RUP 기반의 컴포넌트 식별 방법에 관한 연구

최 미 숙[†] · 윤 용 익[††] · 박 재 년[†††]

요 약

사용자의 요구사항 변경에 따른 반영, 빠른 시스템 구축, 유지 보수 단계의 효율적인 시스템 관리, 소프트웨어의 수정 용이성, 저렴한 비용 등은 컴포넌트 기반 시스템 구축이 필수적인 사항으로 여겨지고 있다. 이러한 컴포넌트 기반 시스템 구축을 위한 기존의 컴포넌트 개발 방법론은 컴포넌트 식별을 위하여 객체를 추출하는 부분이 비 효율적이고 시스템 컴포넌트를 추출하기 위한 방법이 제시되어 있지 않다. 또한 시스템의 전체 도메인을 중심으로 비즈니스 컴포넌트 식별을 위한 절차와 방법만을 제시하고 있다. 그리고 컴포넌트 식별을 위하여 대부분 개발자의 직관과 경험에 의존하는 문제점이 있다. 따라서 본 논문에서는 객체를 추출하는 비 효율적인 부분을 개선하기 위하여 요구 사항 분석단계부터 객체를 추출하는 단계까지 RUP(Rational Unified Process)를 적용한다. 또한 시스템 컴포넌트를 식별하기 위한 방법과 절차를 제안하고 시스템의 전체 도메인을 중심으로 비즈니스 컴포넌트를 식별하는 것이 아니라 추출된 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 추출한다. 개발자의 직관과 경험에 의존하여 컴포넌트를 식별하는 문제점을 보완하기 위하여 응집척도와 결합척도를 제안하고 적용한다. 본 논문에서 제안하는 컴포넌트 식별 방법은 객체 식별의 용이성, 컴포넌트의 기능적 재사용성, 추적성 그리고 컴포넌트의 독립성을 중심으로 좀 더 효율적으로 컴포넌트를 식별한다.

Study about Component Identification Method Based On RUP

Mi-Sook Choi[†] · Yong-Ik Yoon[††] · Jai-Nyun Park[†††]

ABSTRACT

We need a component-based system to reflect software changes in user's requirements, to implement a system at a rapid speed as well as to efficiently manage the system in a maintenance phase and to easily change software. Moreover, the component-based system has a merit in development cost. However, existing component development methodology for implement of component-based system is inefficient in object identification for component identification. Moreover, the existing component development methodology also fails to provide any method to identify system component. It merely provides procedures and methods to identify business component focused on a whole system domain. In addition, it has another problem that it considerably relies on developer's experiences and intuitions for component identification. Therefore, according to this paper, RUP (Rational Unified Process) is applied from a requirement analysis phase to an object identification phase in order to improve the inefficiency of object identification. In addition, this paper provides procedures and methods for system component identification, and identifies business components based on the identified system component, rather than on the whole system domain. This paper also provides and applies cohesion metric and coupling metric so as to overcome the problem that component identification depends on developer's intuitions and experiences. Accordingly, the component identification method proposed in this paper, may identify components more effectively based on facility of object identification, functional reusability of components, traceability, and independence of components.

키워드 : 컴포넌트 식별(Component Identification), 응집척도(Cohesion Metric), 결합척도(Coupling Metric), 추적성(Traceability), 재사용(Reusability), 독립성(Independence)

1. 서 론

오늘날 정보통신의 발달과 전자상거래가 일반화되어 가는 시점에서 직접 기업과 개인간의 전자상거래가 이루어지는 광역적 시장을 형성했다. 시간에 따른 급속한 경쟁이 이루어지면서 기업 이윤의 확대와 위험요소는 증가되었고 다루어야 할 시스템들은 더욱 복잡해졌다. 이렇게 환경이 복잡하고 빠르게 변화함에 따라서 구축될 시스템도 그러한 환경에 대처해야 하므로 제한된 시간에 저렴한 비용으로 원하는 시스템을 구축하는 것이 필수적이다. 또한 사용자의 요구사항이 새롭게 변경될 때마다 소프트웨어에 그 변경이 반영되어야 하며 이러한 변경이 소프트웨어에 반영하는 데는 많은 비용이 소요된다. 따라서 사용자의 요구사항 변경에 따른 반영, 빠른 시스템 구축, 유지 보수 단계의 효율적인 시스템 관리, 소프트웨어의 수정 용이성, 저렴한 비용 등은 컴포넌트 기반 시스템 구축이 필수적이 되

※ 본 논문은 2001년도 숙명여대 교비연구비에 의해서 지원되었음.
† 정 회 원 : 숙명여자대학교 대학원 전산학과
†† 종신회원 : 숙명여자대학교 정보과학부 교수
††† 정 회 원 : 숙명여자대학교 정보과학부 교수
논문접수 : 2001년 8월 20일, 심사완료 : 2001년 12월 12일

도록 한다.

컴포넌트란 한 단위로써 독립적으로 개발, 배포되어 질 수 있고 요구되는 시스템 구성을 위해서 다른 컴포넌트와 연결되어질 수 있는 소프트웨어의 응집력 있는 패키지이고 일정한 기능을 수행할 수 있는 실체화의 단위이어야 한다[2]. 이러한 컴포넌트 기반 시스템 구축을 위하여 가장 중요한 작업은 소프트웨어의 수정이 필요할 경우 수정 영향의 범위를 줄이고 효율적으로 수정이 이루어질 수 있도록 컴포넌트간의 의존성이 적은 독립적인 컴포넌트를 어떻게 추출할 것인가? 컴포넌트는 일정한 기능을 수행할 수 있는 실체화의 단위이어야 하므로 기능적 재 사용성이 가능한 컴포넌트를 어떻게 추출하여야 하는가?, 요구사항 변경의 용이성을 위하여 요구사항 분석단계의 산출물부터 설계단계의 산출물들간의 추적성이 가능한가? 컴포넌트를 식별하기 위한 방법이 효율적인가? 이다.

따라서 본 논문에서는 위에서 제시한 중요한 작업을 중심으로 기존 컴포넌트 개발 방법론들의 컴포넌트 식별 방법과 그들의 문제점을 제시하고 기존의 식별 방법의 문제점을 보완한 확장된 컴포넌트 식별 방법을 제안한다. 본 논문에서 제안하는 식별 방법은 컴포넌트를 구성하는 객체 식별의 용이성, 컴포넌트의 기능적 재사용성, 추적성, 독립성에 중점을 두고 이루어지며 기존의 방법을 보완하여 좀 더 효율적으로 컴포넌트를 식별할 수 있도록 한다.

2. 관련 연구

2.1 컴포넌트의 식별 방법과 문제점

최근에 산업계에서 널리 쓰이고 있는 대표적인 컴포넌트 기반 소프트웨어 개발 방법론은 Rational의 RUP(Rational Unified Process)[3], Computer Associates의 CBD96[4, 7], Compuware의UNIFACE[20], Castek의 CBD/e[21, 22], MTW사의 Progression[23]등이 있다. 또한 Cheesman과 Daniels가 CBD96의 컴포넌트 개발 방법을 확장하여 제안한 UML Components 방법[1]이 있다.

현재 널리 쓰이고 있는 컴포넌트 개발 방법론인 **Computer Associates의 CBD96은** 컴포넌트를 식별하기 위하여 절차와 지침이 다른 방법들에 비해서 좀 더 체계적이고 구체적으로 제시되고 있는 장점이 있지만 몇 가지의 문제점이 존재한다. 그 문제점은 첫째, 요구사항을 파악하기 위하여 현업 담당자가 타입 다이어그램을 그린다. 그러나 객체 다이어그램과 유사한 다이어그램으로서 업무를 파악할 때 의미 있는 개념을 추출하고 그들 사이의 연관관계를 나타냄으로 타입 다이어그램을 그리는데 시스템적 개념이 없는 현업 담당자는 어떠한 단어가 의미가 있는지 없는지 구별하기가 쉽지 않고 그들 사이에 관계성을 도출하기가 쉽지 않다. 둘째, 비즈니스 타입 모델은 시스템 분석가가 현업 담당자가 그린 타입 모델을 보고 시스템 측면에서 불필요한 것, 중복되는 것, 모호한 것, 의미가 없는 것을 제거하고 다시 의미가 있는 객체를 추가하여 비즈니스 타입 모델을 만든다. 그러나 비즈니스 타입 모델의 객체를 도출할 때 객체 추출의 구체적인 지침 없이 도출한다는 것은 경험이 많은 시스템 분석가의 직관에 의해서 객체를 추출해야 하는 모순이 있다. 셋째, 요구사항 변경시 요구사항 분석 단계부터 체계적으로 추적이 가능하지 않다. 왜냐하면 비즈니스 타입 모델을 전체 시스템을 중심으로 도출하기 때문이다. 넷째, 시스템 서비스 측면의 기능을 재 사용할 수 있는 시스템 컴포넌트를 식별하기 위한 지침과 절차가 없고 비즈니스 컴포넌트 식별 방법만을 중심으로 컴포넌트 추출이 이루어진다. Cheesman과 Daniels가 제안한 **UML Components 방법은** CBD96의 첫째, 둘째의 문제점을 가지고 있고 시스템 서비스 측면의 기능적 재사용이 가능하도록 시스템 컴포넌트의 정의가 명확히 명시되어 있지만 이러한 시스템 컴포넌트를 추출하기 위하여 유스케이스 만을 가지고 식별하기 때문에 유스케이스에 포함될 객체들은 고려하지 않는다. 또한 시스템 컴포넌트를 추출하기 위하여 유스케이스를 어떻게 그룹화 할 것인지 명확한 기준이 정의되지 않았다. 그리고 UML Components 방법은 시스템 컴포넌트와 비즈니스 컴포넌트가 명확히 분리되어서 추출되어지기 때문에 비즈니스 컴포넌트를 조합하여 시스템 컴포넌트를 완성하기 위해서는 다시 시스템 컴포넌트에 포함된 각 유스케이스의 이벤트 흐름을 분석하여 어떠한 클래스가 포함될 것인지를 추출하고 추출된 클래스가 어느 비즈니스 컴포넌트에 포함되는지를 다시 한번 고려하여야 하는 번거로움이 있다. **RUP는** 표준화된 UML을 산출물로 정의하고 사용되는 개발 방법론이다. 그러나 CBD96에서 제시한 컴포넌트 식별 방법의 문제점 중 첫 번째, 두 번째, 세 번째 문제점을 많이 보완하고 있는 개발 방법론이다. 즉, RUP는 요구사항 분석 단계의 산출물로 유스케이스 모델을 산출하고 분석 단계에서 각 유스케이스의 이벤트 플로우를 보고 인터페이스 객체, 컨트롤 객체, 엔티티 객체 추출을 통하여 CBD96 보다는 좀더 체계적이고 효율적으로 객체를 도출해 나간다. 클래스 다이어그램을 도출할 경우에도 인터랙션 다이어그램을 통해서 효율적으로 도출한다. 또한 유사한 이벤트를 그룹화 하여 패키지 다이어그램을 도출하고 패키지 다이어그램을 중심으로 클래스 다이어그램을 산출한다. 이는 시스템을 의미적으로 분할하여 클래스 다이어그램을 도출하므로 이해하기가 쉽고 분산환경에서 전개(Deploy)할 경우에 분할이 쉬우며 요구사항 변경시 요구사항 분석단계부터 설계단계까지 각 산출물을 추적할 수가 있어 요구사항 변경을 쉽게 수용할 수 있다. 그러나 이러한 장점을 보유하고 있음에도 컴포넌트를 추출하기 위하여 널리 사용되지 못하고 있는 문제점은 컴포넌트를 식별하기 위한 방법

만 제시하고 있지 제시한 방법을 사용해서 컴포넌트를 식별하기 위한 구체적인 지침과 절차를 제시하고 있지 않으므로 시스템 분석가의 직관과 경험에 의하여 컴포넌트를 식별할 수밖에 없다. Compuware의 UNIFACE는 거의 모든 모델링을 UML의 모델과 표기법을 그대로 사용하고 있어서 CBD96이나 UML Components의 문제점을 많이 보완하고 있는 방법론이다. 또한 상위 유스케이스를 잠재적인 시스템 컴포넌트로 추출하여 각 상위 유스케이스 별로 비즈니스 컴포넌트를 추출하기 위한 모델링이 점진적으로 진행되는 특징이 있어서 각 산출물간에 추적이 가능하고 UML Components의 시스템 컴포넌트와 비즈니스 컴포넌트를 분리하여 추출함으로 일어나는 비효율적인 부분을 많이 보완하고 있는 방법론이다. 그러나 상위 유스케이스를 추출할 때 유스케이스에 포함될 객체를 고려하지 않고 있다. 따라서 추출된 상위 유스케이스 들 사이에 많은 클래스들이 중복될 경우의 문제점이 발생되고 동일한 객체를 포함하는 컴포넌트를 구현할 경우에 대한 컴포넌트 검증 부분이 부족하고 또한 상위 유스케이스 추출, 그리고 컴포넌트 패키지 다이어그램이나 컴포넌트 다이어그램을 추출하기 위한 상세한 추출 방법이나 지침이 제시되어 있지 않다[10].

3. RUP 기반의 컴포넌트 식별 방법의 제안

이 장에서는 위에서 제시한 기존 컴포넌트 식별 방법의 문제점을 보완하여 좀 더 효율적으로 컴포넌트를 식별하기 위한 컴포넌트 식별 방법의 특성, 전체 프로세스, 식별 방법 그리고 절차를 제안한다.

3.1 RUP 기반의 컴포넌트 식별 방법의 특성

컴포넌트 아키텍처는 시스템 서비스 계층과 비즈니스 서비스 계층으로 분리된다. 시스템 서비스 계층은 시스템 인터페이스와 시스템 컴포넌트를 식별하고 비즈니스 서비스 계층은 비즈니스 인터페이스와 비즈니스 컴포넌트를 식별한다. 시스템 서비스 계층의 기능을 수행하기 위해서는 비즈니스 서비스 계층의 비즈니스 컴포넌트를 통하여 실행될 수 있다[1].

다음은 본 논문이 제시하는 컴포넌트 식별 방법의 구체적인 특성을 기술한다. CBD96과 UML Components방법에서 컴포넌트의 식별을 위한 객체의 추출은 개념 모델(Concept Model)을 중심으로 이루어진다. 그리고 컴포넌트 아키텍처의 좋고 나쁨은 이 개념 모델에 많이 의존한다. 만약 개념 모델이 명확하지 않을 경우에 시스템 분석가들이 다시 분석하여 좋은 컴포넌트 아키텍처를 도출하기 위해서는 많은 시간과 노력이 필요하다. 즉, 본 논문의 2.1절에서 기술했듯이 시스템적 개념이 없는 현업 담당자들이 업무에 대하여 전문적인 지식을 갖고 있다 하더라도 개념 모델을 도출하기가 어렵다. 그리고 시스템 분석가는 개념 모델을 시스템적 측면에서 다시 정제하여 비즈니스 타입 모델(Business Type Model)을 완성하고 비즈니스 타입 모델을 구성하고 있는 타입 중 핵심 타입을 중심으로 컴포넌트를 식별한다. 따라서 비즈니스 타입 모델은 개념 모델인 타입 다이어그램(Type Diagram)에 직접적인 영향을 받게 되므로 좋은 아키텍처를 도출하기가 쉽지 않고 잘못 되었을 경우 처음부터 요구사항 분석단계로 피드백 해서 다시 시작해야 하는 문제점이 있다. 그러나 RUP 분석단계는 2.1절에서 제시했듯이 객체를 도출하기 위하여 좀 더 체계적인 절차와 지침이 있고 요구사항 분석단계에서 산출된 유스케이스를 중심으로 분석단계에서 유스케이스 실체화(Usecase Realization) 과정에 의해서 충분히 핵심적이고 일반화된 객체들이 도출된다[5]. 또한 CBD96과 UML Components의 비즈니스 타입 모델은 타입으로 구성되어 있고 이때의 타입은 시스템에서 지속적으로 관리되어야 하는 정보를 나타낸다고 정의하고 있다[1]. Jacobson은 분석 객체를 경계(Boundary), 제어(Control), 엔티티(Entity) 등의 3가지 타입으로 분류하였고 그 중 엔티티 객체는 시스템에 장기간 또는 영구적으로 저장해야 할 정보를 모델링 한 것이다 라고 정의하고 있다[6]. 결론적으로 CBD96과 UML Components의 비즈니스 타입 모델에서의 타입은 Jacobson의 엔티티 객체와 동일한 개념으로 정의할 수 있다. 따라서 비즈니스 타입 모델을 구성하고 있는 타입은 RUP 분석 프로세스의 유스케이스 실체화의 과정을 통하여 충분히 핵심적인 엔티티 객체를 도출할 수 있으므로 CBD96과 UML Components의 개념 모델을 통하여 비즈니스 타입 모델을 도출하는 문제점을 RUP의 요구사항 분석단계와 분석단계를 통하여 충분히 보완할 수 있다.

객체타입 중 경계 객체는 시스템의 기능이기보다는 사용자와의 인터페이스(Interface)를 위한 것이고, 시스템의 환경에 따라 자주 바뀔 수 있는 부분이므로 컴포넌트 추출에서는 고려하지 않는다. 제어 객체는 추출될 비즈니스 컴포넌트의 인터페이스에 포함되어야 할 기능을 명시하고 있으므로 고려하지 않고 엔티티 객체를 중심으로 비즈니스 컴포넌트를 식별하는데 엔티티 객체간의 상호작용 분석이나 비즈니스 컴포넌트의 인터페이스는 RUP 분석 프로세스의 인터랙션 다이어그램(Interaction Diagram)을 통하여 정의된다. 따라서 본 논문에서는 비즈니스 컴포넌트를 식별하기 위하여 RUP의 분석 프로세스를 통하여 추출된 엔티티 객체를 중심으로 비즈니스 컴포넌트를 식별한다.

Castek의 CBD/e나 CBD96은 시스템 서비스 측면에서 독립적인 기능을 재사용할 수 있음에도 일반적으로 비즈니스 컴포넌트만을 추출하기 위한 방법을 제시하고 있다. 그러나 Compuware의 UNIFACE와 UML Components의 방법은

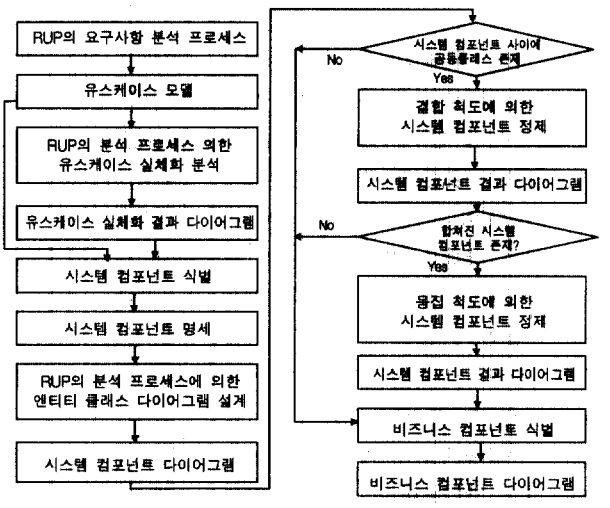
시스템 서비스 측면의 독립적인 기능을 재사용하기 위하여 시스템 컴포넌트를 추출하도록 제시하고 있지만 시스템 컴포넌트를 추출하기 위해서 유스케이스 만을 고려하고 있고 유스케이스 사이에 중복되는 객체에 대하여는 고려하지 않고 있다. 또한 시스템 컴포넌트를 추출하기 위한 상세한 지침이 없다. 컴포넌트란 유사한 기능을 수행하는 관련된 객체들의 그룹으로 이루어진다. 따라서 독립적인 기능적 재사용을 위한 시스템 컴포넌트 추출을 위해서는 유사한 기능을 그룹화 해야 한다는 것이다. 또한 Bunge의 유사성(Similarity) 정의에 의하여 유도되어진 두 메소드 사이의 유사성, 즉 두 개의 메소드가 얼마나 유사한 기능을 가지고 있는가의 척도는 두 메소드가 공통으로 참조하는 인스턴스 변수의 집합에 의하여 결정된다[27]. 즉, 기능의 유사성은 각각의 유스케이스가 공통으로 참조하는 객체의 집합에 의하여 결정되어진다. 따라서 본 논문에서는 시스템 컴포넌트를 식별하기 위하여 의미적으로 관련 있는 유스케이스만 고려하여 그룹화하는 것이 아니라 유스케이스와 각 유스케이스에 포함되는 공통 클래스를 동시에 고려하여 시스템 컴포넌트를 더욱 더 명확하게 추출하고 시스템 컴포넌트를 추출하기 위한 지침을 제시한다.

비즈니스 컴포넌트의 효율적인 추출과 추적성을 위하여 CBD96, UML Components, CBD/e방법처럼 전체 시스템을 중심으로 비즈니스 컴포넌트를 추출하는 것이 아니라 기능적으로 분리된 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 추출한다. 따라서 추출된 시스템 컴포넌트 사이의 중복 객체를 결합척도와 응집척도를 적용하여 시스템 컴포넌트가 포함하고 있는 객체가 중복되지 않도록 재배치하므로 추출된 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 추출하는 것이 가능하다. 전체 시스템의 부분 집합인 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 추출함으로 시스템을 이해하기가 더 쉽고 비즈니스 컴포넌트를 더 효율적으로 추출한다. 또한 요구사항 분석단계부터 추적이 가능하도록 하기 위하여, 또한 2.1절에서 제시한 UML Components의 문제점 중 시스템 컴포넌트와 비즈니스 컴포넌트를 분리해서 추출함으로 일어나는 문제점을 보완하기 위하여, 요구사항과 객체를 분석한 산출물을 가지고 시스템 컴포넌트를 추출한 다음 추출된 시스템 컴포넌트를 기반으로 비즈니스 컴포넌트를 점진적으로 추출한다.

개발자의 직관과 경험에 의하여 컴포넌트를 추출하는 문제점을 보완하기 위하여 또한 보다 독립적인 컴포넌트를 효율적으로 추출하기 위하여 결합척도와 응집척도를 제안하고 적용한다

3.2 컴포넌트 식별을 위한 전체 프로세스

다음은 본 논문에서 제안하고 있는 컴포넌트 식별 방법을 위한 전체적인 프로세스를 제시한다.



(그림 1) 컴포넌트 식별을 위한 전체적인 프로세스

3.3 시스템 컴포넌트의 식별과 시스템 컴포넌트의 정제

RUP의 요구사항 분석프로세스와 분석프로세스의 산출물을 기반으로 독립적인 기능적 재사용을 위하여 시스템 컴포넌트를 효율적으로 식별하기 위한 방법과 비즈니스 컴포넌트의 효율적 식별을 위하여 시스템 컴포넌트의 중복 클래스를 정제하는 방법을 제시한다.

3.3.1 시스템 컴포넌트의 식별 방법

기존의 컴포넌트 식별 방법은 시스템 컴포넌트를 추출하기 위한 구체적인 지침이 없고 의미적으로 유사한 유스케이스만을 그룹화 해서 시스템 컴포넌트를 추출한다. 그러나 본 논문에서는 업무를 잘 모르는 개발자라 하더라도 시스템 컴포넌트를 효율적으로 추출하기 위하여 유스케이스만이 아니라 액터(Actor) 그리고 유스케이스에 포함된 객체도 고려하여 시스템 컴포넌트를 추출한다. 따라서 시스템 컴포넌트를 추출하기 위한 방법은 다음과 같다. 첫째, 액터(Actor)별로 유스케이스를 분류하고, 둘째, 액터별 유스케이스와 이 유스케이스에 참여하는 엔티티 객체와의 관계 테이블을 만든다. 셋째, 관계테이블을 보고 유사한 기능의 유스케이스와 이 유스케이스에 참여하는 엔티티 객체의 공유정도를 고려하여 유스케이스를 그룹화 해서 시스템 컴포넌트를 추출한다. 이때 시스템 컴포넌트에 포함되는 유스케이스는 시스템 인터페이스가 되고 이 유스케이스에 의해서 도출되는 엔티티 객체는 시스템의 기능을 수행하기 위한 정보가 된다.

3.3.2 결합도를 적용한 시스템 컴포넌트 정제

컴포넌트는 재 사용성과 유지 보수 단계의 소프트웨어 수정에 대한 효율적인 관리를 위해서 보다 독립적인 특성을 가지고 정의되어져야 한다. 즉, 잘 정의된 컴포넌트는 컴포넌트 사이의 낮은 결합도와 컴포넌트 내의 높은 응집도를 만족해야 한다. 따라서 컴포넌트의 특성을 적용하여 결합도

를 측정할 수 있는 척도를 정의하고 정의된 결합 척도는 보다 독립적인 시스템 컴포넌트와 비즈니스 컴포넌트를 효율적으로 추출하기 위하여 적용되고 시스템 컴포넌트간의 중복 클래스를 제거한다. Chidamber와 Kermerer[7]가 제안한 객체지향 메트릭스 중 클래스의 결합도를 측정하기 위한 척도는 CBO(Coupling Between Object classes)로써 한 클래스와 결합되어 있는 다른 클래스들의 개수로 정의된다. 따라서 본 논문에서는 한 클래스와 결합되어 있는 다른 클래스의 개수를 Association의 수라고 정의하고 컴포넌트와 컴포넌트 사이의 관계는 상속관계 그리고 집합관계가 존재하지 않고 연관관계만 존재하기 때문에 두 개의 컴포넌트 C_i와 C_j의 결합도 $Cp(C_i, C_j)$는 컴포넌트 C_i와 C_j 사이의 Association수로 구한다. 또한 추출된 시스템 컴포넌트 사이에 공통 클래스 C_i와 시스템 컴포넌트 S_i에 대한 결합도 $Sp(C_i, S_i)$는 시스템 컴포넌트 S_i에 속해 있는 공통 클래스 C_i와 결합되어 있는 다른 클래스의 개수로 구한다. 또한 컴포넌트 내의 클래스들은 연관관계만 존재하는 것이 아니라 상속관계, 집합 연관관계가 존재하므로 클래스들의 구조적인 특징을 고려하여야 한다. 클래스들의 구조적 관계의 특성은 Association < Inheritance < Aggregation < Composition의 순으로 강한 결합을 나타내고 소프트웨어의 수정시 강한 영향을 받는다. 따라서 클래스들의 결합도를 측정하기 위해서는 클래스들의 구조적 특징도 고려하여야 한다. 컴포넌트의 결합도가 3또는 4이상일 경우 컴포넌트의 범위를 다시 고려해 볼 것을 권고[8]하고 있으므로 두 컴포넌트 사이의 연관관계의 수가 3또는 4이상일 경우 본 논문에서는 두 컴포넌트를 결합할 것을 제안한다. 이러한 특성들을 적용하여 독립적인 컴포넌트 식별을 위한 결합척도는 <표 1>에서 정의한다.

다음은 결합 척도를 적용하여 시스템 컴포넌트를 정제하기 위한 방법이다. 시스템 컴포넌트들에 속한 중복 클래스를 제거하기 위해서 시스템 컴포넌트에 속하는 공통 클래스의 배치를 결합 척도를 적용하여 결정한다. 시스템 컴포넌트의 공통 클래스 배치는 다음과 같이 결정된다. 만약 시스템 컴포넌트 사이에 공통 클래스가 존재한다면 결합의 정도에 의하여 결합도가 높은 곳으로 공통 클래스를 포함시키고 두개 이상의 시스템 컴포넌트에 동시에 강한 결합도를 가지고 있다면 공통 클래스가 임의의 컴포넌트로 포함될 수 없으므로 컴포넌트를 합하여 하나의 시스템 컴포넌트를 형성한다. 이렇게 해서 만들어진 컴포넌트가 제대로 추출이 되었는지를 판단하여 그렇지 않은 경우 다시 정제하여야 한다. 따라서 컴포넌트의 결합도가 3또는 4이상일 경우 다시 컴포넌트의 범위를 고려해 볼 것을 권고하고 있으므로 두 컴포넌트의 결합도가 4이상일 경우 본 논문에서는 두 컴포넌트를 결합할 것을 제안한다. 다음 <표 1>은 독립적인 컴포넌트 식별을 위하여 공통 클래스 배치를 결정하기 위한 결합 척도 적용 규칙을 정의한다.

<표 1> 공통 클래스 배치를 결정하기 위한 결합도 적용 규칙

규칙	공통클래스와 컴포넌트 1과의 관계	결합도	공통클래스와 컴포넌트 2과의 관계	공통클래스배치
규칙 1	Composition	>	Association	컴포넌트 1
규칙 2	Aggregation	>	Association	컴포넌트 1
규칙 3	Inheritance	>	Association	컴포넌트 1
규칙 4	Association size ↑ (<5)	>	Association size ↓ (< 5)	컴포넌트 1
규칙 5	Association size = (<5)	>,<	Association size = (<5)	컴포넌트 1이나 컴포넌트 2
규칙 6	Composition Aggregation, Inheritance	∪	Composition Aggregation, Inheritance	컴포넌트 1 + 컴포넌트 2
규칙 7	Association size >= 5	∪	Association size >= 5	컴포넌트 1 + 컴포넌트 2

3.3.3 응집도를 적용한 시스템 컴포넌트 정제

본 논문에서 정의하는 응집 척도는 보다 독립적인 시스템 컴포넌트를 효율적으로 추출하기 위한 시스템 컴포넌트 정제 방법으로 3.3.2절의 결합도를 적용한 시스템 컴포넌트 정제 과정을 통하여 결정된 시스템 컴포넌트 중 공통클래스가 컴포넌트 들 사이에서 강한 결합도를 가지고 있어서 하나의 시스템 컴포넌트로 합친 경우 응집도를 측정하여 다시 시스템 컴포넌트를 정제한다. Chidamber와 Kermerer가 제안한 객체지향 메트릭스[7]중 클래스의 응집도를 측정하기 위한 척도는 LCOM(Lack of Cohesion in Methods)으로 n개의 메소드 $M_1, M_2, \cdots, M_n$을 갖는 클래스 C_i 를 가정할 때 n개의 각 메소드 $M_1, M_2, \cdots, M_n$의해서 사용되는 인스턴스 변수들의 집합을 $\{I_1\}, \cdots, \{I_n\}$라 정의한다. 이때 두 개의 메소드 M_i와 M_j에 의해서 사용되는 인스턴스 변수들의 집합, I_i와 I_j에 대해서 $p = \{ (I_i, I_j) \mid I_i \cap I_j = \emptyset \}$, $q = \{(I_i, I_j) \mid I_i \cap I_j \neq \emptyset \}$라 할 때, 만약 $|p| > |q|$면 $LCOM = |p| - |q|$이고, 그렇지 않으면 $LCOM = 0$ 이다.

본 논문에서는 객체지향 개발 방법에서 클래스의 응집 척도인 LCOM을 기본으로 컴포넌트의 응집 척도를 정의한다. 클래스는 속성인 변수와 메소드로 이루어지지만 컴포넌트는 클래스와 컴포넌트의 기능을 나타내는 인터페이스로 이루어진다. 이때 클래스의 메소드는 변수를 사용하여 기능을 실행하고 컴포넌트의 인터페이스는 컴포넌트 내의 클래스를 사용하여 기능을 실행함으로 클래스의 변수는 컴포넌트의 클래스로 대치되고 클래스의 메소드는 컴포넌트의 인터페이스로 대치된다. 따라서 본 논문에서는 이러한 컴포넌트의 특성을 적용하여 응집도 측정을 위한 척도를 다음과 같이 정의한다.

정의 1. 한 컴포넌트를 구성하고 있는 클래스

한 컴포넌트 P는 m개의 클래스인 $C_1, C_2, \cdots, C_m$의 그룹으로 구성되어 있으므로 하나의 컴포넌트 P를 구성하고 있는 m개의 클래스의 집합을 C_P라고 정의한다.

$$C_P = \{C_1, C_2, \cdots, C_m\}$$

정의 2. 한 컴포넌트를 구성하고 있는 인터페이스

한 컴포넌트 P는 하나 이상의 인터페이스가 존재하고 한 컴포넌트 P에 속하는 인터페이스들은 기능을 수행하는 메소드의 집합으로 구성되어 있으므로 한 컴포넌트의 인터페이스들에 속하는 n개의 메소드의 집합을 M_p라 정의한다.

$$M_p = \{m_1, m_2, \cdots, m_n\}$$

정의 3. 한 컴포넌트의 인터페이스 메소드에 의해서 참조되는 클래스들의 집합

한 컴포넌트 P의 인터페이스 메소드 m_i에 의하여 참조된 클래스의 집합을 I_i 라고 정의하고 메소드 m_i 에 의하여 참조되는 클래스의 집합, 즉 I_i는 정의 1에 의하여 $C_P = \{C_1, C_2, \cdots, C_m\}$의 범위 안에 있다. 따라서 I_I 는 다음과 같이 정의된다.

$$I_i = m_i(C_P) = \{ C_1, C_2, \cdots, C_m\},\ 1 \le i \le n$$

정의 4. 한 컴포넌트의 응집도 척도

한 컴포넌트 P를 구성하는 두개의 인터페이스 메소드들에 대하여 인테페이스 메소드가 참조하는 공유되지 않는 클래스들의 집합을 p라고 정의하고 인터페이스 메소드가 참조하는 공유되는 클래스들의 집합을 q정의한다. 이때 컴포넌트의 응집도 척도 Ch(P)는 집합q의 크기와 집합p의 크기의 차에 의해서 결정되어 진다.

$$p = \{ (I_i, I_j) \mid I_i \cap I_j = \emptyset \},\ 1 \le i, j \le n$$
$$q = \{ (I_i, I_j) \mid I_i \cap I_j \neq \emptyset \},\ 1 \le i, j \le n$$
$$Ch(P) = |q| - |p|$$

3.4 비즈니스 컴포넌트 식별 방법

CBD96이나 UML Components 방법은 시스템의 전체 도메인 모델을 중심으로 비즈니스 컴포넌트를 추출함으로 요구사항 분석단계로부터 비즈니스 컴포넌트 추출단계까지 산출물들간에 추적이 가능하지 않고 중규모 이상의 시스템이라면 도메인 모델을 산출하거나 이해하기도 어려워 개발자가 비즈니스 컴포넌트를 추출하는 것이 비효율적이다. 그러나 본 논문에서는 시스템의 전체 도메인을 중심으로 비즈니스 컴포넌트를 추출하는 것이 아니라 전체 시스템을 기능적으로 분할하여 추출한 독립적인 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 추출한다. 따라서 모델을 산출하거나 시스템을 이해하기가 쉽고 비즈니스 컴포넌트를 더욱 효율적으로 추출할 수 있다. 또한 요구사항 분석 단계부터 비즈니스 컴포넌트 추출까지 추적이 가능하므로 사용자의 요구사항 변경시나 소프트웨어의 수정시 효율적으로 대처할 수 있다[25, 26]. 또한 UML Components 방법의 경우 시스템 컴포넌트와 비즈니스 컴포넌트가 명확히 분리되어서 추출되기 때문에 비즈니스 컴포넌트를 조합하여 시스템 컴포넌트를 완성하기 위해서는 다시 시스템 컴포넌트에 포함된 각 유스케이스의 이벤트 흐름을 분석하여 어떠한 클래스가 포함될 것인지를 추출하고 추출된 클래스가 어느 비즈니스 컴포넌트에 포함되는지를 다시 한번 고려하여야 하는 문제점이 있으나 본 논문에서 제안한 방법은 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 추출하기 때문에 시스템 컴포넌트를 완성하기 위해서 다시 분석 할 필요가 없다. 또한 비즈니스 컴포넌트를 추출하기 위하여 핵심 클래스를 중심으로 어떻게 그룹화 해야 하는지에 대한 명확한 지침이 나와있지 않다. 이러한 문제점을 보완하기 위하여 <표 1>에서 정의하고 있는 결합 척도를 적용하여 비즈니스 컴포넌트를 식별한다.

3.5컴포넌트 식별을 위한 상세 절차

이 장에서는 RUP의 요구사항 분석단계와 분석단계를 기반으로 본 연구에서 제안한 컴포넌트 식별 방법을 적용한 상세 절차를 제시한다.

절차 1. 요구사항을 분석하여 유스케이스를 추출하고 RUP의 요구사항 분석모델을 완성한다.

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

절차 3. 추출된 유스케이스 중 시스템 서비스 관점에서 유사한 기능을 가진 유스케이스와 동일한 엔티티 객체를 참조하는 정도를 고려하여 유스케이스를 그룹화 한다.

절차 3에 대한 상세 절차는 다음과 같다.

절차 3-1. Actor별 유스케이스와 이 유스케이스에 참여하는 엔티티 객체와의 관계 테이블을 만든다.

절차 3-2. Actor별로 관계테이블을 보고 유사한 기능의 유스케이스와 유스케이스에 참여하는 엔티티 객체의 공유정도를 고려하여 유스케이스를 그룹화 한다.

절차 3-3. Actor들이 공동으로 참여하는 유스케이스나 다른 Actor와의 유스케이스와 공유 객체를 다시 고려하여 유스케이스를 재 그룹화 한다.

절차 4. 절차 3에서 그룹화 된 유스케이스를 시스템 컴포넌트라 정의하고 RUP의 요구사항 분석 프로세스를 통하여 산출된 절차1, 2, 3의 결과와 RUP의 분석 프로세스를 통하여 시스템 컴포넌트를 중심으로 엔티티 클래스 다이어그램을 설계한다.

절차 5. 만약 도출된 시스템 컴포넌트 사이에 공통 클래스가 없다면 시스템 컴포넌트의 클래스로 결정되고 절차 8로 가서 비즈니스 컴포넌트를 식별한다. 그러나 만약 공통 클래스가 존재한다면 공통 클래스의 배치를 고려하기 위하여 절차 6을 진행한다.

절차 6. 두 개 이상의 컴포넌트에 공통적으로 속하는 클래스들은 <표 1>에서 제시한 결합 척도를 적용하여 공통 클래스의 배치를 결정한다.

절차 6에 대한 상세 절차는 다음과 같다.

절차 6-1. 공통 클래스들의 결합도를 측정하여 결합도가 높은 쪽으로 클래스들의 컴포넌트 배치가 결정된다.

절차 6-2. 공통 클래스가 각 컴포넌트에 동시에 강한 결합도를 가지고 있지 않다면 시스템 컴포넌트의 클래스로 결정되고 절차 8로 가서 비즈니스 컴포넌트를 식별한다.

절차 6-3. 공통 클래스가 각 컴포넌트에 대해서 동시에 강한 결합도를 가지고 있다면 강한 결합도를 가지고 있는 컴포넌트를 합친 후 절차 7로 가서 시스템 컴포넌트를 다시 정제한다.

절차 7. 합친 시스템 컴포넌트의 경우 컴포넌트들의 아키텍처를 결정하여 아키텍처별로 컴포넌트들을 분리한다. 각 아키텍처 당 컴포넌트의 응집도를 계산하여 평균적으로 가장 응집도가 좋은 아키텍처가 시스템 컴포넌트로 결정된다.

절차 7에 대한 상세 절차는 다음과 같다.

절차 7-1. Inheritance, Composition, Aggregation의 관계가 있는 클래스들은 하나의 클래스로 가정하고 공통 클래스C'으로 정의한다.

절차 7-2. 이 클래스들을 패키지별로 분리한 아키텍처와 공통 클래스 배치를 모든 경우의 수를 고려하여 배열한다.

절차 7-3. 각각의 아키텍처에 대하여 응집도를 계산한다.

절차 7-4. 가장 응집도가 높은 아키텍처로 해당 컴포넌트를 분리한다.

절차 8. 시스템 컴포넌트가 결정되면 시스템 컴포넌트 내의 엔티티 클래스를 중심으로 비즈니스 컴포넌트를 추출하기 위하여 다른 클래스와 필수 연관 관계가 존재하지 않고 유일한 식별자를 가진 엔티티 클래스를 핵심클래스(Core-Class)로 식별하고 이러한 핵심 클래스를 기준으로 클래스 사이의 결합도, 즉 <표 1>의 결합 척도를 적용하여 비즈니스 컴포넌트를 식별한다.

4. 사례 연구 및 비교 평가

이 장에서는 본 연구에서 제안한 컴포넌트 식별 방법과 절차를 치과 병원 관리 시스템을 통하여 적용함으로 그 효율성을 평가한다.

4.1 본 연구에서 제안한 방법에 의한 컴포넌트 식별 적용 사례

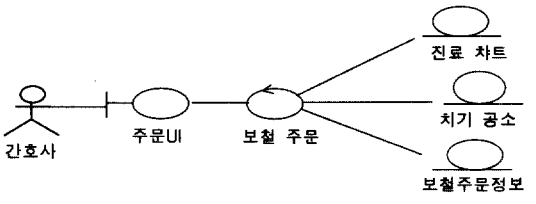
다음은 본 논문의 3절에서 제안한 컴포넌트 식별 방법과 절차에 따라 치과 병원 관리 시스템을 적용한다.

절차 1. 치과병원 관리 시스템의 요구사항을 분석하여 추출된 유스케이스는 다음과 같다.

〈표 2〉 치과병원 관리 시스템의 유스케이스

액 터	유스케이스	
간호사	접수를 하다. (Extends)	환자의 신환 접수를 하다.
		환자의 기환 접수를 하다.
	환자의 기본 정보를 관리하다.	
	진료비를 산출하다.	
의사	진료를 하다. (Include)	병명을 입력하다.
		치식을 입력하다.
		처치항목을 입력하다.
		처탕항목을 입력하다.
	진료를 하다. (Extends)	검사를 하다.
의사, 간호사	의료 보험료를 청구하다.	
	보험료 현금 입고정보를 조회하다.	
	의료 보험 청구정보를 관리하다.	
	비보험 적용항목을 조회하다.	
	비보험 진료비를 조회하다.	
	보험 진료비를 조회하다.	
	보철물을 주문하다.	
	보철물 주문정보를 관리하다.	
	보철물 입고정보를 관리하다.	
간호사, 환자	예약을 관리하다.	
	예약을 하다.	

절차 2. 절차2는 각 유스케이스의 시나리오, 즉 이벤트의 흐름에 따라 유스케이스 실체화 과정을 통해서 객체를 도출하는데 다음은 "보철물을 주문하다." 유스케이스의 실체화 과정에 의해서 도출된 객체이다.



(그림 2) 유스케이스 실체화 과정

절차 3. 유스케이스를 추출하고 유스케이스의 유사한 기능과 동일한 객체를 참조하는 정도를 고려하여 유스케이스를 그룹화 한다. 위의 정의에 의하여 도출된 유스케이스를 그룹화한 시스템 컴포넌트는 다

음과 같다.

(1) 3.5절의 절차3-1에서 3-3까지의 적용 사례는 다음과 같고 <표 3>의 엔티티 객체를 대치하는 이름은 <표 5>에 의하여 정의된다.

〈표 3〉 액터 그리고 유스케이스와 엔티티 객체의 관계 테이블

액터	유스케이스	엔티티 객체										
		C19	C2	C1	C5	C8	C9	C10	C11	C12	C13	C14
간호사, 환자	예약을 하다.	√		√								
	예약을 관리하다	√		√								
간호사	신환접수를 하다.		√	√	√							
	기환접수를 하다.		√	√	√							
	환자정보를 관리하다.			√								
	진료비를 산출하다.				√	√	√			√	√	
의사	진료를 하다.				√	√	√					
	검사를 하다.				√	√	√			√	√	√

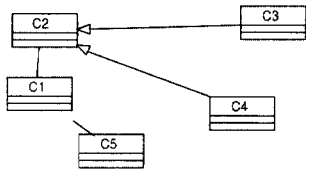
(2) 다음은 (1)의 결과에 의해서 정의된 시스템 컴포넌트 결과 명세이다.

〈표 4〉 치과병원 관리 시스템의 시스템 컴포넌트

시스템 컴포넌트				
접수	진료	보험청구	보철주문	예약

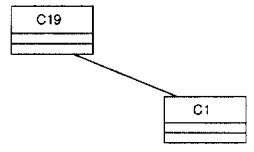
절차 4. 도출된 시스템 컴포넌트 중 예약, 접수, 진료에 대하여 절차 2의 실체화 과정에 의해서 도출된 객체를 중심으로 엔티티 클래스 다이어그램을 완성한다.

(1) 접수 엔티티 클래스 다이어그램



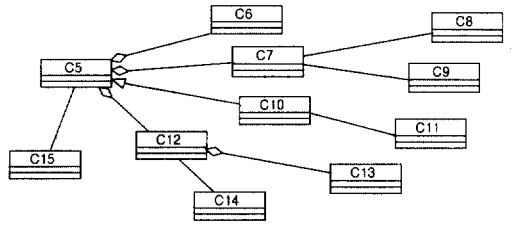
(그림 3) 접수 엔티티 클래스 다이어그램

(2) 예약 엔티티 클래스 다이어그램



(그림 4) 예약 엔티티 클래스 다이어그램

(3) 진료 엔티티 클래스 다이어그램



(그림 5) 진료 엔티티 클래스 다이어그램

다음 <표 5>는 다이어그램들에서 제시되는 엔티티 클래스와 연관된 클래스 명을 나타낸다.

〈표 5〉 엔티티 클래스 목록

C1	환자의 기본정보	C8	보험 적용항목	C15	진료비 내역정보
C2	접수	C9	비보험 적용항목	C16	예약정보
C3	신환 접수	C10	처방항목		
C4	구환 접수	C11	약품정보		
C5	진료 차트	C12	검사항목		
C6	치식	C13	검사결과		
C7	처치항목	C14	검사목록		

절차 5. 시스템 컴포넌트 사이에 공통 클래스가 존재하므로 공통 클래스의 배치를 고려하기 위하여 절차 6을 진행한다.

절차 6. 시스템 컴포넌트 들 사이에 공통 클래스가 있으므로 <표 1>의 결합도 적용 규칙에 의하여 공통 클래스를 재배치하거나 시스템 컴포넌트를 합한다. 따라서 절차 6에 의한 결과는 다음 <표 6>과 같다.

① 공통 클래스 C1의 경우

<표 1>의 규칙 2에 의하여 예약 시스템 컴포넌트의 공통 클래스 C1는 접수 시스템 컴포넌트로 배치된다.

② 공통 클래스 C5의 경우

<표 1>의 규칙 6에 의하여 접수 시스템 컴포넌트의 공통 클래스 C5와 진료 시스템 컴포넌트의 공통클래스 C5는 동시에 강한 결합도를 가지고 있으므로 접수 시스템 컴포넌트와 진료 시스템 컴포넌트를 합하여 (접수 + 진료) 시스템 컴포넌트를 도출한다.

〈표 6〉 절차 6에 의한 시스템 컴포넌트

시스템 컴포넌트	포함된 클래스
예약	C19
접수 + 진료	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15

절차 7. 합친 시스템 컴포넌트는 컴포넌트들의 아키텍처를 결정하여 아키텍처별로 컴포넌트들을 분리하고 본 논문의 3.3.3절에서 정의한 응집척도를 적용하여 평균적으로 가장 응집도가 좋은 아키텍처에 의하여 시스템 컴포넌트가 결정된다.

본 논문의 3.5절에서 제안한 절차 7의 구체적인 적용 사례는 다음과 같다.

(1) 분리된 아키텍처는 다음과 같다.

Architecture 1 (접수), (진료)

(2) 3.5절의 절차 7-1에 의하여 상속, 포함관계에 있는 클래스들은 하나의 클래스로 간주하고 C'로 정의 한다. 즉, **C1' = { C2, C3, C4}, C2' = {C1, C5, C6, C7, C10, C12, C13}**이다.

(3) 3.5절의 절차 7-2에 의하여 2)에서 나온 결과 중 2개의 시스템 컴포넌트에 공통 클래스 묶음인 C2'을 어디에 배치할 것인지를 고려하여 아키텍처를 다 다시 정의한다.

- **C2'을 접수 시스템 컴포넌트에 배치할 경우** : 아키텍처 1-1

  접수 = (C1', C2'), 진료 = (C8, C9, C11, C15, C14)

- **C2'을 진료 시스템 컴포넌트에 배치할 경우** : 아키텍처 1-2

  접수 컴포넌트 = (C1'), 진료 컴포넌트 = (C2', C8, C9, C11, C14, C15)

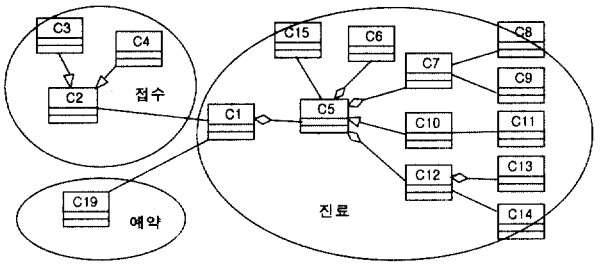
(4) 3.5절의 절차 7-3에 의하여 아키텍처별로 응집도를 계산한다.

(5) 3.5절의 절차 7-4에 의하여 응집도가 좋은 컴포넌트 아키텍처는 1-2이므로 결정된 시스템 컴포넌트는 컴포넌트 아키텍처 1-2이고 그 결과는 <표 7>과 같다.

〈표 7〉 결정된 시스템 컴포넌트와 포함된 엔티티 클래스

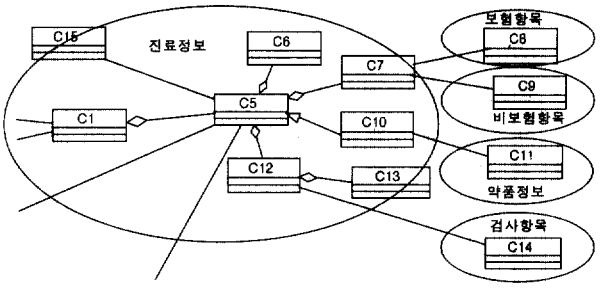
시스템 컴포넌트	클래스
예약	C19
접수	C1'
진료	C2', C8, C9, C11,C14,C15

(6) 본 논문의 3.5절에서 제안한 시스템 컴포넌트 식별 절차1에서 절차 7까지에 의하여 식별된 시스템 컴포넌트 결과 다이어그램은 다음과 같다.



(그림 6) 시스템 컴포넌트 결과 다이어그램

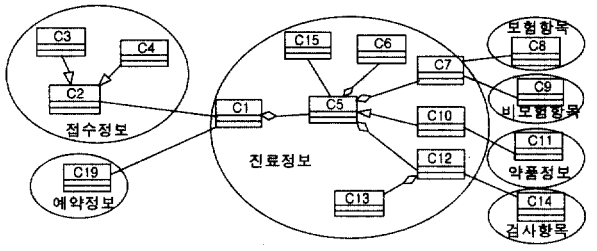
절차 8. 절차 7에 의해서 식별된 시스템 컴포넌트에 대한 비즈니스 컴포넌트의 식별은 클래스간의 결합도 척도를 적용하여 효율적으로 독립적인 비즈니스 컴포넌트를 식별한다. 진료 시스템 컴포넌트에 대한 비즈니스 컴포넌트 식별 결과는 다음과 같다.



(그림 7) 진료 시스템 컴포넌트에 대한 비즈니스 컴포넌트

4.2 CBD96의 컴포넌트 식별 방법에 의한 치과병원관리시스템 적용 사례

다음은 CBD96의 컴포넌트 식별 방법에 의하여 도출된 비즈니스 타입 모델의 타입 중 핵심타입(Core Type)을 중심으로 비즈니스 컴포넌트를 식별한 결과 다이어그램이다.



(그림 8) CBD96의 컴포넌트 식별 방법에 의한 비즈니스 컴포넌트

핵심타입(Core Type)은 접수정보의 C2, 예약정보의 C19, 진료의 C1, 보험항목의 C8, 비보험 항목의 C9, 약품정보의 C11, 검사항목의 C14이다.

4.3 추출된 시스템 컴포넌트와 비즈니스 컴포넌트

본 연구에서 제안한 방법에 의하여 추출된 시스템 컴포넌트와 비즈니스 컴포넌트는 <표 8>와 같다.

〈표 8〉 추출된 시스템 컴포넌트와 비즈니스 컴포넌트

본 연구에서 제안한 방법	
시스템 컴포넌트	비즈니스 컴포넌트
예약	예약정보
접수	접수정보
진료	진료, 보험항목, 비보험 항목, 약품정보, 검사항목
보철주문	...
보험청구	...

4.4 비교 평가

본 논문에서 제안한 방법을 중심으로 치과병원 관리 시스템에 의해 사례를 제시하였다. 사례를 통하여 검증 되었듯이 본 논문에서 제시한 컴포넌트 식별 방법의 기대효과는 다음과 같다.

첫째, 사례에서 제시했듯이 비즈니스 타입 모델에서 도출된 타입과 RUP 프로세스에 의하여 도출된 엔티티 객체가 같으므로 RUP 프로세스를 적용하여 요구사항을 분석하고 객체를 도출하는 것이 효율적임을 알 수 있다.

둘째, 시스템 컴포넌트 식별 방법을 제시함으로 시스템의 기능적 측면의 재사용이 가능하고 시스템 컴포넌트 추출이 용이하다. 또한 그 컴포넌트가 가지고 있는 기능이 사용자들에게 명확하게 제공되고, 컴포넌트에 대한 이해성도 높아져 이를 사용하고자 하는 사용자들이 쉽게 컴포넌트를 선택할 수 있다.

셋째, 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 식별하기 때문에 요구사항 분석단계의 유스케이스로 부터 비즈니스 컴포넌트까지 추적이 가능하므로 요구사항의 변경이나 소프트웨어의 수정이 일어났을 경우 쉽게 수용할 수 있다.

넷째, 비즈니스 컴포넌트를 추출하기 위하여 전체 도메인 모델을 중심으로 추출하는 것이 아니라 시스템의 부분 집합인 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 추출함으로 모델을 산출하거나 시스템을 이해하기가 쉬워 비즈니스 컴포넌트를 더욱 효율적으로 추출할 수 있고 비즈니스 컴포넌트를 조합하여 시스템 서비스 측면의 기능을 실행하는 컴포넌트를 생성하는 부분을 고려할 필요가 없이 추출된 시스템 컴포넌트를 재사용하여 원하는 기능을 바로 실행할 수 있다.

다섯째, 컴포넌트를 식별하기 위한 결합 척도와 응집 척도를 정의하고 적용함으로 좀 더 효율적으로 독립적인 컴포넌트를 식별할 수 있고 본 논문에서 정의한 응집 척도는 다른 컴포넌트 방법론의 컴포넌트 식별을 위해서 일반적으로 적용되어 질 수 있다.

따라서 본 논문에서 제안한 방법과 기존의 컴포넌트 개발 방법론들과의 총체적인 비교 평가 결과는 다음과 <표 9>과 같고 본 논문에서 제안하는 컴포넌트 식별 방법은 기존 컴포넌트 개발 방법론의 단점을 보완한 방법으로 효율적인 객체의 추출, 시스템의 기능적 재 사용성, 시스템 컴포넌트의 효율적 추출, 추적성 그리고 비즈니스 컴포넌트의 효율적인 추출에 중점을 둔 방법으로 이러한 항목을 중심으로 타 방법론과 비교평가를 제시한다.

〈표 9〉 총체적인 비교 평가

식별항목 / 비교항목	RUP	CBD96	UML Components	UNIFACE	제안한 방법
객체추출 지침과 용이성	○	×	×	△	○
시스템 컴포넌트의 식별지침과 절차	×	×	×	×	○
시스템 컴포넌트 기반의 비즈니스 컴포넌트 식별	×	×	×	○	○
컴포넌트간의 중복객체 고려	×	○	○	×	○
효율적인 컴포넌트 추출을 위한 응집척도 적용	×	×	×	×	○
효율적인 컴포넌트 추출을 위한 결합척도 적용	×	×	×	×	○
기능적 재사용성 고려	○	×	×	○	○
추적성	△	△	△	○	○

5. 결 론

본 논문에서는 기존 방법론들의 컴포넌트 식별에 관한 문제점을 보완한 새로운 컴포넌트 식별 방법과 절차 그리고 독립적인 컴포넌트를 좀 더 효율적으로 식별하기 위해 응집 척도와 결합 척도를 정의하고 제안하였다. 또한 본 연구에서 제안한 방법을 치과 병원 관리 시스템의 사례에 적용하여 그 효율성을 검증하였고 기대 효과를 제시하였다.

본 논문에서 제안한 컴포넌트 식별 방법은 CBD96과 UML Components방법의 문제점인 개념 모델과 비즈니스 타입 모델의 타입 추출의 방법을 RUP 프로세스의 요구사항 분석 단계와 분석단계의 절차를 적용하여 객체를 추출하도록 하였고 사례에 적용해본 결과 훨씬 효율적으로 객체가 추출됨을 알 수 있었다.

시스템 서비스 측면의 기능을 재 사용할 수 있음에도 일반적으로 기존의 컴포넌트 개발 방법론들인 CBD96이나 CBD/e는 재사용의 단위를 비즈니스 컴포넌트로 정의하고 있고 비즈니스 컴포넌트 식별에 중점을 두고 있다. 또한UML Components나 UNIFACE 방법 같이 시스템 컴포넌트의 정의가 있다 하더라도 시스템 컴포넌트를 추출하는 구체적인 지침이 없다. 따라서 본 논문에서는 시스템 서비스 측면의 기능적 재사용을 위하여 시스템 컴포넌트를 효율적으로 식별하는 방법과 절차를 제안하였다. 사례에 적용해본 결과 치과병원 관리 시스템에서는 예약, 접수, 진료, 보험청구, 보철주문 등의 시스템 컴포넌트가 식별되었다. 따라서 추출된 시스템 컴포넌트를 재 사용하여 원하는 기능을 바로 실행할 수 있는 장점이 있다.

CBD96이나 UML Components 방법은 전체 시스템 도메인 모델을 중심으로 비즈니스 컴포넌트를 추출함으로 요구사항 분석단계로부터 비즈니스 컴포넌트 추출단계까지 산출물들간에 추적이 가능하지 않고 중규모 이상의 시스템이라면 도메인 모델을 산출하거나 이해하기도 어려워 개발자가 비즈니스 컴포넌트를 추출하는 것이 비효율적이다. 또한 UML Components 방법의 경우 시스템 컴포넌트와 비즈니스 컴포넌트가 명확히 분리되어서 추출되어지기 때문에 비즈니스 컴포넌트를 조합하여 시스템 컴포넌트를 완성하기 위해서는 다시 시스템 컴포넌트에 포함된 각 유스케이스의 이벤트 흐름을 분석하여 어떠한 클래스가 포함될 것인지를 추출하고 추출된 클래스가 어느 비즈니스 컴포넌트에 포함되는지를 다시 한번 고려하여야 하는 문제점이 있다. 따라서 본 논문에서는 비즈니스 컴포넌트를 추출하기 위하여 전체 도메인 모델을 중심으로 추출하는 것이 아니라 시스템의

부분 집합인 시스템 컴포넌트를 중심으로 비즈니스 컴포넌트를 추출함으로 이러한 문제점을 해결하였다. 따라서 본 논문에서 제안한 비즈니스 컴포넌트 식별은 사례에서 제시했듯이 예약, 접수, 진료의 시스템 컴포넌트를 중심으로 예약정보, 접수정보, 진료, 보험항목, 비 보험항목, 약품정보, 검사 항목 등의 비즈니스 컴포넌트가 추출되었고 CBD96이나 UML Components의 전체 도메인 모델을 중심으로 비즈니스 컴포넌트를 추출한 결과와 같음을 보였다.

보다 독립적인 컴포넌트를 효율적으로 추출하기 위하여 그리고 시스템 컴포넌트간의 중복 클래스 제거를 위하여 결합 척도와 응집 척도를 정의하여 적용한 결과 보다 독립적인 컴포넌트가 추출되고 중복 클래스가 제거됨을 사례를 통하여 보였다. 또한 본 연구에서 제안한 응집도 척도와 결합도 척도는 컴포넌트의 특성을 고려하여 정의하였으므로 일반적인 컴포넌트 식별 방법에서도 적용 가능하다.

향후 연구 과제로는 좀더 많은 실 사례 검증을 통하여 본 논문에서 제시한 방법을 보완해 나가는 과정이 필요하고 효율적인 컴포넌트의 추출을 위하여 컴포넌트 식별을 지원하는 자동화 도구가 요구된다.

참 고 문 헌

[1] John Cheesman, John Daniels, "UML Components," Addison-Wesley, pp.67-120, 2001.

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

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

[4] Computer Associates, CBD96, http : //www.sterling.com/cbdedge1/cbd96.htm.

[5] Chris Marshall, "Enterprise Modeling with UML," Addison-Wesley, 2000.

[6] Ivar Jacopson, "The Object Advantage," ACM Press, Addison-Wesley, 1995.

[7] S. R. Chidanber and C. F. Kemerer, "Towards a Metrics Suite for Object-Oriented Design," OOPSLA'91, Phoenix, Arizona, USA, pp.197-211, 1991.

[8] John Dodd, "Identifying & Scoping CBD96 Components," Texas Instruments Inc., 1999.

[9] 조진희, 이우진, 김민정, 신규상, "CBD 방법론의 컴포넌트 식별 방법의 비교", 정보처리학회지, 제7권 제2호, pp.515-518, 2000.

[10] 유영란, 김수동, "Use Case 및 클래스의 가중치 분석에 의한 컴포넌트 추출기법", 숭실대학교 학위 논문, 2000.

[11] Martin Folower, "UML Distilled," Addison-Wesley, 1999.

[12] Seok-Jin Yoon, Gyu-Sang Shin, "Extraction Component from Object-Oriented Programs," Proceedings of the 14th KIPS Fall Conference, 2000.

[13] Eriksson, Penker, "UML Toolkit," Wiley, 1998.

[14] Peter Herzum and Oliver Sims, 'Business Component Factoring ; A Comprehensive Overview of Component-Based Development for Enterprise," John Wiley & Sons. Inc., 2000.

[15] Peter Herzun and Oliver Sims, "The Business Component Approach ; Business Object Design and Implementation II," OOPSLA'96, OOPSLA '97, OOPSLA '98 Workshop Proceedings.

[16] Cool software korea, "CBD Project Guide using Cool : Joe ver 1.0," Cool software korea, 2000.

[17] Doug Rosenberg, Kendall scott, 'UML Object Modeling,' Addison-Wesley, 2000.

[18] 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, June, 1994.

[19] M. Lorenz, and J. Kidd, "Object-Oriented Software Metrics : A Practical Guide," Prentice-Hall, 1994.

[20] Compuware corp., "UNIFACE Development Methodology : UNIFACE V7.2," Compuware corp., 1998.

[21] kirby Mclnnis, "An Overview of CBD/e," Technical Reports, Castek corp., 1999.

[22] W. Burg, S. Hawker, D. Hale, K. Mclnnis, A. Parrish, S. Sharpe, R. Woolridge, "Exploring a Comprehensive CBD Method : Use of CBD/e in Practice," The 3nd International Workshop on Component-Based Software Engineering, 2000.

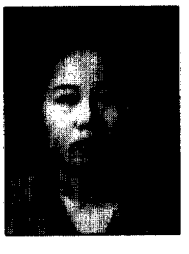
[23] MTW Corp., "A Proven Path to Delivering Mission Critical Enterprise Systems," White Paper, MTW corp., 2001.

[24] Misook Choi, Seojung Lee, Jaenyun Park, "System Analysis Model Applying the Property of Business Domain," IAWTIC International Conference Proceedings, 2001.

[25] Misook Choi, Hyunhee Koh, Yongik Yoon, Jaenyun Park, "Algorithm and Graph for Change Management by Software Architecture," ICIS'01 International Conference Proceeding, 2001.

[26] Misook Choi, Kyunghee Kim, Seojung Lee, Yongik Yoon, Jaenyun Park, "Change Impact Analysis based on UML," CIMCA International Conference, Proceeding, 2001.

[27] Brian Henderson-Sellers, 'Object-Oriented Metrics,' Prentice Hall, 1996.



최 미 숙

e-mail : cms@cs.sookmyung.ac.kr

1990년 전북대학교 졸업(학사)

1994년 숙명여자대학교 대학원 전산학과 (이학석사)

1995년~1999년 나주대학 소프트웨어개발과 전임강사

1997년~현재 숙명여자대학교 대학원 전산학과 박사과정 수료

관심분야 : 컴포넌트 개발방법론, 품질평가, 소프트웨어 테스팅

윤 용 익

e-mail : yiyoon@cs.sookmyung.ac.kr

1983년 동국대학교 졸업(학사)

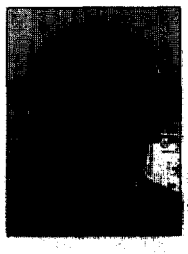
1985년 한국과학기술원 전산학과(이학석사)

1994년 한국과학기술원 전산학과(이학박사)

1985년~1997년 한국 전자통신연구소 책임연구원

1997년~현재 숙명여자대학교 정보과학부 교수

관심분야 : 정보통신, 멀티미디어통신, 분산시스템, 실시간 처리 시스템, 분산 미들웨어 시스템, 분산 데이터베이스 시스템, 실시간 OS/DBMS



박 재 년

e-mail : jnpark@cs.sookmyung.ac.kr

1966년 고려대학교 졸업(학사)

1969년 고려대학교 이학석사

1981년 고려대학교 이학박사

1979년~1983년 전남대학교 전산통계학과 교수

1983년~현재 숙명여자대학교 정보과학부 교수

관심분야 : 시스템 개발방법론, 품질 평가, 메타 DB, 시뮬레이션