CBD 에서 컴포넌트 추출을 위한 자동화 도구 구현

한만집, 장수호, 김수동
숭실대학교 대학원 컴퓨터학과
{mjhan, shchang}@otlab.ssu.ac.kr, sdkim@ssu.ac.kr

Implementing a Tool to Automate Component Identification in CBD

Man Jib Han, Soo Ho Chang, and Soo Dong Kim
Dept. of Computer Science, Soongsil University

요 약

컴포넌트 기술은 소프트웨어를 개발하는데 있어서 재사용성을 높이는 효과적인 기술로 널리 사용되고 있다. 특히, 이 기술은 응용프로그램을 개발하는데 있어서 모듈을 조립하는 개발 패러다임의 변화에 공헌이 있다. 효과적인 사용을 위해서, 컴포넌트는 응용프로그램이 원하는 기능을 조립이 용이하게 제공하여야 한다. 그러나 이러한 컴포넌트의 기능을 할당하여 컴포넌트를 추출하는 방법은 도메인 전문가에 의하거나 Ad-hoc 방식으로 이루어 지고 있다.

본 논문에서는 컴포넌트의 기능성 추출을 위해 프로세스와 이를 구현한 툴을 보여주어 효과적인 기능단위의 컴포넌트를 추출될 것으로 기대된다.

1. 서론

컴포넌트 기반 개발 (CBD) 기술은 대표적인 재사용기술로 보편화되고 있다. CBD에서 재사용의 기본 단위는 컴포넌트로서 도메인의 공통적인 기능을 제공한다. 효과적인 재사용을 위해서 컴포넌트는 주변 컴포넌트와의 결합도가 낮고 그 자체의 응집도는 높아야 한다. 도메인의 개발요구사항에서 추출된 컴포넌트의 품질이 CBD 프로젝트에서 중요한 성공 요소이다.

도메인의 공통 컴포넌트 추출을 위한 여러 기법들이 개발되어 있으며, 이들은 대부분은 개발자가 정해서 적용해야 하는 여러 개의 가중치(Weight Value)를 사용하고 있다. 따라서, 이 기법들을 개발자가 수작업으로 적용하는데 많은 노력이 소요되고, 가중치를 조정할 때 마다 기법을 다시 적용해야 하는 등 실용적 측면에서 적용이 제한적이었다. 특히, 유즈케이스나 클래스가 100여 개 이상이 되는 중, 대형 과제에 이 기법을 수작업으로 적용하는 것은 매우 어렵다. 또한, 개발자의 경험과 도메인 지식을 기반으로 수작업의 계산에 의해 추출된 컴포넌트의 품질, 즉 응집도와 결합도도 향상하기 어렵다. 이런 문제들을 해결하기 위한 가장 효과적인 방법은 잘 정립된 기법 즉 알고리즘을 기반으로 자동화된 도구를 개발하여 이를 이용한 컴포넌트의 자동 추출이다.

본 논문에서는 컴포넌트 식별을 위한 대표적 기법들 [1],[2],[3] 의 공통적인 알고리즘을 채택한 자동화 도구의 설계 및 구현을 소개한다. 2장에서는 컴포넌트 추출을 위한 기존의 관련연구들을 조사하고, 3장에서 양질의 컴포넌트 추출을 위한 체계적인 프로세스 및 핵심 알고리즘을 제안하며 4장에서는 이를 구현한 설계와 사례 연구를 보여준다.

2. 관련 연구

Hemant Jain의 연구에서는 컴포넌트를 추출하기 위해 필요한 요소로서 정적 관계(Static Relationship)과 동적 관계(Dynamic Relationship)을 제안한다[4]. 정적 관계는 클래스간의 연관관계의 수와 가중치로 계산하며 동적 관계는 클래스와 클래스를 사용하는 유즈케이스, 유즈케이스의 가중치, 실행시의 메시지 전송 수를 이용하여 계산한다. 이렇게 Jain의 연구에서는 클래스간의 관계의 정도에 따라 묶어 컴포넌트를 식별하는 방법을 제안하고, 관계의 정도를 계산하기 위한 구체적인 공식도 제안한다. 그러나 이 방법론에는 컴포넌트를 식별하기 위한 자동화는 되어 있지 않다.

[1]방법론은 도메인 분석단계부터 공통 기능성을 추출하여 컴포넌트를 생성하는 클러스터링 알고리즘, 컴포넌트 설계, 명세의 전반적인 프로세스와 지침을 제안하고 있다. 또한 이 방법론에서 제안하는 컴포넌트 클러스터링 방법은 유즈케이스간의 관계성의 정도에 따라 유즈케이스를 클러스터링하며 클러스터링 된 유즈케이스와 클래스와의 관계에 따라 클래스를 할당하여 컴포넌트를 추출한다. 유즈케이스 모델과 클래스 모델은 후보 컴포넌트를 식별하기 위해 사용 된다. 컴포넌트를 식별하기 위해 두 단계의 클러스터링 알고리즘을 사용한다. 첫 단계는 유즈케이스를 묶는 것이고, 두 번째 단계는 유즈케이스와 관련된 클래스들을 유즈케이스/클래스 측정 표를 통하여 컴포넌트 안으로 묶는 것이다. 유즈케이스/클래스 측정 표는 CRWD(Create /Read /Write /Delete) 관계를 통하여 만들어진다. 결정된 측정을 사용해서, 하나의 컴포넌트에 대해 각 클러스터를 정의한다. 비즈니스 객체들은 보통 영구적인 데이터를 가지고, 시스템 레벨의 객체나 컨트롤러로부터 호출된다. 그래서 비즈니스 객체들을 적합한 컴포넌트에 할당하는 것이 필요하다. 또한 이 방법론을 적용할 수 있는 도구가 필요하다.

3. 컴포넌트 추출 프로세스 및 알고리즘

3.1. 컴포넌트 추출 프로세스

본 절에서는 효율성 있는 컴포넌트를 추출하기 위한 프로세스를 그림 1과 같은 방식을 기반으로 한다[3].

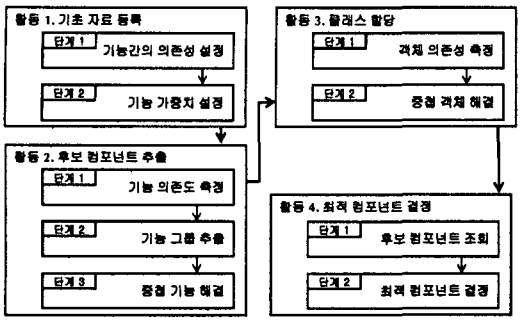

그림 1. 컴포넌트 추출을 위한 프로세스

활동 1. 기초 자료 등록에서는 기초 자료는 기능간의 의존성을 측정하기 위한 요소로, 서브시스템, 액터, 클래스, 유즈케이스를 제안한다. 활동 2. 후보 컴포넌트 추출에서는 관련도가 높은 유즈케이스들을 묶어 기능 그룹을 생성한다.활동 3. 클래스 할당에서는 추출된 기능 그룹에 클래스를 할당하여 후보 컴포넌트를 생성한다. 활동 4. 최적 컴포넌트 결정에서는 최종 컴포넌트를 결정하고 선택된 컴포넌트를 생성한다.

3.2. 알고리즘

본 절에서는 전체 프로세스 중 활동 2의 단계 1인 기능 그룹을 추출하기 위해 기준이 되는 기능 의존도를 측정하는 알고리즘에 대하여 설명한다. 측정하는 알고리즘은 다음과 같다.

유즈케이스 U_i와 U_j가 같은 서브시스템 내에 속하면 U_i와 U_j는 서로 연관관계가 있다. 서브시스템 항목에서는, 이러한 분류에 따라 U_i와 U_j가 같은 서브시스템에 속하면 U_i와 U_j에 대한 서브시스템 값은 1값을 같은 서브시스템에 속하지 않으면 0값을 가진다.

U_i와 U_j를 같은 액터가 초기화 하면 U_i와 U_j는 연관관계가 있다. 액터 항목에서는, U_i와 U_j의 액터가 같은지의 여부에 따라 같으면 U_i와 U_j에 대한 액터 값은 1값을 다르면 0값을 가진다. 유즈케이스를 초기화 할 수 있는 액터가 여러 개일 경우는 (U_i 액터 ∩ U_j 액터)의 수/ (U_i 액터 ∪ U_j 액터)의 수의 값을 할당한다.

U_i와 U_j가 공유하는 데이터를 가지면 U_i와 U_j는 서로 연관관계가 있다. 이러한 영향을 연관관계 값에 할당하기 위해 (U_i 객체∩ U_j 객체)의 수/ (U_i 객체∪ U_j 객체)의 수로 계산한다. U_i와 U_j가 모든 데이터를 공유하면 1값을 가지고 공유하는 데이터가 없다면 0값을 가진다.

U_i와 U_j가 <<extend>>나 <<include>>또는 <<generalization>>으로 연결되어 있다면 U_i와 U_j는 서로 연관관계가 있다. U_i와 U_j가 관계가 있다면 1값을 가지고 연결되어 있지 않으면 0값을 가진다.

끝으로 측정된 각 요소에 대한 할당치에 대해 주어진 가중치를 곱하여 요소에 대한 U_i와 U_j의 기능 의존도를 구한다.

4. 프로그램 설계 및 구현

본 장에서는 프로그램의 설계에 대하여 설명한다. 프로그램의 클래스 다이어그램은 다음과 같다.

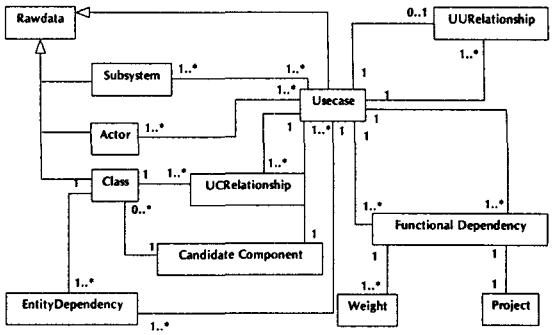

그림 2. 시스템의 클래스 다이어그램

Subsystem, Class, Actor, Usecase는 기초 자료를 등록하고 등록된 기초자료와 유즈케이스간의 관계에 대한 데이터를 관리하는 클래스이다. 그리고 Rawdata는 기초 자료의 상위 추상 클래스로 기초 자료가 가지는 공통 기능과 속성을 가진다. UURelationship는 유즈케이스와 유즈케이스 간의 관계와 기능 의존도에 대한 정보를 관리하는 클래스이다. UCRelationship는 유즈케이스와 클래스 간의 공유 데이터의 관계와 객체 의존도에 대한 정보를 관리하는 클래스이다. Weight, FunctionalDependency는 기초 자료를 이용하여 기능 의존도를 구하고 기능 그룹을 만들기 위한 클래스이다. EntityDependency는 기능 그룹에 대한 객체 의존도와 클래스를 할당 해주기 위한 클래스이다. CandidateComponent는 최종적으로 나온 후보 컴포넌트들의 데이터를 관리하는 클래스이다.

다음은 활동 3의 산출물인 후보 컴포넌트를 생성하는 알고리즘이다. 이 알고리즘의 다음과 같다. 처음 단계는 모든 유즈케이스를 찾은 것이다. 다음 단계는 찾은 유즈케이스들을 이용하여 각 유즈케이스마다 관련된 유즈케이스의 기능적 의존도 값이 입력한 T 값보다 큰 관계만을 뽑아 기능 그룹을 생성한다. 다음 단계는 모든 클래스를 찾고, 찾은 각 클래스마다 가장 높은 객체 의존도를 가지는 기능 그룹에 할당함으로써 모든 후보 컴포넌트들을 생성한다.

```
computeCandidateComp(Weight fdw, Weight edw, float t)
{
  // 계산하기 위해 필요한 변수들
  ...
  //T 값보다 큰 값을 가지는 연관된 유즈케이스 선별
  allUsecase = Usecase.searchByProject();
  for(int i = 0; i < allUsecase.Count; i++){
        ucList = clusterUsecaseByT(temp[i],fdw,t);
        ucListSet.Add(ucList.Clone());
  }
  //중첩된 유즈케이스의 집합 제거
  compUcList = removeSubset(ucListSet);
  //후보 컴포넌트 생성
  for(int i = 0; i < compUcList.Count; i++)    {
        cc = new CandidateComponent(i,t,fdw,edw);
        cc.UCList = (ArrayList)compUcList[i];
        cComp.Add(cc);
  }
  // 클래스마다 연관된 유즈케이스 선별
  allClass = Class.searchByProject();
```

```
for(int i = 0; i < allClass.Count; i++){
        cList = clusterClass(temp[i],edw);
        cListSet.Add(cList.Clone());
}
// 컴포넌트에 클래스 할당
assignClass(cComp,cListSet,edw,t);
// 최종 후보 컴포넌트
return cComp;
}
```

5. 도구 실행 사례

본 절에서는 대여관리 도메인을 사례연구로 컴포넌트를 자동 추출하는 과정을 나타낸다.

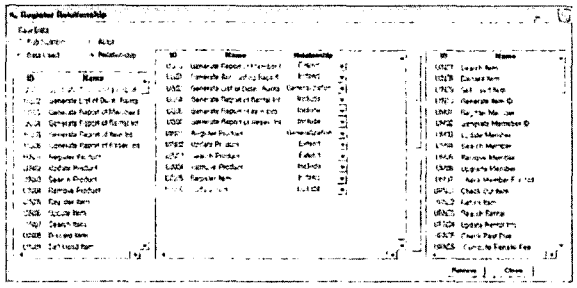

그림 3. 기초자료 관계 등록

그림 3은 활동 1의 작업으로 유즈케이스 간의 기능적 의존도 (Functional Dependency)를 산출할 수 있는 4가지의 요소 값 [3]을 입력한다. 그림 3의 좌측 상단에 이 4가지 요소를 선택하여 입력하게 하였다. 내부에는 3개의 목록이 있는데 왼쪽에서부터 첫 번째 목록은 의존성이 설정될 유즈케이스의 리스트이고, 두 번째 목록은 실제 의존성을 가질 자료이다. 마지막 목록은 의존성을 연결할 수 있는 후보 자료들이다.

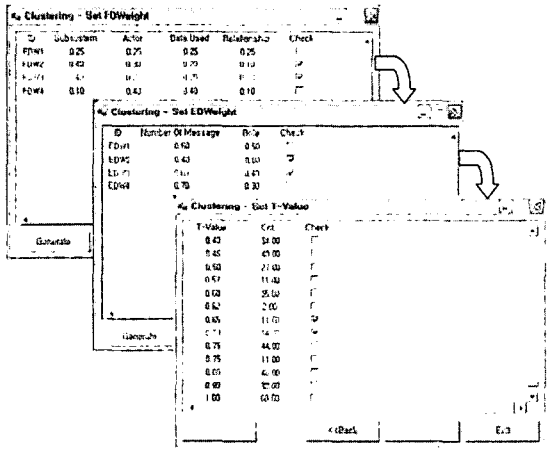

그림 4. 후보 컴포넌트 추출

그림 4는 활동 2와 3의 작업으로 후보 컴포넌트를 생성하고자 하는 기능적 가중치, 객체 가중치, 그리고 T 값을 선택하여 후보 컴포넌트를 생성하는 일련 된 작업이다. 첫 번째 작업에서 기능적 가중치를 선택하여 모든 기능적 의존도를 생성한다. 다음 작업에서 객체 가중치를 선택하여 객체 의존도를 생성한다. 마지막 작업에서는 T값을 정하여 기능적 의존도와 비교하여 기능 그룹을 생성하고 객체 의존도에 의하여 기능 그룹에 클래스를 할당함으로 후보 컴포넌트를 생성한다.

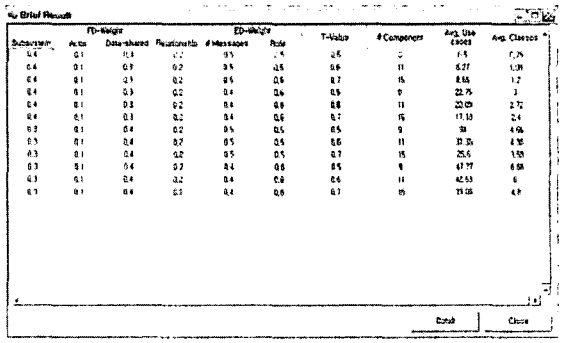

그림 5. 후보 컴포넌트 요약 정보

그림 5은 활동 4의 작업의 결과로부터 개발자에 맞는 최적 컴포넌트를 선택함으로써 컴포넌트를 결정하는 작업이다. 만들어진 후보 컴포넌트들을 목록을 각 가중치와 T값에 따라 표시하여 개발자가 원하는 최적의 컴포넌트 목록을 선택할 수 있다.

6. 결론

컴포넌트 기반 소프트웨어 개발 기술은 재사용 가능한 컴포넌트를 조합하는 방식의 응용프로그램 개발로 소프트웨어 개발의 경제적 효과를 높여주는 기술로 정착되고 있다. 이러한 용도로 사용되는 컴포넌트는 그 품질에 따라 재사용성이 달라질 수 있다. 따라서 컴포넌트의 식별기술은 소프트웨어 높은 재사용을 위해 필요한 중요한 활동 중에 하나이다.

본 논문에서는 컴포넌트 식별을 위한 프로세스와 이를 구현한 툴을 제안하였다. 이러한 툴을 이용하여 컴포넌트 추출 효과적으로 개발할 수 있고 그에 대한 재사용성의 효과, 그리고 개발 비용감소가 기대된다.

7. 참고문헌

[1] Cho, E., Kim, S., and Rhew, S., "A Domain Analysis and Modeling Methodology for Component Develoment," International Journal of Software Engineering and Knowledge Engineering, Jan. 2003

[2] Choi, S., Chang, S., and Kim, S., "A Systematic Methodology for Developing Component Frameworks," Lecture Notes in Computer Science 2984, Proceedings of 7th Fundamental Approaches to Software Engineering (FASE'04) Conference, 2004.

[3] Kim, S., and Chang, S., "A Systematic Method to Identify Software Components," 11th Asia-Pacific Software Engineering Conference (APSEC 2004), Nov. 30 - Dec. 3, 2004.

[4] Jain, H. and Chalimeda, N., "Business Component Identification – A Formal Approach," Proceedings of the IEEE Fifth International Enterprise Distributed Object Computing Conference, 2001