의 공유정도를 고려하여 유스케이스를 그룹화하고 시스템 컴포넌트라 정의한다.

절차 5. 만약 도출된 시스템 컴포넌트 사이에 공통 클래스가 없다면 시스템 컴포넌트의 클래스로 결정되고 비즈니스 컴포넌트를 식별한다. 그러나 만약 공통 클래스가 존재한다면 표 1에서 제시한 적용 규칙을 이용하여 공통 클래스 배치를 결정한다.

절차 5-1. 시퀀스 다이어그램을 참조하여 유스케이스와 클래스 간의 CRWD 메소드의 유형을 적용한 매트릭스 테이블을 만든다(C:생성, D:삭제, W:수정, R:참조).

절차 5-2. 공통 클래스가 각 시스템 컴포넌트에 동시에 강한 결합도를 가지고 있지 않다면 표 1에서 제시한 공통 클래스의 배치 적용 규칙 1,2,3,4에 의하여 공통 클

래스의 배치를 결정한다.

절차 5-3. 공통 클래스가 각 시스템 컴포넌트에 대해서 동시에 강한 결합도를 가지고 있다면 표 1에서 제시한 공통 클래스의 배치 적용 규칙 5에 의해서 시스템 컴포넌트를 합친 후 비즈니스 컴포넌트를 식별한다.

3.4 비즈니스 컴포넌트 식별 방법

다음은 시스템 컴포넌트를 기반으로 비즈니스 컴포넌트를 식별하는 방법과 절차를 제시한다. 3.3절의 절차에 의하여 식별된 시스템 컴포넌트 사이에 포함하는 클래스들이 중복되지 않으므로 추출된 시스템 컴포넌트를 기반으로 비즈니스 컴포넌트의 식별이 가능하다. 비즈니스 컴포넌트의 식별은 시스템 컴포넌트가 포함하는 클래스들 간에 3.1절과 3.2절에서 제시한 클래스 간의 종속적 특성에 의하여 클래스들을 그룹화하여 비즈니스 컴포넌트로 식별한다.

절차 6. 3.1절과 3.2절에 의해 시스템 컴포넌트 내의 클래스들 중에 포함관계와 상속관계의 클래스들은 강한 종속관계가 존재함으로 하나의 클래스로 그룹화 하여 클래스 다이어그램을 정제한다.

절차 7. 절차 6에서 도출된 클래스 다이어그램을 기반으로 클래스들 사이의 메시지 호출관계 그래프는 시퀀스 다이어그램을 참조하여 정의한다.

절차 8. 3.1과 3.2절에 의하여 메시지 호출관계에서 생성과 삭제의 관계인 클래스들은 하나의 클래스로 그룹화 한다. 식별된 각각의 클래스들은 하나의 후보 비즈니스 컴포넌트로 식별된다.

절차 9. 3.1절과 3.2절에서 제시한 특성을 고려하여 상세 정제한 후 완전한 비즈니스 컴포넌트를 식별한다.

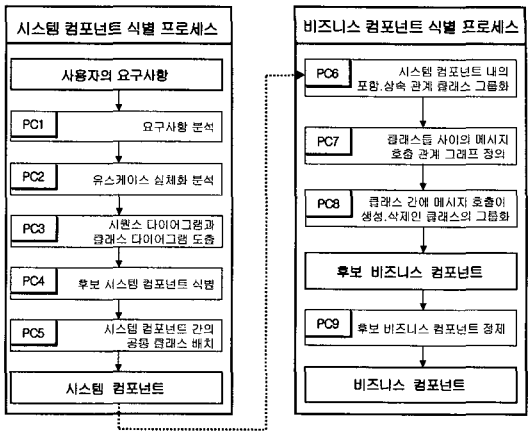


그림 2 컴포넌트 식별 프로세스

4. 사례 연구 및 평가

인터넷과 웹 애플리케이션 기술의 급속한 발전으로 웹 애플리케이션 유형들 가운데 급성장하고 있는 분야는 전자상거래 애플리케이션이다. 따라서 본 논문에서는 전자상거래 도메인을 선택하여 컴포넌트 식별 사례를 제시하고 그 효과성을 검증한다.

4.1 사례연구

절차 1. 요구사항을 분석하여 유스케이스를 추출하고 요구사항 분석 모델을 그림 3과 같이 완성한다.

절차 2~3. 각 유스케이스의 시나리오, 즉 이벤트의

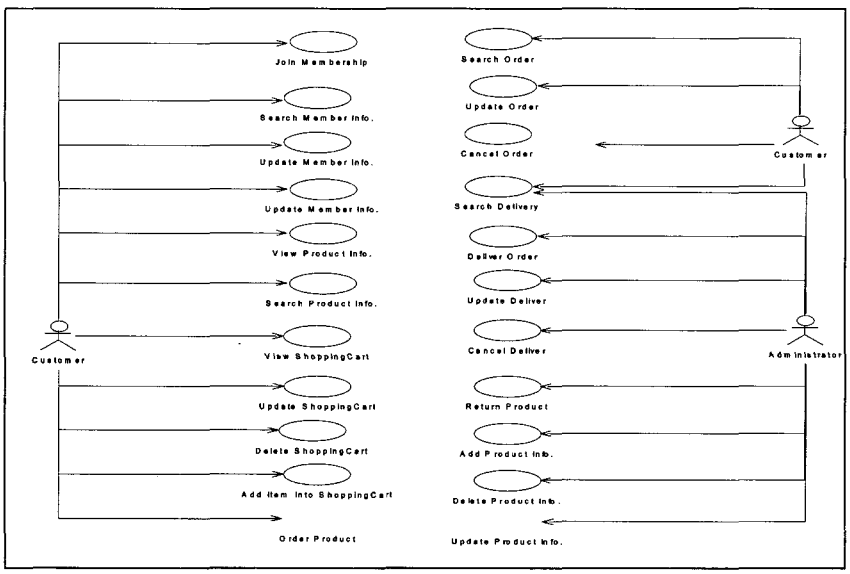


그림 3 유스케이스 다이어그램

흐름에 따라 RUP의 분석 프로세스인 유스케이스 실체화 과정을 통해서 객체를 도출한 후 도출된 객체를 중심으로 시퀀스 다이어그램과 클래스 다이어그램을 그림 4, 5와 같이 완성한다.

절차 4. 추출된 유스케이스 중 포함관계(include)와 확장관계(extend)의 유스케이스를 그룹화하고 이 유스케이스들 중에서 시스템 서비스 관점에서 유사한 기능을 가진 유스케이스와 동일한 객체를 참조하는 정도 그리고 참여 액터(Actor)를 고려하여 유스케이스를 그룹화 한다.

그림 4에서 제시한 클래스 다이어그램의 클래스 중 Customer, Shopping Cart, Product, Order, Payment, Delivery를 중심으로 분석한다. 표 3에서 제시한 유스케이스와 클래스 간의 관계에서 Shopping Cart를 처리하기 위한 유스케이스들은 Shopping Cart 클래스와 Product클래스를 참조하지만 이들은 주문을 하기 위해

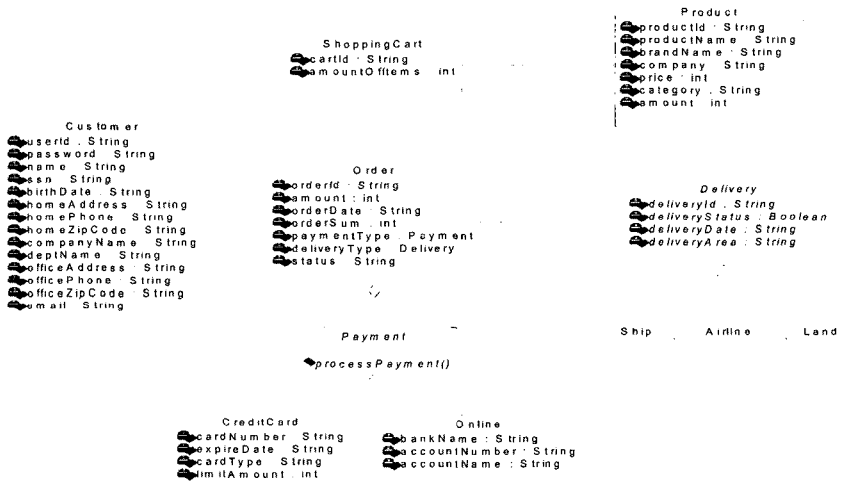


그림 4 클래스 다이어그램

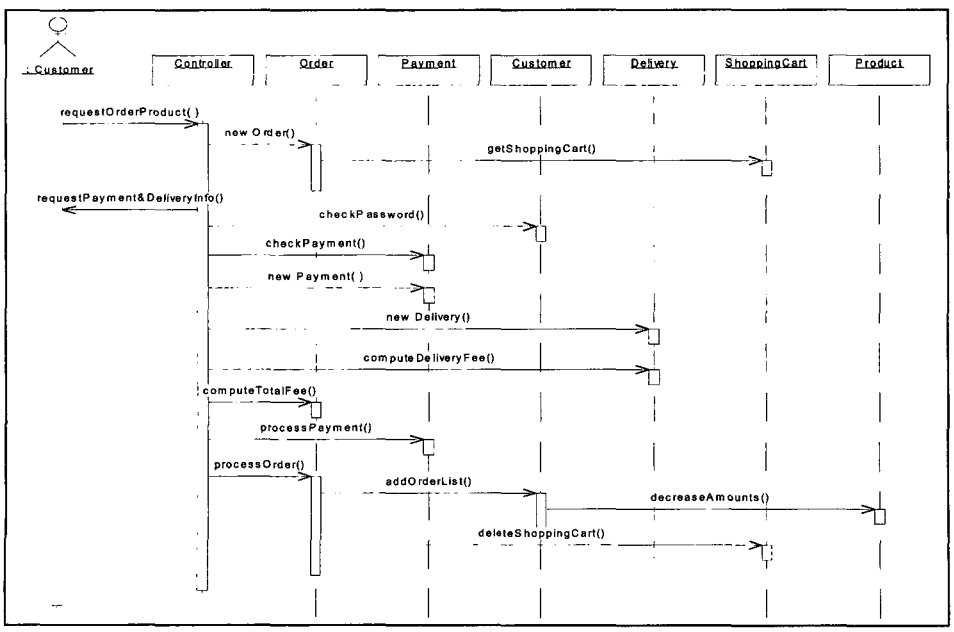


그림 5 시퀀스 다이어그램

표 3 유스케이스 다이어그램과 클래스 관계 테이블

시스템 컴포넌트	유스케이스	customer	Order	Payment	Delivery	Shopping Cart	Product
회원관리	Join Membership	V					
	Delete Membership	V					
	Update Membership	V					
	Search Membership	V					
주문관리	Order Product	V	V	V	V	V	V
	Cancel Order	V	V	V	V	V	V
	Update Order	V	V	V	V	V	V
	Search Order	V	V	V	V		
	Search Delivery	V	V	V	V		
	Deliver Order	V	V		V		
	Update Deliver	V	V	V	V		
	Cancel Deliver	V	V	V	V		
	Return Product	V	V				V
	View ShoppingCart					V	
	Delete ShoppingCart					V	
	Update ShoppingCart					V	V
	Add Item to Shopping					V	V
상품관리	Add Product						V
	Update Product						V
	Delete Product						V
	View Product						V
	Search Product						V

서 처리되는 기능들이므로 당연히 주문관리 시스템에 포함되어야 한다. 따라서 식별된 후보 시스템 컴포넌트는 표 4에서 제시했듯이 회원관리, 주문관리, 상품관리다. 또한 식별된 후보 시스템 컴포넌트가 포함하는 클래스들을 보면 시스템 컴포넌트 사이에서 Customer 클래스와 Product 클래스가 공유되어진다. 즉, 회원관리 시스템과 주문관리 시스템 사이에 Customer 클래스가 중복되었고, 주문관리 시스템과 상품관리 시스템 사이에 Product 클래스가 중복되었다. 따라서 식별된 시스템 컴포넌트와 포함된 클래스는 다음 표 4와 같다.

표 4 식별된 후보 시스템 컴포넌트와 포함된 클래스

식별된 시스템 컴포넌트	포함된 클래스
회원관리	Customer
주문관리	Customer, Order, Payment Delivery, Shopping Cart, Product
상품관리	Product

절차 5. 절차 4에 의해서 공통 클래스가 존재하므로 시스템 컴포넌트 사이에 존재하는 공통 클래스가 어느 시스템 컴포넌트에 포함되어야 할지 결정해야 한다. 따라서 공통 클래스 배치는 3.3절에서 제시한 적용 규칙을 적용하여 Customer 클래스와 Product 클래스가 어느 시스템 컴포넌트에 포함되어져야 하는지를 결정한다. 식별된 시스템 컴포넌트와 클래스, 클래스들 간의 관계 그리고 유스케이스와 클래스 간의 CRWD매트릭스 테이블을 보면 3.3 절의 표 1에서 제시한 적용 규칙 1,2,3,5에는 해당되지 않고 규칙 4에 해당한다. 즉, Customer는 회원관리 시스템에서 생성되고 Product는 상품관리 시스템에서 생성되므로 Customer는 회원관리에 포함되고 Product는 상품관리에 포함된다.

따라서 결정된 시스템 컴포넌트와 포함된 클래스는 다음 표 6에서 제시되어진다.

이처럼 식별된 시스템 컴포넌트들과 각 시스템 컴포넌트에 포함된 클래스들을 다이어그램으로 표현한 것이 그림 6에 제시되어 있다.

절차 6. 시스템 컴포넌트가 식별되었으므로 시스템 컴포넌트 단위로 비즈니스 컴포넌트를 식별한다. 회원관리와 상품관리 시스템을 보면 포함된 클래스가 각각 하나이고 그 자체가 비즈니스 컴포넌트이므로 다시 비즈니스 컴포넌트 식별을 고려할 필요가 없고 주문관리 시스템만 비즈니스 컴포넌트를 식별하면 된다. 이 프로세스는 3.1절과 3.2절에 의해 시스템 컴포넌트 내의 클래스들 중에 포함관계와 상속관계의 클래스들은 강한 종속관계가 존재함으로 하나의 클래스로 그룹화 한다. 따라서 Order 클래스는 Payment 클래스와 Delivery 클래스를 포함하고 있으므로 그들 클래스를 하나의 클래스로 그룹화하고 그 클래스를 Order라고 정의한다. 따

표 5 유스케이와 참조하는 클래스와의 CRWD관계 테이블

시스템 컴포넌트	유스케이스	Customer	Order	Payment	Delivery	Shopping Cart	Product
회원관리	Join Membership	C					
	Delete Membership	D					
	Update Membership	W					
	Search Membership	R					
주문관리	Order Product	R	C	C	C	R	W
	Cancel Order	R	D	D	D	D	W
	Update Order	R	W	W	W	W	W
	Search Order	R	R	R	R		
	Search Delivery	R	R	R	R		
	Deliver Order	R	R		W		
	Update Deliver	R	W	W	W		
	Cancel Deliver	R	W	W	D		
	Return Product	R	W				W
	View ShoppingCart					R	
	Delete ShoppingCart					D	
	Update ShoppingCart					W	R
	Add Item to Shopping					C	R
상품관리	Add Product						C
	Update Product						W
	Delete Product						D
	View Product						R
	Search Product						R

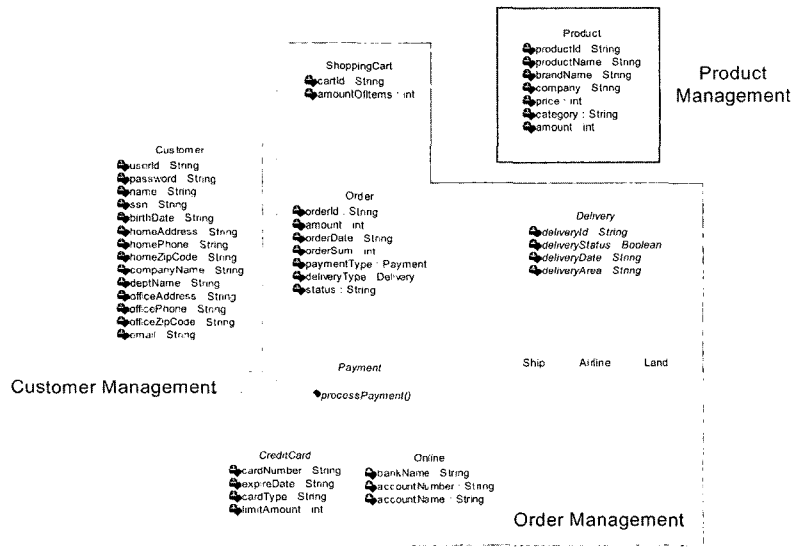


그림 6 식별된 시스템 컴포넌트 다이어그램

표 6 식별된 시스템 컴포넌트와 포함된 클래스

식별된 시스템 컴포넌트	공통클래스 배치에 의해서 재 배열된 클래스
회원관리	Customer
주문관리	Order, Payment, Delivery, Shopping Cart
상품관리	Product

라서 Order 클래스 = Order 클래스 + Payment 클래스 + Delivery 클래스이다. 그러면 고려해야 할 클래스는 Order 클래스와 Shopping Cart클래스이다.

절차 7. 절차 7은 3.1절과 3.2절에 의해 비즈니스 컴포넌트를 식별하기 위해서 시퀀스 다이어그램을 참조하여 시스템 컴포넌트 내 클래스들 사이의 메시지 호출관

게 그래프를 정의해야 한다. 따라서 비즈니스 컴포넌트를 식별하기 위해서 고려해야 할 클래스는Order 클래스(Order Class + Payment Class +Delivery Class)와 Shopping Cart 클래스이므로 그들 사이에 메시지 호출 관계 그래프를 그림 5의 시퀀스 다이어그램을 참조하여 정의하면 다음 그림 7과 같다.

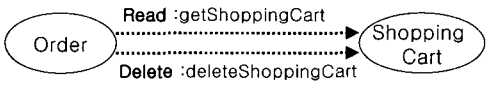


그림 7 클래스 간에 메시지 호출관계 그래프

절차 8~9. 3.1과 3.2절에 의하여 클래스들 간에 메시지 호출 관계에서 생성과 삭제 관계인 클래스들은 하나의 클래스로 그룹화하여 후보 비즈니스 컴포넌트로 정의한다. 따라서 절차 7의 메시지 호출 관계 그래프에 의해서 Order 클래스와 Shopping Cart 클래스가 삭제 관계이기 때문에 그룹화 되어 후보 비즈니스 컴포넌트로 도출된다. 그리고 다시 정제할 부가적인 부분이 없기 때문에 완전한 비즈니스 컴포넌트로 식별된다. 따라서 절차 1에서 절차 9까지의 과정에 의하여 전자상거래 도메인에서 식별된 시스템 컴포넌트와 비즈니스 컴포넌트는 다음 표 7과 같다.

표 7 식별된 시스템 컴포넌트와 비즈니스 컴포넌트 그리고 포함된 클래스

식별된 시스템 컴포넌트	식별된 비즈니스 컴포넌트	포함된 클래스
회원관리	Customer	Customer
주문관리	Order	Order, Payment, Delivery, Shopping Cart
상품관리	Product	Product

따라서 본 논문에서 제안한 방법과 절차에 의해서 식별하게 되면 서브시스템의 단위가 되는 시스템 컴포넌트가 식별되고 시스템 컴포넌트 내부에 포함되어져야 할 비즈니스 컴포넌트가 자연스럽게 식별된다. 여기서 각 서브 시스템 단위로 개발할 경우에 그대로 적용하면 되고 다른 시스템을 개발할 경우에는 식별된 비즈니스 컴포넌트를 조립하여 구축하면 된다. 본 논문에서 사례로 제시하고 있는 전자상거래 시스템을 비즈니스 컴포넌트를 기반으로 구축한다면 전자상거래 시스템의 컴포넌트 아키텍처와 컴포넌트 인터페이스는 다음 그림 8과 그림 9와 같다.

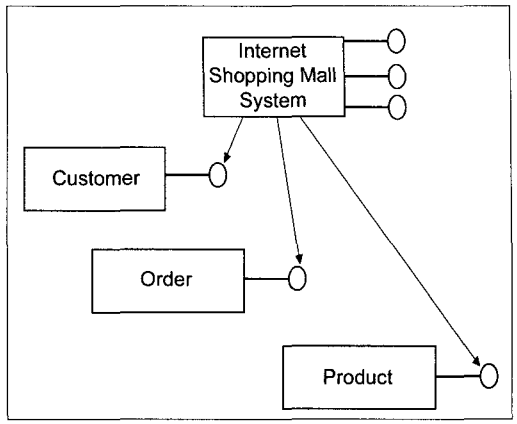


그림 8 시스템 컴포넌트 아키텍처

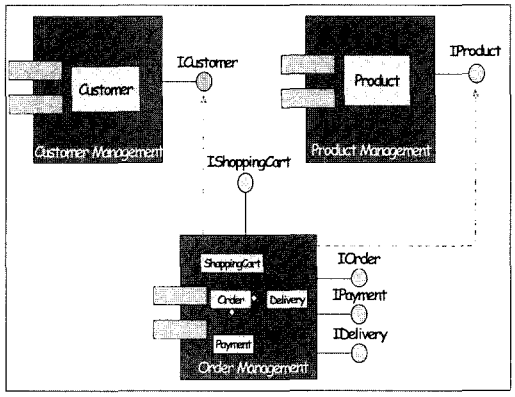


그림 9 컴포넌트 인터페이스

4.2 분석 및 평가

본 논문이 제안한 시스템 컴포넌트 기반의 비즈니스 컴포넌트 식별 방법이 유용하다는 것을 증명하기 위해서 CRWD인자를 컴포넌트 식별에 적용하였을 경우와 적용하지 않았을 경우로 나누어 결합도를 평가하였다. 클래스 간의 수평적 관계인 연관관계에서 메시지 호출 유형에 따라서 클래스 간의 의존의 정도가 달라지므로 메시지 호출 유형인 생성, 삭제 > 수정 > 조회 순으로 가중치의 크기를 달리하고 또한 시퀀스 다이어그램에서 제시된 클래스들 사이의 메시지 호출 방향과 수를 고려하여 결합도를 측정하였다.

다음 표 8은 클래스들 사이의 메시지 유형과 개수 그리고 메소드 호출 방향을 제시하였다.

표 8 클래스 간의 메시지 호출 타입과 호출 수

열Class / 행Class	Customer(A)	Order(B)	Shopping Cart(C)	Product(D)
Customer(A)				
Order(B)	Read(9)		Read(1), Delete(1)	Write(3)
Shopping Cart(C)		Write(1)		Read(2)
Product(D)				

메시지 호출 방향은 행 클래스→열 클래스이고 셀 내부는 메시지 호출 유형(개수)를 표시하였다. 즉 예를 들면 표 8의 3행 2열에서 메시지 호출 방향은Order→Customer이고 메시지 호출 유형(개수)는 Read(9)를 의미한다.

다음 표 9는 가중치를 적용하지 않고 메소드 호출 방향과 개수 만을 고려하여 결합도를 측정한 결과 표이다. A 클래스는 Customer로 B클래스는 Order로 C 클래스는 Shopping Cart로 D 클래스는 Product 클래스를 의미한다.

표 9 가중치를 적용하지 않은 경우의 결합도

클래스	A↔B	B↔C	C↔D	B↔D
메시지 호출 수	9	3	2	3
결합도	9	3	2	3

표 9의 결과는 클래스 간에 메시지 호출 수가 가장 많은 A클래스와 B클래스 간에 결합도가 가장 높게 나왔으므로 표 9의 결과에 의하여 A클래스와 B 클래스가 합쳐진 비즈니스 컴포넌트가 식별되어야 타당한 것이다. 그러나 그 결과는 의미적으로 우리의 직관과 일치하지 않는다.

다음 표 10은 메소드 호출 방향과 개수 그리고 메소드 호출 유형에 따른 가중치를 적용하여 결합도를 측정한 결과 표이다.

표 10 가중치를 적용한 경우의 결합도

클래스	A↔B	B↔C	C↔D	B↔D
메시지 호출 수	9	3	2	3
결합도	9	24	2	9

가중치 : Create, Delete(20) > Write(3) > Read(1)를 부여하여 측정하였다.

표 10의 결과는 메시지 호출 유형에 따른 클래스 간의 의존도를 가중치로 부여하여 측정하였으므로 클래스 간에 메시지 호출 수가 적다고 해도 B 클래스와 C 클래스 사이의 결합도가 가장 높게 나왔다. 표 10의 결과에 의하여 B클래스와 C 클래스가 합쳐진 비즈니스 컴포넌트가 식별되어야 타당한 것이다. 그리고 그 결과는 의미적으로 우리의 직관과 일치한다. 다음은 클래스들을 조합하여 비즈니스 컴포넌트로 정의하고 정의된 비즈니스 컴포넌트들을 조합하여 가능한 모든 경우의 후보 시스템 컴포넌트를 도출 하였다.

도출된 후보 시스템 컴포넌트들을 중심으로 표9와 10에서 제시한 결합도를 적용하여 표 11에서 평균 결합도를 측정하였다.

표 11 후보 시스템 컴포넌트의 평균 결합도

후보 시스템 컴포넌트	가중치를 적용하지 않았을 경우 : 평균 결합도	가중치를 적용하였을 경우 : 평균 결합도
S1(A, B, C, D)	**4.3 (17/4)**	11 (44/4)
S2(AB, C, D)	2.67 (8/3)	8.67 (26/3)
S3(A, BC, D)	4.67 (14/3)	**6.67 (20/3)**
S4(A, B, CD)	5 (15/3)	14 (42/3)
S5(AB, CD)	3 (6/2)	16.5 (33/2)
S6(ABC, D)	2.5 (5/2)	4.5 (9/2)
S7(A, BCD)	4.5 (9/2)	4.5 (9/2)

S1(B1,…,Bn)은 후보시스템 컴포넌트1(비즈니스 컴포넌트1,.., 비즈니스 컴포넌트n)을 의미한다. 시스템 컴포넌트의 평균 결합도인 ACSC(Average Coupling of System Component)는 비즈니스 컴포넌트 간의 결합도(Coupling between Business Components)의 합/비즈니스 컴포넌트의 개수로 계산하였다. 따라서 비즈니스 컴포넌트 BC_i(i=1..m), BC_j(j=1..n)에 대하여 비즈니스 컴포넌트 간의 결합도를 CBC_x(x=1..u)라 할 때 시스템 컴포넌트의 평균 결합도인 ACSC(Average Coupling of System Component)를 계산하기 위한 수식은 다음과 같이 정의된다.

$$ACSC(S_p) = \frac{\sum_{x=1}^{u} CBC_x(BC_i,\ BC_j)}{u}$$

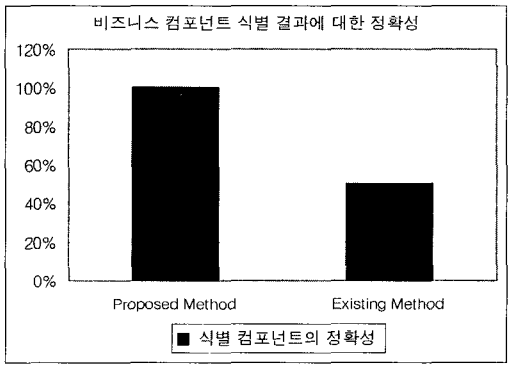
가중치를 적용하였을 경우의 평균 결합도가 가장 낮게 나온 후보 컴포넌트 시스템은 S6와 S7이나 시스템 컴포넌트 식별 단계에서 A 클래스인 Customer 클래스와 D 클래스인 Product 클래스는 이미 비즈니스 컴포넌트로 식별되었으므로 표 10에서 제시한 후보 시스템 컴포넌트들 중에서 A 클래스와 D 클래스가 비즈니스 컴포넌트로 되어 있는 S1과 S3만 고려하면 된다. B 클래스인 주문 클래스와 C 클래스인 Shopping Cart 클래스는 당연히 주문 시스템에서 서로 독립적으로 존재할 수 없는 밀접한 관계를 유지하고 있다. 즉 주문은 쇼핑카트에 담아져 있는 정보를 통해서 생성되고 수정되어지므로 의미적으로 판단해 볼 때 당연히 주문 클래스와 쇼핑카트는 떨어질 수 없는 관계이다. 표 11의 결과에서 메시지 호출 수로만 평균 결합도를 측정하였을 경우 B 클래스와 C 클래스가 떨어진 비즈니스 컴포넌트들의 조합으로 만들어진S1 시스템이 낮게 나와 우리의 직관과 일치하지 않는 결과가 도출됨을 확인하였다. 또한 가중치를 적용하였을 경우는 B클래스와 C 클래스가 합해져서 비즈니스 컴포넌트로 도출된 S3 시스템이 평균 결합도가 적게 나와 우리의 직관과 일치하는 결과가 도출됨을 확인하였다. 따라서 클래스와 클래스 간의 메시지 호출이 생성이나 삭제의 유형이라면 메시지 호출 수가 1

이라 하더라도 강한 연관관계를 가지고 있으므로 그룹화 되어야 하는데 기존의 컴포넌트 식별 방법은 클래스와 클래스 간의 연관 관계를 단순히 클래스 간의 메시지 호출 수에 의해서만 결정하므로 측정해 본 결과 비즈니스 컴포넌트가 올바르게 식별될 수 없다는 것을 알 수 있었다. 또한 관련연구에서 제시했듯이 기존의 컴포넌트 개발 방법론인 [7]의 방법은 비즈니스 컴포넌트를 식별하기 위한 기준을 설정하였다 하더라도 클래스 간의 메시지 호출 수 만을 고려하여 그룹화 하였으므로 올바르게 비즈니스 컴포넌트가 식별되지 않음을 알 수 있다. 또한 마르미 III의 UDA(Usecase Data Access) 테이블을 통한 컴포넌트 식별 방법과 [8]의 방법도 CRWD 인자를 적용하였다 하더라도 클래스와 클래스 간의 메시지 호출 유형과 방향을 고려하지 않고 유스케이스가 기능을 실행하기 위해서 어느 클래스를 어떻게 사용하는지 만 고려했기 때문에 올바르게 비즈니스 컴포넌트를 식별할 수 없음을 알 수 있다. Advisor와 CBD 96 그리고 마르미 III의 객체 모형을 통한 컴포넌트 식별 방법은 핵심 클래스를 중심으로 상호 관련이 많은 주변 클래스를 그룹화하여 비즈니스 컴포넌트를 식별한다고 정의하였지 명확한 기준을 정의하지 않아서 대다수 개발자의 직관과 경험에 의존한다.

또한 시스템 컴포넌트는 비즈니스 컴포넌트들을 포함하고 있고 비즈니스 컴포넌트들의 상호작용에 의해서 수행된다. 따라서 시스템 컴포넌트와 비즈니스 컴포넌트를 연관 지어서 식별하는 것은 필연적인 것이다. 그러나 기존의 방법론들은 전체 도메인을 중심으로 비즈니스 컴포넌트만을 식별하는 방법에만 중점을 두고 있고 Advisor 방법 같이 시스템 컴포넌트가 정의되어 있다고 해도 시스템 컴포넌트를 식별하는 방법과 기준이 제시되어 있지 않고 시스템 컴포넌트와 비즈니스 컴포넌트를 분리해서 식별하기 때문에 비즈니스 컴포넌트를 조합하여 시스템 컴포넌트를 완성하기 위해서는 다시 시스템 컴포넌트의 인터페이스인 각 유스케이스를 분석하여 어떠한 클래스가 어느 시스템 컴포넌트에 포함될 것인지를 결정해야 한다. 또한 그 클래스는 어느 비즈니스 컴포넌트로 포함되어야 하는지를 다시 한번 고려하여야 한다. 그러나 본 논문은 시스템 컴포넌트를 기반으로 비즈니스 컴포넌트를 식별하기 때문에 이러한 문제점은 존재하지 않고 시스템 컴포넌트를 식별하기 위한 방법과 기준을 본 논문에서 제시하였으므로 시스템 컴포넌트를 효율적으로 식별할 수 있다. 식별된 시스템 컴포넌트 사이에 공통 클래스가 존재하지 않으므로 시스템 컴포넌트를 기반으로 비즈니스 컴포넌트를 식별할 수 있다. 또한 비즈니스 컴포넌트를 식별하기 위해서 전체 시스템을 서브 시스템 단위인 시스템 컴포넌트로 분리하고 시스템 컴포넌트 안에 포함된 클래스들 중에 상속관계와 포함관계의 클래스들은 컴포넌트의 특성상 분리되어 질 수 없는 관계이므로 그러한 클래스를 그룹화하여 하나의 클래스로 간주하여 비즈니스 컴포넌트를 식별하기 때문에 식별해야 할 클래스의 개수가 줄어들어 개발자의 노력과 시간을 절감할 수 있다.

다음은 표 11에서 제시한 결합도를 기준으로 본 논문에서 제안한 방법인 메소드 호출방향과 호출유형을 적용한 비즈니스 컴포넌트 식별 결과와 메소드 호출방향과 호출유형을 적용하지 않은 기존의 방법에 대한 비즈니스 컴포넌트 식별 결과에 대한 정확성을 비교한 결과를 다음 표 12에서 제시하였다.

표 12 비즈니스 컴포넌트 식별 결과에 대한 정확성



다음은 사례에서 제시한 전자상거래 도메인을 기준으로 비즈니스 컴포넌트를 식별하기 위해서 고려해야 할 최대 개수를 본 논문에서 제안한 방법과 기존의 방법과의 비교 결과를 다음 표 13에서 제시하였다. 표 13에서 제시하듯이 전체 클래스를 기준으로 비즈니스 컴포넌트를 식별하는 기존의 방법 보다 시스템 컴포넌트를 기반으로 비즈니스 컴포넌트를 식별하면 고려해야 할 클래스의 개수는 50% 정도 감소되는 것을 확인할 수 있다.

표 13 비즈니스 컴포넌트 식별을 위해서 고려해야 할 클래스의 최대 개수

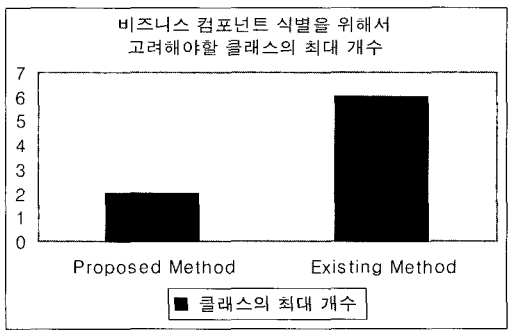


표 14 컴포넌트 식별 방법 비교 평가

비교항목 \ 식별방법	CBD 96	Advisor	RUP	MARMI III	[7]	[8]	Our Method
시스템 컴포넌트 식별	1	2	2	1	1	2	4
비즈니스 컴포넌트 식별	4	4	1	4	4	4	4
시스템 컴포넌트 기반의 비즈니스 컴포넌트 식별	1	1	1	1	1	2	4
클래스 간에 메소드 호출유형 적용	1	1	1	1	1	1	4
클래스 간에 메소드 호출방향 적용	1	1	1	1	1	1	4
클래스 간의 구조적 관계의 특성 적용	3	3	2	3	3	4	4
유스케이스와 클래스 간의 관계 적용	1	1	1	4	3	4	4
유스케이스와 클래스 간의 메소드 유형 적용	1	1	1	4	1	4	4
합계	**13**	**14**	**10**	**19**	**15**	**22**	**32**

다음은 컴포넌트를 식별하기 위해서 적용한 방법을 기준으로 기존의 방법론들과 비교평가한 결과를 다음 표 14에서 제시하였다.

표 14의 평가를 위한 기준은 다음과 같다.

1: 방법을 사용하지 않음

2: 방법을 사용하였지만 정의되지 않음

3: 방법이 사용되었고 정의 되었지만 상세 절차와 기준이 미비

4: 방법이 사용.정의 되었고 상세 절차와 기준도 제시

5. 결 론

본 논문에서 제안하는 컴포넌트 식별 방법은 전체 도메인을 중심으로 비즈니스 컴포넌트를 식별하는 것이 아니라, 서브 시스템의 단위가 되는 시스템 컴포넌트를 식별하고 식별된 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 식별하였다. 기존 방법론들의 시스템 컴포넌트 식별이 쉽지 않은 문제점을 보완하여 시스템 컴포넌트를 식별하기 위한 상세 지침과 절차를 제시하였다. 또한 개발자의 직관과 경험에 의존하는 비즈니스 컴포넌트 식별 방법을 보완하여 분석 클래스 간에 정적.동적 특성을 적용하여 비즈니스 컴포넌트를 식별하는 지침을 제시하였고 분석 및 평가 단계를 통해서 그 효과성을 검증하였다. 우리는 실제로 본 논문에서 제안하는 방법을 토대로 병원관리 시스템, 뱅크 시스템, 경매 시스템, 주문 시스템에 적용해서 분석한 결과 우리의 직관과 일치하는 결과가 도출됨을 확인할 수 있었다.

향후 연구과제는 컴포넌트를 효과적으로 식별할 수 있는 좀 더 상세한 특징을 찾아내어 컴포넌트 개발 방법론에 적용하는 것이다.

참 고 문 헌

[1] Desmond Francis Dsouza and Alan Cameran wills, Objects, Component, and Frameworks with UML: the Catalysis approach , Addison Wesley, 1999.

[2] Compuware, About Uniface, http://www.compuware.com/products/uniface/about.htm, 2001.

[3] Computer Associates, COOL:Joe 2.0, http:/www.-cai.com/products/cool/joe/cooljoe_pd.pdf, 2001.

[5] John Cheesman and John Daniels, UML Components: A Simple Process for Specifying Component-Based Software, Addison-Wesley, 2001.

[5] Ivar Jacobson, Grady Booch and James Rumbaugh, The Unified Software Development Process, Addison Wesley, 1999.

[6] ETRI, 컴포넌트 개발방법론 마르미 III, Technical Report, 2002.

[7] Hyung Ho Kim and Doo Whan Bae, Component Identification via Concept Analysis, Journal of Object Oriented Programming, 2001.

[8] Jong Kook Lee, Seung Jae Jung and Soo Dong Kim, Component Identification Method with Coupling and Cohesion, Proceedings of Asia Pacific Software Engineering Conference, pp. 79-88, 2001.

[9] Eun Sook Cho, Soo Dong Kim and Sung Yul Rhew, A Domain Analysis and Modeling Methodology for Component Development, International Journal of Software Engineering and Knowledge Engineering, Vol.14, No.2, April, 2004.

[10] David C. Kung, Jerry. Gao, Pei Hsia, F. Wem, Y. Toyoshima and C. Chen, Change Impact Identification in Object Oriented Software Maintenance, Proceedings International Technical Conference on Ciecuit/Systems, Computers and Communications, 1999.

[11] David C. Kung, Jerry Gao and Pei Hsia, Class Firewall, Test Order, and Regression Testing of Object-Oriented Programs, Journal of Object-Oriented Programming, pp. 51-65, 1995.
[12] Henderson-Sellers, Brian, Object-Oriented Metrics, Prentice-Hall, 1996.
[13] Clemens Szyperski, Dominik Gruntz and Stephan Murer, Component Software: Beyond ObjectOriented Programming, 2nd Edition, AddisonWesley, 2002.
[14] George T. Heineman and William T. Council, Component Based Software Engineering : Putting the Pieces Together, Addison-Wesley, 2001.
[15] S.R. Chidamber and C.F. Kemerer, A Metric Suite for Object-Oriented Design, IEEE Transactions on Software Engineering, vol. 17, No. 6, pp. 636-638, 1994.

최 미 숙

1990년 전북대학교 졸업(이학사). 1994년 숙명여자대학교 일반대학원 컴퓨터과학과(이학석사). 2002년 숙명여자대학교 일반대학원 컴퓨터과학과(이학박사). 1995년~1999년 나주대학 소프트웨어 개발과 전임교수. 2004년~현재 우석대학교 컴퓨터공학과 초빙교수. 관심분야는 소프트웨어 개발 방법론, CBD 방법론, 소프트웨어 아키텍쳐

조 은 숙

1993년 동의대학교 자연과학대학 전산통계학과 졸업(이학사). 1996년 숭실대학교 공과대학 컴퓨터학과 졸업(공학석사). 2000년 숭실대학교 공과대학 컴퓨터학과 졸업(공학박사). 2002년~2003년 한국전자통신연구원 초빙연구원. 2000년~현재 동덕여자대학교 정보학부 강의전임교수. 관심분야는 소프트웨어모델링, CBD방법론, 소프트웨어 아키텍처, 웹서비스 컴퓨팅

박 재 년

1966년 고려대학교 학사. 1969년 고려대학교 석사. 1981년 고려대학교 박사 1979년~1983년 전남대학교 전산통계학과 교수. 1983년~현재 숙명여자대학교 정보과학부 컴퓨터과학전공교수. 관심분야는 소프트웨어개발방법론, 소프트웨어 품질평가, 컴포넌트기반 소프트웨어공학

하 종 성

1984년 서울대학교 컴퓨터공학과(학사) 1986년 한국과학기술원 전산학과(석사) 1996년 한국과학기술원 전산학과(박사) 1986년~1989년 (주)현대전자산업 근무 1990년~현재 우석대학교 컴퓨터공학과 교수. 2001년 미국 조지워싱턴대학교 방문교수. 관심분야는 응용계산기하학, 컴퓨터그래픽스, CAD/CAM 등임