객체 지향 도메인 모델을 이용한 컴포넌트 식별 도구 개발

(Tool Development for Identifying Components using Object-Oriented Domain Models)

이 우 진 † 권 오 천 ††
(Woo-Jin Lee) (Oh-Cheon Kwon)

요 약 소프트웨어 재사용에 중점을 두고 있는 컴포넌트 기반 개발(CBD: Component-Based Development) 기술은 생산성을 극대화하려는 소프트웨어 개발 업체로부터 많은 관심을 끌고 있다. 하지만, CBD 기술의 핵심 프로세스인 컴포넌트 식별 프로세스는 주로 도메인 전문가의 경험과 직관에 의존하여 지원 도구 개발에 어려움이 많았다. 이 논문에서는 객체 의존성과 객체 사용 패턴 정보를 이용하여 체계적인 컴포넌트 식별 과정을 제안하고 이를 지원하는 도구를 설계 및 구현한다. 객체 지향 도메인 모델에서는 다양한 관점의 다이어그램들이 존재하므로 이들로부터 객체 간의 연관성 정보를 추출하고 통합하여 객체 의존 네트워크로 나타내고 이를 기반으로 수행되는 컴포넌트 식별 알고리즘을 제안한다. 마지막으로 컴포넌트 식별 프로세스 및 도구에 대한 적용성을 평가하기 위해 인터넷 뱅킹 시스템에서의 컴포넌트 식별 과정을 설명한다.

키워드 : 컴포넌트, 컴포넌트 식별, 소프트웨어 재사용, 컴포넌트 식별기, 객체지향 도메인 모델

Abstract Component-based Development(CBD) based on the software reuse has been more attractive from software companies that want to enhance software productivity. However, since component identification process is mainly dependent on domain expert's intuition and experience, it was very difficult to develop tools for supporting the component identification process. In this paper, we propose a systematic procedure of identifying reusable component by using object dependencies and object usages and provide a design and implementation of its supporting tool. In object-oriented domain models, there exists several diagrams which are described in different viewpoints. From these diagrams, object dependency and object usages are extracted and merged into an object dependency network, which is a basis for performing a component identification algorithm. Finally, through a case study of internet banking system, we evaluate the applicability of the proposed identification process and tool.

Key words : Component, Component identification, Software reuse, Component identifier, Object-Oriented domain model

1. 서 론

소프트웨어 산업이 급속하게 발전해감에 따라 정보기술 업체간 경쟁이 더욱 심화되어 소프트웨어 재사용성, 적시성, 유지 보수성 등이 업체의 생명력으로 대두되면서 컴포넌트 기반 개발(CBD: Component-Based Development) 기술이 점차 각광을 받기 시작하였다. CBD 기술은 기존 컴포넌트를 재사용하거나 컴포넌트들의 병행적 개발을 통해 개발 기간 단축 효과와 비용 절감 효과를 가져올 수 있다. 재사용성이 높은 컴포넌트는 CBD 기술의 핵심으로 도메인 모델로부터 여러 어플리케이션에 공통으로 사용할 수 있는 부분들을 추출한 후, 독립적으로 배포 가능하도록 인터페이스를 명확히 정의한 소프트웨어 단위이다[1].

지금까지 컴포넌트 식별 프로세스는 주로 도메인 전

· 본 연구는 과학기술부 국가지정연구실 사업으로 수행되었음
† 종신회원 : 경북대학교 컴퓨터과학과 교수
woojin@knu.ac.kr
†† 비 회 원 : 한국전자통신연구원 S/W · 컨텐츠연구부 연구원
ockwon@etri.re.kr
논문접수 : 2002년 10월 18일
심사완료 : 2003년 4월 2일

문가의 경험과 직관에 주로 의존해 왔다. RUP(Rational Unified Process)[2]에서는 순차도 등에서 특정 제어가 머무르는 부분, 클래스 다이어그램에서 연관성이 높은 객체들, 아키텍처 수준의 서브 시스템 등을 컴포넌트화할 수 있는 후보 컴포넌트로 보는 추상적인 가이드라인을 제공하고 있다. RUP 방법을 확장한 방법[3]에서는 쓰임새(Use Case)를 중심으로 시스템 서비스 컴포넌트를 추출하고 시스템 서비스 컴포넌트를 포함하는 객체들을 중심으로 비즈니스 컴포넌트를 추출하는 방법을 제공하고 있으나 도구 지원에 대해서는 언급하지 않고 있다. CBD 기술을 지원하는 CASE 도구 중에서, Cool:Joe[4] 도구는 사용자가 핵심 객체(Core Type)를 선택하면 나머지 객체들을 연관된 핵심 객체에 추가하는 방법으로 EJB 컴포넌트를 만드는 과정을 지원하지만 이는 EJB 컴포넌트에 특화되어 있으며 사용자 중심으로 수행된다. 또다른 CBD 도구인 TogetherSoft's Together[5]와 Compuware's Uniface[6]는 컴포넌트 식별 과정을 지원하지 않고 있다. 현재까지, 컴포넌트 식별 분야에서는 도구화할 수 있을 정도로 체계화된 식별 과정을 제공하고 있지 못하며 이를 지원하는 도구도 또한 제대로 없는 실정이다.

이 논문에서는 널리 알려진 객체 지향 도메인 모델들로부터 체계적인 방법을 통해 재사용성이 높은 도메인 컴포넌트를 추출하는 방법을 제시하고 이를 지원하는 도구를 개발한다. 이 논문에서는 공통성 및 가변성 등의 도메인 분석은 이미 행해졌다고 가정하며 도메인 분석 결과로 객체지향 도메인 모델이 구성되어 있다고 가정한다. 객체 지향 도메인 모델로는 시스템의 함수적인 특성을 나타내는 쓰임새 다이어그램(Use Case Diagram), 구조적인 특성을 나타내는 객체 다이어그램(Class Diagram), 행위적인 특성을 나타내는 순차도(Sequence Diagram)가 주로 이용된다. 이 세 가지 모델은 서로 다른 관점에서 객체간의 연관성을 나타내고 있으므로 이들을 서로 연계하여 하나의 통일된 관점으로 나타내어 객체간의 의존 관계를 명확히 기술하는 것이 필요하다. 이 연구에서는 객체 다이어그램에서 객체들의 구조적인 연관관계를 추출하고 쓰임새와 순차도 정보로부터 객체간의 이용 관계를 명확히 추출한다. 그리고 서로 상이한 객체간의 의존정보를 행위자 및 객체(Actor and Object : AO) 연관 그래프에 취합하여 기술한다. 그리고 객체 간의 의존 정도를 비교하기 위해 객체 의존값을 수치화하여 객체 의존 네트워크(Object Dependency Network)로 나타내고 이를 바탕으로 컴포넌트 식별 알고리즘을 수행한다. 그리고 이러한 컴포넌트 식별과정을 효율적으로 지원할 수 있는 도구인 컴포넌트 식별기(Component Identifier)의 설계 및 구현에 대해 설명한다. 현재 컴포넌트 식별기는 한국전자통신연구원에서 개발한 EJB 컴포넌트 개발 도구인 COBALT (Component Based Application develOpment Tool) Constructor[7]에 통합되어 동작한다.

이 논문은 다음과 같이 구성된다. 먼저 2절에서는 컴포넌트 개발 프로세스에 대해 간략히 언급하고 3절에서는 컴포넌트 식별기의 구조와 COBALT 도구와의 연계 방법에 대해 다룬다. 4절에서는 행위자 및 객체 연관 그래프와 객체 의존 네트워크에 대해 정의하고 객체 의존 네트워크 기반의 컴포넌트 식별 과정에 대해 자세히 다룬다. 5절에서는 사례 연구로 컴포넌트 식별기를 이용한 인터넷 뱅킹 시스템에 대한 컴포넌트 식별 과정에 대한 간략히 다루고 마지막으로 6절에서는 결론을 맺는다.

2. 컴포넌트 기반 개발(CBD) 프로세스

CBD 프로세스는 일반적으로 재사용 가능한 컴포넌트를 개발하는 컴포넌트 개발(Component Development) 프로세스와 기 개발된 컴포넌트들을 조립하여 새로운 어플리케이션을 개발하는 컴포넌트 기반 소프트웨어 개발(CBSD: Component-Based S/W Development) 프로세스로 구성된다. 도메인 모델을 이용하여 재사용성이 높은 컴포넌트를 식별하는 과정은 컴포넌트 개발 프로세스와 연관이 있으므로 이 논문에서는 컴포넌트 개발 프로세스에 대해서만 설명한다. 그림 1은 컴포넌트 개발 프로세스의 개략적인 설명을 보여주고 있다.

• 도메인 분석 : 도메인 분석은 공통성 및 가변성 분

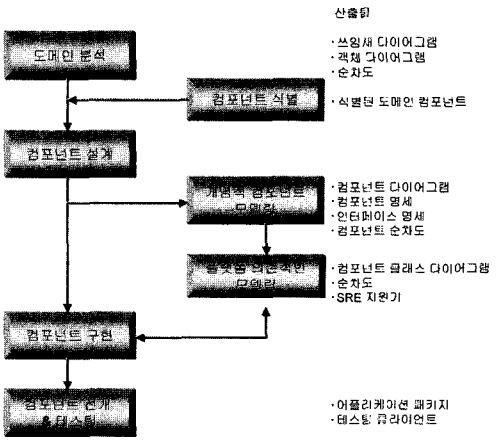


그림 1 컴포넌트 개발 프로세스

석을 통해 도메인 시스템들에 포함되어 있는 객체들을 식별하고 그들간의 의존관계를 파악하여 도메인 모델로 기술한다. 이러한 도메인 모델들은 컴포넌트 식별 과정의 입력 정보로 활용된다. 이 논문에서는 널리 알려진 객체 지향 기법을 이용해 도메인 모델을 기술한다고 가정하며 객체 지향 도메인 모델은 쓰임새 다이어그램, 객체 다이어그램, 순차도로 구성된다고 본다.

- 컴포넌트 식별 : 컴포넌트 식별 과정은 도메인 모델을 바탕으로 재사용 가능한 부분을 추출하여 컴포넌트화 하는 과정이다. 주로 경험이 풍부한 도메인 전문가에 의해 직관적으로 행해져 왔으나 보다 체계적인 식별 방법을 통해 컴포넌트를 추출하는 것이 필요하며 또한 효율적인 적용을 위해 도구 지원이 요구된다.
- 컴포넌트 설계 : 컴포넌트 설계 과정은 개념적인 컴포넌트 모델링 과정과 플랫폼 의존적인 컴포넌트 설계 과정으로 구분한다[10]. 개념적인 컴포넌트 모델링은 식별과정을 통해 얻은 컴포넌트 정보를 바탕으로 컴포넌트의 인터페이스를 명확히 정의하고 컴포넌트 간의 의존성 등을 컴포넌트 다이어그램으로 기술한다. 플랫폼 의존적인 설계는 컴포넌트가 구현되는 플랫폼에 맞게 개념적인 컴포넌트들을 상세 설계하는 과정을 말한다. 예를 들어, 개념적인 컴포넌트를 EJB 아키텍처[11] 상에서 수행시킬 경우는 EJB의 엔티티 빈(Entity Bean), 세션 빈(Session Bean) 등의 플랫폼 특성들을 이용하여 상세 설계 과정을 거치게 된다. 이 단계 컴포넌트 설계의 주된 목적은 수행 플랫폼에 무관한 개념적인 컴포넌트를 통해 컴포넌트의 재사용성을 한층 더 높이기 위함이다.
- 컴포넌트 구현 : 컴포넌트 구현은 플랫폼의 특성을 고려하여 컴포넌트의 내부 비즈니스 로직을 코딩한다.
- 컴포넌트 전개 및 테스팅 : 컴포넌트의 구현이 완료되면 컴포넌트를 패키지로 묶어서 어플리케이션 서버에 전개하고 컴포넌트의 기능들이 정상적으로 동작하는지 여부를 검사하기 위해 다양한 테스팅 과정을 거친다. 이렇게 테스팅 과정을 통과하여 인증된 컴포넌트는 외부로 배포되어진다.

이 논문에서는 컴포넌트 식별 과정만을 제안하며 영역 분석 과정을 포함한 나머지 개발 프로세스들은 *COBALT Constructor* 도구를 통해 수행한다. 즉, 공통성(Commonality) 및 가변성(Variability) 분석 등의 영역 분석 결과로 얻어진 객체 지향 도메인 모델을 COBALT Constructor 도구의 도메인 모델링 도구를 이용하여 나타내고, 컴포넌트 식별기에서는 객체 지향 도메인 모델을 입력받아 후보 컴포넌트를 식별한다. 그리고 컴포넌트들의 상세설계, 구현, 전개 및 테스팅 등의 개발공정은 COBALT Constructor 도구를 통해 이루어진다.

3. 컴포넌트 식별기의 구조

컴포넌트 식별기는 객체 지향 도메인 모델을 분석하여 재사용 가능한 컴포넌트를 식별하는 기능을 수행한다. 컴포넌트 식별기의 입력으로는 쓰임새 다이어그램, 객체 다이어그램, 순차도 정보가 있으며 이들은 COBALT 도구의 도메인 모델러로부터 입력받는다. 그리고 식별된 컴포넌트 정보는 COBALT의 컴포넌트 모델러로 출력되어 상세 설계, 구현, 전개 및 테스팅 과정을 거치게 된다. 그림 2는 컴포넌트 식별기의 구조를 보여준다. 도메인 모델 변환기는 도메인 모델을 입력받아 객체 간의 의존 정보 및 객체 사용 정보를 나타내는 행위자 및 객체(Actor and Object : AO) 연관 그래프로 변환하는 기능을 수행하며 객체 의존 네트워크 생성기는 AO 연관 그래프를 객체 의존 네트워크로 변환하는 기능을 수행한다. 그리고 객체 클러스터링 엔진은 연관성이 높은 객체들을 묶어서 후보 컴포넌트로 식별하는 기능을 수행한다. 객체 사용 예 관리 위저드와 알고리즘 수행 위저드는 사용자 입력을 다루는 GUI 위저드이다.

객체간의 사용예 정보는 주로 도메인 모델의 순차도 정보를 통해서 얻으며 만약 순차도 정보가 없거나 부실할 경우는 사용자가 객체 사용예 관리 위저드를 통해 추가적인 객체 사용예 정보를 입력하거나 기존 객체 사용예를 수정한다. 또한 쓰임새에 대한 가중치를 명시함으로써 사용예에 대한 가중치를 부여한다. 알고리즘 수행 위저드에서는 핵심 객체 지정 임계값(Seed Object

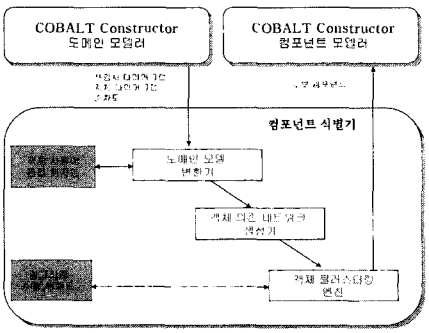


그림 2 컴포넌트 식별기의 구조

Threshold : SOT) 또는 클러스터링 기준 임계값(Clustering Threshold : CT)을 설정할 수 있으며, 사용자가 이들 임계값을 변경시키면서 반복적인 식별 작업을 통해 식별과정을 정제할 수 있다.

4. 컴포넌트 식별 방법

컴포넌트를 체계적인 방법으로 식별하기 위해서는 우선 도메인 모델 정보를 엄격하게 표현하여야 한다. 쓰임새 다이어그램, 클래스 다이어그램, 순차도 정보는 서로 다른 관점에서 기술된 것이므로 우선적으로 이들을 하나로 연계시켜 표현할 필요가 있다. 우선 AO 연관 그래프로 객체간의 구조적인 관계와 객체들간의 사용예 정보들을 표현한다. 그리고 이 정보들은 객체 클러스터링에 부적합함으로 이들을 다시 객체간의 수치화된 의존값으로 표현한 객체 의존 네트워크로 변환한다. 이 절에서는 AO 연관 그래프와 객체 의존 네트워크에 대해 정의하고 이들의 간단한 예제를 다루고 객체 클러스터링 알고리즘을 소개한다.

4.1 도메인 정보의 표현

객체 지향 도메인 모델링은 구조적, 기능적, 행위적인 관점에서 행해진다. 객체 모델링에서는 시스템을 구성하고 있는 핵심 도메인 객체들을 정의하고 이들간의 관계를 정의하며 쓰임새 모델링에서는 시스템의 기능들을 쓰임새로 나타낸다. 그리고 순차도에서는 특정 시나리오에 대한 객체간의 사용 정보 즉, 행위적인 요소를 명확히 정의한다. 이러한 객체 지향 도메인 모델 정보는 여러 관점에서 기술된 것이므로 하나의 표기법으로 나타내고 하나의 통합된 관점으로 합성하는 것이 중요하다. 이 논문에서는 객체간의 의존 가중치 관점에서 도메인 모델 정보를 통합하고자 한다.

먼저, 객체 모델에서는 객체간의 의존 관계가 명확한 일반화(Generalization), 집합연관(Aggregation), 복합연관(Composition) 관계를 뽑아낸다. 하지만 연관관계(Association)와 의존관계(Dependency)에는 도메인 고유의 행위 특성들이 포함되어 있어 구조적인 연관관계 정보만으로 두 객체간의 결합 강도를 객관적으로 가늠하기 어렵다. 대신 객체간의 연관 또는 의존관계는 쓰임새 다이어그램에 의해 유추된 객체간 사용 정보로 나타내거나 순차도에 나타난 객체간 사용 패턴들을 추출하여 표현한다. 그리고 도메인 전문가에게 쓰임새의 중요도를 명시하게 함으로써 순차도에 나타난 객체 사용예의 가중치를 기술한다. 즉, 객체간의 이용 정보의 가중치는 순차도와 연관된 쓰임새의 가중치를 그대로 물려받는다. 이러한 객체간의 구조적 관계 정보, 객체간의 축적된 이용정보, 그리고 가중치 등을 하나의 표기법으로 나타내기 위해 AO 연관 그래프를 제안한다. 다음은 AO 연관 그래프의 정의를 나타낸다.

정의 1. 행위자와 객체간(Actor and Object: AO) 연관 그래프

행위자와 객체(AO) 연관 그래프는 방향 그래프, G = (V, E, w)로 정의되며 아래 조건을 만족한다.

- V = A∪O : 노드의 집합은 행위자의 집합 A와 객체 집합 O의 합집합으로 나타낸다.
- E : 행위자와 객체, 객체와 객체간의 의존성을 아크로 표현한다.
- w(e) : 각 아크는 일반화, 집합연관, 복합연관으로 이루어진 구조적 연관관계를 가지거나 가중치가 있는 사용 유형들(Create-Destroy, Create, Destroy, Update, Reference)을 가진다. 하나의 아크는 여러 개의 사용 유형을 포함할 수 있다.

비록 행위자는 객체간의 의존 정보에는 직접적으로 영향을 미치지 않음에도 불구하고 AO 연관 그래프에서 행위자를 나타내는 이유는 행위자와 객체사이에서 객체 이용 정보가 나타나며 또한 행위자가 객체 이용 시나리오의 출발점 역할을 가지기 때문이다. AO 연관 그래프에서 객체는 실선으로 된 원으로 표시하고 행위자는 객체와 구분하기 위해 점선으로 된 원으로 표시한다. 그림 3은 AO 연관 그래프의 예를 보여준다.

AO 연관 그래프에서 사용되는 객체간 또는 행위자와 객체간의 의존 관계는 세가지 유형의 구조적인 연관관계(Generalization, Aggregation, Composition)와 5가지의 객체 사용 유형(Create-Destroy, Create, Destroy, Update, Reference)으로 표현된다. 그림 3에서와 같이 객체 사용 정보는 객체 사용 유형 가중치에 해당 쓰임새의 가중치를 곱한 형태로 표현된다. 예를 들어,

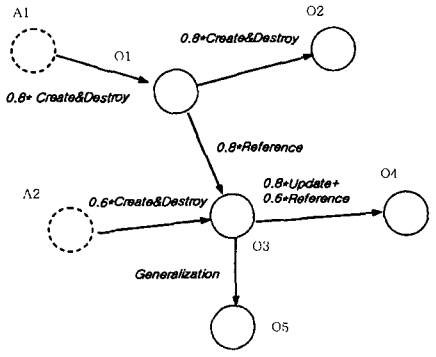


그림 3 AO 연관 그래프의 예

행위자 A1의 쓰임새 가중치를 0.8로 가정하고 객체사용 유형이 "Create-Destroy"일 때 객체 사용 정보는 *"0.8* Create-Destroy"*로 계산된다. 노드 간의 객체 사용 정보는 *"0.8*Update + 0.6*Reference"*와 같이 시나리오에서 나타날 때마다 덧셈을 이용하여 누적된 형태로 표현된다.

비록 AO 연관 그래프가 객체간의 구조적인 관계와 객체 이용관계를 하나의 표기법으로 나타내고 있지만 객체간의 의존성을 비교할 수 없어 객체 클러스터링 과정에 활용하기에는 부족한 점이 있다. 그래서 이러한 의존 정보들을 비교 가능한 가중치 값으로 변환하여 나타낸 것이 객체 의존 네트워크이다. 객체 의존 네트워크에서는 객체간의 의존 정보만을 표현하기 위해 AO 연관 그래프에서 행위자에 해당하는 점선 노드를 생략하고 대신 행위자와 연결된 객체에 셀프 루프(self-loop)를 추가하고 행위자와 객체간의 객체 사용 정보는 셀프 루프 상에 기술한다. 객체 의존 네트워크는 방향성을 가지는 아크에 가중치 값을 두는 가중치 유향 그래프[13]의 하나로 노드에 추가적으로 가중치를 두고 있는 점이 일반 가중치 유향 그래프와 다르다. 객체 의존 네트워크의 정의는 다음과 같다.

정의 2. 객체 의존 네트워크

객체 의존 네트워크는 방향성 그래프 G = (V, E, *imp*, *w*)로서 아래 조건을 만족한다.

- V : 노드는 객체의 집합이다.
- E : 의존 값을 가지는 방향 아크의 집합이다.
- *imp(e)* : 아크의 가중치를 나타낸다.
- *w(v)* : 객체의 가중치 함수로 객체의 중요도를 나타낸다.

그림 4는 객체 간의 의존 정도를 수치 값으로 표현한 객체 의존 네트워크의 예를 보여준다. 객체 간의 의존 값(Dependency Degree: DD)은 의존 관계를 나타내는 아크 위에 표현된다. 그리고 행위자로부터 객체 사용 정보를 나타내는 셀프 루프 또한 의존 값을 가진다. 의존 값은 가장 큰 값을 1.0로 변환한 상대 값으로 0.0~1.0 사이의 값을 가진다. 객체 의존 네트워크에서는 객체 간의 의존 값뿐만 아니라 객체의 중요도(Importance Degree: ID)를 노드 안에 표현한다. 객체의 중요도는 특정 객체가 얼마나 다른 객체에게 중요하게 이용되는지를 나타내는 척도로 해당 객체에 입력되는 아크의 의존 값들을 더한 것이다. 객체의 중요도 또한 0.0~1.0 사이의 상대 값으로 표현된다.

AO 연관 그래프의 누적된 아크 가중치 정보는 "쓰임새가중치*의존유형" 형태의 객체 이용 정보들의 집합으로 표현된다. 이러한 객체간의 의존 값을 수치 값으로 계산하는 방법은 수식 (1)과 같다. 쓰임새별로 특정 객체간의 이용 정보를 분류하여 쓰임새 가중치와 의존유형의 가중치를 곱하여 누적한 것이다. 어떤 쓰임새에서는 특정 객체간의 이용 정보가 없을 수도 있고 어느 쓰임새에서는 여러 개가 있을 수도 있다.

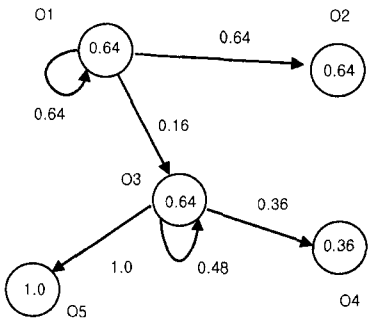


그림 4 객체 의존 네트워크의 예

$$\sum_{i=1}^{\text{쓰임새의개수}} \sum_{j=1}^{\text{객체이용정보의 개수}} W(\text{쓰임새}_i) * Weight(\text{의존유형}_j) \quad (1)$$

표 1 의존 유형에 따른 가중치 값

의존 유형	가중치
집합연관(Aggregation)	1.0
복합연관(Composition)	1.0
생성/소멸(Create/Destroy)	0.8
생성(Create)	0.7
소멸(Destroy)	0.6
변경(Update)	0.3
참조(Reference)	0.2

그리고 각 의존 유형에 따른 가중치 값은 표 1과 같이 나타내고 있다. 표 1의 의존 유형의 가중치는 0.0~1.0 사이의 값으로 의존 강도에 따라 디폴트로 정의한 값이다. 이러한 의존 가중치 값은 사용자에 따라, 적용 도메인 특성에 따라 조금씩 조정 가능하다. 컴포넌트 식별기에서는 의존 유형별 가중치 값을 도메인 특성에 따라 사용자가 유연하게 재조정할 수 있는 옵션을 제공하고 있다. 표 1에서 한 가지 유의사항은 의존 유형에 일반화에 대한 가중치를 나타내고 있지 않다는 점이다. 일반화는 객체간의 가장 강한 결합도를 보이는 것으로 상속받은 객체는 반드시 부모 클래스들을 가지고 있어야 하기 때문에 객체간의 의존관계에는 포함하지 않고 추후 클러스터링 알고리즘에서 특별히 다룬다.

4.2 핵심 객체와 클러스터링 정보

재사용 가능한 시스템의 부분들을 찾아내는 한가지 방법으로는 시스템에서 핵심적인 역할을 수행하는 객체를 찾아서 그 객체와 연관된 부분들을 함께 묶는 방법이 있을 수 있다. 여기에서는 이러한 핵심 객체를 찾는 몇 개의 방법을 고려해 본다. 핵심 객체를 판단하는 하나의 기준으로 외부의 요구를 받아들여 어떠한 시스템의 행위를 시작하는 시작 객체를 들 수 있다. 시작 객체는 행위자로부터 어떠한 요구사항을 받아 들여 내부의 다른 객체들을 호출하여 시스템의 행위를 수행하게 한다. 또 하나의 기준으로는 다른 객체에 공통적으로 중요하게 이용되는 재사용 객체를 들 수 있다. 재사용 객체는 여러 객체들에게 공통으로 중요하게 이용되는 객체이다. 아래는 핵심 객체를 판별하는 기준을 제시한다.

- 셀프 루프를 가지며 핵심객체 판별 임계값(SOT) 이상의 객체 중요도(ID)를 가진 객체이거나,
- 핵심객체 판별 임계값(SOT) 이상의 의존값(DD)을 가진 입력 아크가 두개 이상 있는 객체

만약 SOT의 값이 0.4라고 가정하면, 그림 5에서 객체 O_1와 O_5는 셀프 루프를 가지고 있으며 각각 0.4 보다 큰 객체 중요도를 가지므로 핵심 객체로 식별된다. 그리고 객체 O_3는 두 개 입력 아크의 의존 값이 0.4 이상이므로 두번째 핵심 객체 조건을 만족한다. 그림 5에서와 같이 O_1, O_3, O_5의 핵심 객체들을 식별하고 핵심 객체를 중심으로 밀접하게 연관된 이웃 객체들을 묶는 객체 클러스터링 과정을 수행한다.

객체 클러스터링을 수행하는 간단한 방법은 사용자가 지정한 클러스터링 임계값(Clustering Threshold : CT) 보다 크거나 같은 의존 값으로 핵심 객체와 연결된 객체들을 동일 그룹으로 묶으면 된다. 하지만 이러한 방법으로는 특정 객체만 사용하는 객체(*'몰두 객체'*라 부르자)들을 묶을 수 없다. 그림 5에서 O_2와 O_4는 다른 객체들에 이용되지 않고 오직 객체 O_1과 O_3에 의해서만 이용되는 몰두 객체이다. 이러한 몰두 객체들을 클러스터링 하기 위해서는 클러스터링 기준을 조금 변경하여야 한다. 객체간의 의존값을 곧바로 CT와 비교하지 않고 대상 객체의 중요도로 나눈 상대 의존값 (즉, *"객체간의 DD"* / *"대상객체의 ID"*)으로 비교하면 몰두 객체들은 모두 1.0 값을 가지므로 클러스터링에 포함될 수 있다. 이와 같이 상대 의존값으로 클러스터링 기준을 바꿀 경우 그림 5의 몰두 객체 O_2와 O_4는 각각 핵심객체 O_1과 O_3에 포함된다.

4.3 객체 클러스터링 알고리즘

객체 클러스터링 알고리즘은 앞서 설명한 핵심 객체

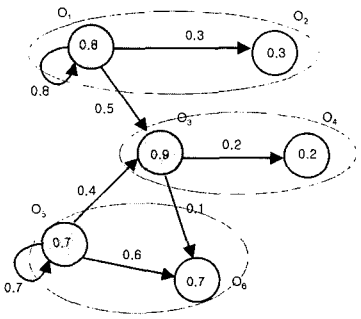


그림 5 핵심 객체의 결정과 객체 클러스터링

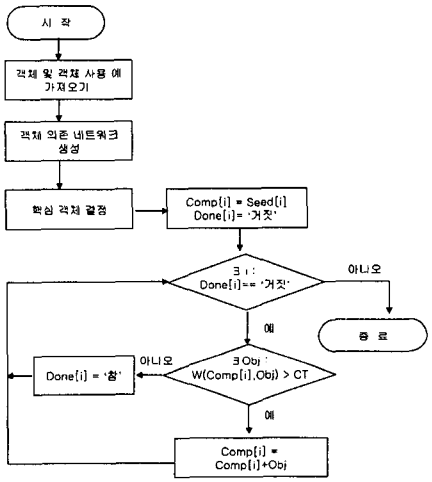


그림 6 핵심 객체 알고리즘의 흐름도

를 중심으로 밀접하게 연관된 객체들을 묶는다. 핵심 객체 선정을 위해 SOT 값을 사용자로부터 입력받고 객체 클러스터링을 위해 CT 값을 입력받아 수행하며 반복적인 임계값 설정을 통해 컴포넌트 식별 과정을 반복할 수 있다. 그림 6은 객체 클러스터링 알고리즘의 흐름도를 나타내고 있으며 아래와 같은 과정으로 수행된다.

- **단계 1**: 객체 모델 정보를 입력받는다. 여기에서 상속관계, 집합연관, 복합연관만을 추출하여 AO 연관 그래프로 나타낸다. 그리고 순차도 정보로부터 객체 사용 정보를 추출하여 AO 연관 그래프에 추가한다. 생성과 소멸에 관련된 유형 정보(Create-Destroy, Create, Destroy)는 순차도에서 추출하여 나타낸다. 하지만 변경(Update)와 참조(Reference) 정보는 순차도에 나타나 있지 않으므로 오퍼레이션의 이름으로 유추하여 분류한다. 예를 들어, set-XXX(), updateXXX(), changeXXX(), register-

XXX(), modifyXXX() 등과 같이 변경을 의미하는 단어가 이름에 포함되어 있으면 변경(Update) 정보로 나타내고 나머지 경우는 참조(Reference) 정보로 나타낸다. 사용자는 사용 예 관리 위저드를 통해 새로운 객체 사용 정보를 추가하거나 기존 객체 사용 예 정보를 수정할 수 있으며 또한 쓰임새의 가중치를 지정하며 객체 사용 가중치를 나타낸다.

- **단계 2**: AO 연관 그래프의 모든 아크들에 대해, 누적된 객체 사용 정보에 대해 표 1의 유형 가중치를 참조하고 수식 (1)의 계산 방법을 이용하여 객체간의 의존값(DD)를 계산한다. 그리고 이러한 DD 값을 바탕으로 각 객체의 중요도 값(ID)을 계산한다. 그리고 행위자 노드를 없애고 관련 객체에 셀프 루프를 추가하여 최종적으로 객체 의존 네트워크를 완성한다.
- **단계 3**: 핵심 객체를 판별하기 위해 객체 의존 네트워크의 모든 객체를 둘러보며 SOT를 이용하여 앞절에서 설명한 핵심 객체 조건을 만족하는지 여부를 검사한다. 핵심 객체로 판별되면 핵심 객체 배열인 Seed 배열에 기록한다.
- **단계 4**: 각 핵심 객체를 후보 컴포넌트의 초기 값으로 설정한다. 그리고 각 컴포넌트에 연결된 객체들에 대한 검색 종료 여부를 나타내는 조건 배열인 Done을 '거짓'으로 초기화한다. 조건 배열이 참인 경우는 해당 컴포넌트로부터는 더 이상 객체 검색이 필요치 않은 경우를 나타낸다.
- **단계 5**: 식별 과정을 종료시킬지 여부는 모든 컴포넌트의 조건 배열이 참이냐에 따라 결정된다. 조건 배열이 거짓인 컴포넌트(즉, 아직 검사할 객체가 남아 있는 컴포넌트)를 선택하여 연결된 객체가 포함될지 여부를 판단한다. 더 이상 조건 배열이 거짓인 컴포넌트가 없으면 식별 과정을 종료하고 나머지 경우는 단계 6으로 진행한다.
- **단계 6**: 컴포넌트에 포함된 객체에 연결된 다른 객체 중에서 클러스터링 임계값(CT) 이상의 상대 의존값을 가진 객체가 있는지 여부를 판단하고 이러한 객체가 있으면 컴포넌트에 추가한다. 만약 이러한 객체가 존재하지 않으면 해당 컴포넌트의 조건 배열을 참으로 설정한다. 그리고 단계 5로 올라가 반복적인 검사 과정을 거친다.

후보 컴포넌트에 포함되는 객체들이 결정되면, 컴포넌트 식별의 마지막 단계로 컴포넌트 내부의 각 객체에 대해 상속관계에 있는 모든 부모 객체들을 동일 컴포넌트에 포함시킨다. 객체지향 언어에서는 상속받은 객체를 수행하기 위해서는 모든 부모 객체들이 있어야 하기 때문이다.

5. 사례 연구

이 절에서는 실제 도메인 모델에 컴포넌트 식별기를 적용하여 식별 기능의 타당성을 검사하고자 한다. 사례 연구로는 2000년에 한국소프트웨어 컴포넌트 컨소시엄(KCSC) 주관으로 수행되었던 파일럿 프로젝트인 인터넷 기반 뱅킹 시스템의 도메인 모델을 사용한다. 인터넷 기반 뱅킹 시스템의 기능으로는 인터넷을 통한 시스템 관리, 고객 업무, 수신 업무, 일계 업무 등으로 구성되어 있다. 파일럿 프로젝트 수행 당시, 작성된 객체 모델, 쓰임새 모델, 순차도 모델들을 COBALT 도구의 도메인 모델러를 이용하여 다시 모델링한다. 그림 7은 도메인 분석 과정을 거쳐 식별된 도메인 객체들을 클래스 다이어그램으로 나타내고 있다. 주요 도메인 객체로는 *Customer, PrivateCustomer, CorporateCustomer, Account, OrdinaryAccount, BookKeeping, TX, OrdinaryTX, CustomerTX, Journaling, DepositJournal, CustomerJournal*이 있다. 그림 7의 클래스 다이어그램에서는 객체간의 상속성과 연관성만을 간단히 나타내고 있다.

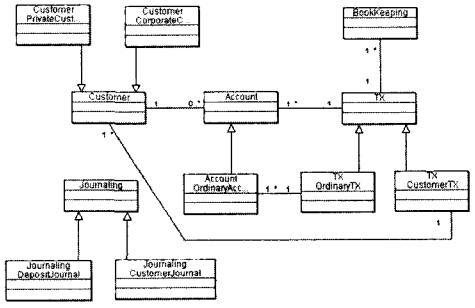
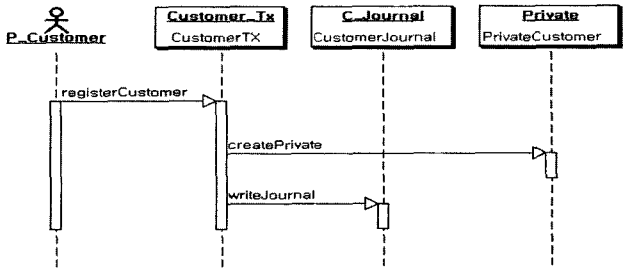
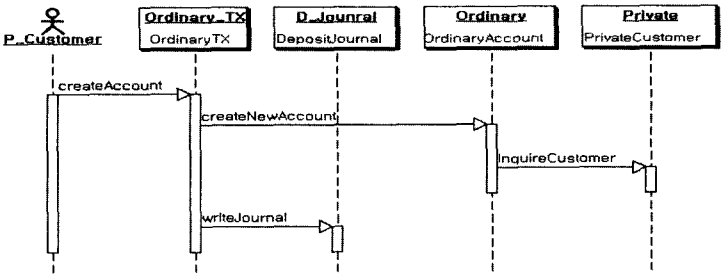


그림 7 뱅킹 시스템의 도메인 객체들을 나타내는 클래스 다이어그램

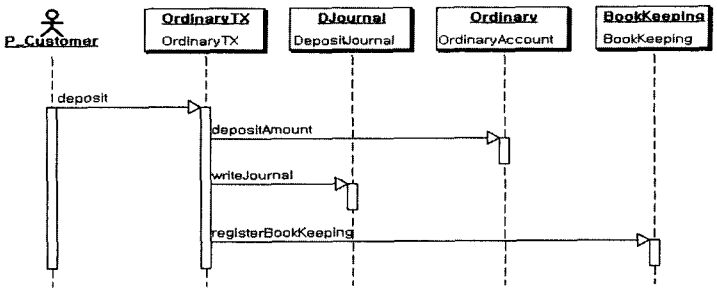
뱅킹 시스템에는 "Register Private Customer", "Create Account", "Deposit", "Withdraw" 등의 13개의 쓰임새가 있으며 쓰임새를 구체화하여 객체간의 의존관계를 나타내는 순차도는 그림 8에 나타난 3개의 순차도를 포함하여 18개의 순차도가 있다. 그림 8에서는 "Register Private Customer", "Create Account", "Deposit"의 쓰임새에 해당하는 개인 고객 등록 순차도, 새로운 계좌 생성 순차도, 계좌 예금 순차도를 보여준다.



(a) 개인 고객 등록 순차도



(b) 새로운 계좌 생성 순차도



(c) 계좌 예금 순차도

그림 8 뱅킹 시스템의 순차도

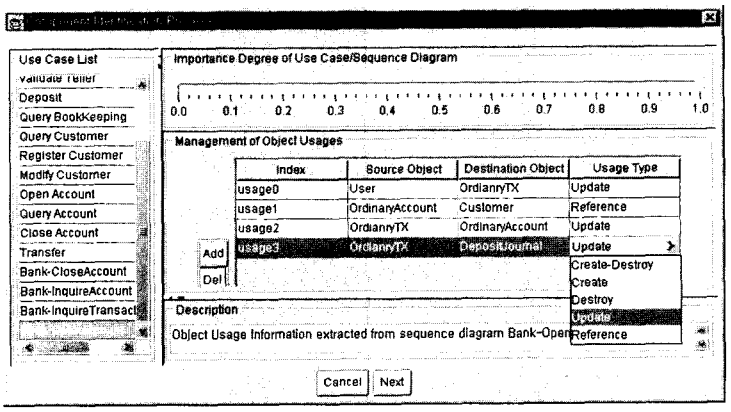


그림 9 사용예 관리 위저드의 실행 화면

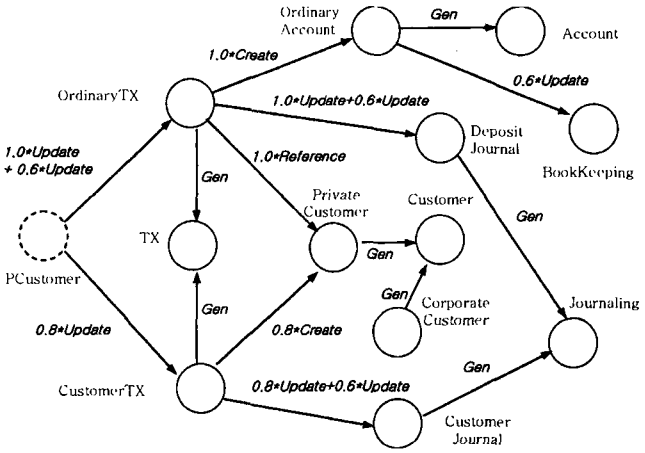


그림 10 AO 사용예 그래프의 예

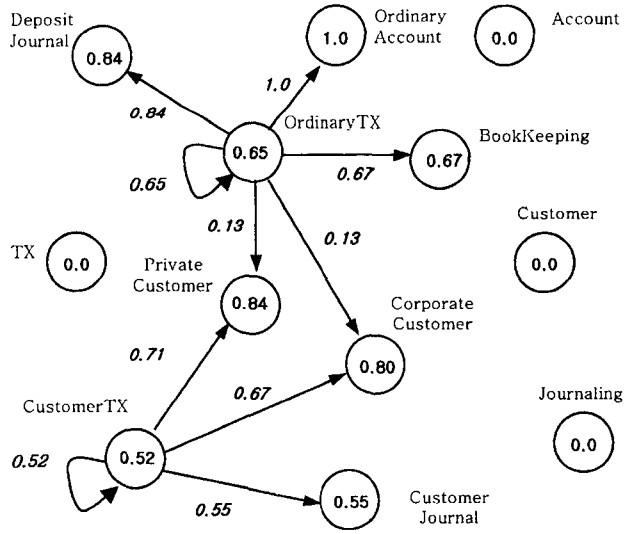


그림 11 뱅킹 시스템의 객체 의존 네트워크

뱅킹 시스템의 객체 지향 도메인 모델들은 COBALT 도구의 도메인 모델러를 통해 구성되며 이들 모델 정보들은 자동으로 컴포넌트 식별기에 입력된다. 도메인 전문가는 그림 9에 나타난 사용예 관리 위저드를 통해 사용예에 대한 정보 변경 및 추가 기능을 수행할 수 있다. 또한 그림 9에서 보는 바와 같이 슬라이딩 바를 통해 각 쓰임새 혹은 순차도의 중요도를 입력할 수 있다. 그림 9는 새로운 계좌를 만드는 순차도에서 추출한 객체 이용 정보를 보여주고 있으며 순차도의 중요도는 1.0으로 설정되어 있다.

사용예 관리 위저드를 통해 입력된 사용예 정보는 그림 10의 AO 사용 그래프로 표현된다. 그림 10는 그림 8에 나타난 3개의 순차도의 사용예 정보를 AO 사용 그래프로 변환한 것으로 각 순차도의 중요도를 각각 1.0, 0.8, 0.6으로 설정한 것으로 가정하고 있다.

AO 사용예 그래프는 내부적으로 객체 클러스터링 알고리즘을 수행하기 위해 객체 의존 네트워크로 변환된다. 그림 11은 뱅킹 시스템에 존재하는 모든 순차도 정보들을 AO 사용예 그래프로 나타내고 이를 다시 객체 의존 네트워크로 변환한 것이다. 객체 의존 네트워크에 나타난 가중치들은 최대 의존값으로 나눈 상대값으로 0.0~1.0 사이의 값을 가진다. 객체 의존 네트워크에서는

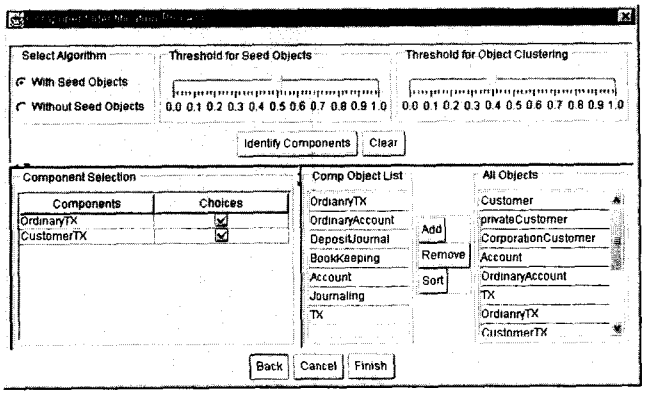


그림 12 알고리즘 수행 위저드

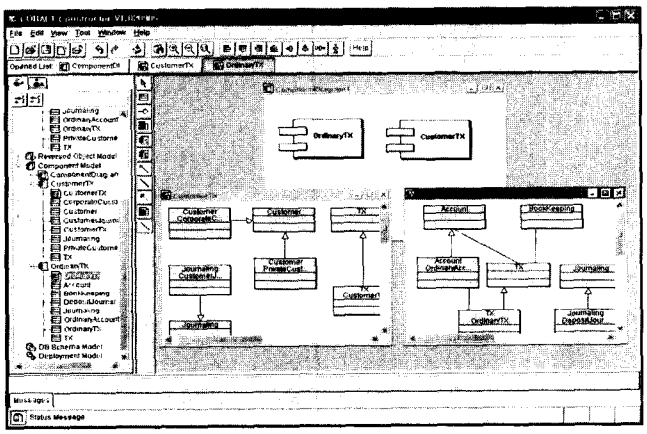


그림 13 컴포넌트 다이어그램의 예

일반화 관계를 표현하고 있지 않으므로 그림 11에서 Account, Customer, Journaling, TX 객체들은 의존관계를 가지지 않고 있다.

객체 클러스터링 알고리즘을 수행하기 위해서는 그림 12의 알고리즘 수행 위저드를 통해 핵심 객체를 선정하기 위한 임계값(SOT)과 객체 클러스터링 임계값(CT)을 입력 받는다. 그림 12와 같이 SOT 값을 0.5로 CT 값을 0.4로 설정하였다고 가정하면, 그림 11의 객체 의존 네트워크에서는 OrdinaryTX와 CustomerTX가 핵심 객체로 식별된다. 그리고 핵심 객체를 중심으로 상대적인 의존값이 CT보다 큰 객체들을 묶어 나간다. OrdinaryTX 핵심 객체에는 DepositJournal, OrdinaryAccount, BookKeeping 객체가 묶여지고 CustomerTX 핵심객체에는 PrivateCustomer, CorporateCustomer, CustomerJournal 객체가 묶여진다. 클러스터링 알고리즘 수행 결과는 그림 12의 아래 부분에 나타난다. 예제에서는 OrdinaryTX 컴포넌트와 CustomerTX 컴포넌트가 식별되었다. 각 컴포넌트에 속하는 객체들은 아래 부분의 오른쪽 창에 나타나 있다. 이 부분에서는 상속관계에 있는 객체들이 모두 포함되어 있다.

컴포넌트 식별기에서 사용자는 임계값들을 조정하면서 반복적으로 알고리즘을 수행할 수 있으며 **Back** 버튼을 이용하여 이전 단계인 사용예 관리 위저드로 되돌아 가서 추가 정보를 입력하고 다시 알고리즘 수행 과정을 반복할 수 있다.

알고리즘 수행 위저드에서 **Finish** 버튼을 누르면 식별된 후보 컴포넌트는 COBALT 도구의 컴포넌트 모델러의 입력으로 들어가서 그림 13과 같은 컴포넌트 다이어그램을 형성하게 되고 내부에 포함된 객체들은 컴포넌트 클래스 다이어그램에 나타난다. 그림 13에서 아래 클래스 다이어그램은 각 컴포넌트에 속하는 객체들의 관계를 보여준다. 이후 단계인 컴포넌트 인터페이스의

상세화, 각 컴포넌트의 상세 설계, 구현, 전개, 시험 등은 COBALT Constructor 도구를 통해 이루어진다.

6. 결론 및 향후 연구

컴포넌트 식별 프로세스는 컴포넌트 재사용성과 밀접하게 연관되어 컴포넌트 개발 프로세스의 핵심부분이 되나, 아직까지 도구화하기에 충분한 체계적인 절차와 방법을 제공하지 못하고 있었다. 이 논문에서는 객체 지향 도메인 모델을 바탕으로 체계적인 컴포넌트 식별 방법을 제안하고 이를 효과적으로 지원하는 도구인 컴포넌트 식별기를 설계하고 구현하였다. 컴포넌트 식별기의 입력으로는 널리 사용되고 있는 객체 지향 모델을 이용하므로 이미 만들어 놓은 객체 지향 도메인 모델을 그대로 이용할 수 있는 장점이 있다. 그리고 컴포넌트 식별기는 컴포넌트 개발 도구인 COBALT 도구와 연계하여 동작하므로 도메인 모델링, 컴포넌트 모델링, 컴포넌트 설계 및 구현 등의 컴포넌트 개발 전 단계를 체계적이고 유기적으로 지원할 수 있게 되었다. 컴포넌트 식별기는 도메인 전문가의 경험과 직관에 기초한 식별 방법을 대치하기 위함이 아니라 보다 정확한 수치화된 객체 의존값들을 제시함으로써 도메인 전문가의 식별 과정을 보완해주는 보조 도구로 활용될 수 있으리라 생각된다.

현재 여러 도메인에서 작성된 객체 지향 모델들을 컴포넌트 식별기에 적용하여 후보 컴포넌트를 식별하는 과정을 수행하면서 식별 알고리즘을 조율하고 있다. 향후 연구로는 식별 과정에 사용자와의 상호작용 부분에 좀더 많은 유연성을 부여하여 사용자가 손쉽게 도구를 이용할 수 있도록 하고 객체 지향 도메인 모델에서 보다 많은 유용한 객체 간의 연관 관계 정보들을 추출하여 객체간의 의존 관계를 더욱 명확하게 기술하는 방법에 대한 연구가 필요하다. 또한 컴포넌트 개발 도구인 COBALT Constructor와 함께 개발된 비쥬얼한 컴포넌트 조립 도구인 COBALT Assembler[14] 에 컴포넌트 식별기를 통합시켜 아키텍처 기반의 Plug-&-play 방식의 컴포넌트 조립 도구에도 영역 분석 기능을 추가하여 CBSD 프로세스의 전 단계 기능을 지원하도록 할 것이다.

참 고 문 헌

[1] D. F. D'Souza and A. C. Wills, *Objects, Components, And Frameworks with UML - the Catalysis approach*, Addison Wesley, 1999.

[2] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, Addison Wesley, 1999.

[3] 최미숙, 윤용익, 박재년, "RUP 기반의 컴포넌트 식별 방법에 관한 연구", 정보처리학회논문지 D, Vol. 9-D, No. 1, 2002년 2월.

[4] Computer Associates, "COOL:Joe 2.0 Product Descriptions," *http://www.cai.com/products/cool/joe/cooljoe_pd.pdf*, 2001.

[5] TogetherSoft, "Features of Together," *http://www.togethersoft.com/products/controlcenter/features.jsp*, 2001.

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

[7] M. J. Kim, W. J. Lee and G. S. Shin, "Development of COBALT (COmponent-Based AppLication deveLopment Tool) for Modeling and Constructing EJB based Components," IASTED-AI2002, Feb. 2002.

[8] Object Management Group, "CORBA Components," *http://www.omg.org*, March 1999.

[9] Object Management Group, "Introduction to OMG's Unified Modeling Language(UML™)," *http://www.omg.org/gettingstarted/what_is_uml.htm*, 2002.

[10] 이우진, 김민정, 정양재, 윤석진, 최연준, 이지현, 신규상, "EJB 기반 컴포넌트의 모델링 및 생성지원 도구 개발", 정보처리학회논문지 D, Vol. 8, No. 6, 2001년 8월.

[11] R. Monson Haefel, *Enterprise JavaBeans*, second edition, O'Reilly, 2000.

[12] P. Herzum and O. Sims, *Business Component Factoring: A Comprehensive Overview of Component Based Development for Enterprise*, John Wiley & Sons. Inc., 2000.

[13] J.A. Bondy and U.S.R. Murty, *Graph Theory with Application*, Second Edition, North Holland, 1976.

[14] You-Hee Choi, Oh-Cheon Kwon and Gyu-Sang Shin, "An Approach to Composition of EJB Components Using C2 style," *Proceedings of the 28th EUROMICRO Conference(EUROMICRO 2002)*, Dortmund, Germany, Sep. 2002.

이 우 진

1992년 경북대학교 컴퓨터과학과 졸업(학사). 1994년 한국과학기술원 전산학과 졸업(공학석사). 1999년 한국과학기술원 전산학과 졸업(공학박사). 1999년~2002년 2월 한국전자통신연구원 선임연구원 2002년 3월~현재 경북대학교 컴퓨터과학과 전임강사. 관심분야는 컴포넌트 공학, 요구 공학, Petri nets, 실시간 시스템 모델링 및 분석

권 오 천

1994년 영국 Teesside 대학교 대학원 S/W공학과 졸업(공학 석사). 1998년 영국 Durham 대학교 대학원 전산과학과 졸업(공학 박사). 1991년 미국 RTP IBM 연구소 객원 연구원. 1993년 시스템공학연구소 선임연구원. 1998년~현재 ETRI 컴퓨터·소프트웨어연구소 S/W·컨텐츠연구부 책임연구원. 관심분야는 S/W 재사용, 컴포넌트 기반 개발(CBD), MDA, 제품계열(Product Line)