UML 기반의 컴포넌트 인터페이스 추출 기법

°유영란, 김수동
숭실대학교 대학원 컴퓨터학과

A UML-based Component Interface Extraction Method

°Young Ran Yu, Soo Dong Kim
Dept. of Computing, Soongsil University

요 약

소프트웨어의 경제성, 시장 경쟁력 확보를 위한 소프트웨어의 재사용은 소프트웨어 공학의 주요 이슈가 고 있다. 그 중 컴포넌트와 컴포넌트 기반의 소프트웨어 개발은 재사용성을 확보할 수 있는 가장 주목 받는 방안으로 제시되고 있으며 많은 기법이나 지침들이 제안되고 있다 본 논문에서는 컴포넌트 개발에서 UML에 기반하여 컴포넌트의 인터페이스를 추출하는 기법을 제시하고자 한다. 분석 단계에서 컴포넌트의 식별이 이루어졌다고 가정하고 분석 단계에서 나온 산출물 중, Use Case 모델과 클래스 다이어그램을 이용하여 컴포넌트의 메소드들을 식별하고, 인터페이스로 정의한다. 그리고 사용자 요구사항에 근거하여 *Hot Spot*을 식별한 후, 컴포넌트의 커스터마이즈를 위한 메소드와 인터페이스를 정의한다.

1. 서 론

학계나 산업계가 컴포넌트의 경제성에 주목하면서, 주어진 업무 도메인에서의 컴포넌트 추출, 컴포넌트 명세, 컴포넌트 기반의 어플리케이션 구축을 위한 방법론 등이 주요 연구 대상이 되고 있다. CBSD에서 가장 기본적인 작업은 컴포넌트를 식별하여 정의하는 하는 것이라고 할 수 있다. 재사용성을 최대한 확보하면서, 잘 정의된(well-defined) 컴포넌트만이 컴포넌트를 조립하여 구축할 시스템의 안정성과 성능 및 융통성을 보장할 수 있기 때문이다.

컴포넌트를 구축하는 일은 도메인의 공통 업무를 컴포넌트로 추출하는 일과 추출된 컴포넌트를 정의하는 작업으로 나뉠 수 있으며, 컴포넌트의 추출은 분석 단계에서, 컴포넌트에 대한 명세 작업은 설계 단계에서 이루어질 수 있다. 본 논문에서는 도메인 분석 단계에서 컴포넌트가 추출되었다는 가정하에 UML의 모델들을 이용하여 컴포넌트의 메소드를 찾고 인터페이스를 정의하는 기법을 제시하고자 한다.

본 논문의 2장에서는 관련 연구로서 CBSD와 사례 연구의 도메인인 전자상거래에 대해서 설명하고, 3장에서는 전자상거래 시스템을 위해 추출된 컴포넌트에 대한 메소드 및 인터페이스 식별 기법을 제시한다. 마지막으로 4장에서는 결론을 내리고자 한다.

2. 관련 연구

2.1. CBSD

CBSD는 기존의 소프트웨어 컴포넌트를 결합시켜 대규모의 소프트웨어 시스템을 구축하는데 초점을 맞추고 있다. CBSD는 전사적 규모의 소프트웨어 시스템의 어떤 부분은 충분한 규칙성을 가지고 반복해서 사용된다는 가정을 전제로 하고 있으며, 공통된 시스템을 반복하여 작성하는 대신 재사용을 통하여 조립하겠다는 것이 CBSD의 출발점이다[1]

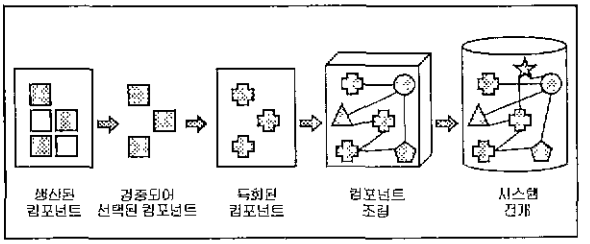

그림 1. 컴포넌트 기반의 개발 단계

컴포넌트 기반의 개발 접근 방법은 다음 4개의 주요 단계로 특징지어질 수 있다[2].

- 컴포넌트 검증 : 시스템에서 원하는 용도에 적합한가를 검증한다.

- 컴포넌트 적용 : 대상 컴포넌트를 시스템의 환경이나 용도에 맞게 커스터마이즈 한다.
- 컴포넌트 조립 : 컴포넌트들이 잘 정의된 하부구조를 통하여 통합된다.
- 시스템 전개 · 구축된 시스템을 요청된 변경이나 추가 기능들을 위해 변환시키고 적응시킨다.

CBSD 의 가장 큰 장점은 소프트웨어의 재사용성 확보이다. 소프트웨어의 재사용성 확보는 개발 기간의 단축, 개발 비용의 절약, 생산성 향상, 위험요소 축소, 향상된 일관성이라는 장점들로 확대된다. 그러나 재사용을 고려하지 않더라도 컴포넌트에 기반한 개발은 가치 있는 다른 장점들을 또한 제공한다. 먼저 전체 프로젝트에서 복잡도를 감소시키고 대량의 병렬 개발을 지원하며, 완성된 컴포넌트 정의는 시스템의 적응력(Flexibility)을 향상시킨다. 또한 점진적인 시험을 가능하게 하며 변경이 다른 부분에 영향을 주는 범위를 한정하여 유지보수를 쉽게 한다[3].

2.2. 전자상거래 시스템

인터넷이 상업적으로 가장 많이 활용되는 분야 중 하나가 전자상거래(Electronic Commercial) 분야이다. 인터넷 서점에서부터 대형 쇼핑몰에 이르기까지 그 대상은 실제 상거래의 전 품목에 이르며, 그 발전 속도나 실물 경제에서 차지하는 비중도 급속하게 증가할 것으로 예상되고 있다

3. 컴포넌트 인터페이스 추출 기법

3.1. 가정

본 논문에서는 컴포넌트 설계 작업 중, 컴포넌트 인터페이스 식별 기법만을 다루기 때문에 아래의 분석 결과들을 가정(assumption)으로 하고 있다.

- Use Case 모델과 클래스 다이어그램을 이용하여 컴포넌트가 식별되었다.
- 컴포넌트에 동일한 클래스가 동시에 할당될 수 없다
- 컴포넌트에 동일한 Use Case 가 동시에 할당될 수 없다.

이미 작업된 분석 단계에서는, 전자상거래 도메인에서 다음 4 개의 컴포넌트가 추출되었다고 가정한다. 각 컴포넌트들의 기능은 다음과 같다.

- □ 고객 관리 : 거래를 하는 고객들의 정보를 관리한다.
- □ 상품 관리 : 거래 대상이 되는 상품의 정보를 관리하고 상품 정보를 고객들에게 제공한다.
- □ 주문 관리 : 주문 입력, 조회, 수정, 삭제를 관리하며, 주문에 따른 결제 정보도 관리한다.
- □ 배달 관리 : 주문에 따른 배달 정보를 관리한다 배달된 주문 내역에 한해서 결제 확인 정보를 관리한다.

3.2. 업무 기능 메소드 추출

컴포넌트 인터페이스를 Use Case 다이어그램과 클래스 다이어그램에서 추출한다.

Use Case 다이어그램의 경우, 실제 엑터(Actor)와 연결된 Use Case 와 다른 컴포넌트 내의 Use Case 를 *"uses"*하는 경우, 컴포넌트의 메소드로 추출된다. 엑터과 연결된 Use Case 를 포함하고 있는 컴포넌트는 해당 Use Case 의 작업을 어플리케이션에 제공해 주어야 하므로 해당 Use Case 의 기능을 제공(provided) 메소드로 가지게 된다. use Case 를 *"uses"* 하는 경우도 *"uses"* 되는 Use Case 를 가지는 컴포넌트에서 *"uses"* 되는 Use Case 의 기능을 제공 메소드로 정의하여 다른 컴포넌트들이 이용할 수 있도록 하여야 한다.(그림 2 참조)

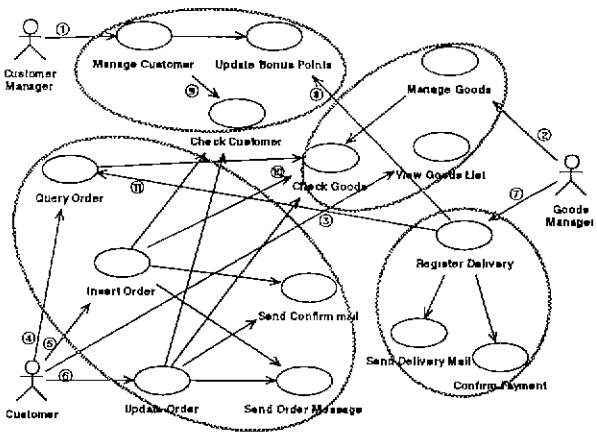

그림 2. Use Case 다이어그램에서의 컴포넌트 메소드 식별

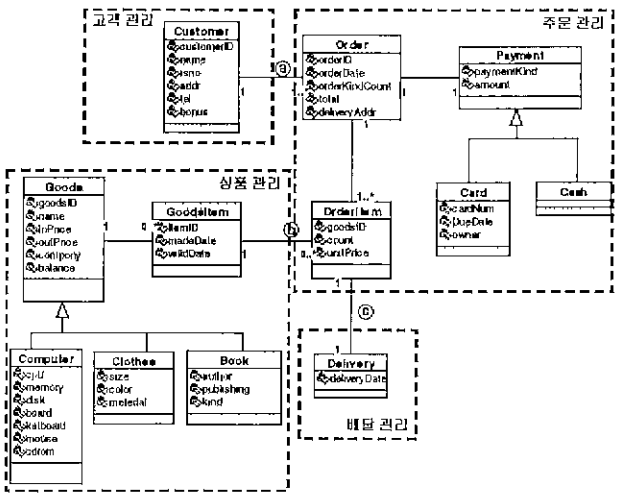

그림 3. 개념 객체 모델에서의 컴포넌트 메소드 식별

한편, 클래스 다이어그램에서는 서로 다른 컴포넌트 내의 클래스들이 관계(association)을 갖는 경우, 관계의 카디날리티(Cardinality)를 따져서 참조되어지는 컴포넌의 제공 메소드로 추출하여야 한다 (그림 3 참조)

추출된 컴포넌트의 메소드들은 다음과 같다.

- □ 고객 관리 ① 고객 등록, 수정, 조회, 삭제 (*addCustomer, selectCustomer, updateCustomer, removeCustomer*)
  ⑧ 고객 보너스 점수 수정 (*addCustomerBonus, minusCustomerBonus*)
  ⑨ 고객 코드 확인(*checkCustomer*)
  ⓐ 고객 조회(*searchCustomer*)
- □ 상품 관리 ② 상품 등록, 조회, 수정, 삭제

(*addGoods, selectGoods, updateGoods, removeGoods*)
③ 상품 목록 조회(*listGoods*)
⑩ 상품 코드 확인(*checkGoods*)
⑤ 상품 조회(*searchGoods*)

□ 주문 관리 ④ 주문 조회(*selectOrder*)
⑤ 주문 입력(*addOrder*)
⑥ 주문 수정 +⑪ 주문 내역 조회(*updateOrder*)
ⓒ 주문 목록 조회(*searchOrder*)

□ 배달 관리 ① 배달 등록(registerDelivery)

식별된 일부 메소드들 중 중복되는 부분은 사용자 요구시항에 근거하여 하나로 합칠 수도 있다.

3.3. 커스터마이즈 메소드 추출

대상 컴포넌트 중 가장 비중이 큰【주문 관리】에 대한 Hot Spot은 요구사항 명세서를 통하여 다음과 같이 추출되었다

□ 주문 관리 : 일단 주문이 된 후, 수정이 필요할 때 그 방법이 각 쇼핑몰마다 다를 수 있다. 전체 취소만 가능하다던지, 배달 이전의 상품에 한해서 가능하다던지 하는 경우이고, 현금(지로)으로 지불된 경우 환불 방법도 고려되어야 한다.(*setCancelOrderStrategy*) 그리고 주문된 내역에 대한 확인 메일을 고객에게 발송한다던지, 주문 내역을 배달부서에 전송하는 것도 각 쇼핑몰마다 다를 수 있다. (*setOrderConfirmMail, setOrderMessage*)

3.4. 컴포넌트 인터페이스 정의

컴포넌트의 인터페이스는 위 작업들에서 추출된 메소드들의 집합이다. 본 연구에서는 업무 기능에 관한 메소드들과 커스터마이즈를 위한 메소드들을 분리하여 두 개의 인터페이스를 정의하도록 하였다. 이렇게 분리하므로서 커스터마이즈에 대한 권한별 접근을 제어할 수 있으며, 기능에 변화에 따른 메소드 및 인터페이스의 수정과도 독립적일 수 있기 때문이다. 【주문 관리】컴포넌트에 대한 인터페이스 정의는 그림 4 와 같다. 인터페이스 내, 각 메소드 별 리턴 값 및 변수가 정의되어야 하고 타입이 결정되어야 한다.

```
OrderManagement

interface Order {
    status addOrder(string[] goodsID, int[] count);
    status updateOrder(string[] goodsID, int[] count);
    status selectOrder(string CustomerID, String OrderID);
    status searchOrder(string OrderID);
}
interface CustomizeOrder {
    status setCancelOrderStrategy(int cancelOrderFlag),
    status setOrderConfirmMail(boolean confirmMailFlag),
    status setOrderMessage(boolean orderMsgFlag);
}
```

그림 **4. OrderManagement** 컴포넌트 인터페이스 명세서

3.5. 인터페이스 메소드들의 클래스 할당

정의된 인터페이스의 메소드들은 결국 컴포넌트 내의 클래스들에게 할당된다. 각 메소드의 중심 역할을 하는 클래스에 할당한다. 그리고 이 메소드별로 UML의 순차도(sequence Diagram)을 작성하므로서 내부 클래스들에 대한 메소드들도 추출할 수 있다. 【주문 관리】컴포넌트의 *addOrder* 메소드에 대하여 순차도를 작성하고, *addOrder* 와 순차도에서 발견된 클래스의 메소드들을 할당한다.

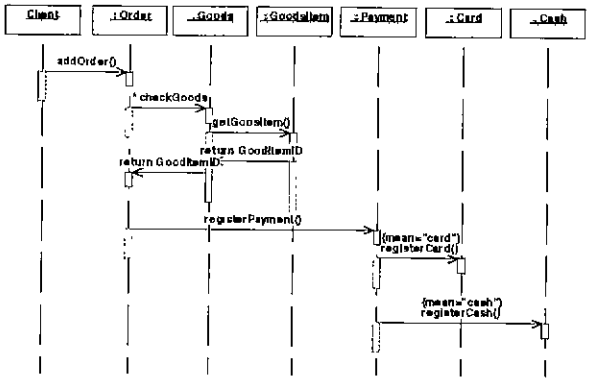

그림 **5.** ***addOrder*** **의 순차도**

- 클래스 Order · *addOrder*
- 클래스 OrderItem : *getGoodsItem*
- 클래스 Payment : *registerPayment*
- 클래스 Card . *registerCard*
- 클래스 Cash : *registerCash*

그림 5 의 예에서 클래스 Goods 나 GoodsItem 에 보내지는 메시지는 차후 Goods 나 GoodsItem 이 속한 컴포넌트의 제공(provide) 인터페이스로 추가되어야 할 부분이다. 컴포넌트 내의 각 클래스의 메소드는 위의 순차도와 같은 자료들로부터 발견될 수 있다.

4. 결론 및 향후 연구 과제

본 논문에서는 전자상거래 시스템을 사례 연구 도메인으로 하여, UML 표기법에 기반하여 추출된 컴포넌트에 대한 메소드와 인터페이스 추출 기법을 제안하였다. 본 논문에서 제시된 방법이 컴포넌트의 모든 인터페이스 메소드들을 추출할 수 있는 것은 아니지만, 적어도 주요 기능들에 대한 추출은 이루어지도록 하므로서, 이후 추가 내지는 수정 작업의 기반을 제시할 수 있도록 하였다. CBSD 에서 컴포넌트의 추출과 인터페이스 식별은 주요 연구 과제로 제안될 것이다.

5. 참고문헌

[1] Capt Gary Haines, David Carney, John Foreman, *Component-Based Software Development / COTS Integration, CMU Software Technology Review*, 1997.

[2] Alan W. Brown, Kurt C. Wallnau, *Engineering of Component-Based Systems*, 7-15. *Component-Based Software Engineering:Selected Papers from the Software Engineering Institute*. Los Alamitos, CA:IEEE Computer Society Press, 1996.

[3] Keith Short, *CBD and Object Modeling, Sterling Software CBD White Paper version 1.0*, February, 1997.

[4] Martin Fowler, *UML Distilled*, Addison Wesley, 1997.

[5] Clemens Szyperski, *Component Software*, Addison Wesley, 1998