UML을 이용한 컴포넌트 재사용을 위한 인터페이스 명세 시각화 방법

°이건화, 김강태, 이경환
중앙대학교 컴퓨터공학과 소프트웨어공학 연구실

Visualization of interface description for component reuse using UML

Keon-hwa Lee, Kang-tae Kim, Kyung-whan Lee
SE Laboratory, Dept of Computer Science & Engineering, Chung-Ang University

요 약

컴포넌트 재사용을 위해서 필요한 정보를 제공하는 일반적인 방법으로 인터페이스를 명세한다. 그러나 일반적인 인터페이스 명세는 단순히 외부적인 관점에서만 표현하므로, 컴포넌트간의 상호작용과 협력관계를 나타내지 못한다. 그러므로 컴포넌트 개선(evolution) 및 합성(composition)시에 발생할 수 있는 문제점(conflict)에 대한 충분한 정보를 제공할 수 없다.

본 논문에서는 외부적으로 요구되는 컴포넌트 인터페이스뿐만 아니라 각 컴포넌트 사이에 발생하는 상호작용을 나타낼 수 있는 방법으로 기존의 계약(*contract*) 개념을 바탕으로 컴포넌트에 적합하도록 변형하여 이를 시각화하였다. 시각화를 위해서 일반적인 산업표준으로 자리잡아가고 있는 UML의 확장 메커니즘의 하나인 stereotype을 이용하여 기본 컴포넌트 계약 재사용 타입을 나타내고 정의하였다. 그리고 재사용 타입간의 관계를 통하여 컴포넌트 개선 및 합성시에 발생할 수 있는 문제점을 지적하고 이를 감지하는 방법을 제공했다.

1. 서 론

소프트웨어의 위기이래 문제 해결을 위해서 많은 연구가 진행되어 왔다[7]. 소프트웨어의 생산성, 품질, 효율성을 위해서 제시된 새로운 개념 중 하나가 부품화와 조립의 특성을 지닌 컴포넌트 기반 개발이다[1]. 컴포넌트를 부품처럼 조립하고 기능이 개선된 부품을 재조립하는 재사용 방식을 가지게 된다[2].

컴포넌트 재사용을 위해서 필요한 정보를 제공하는 일반적인 방법으로 인터페이스를 명세한다[8]. 그러나 일반적인 인터페이스 명세는 단순히 외부적인 관점에서만 표현하고, 컴포넌트간의 상호작용과 협력관계를 나타내지 못한다. 그러므로 컴포넌트의 개선 및 합성시에 발생할 수 있는 문제점에 대한 충분한 정보를 제공할 수 없다. 이를 해결하기 위해서 ADL, Cook Book, Design Pattern, 인터페이스 상세화, 계약(contract) 등의 연구가 진행되고 있다[8].

본 연구에서는 기존의 계약 연구[3,4,5]를 바탕으로 컴포넌트 기반 개발의 특성에 적합하도록 명세 방법을 연구하였다. 기존의 명세 방식의 읽기 어려운 점으로 인해 개발자나 사용자가 실제적으로 적용하기 힘든 단점을 보완하고자 시각화한다. 이를 위해서 일반적인 산업표준으로 자리잡아가고 있는 UML[6]의 확장 메커니즘의 하나인 stereotype을 이용하여 기본적인 컴포넌트 계약 재사용 타입을 나타내고 정의하였다. 그리고 재사용 타입간의 관계를 통하여 컴포넌트 개선 및 합성시에 발생할 수 있는 문제점을 지적하고 이를 감지하는 방법을 제시했다.

2. 컴포넌트 계약

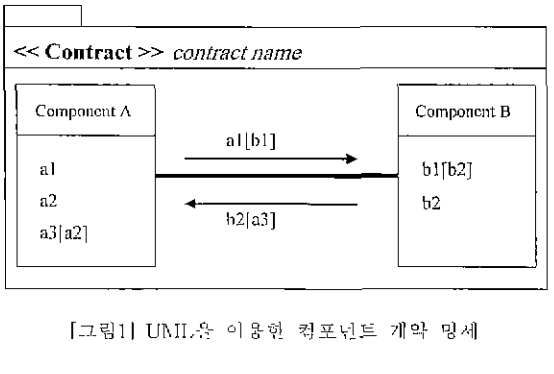
앞서 언급한 바와 같이 컴포넌트 명세에는 인터페이스뿐만 아니라, 컴포넌트들 간의 협력 관계 및 상호작용을 나타낼 수 있어야 한다. 그러나 컴포넌트 개선 및 합성시에 필요한 모든 정보를 모두 나타내려면 개발자에게 많은 부담이 될 수 있다. 그러므로 본 연구에서는 이러한 trade-off로써 기존의 계약 연구[3,4,5]를 바탕으로 이를 정형적인 표현으로 나타내었다. 정형적인 컴포넌트 계약 표기는 다음과 같다.

```
Contract contract name
        interface-name1
        interface-name2 { specialization clause }
        ...
End Contract
```

용어	의미
Contract End Contract	계약의 시작과 끝
Contract name	계약의 이름 다른 계약과는 구별되는 유일한 이름을 가져야 한다.
interface-name1 *interface-name2*	인터페이스(오퍼레이션) 이름 다른 오퍼레이션과는 구별되는 유일한 이름을 가지야 한다 뒤에 *specialization clause*를 하나 이상 명세 가능하다. 이를 통해서 컴포넌트내에서의 오퍼레이션간의 관계와 다른 컴포넌트와 연계된 상호작용을 표현한다.
specialization clause	invoke되는 오퍼레이션의 이름. 같은 컴포넌트 내에서 invoke되는 컴포넌트 오퍼레이션 이름이거나 다른 컴포넌트에 있는 invoke되는 오퍼레이션 이름이다

[표1] 컴포넌트 계약의 정형적 표기 용어의 각 의미

즉, [표1]에서와 같이 속한 컴포넌트의 인터페이스만을 표현하는 것이 아니라 컴포넌트 내부에서 사용되는 인터페이스(인터페이스)간의 관계를 나타낼 수 있다. 그리고 다른 클래스와의 협력 및 상호 관계를 표현하게 된다 그러나 이는 텍스트 형식으로 읽기 어렵기 때문에, 개발자나 사용자가 실제로 사용하기에는 어렵다는 단점을 지니게 된다. 이를 개선하기 위해서는 시각화를 통해 제공하는 방법이 가장 일반적이다 시각화 표현은 개발자가 쉽게 인식할 수 있고, 다른 컴포넌트와의 연계를 그림을 통해서 나타내기 때문에 각 컴포넌트가 다른 컴포넌트와 어떻게 연동되는지를 구별하여 표현할 수 있다 이를 위해서 시각화에 사용하는 방식으로 UML Notation을 기반으로 사용한다 [그림1]은 UML을 이용한 컴포넌트 계약 명세의 시각화를 보여준다 각 계약은 팩키지로 표현되고, stereotype으로 "<< Contract >>"를 표기한다. 각 컴포넌트간의 관계는 선으로 연결한다 컴포넌트간의 상호작용은 화살표로 표현하고, 컴포넌트내의 상호작용은 컴포넌트 계약 박스 내에 표기한다



[그림1] UML을 이용한 컴포넌트 계약 명세

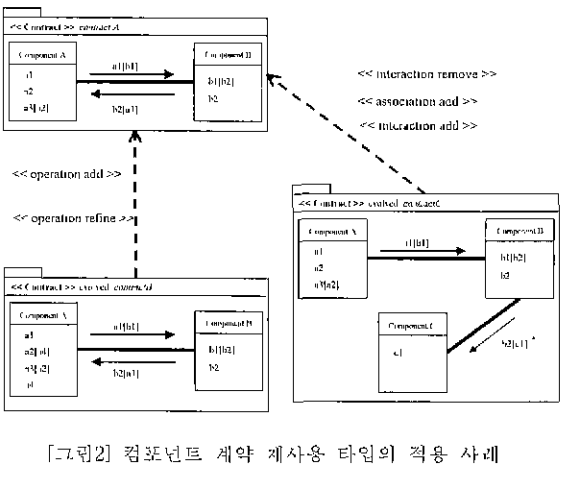
3. 컴포넌트 계약 재사용 타입

2장에서 언급한 바와 같이 컴포넌트 계약을 사용하면 각 컴포넌트의 인터페이스뿐만이 아니라 컴포넌트 내부에서의 오퍼레이션 연계를 고려할 수 있게 된다[3]. 그러나 실제로 컴포넌트 개신 및 합성 시에 발생하는 문제를 감지하기 위해서는 각 컴포넌트 계약을 실제로 비교해야 하는 부담이 있다 이를 위해 컴포넌트 전개의 종류를 정의하고 이에 따른 합성의 재사용 타입을 정의하여 변경의 종류를 미리 알 수 있도록 하였다 그리고 재사용 타입의 비교만으로 미리 발생할 수 있는 문제점을 지적함으로써 발생할 수 있는 문제에 대한 감지를 쉽게 또는 자동적으로 하도록 하고자 한다.

기본 컴포넌트 계약 재사용의 타입의 정의 및 의미는 [표2]와 같이 10가지로 나누었다

컨트랙트 재사용 타입	의미
<< operation add >>	하나의 컴포넌트 내에 새로운 오퍼레이션을 추가
<< operation remove >>	하나의 컴포넌트 내에서 오퍼레이션을 삭제
<< operation refine >>	하나의 컴포넌트 내에 새로운 오퍼레이션 의존성 추가
<< operation separate >>	하나의 컴포넌트 내에서 오퍼레이션의 의존성 삭제
<< interaction add >>	계약내의 새로운 컴포넌트간의 상호 관계 추가
<< interaction remove >>	계약내의 컴포넌트간의 상호 관계 삭제
<< interaction refine >>	상호작용 내에서의 오퍼레이션 의존성 추가
<< interaction separate >>	상호작용 내에서의 오퍼레이션 의존성 삭제
<< association add >>	계약내에 새로운 참여 관계를 가진 컴포넌트 추가
<< association delete >>	계약 내에 참여 관계를 가진 컴포넌트 삭제

[표2] 기본 컴포넌트 계약 재사용 타입



[그림2] 컴포넌트 계약 재사용 타입의 적용 사례

[그림 2]는 컴포넌트 계약 재사용 타입을 UML을 이용한 시각화에서 쓰이는 방식의 한 예를 보여주고 있다

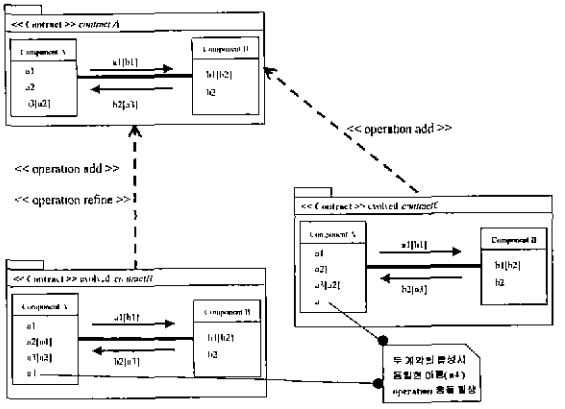
4. 컴포넌트 개선 및 합성시의 문제점과 계약 재사용 타입을 이용한 감지 방법

컴포넌트 전개 후 이를 합성하여 재사용하고자 하는 경우에 둘간의 호환성을 체크할 필요성이 있다.

예로, [그림 2]에서와 같이 *contractA*를 수정하여 *evolved contractB*가 만들어졌는데, ComponentA에 새로운 오퍼레이션인 a4를 추가하고(<< operation add >>), 또한 a2를 a2[a4]로 바꾸었다(<< operation refine >>). 또한 *contractA*를 수정하여 *evolved contractC*가 만들어졌는데, ComponentA와 ComponentB간의 상호작용 중 b2[a3]을 삭제하고(<< interaction remove >>), 새로운 ComponentC가 추가되어 ComponentB와의 관계가 추가되고(<< association add >>), 또한 ComponentA와 ComponentB 사이의 새로운 상호작용으로 b2[c1]을 추가하였다(<< interaction add >>).

만일, *evolved contractB*와 *evolved contractC*를 합성하여 재사용하고자 하는 경우, *evolved contractB*에서의 상호작용 b2[a3]가 있지만, *evolved contractC*에서는 상호작용 b2[a3]가 없고 대신에 새로운 ComponentC와의 상호작용 b2[c1]이 있기 때문에, 이 둘을 합성할 경우에 각 contract에서 원하던 기능이 제대로 수행되지 않는 문제가 생길 수 있다.

또한 [그림 3]에서와 같은 재사용 문제가 발생하기도 한다.



[그림 3] 계약을 이용한 합성시 이름 충돌의 예

이외에도 재사용시에 내부적인 상호 작용 관계로 인해서 여러 가지 문제가 발생할 수 있다[3]. 이러한 재사용 문제점들을 감지하는 방법으로는 2가지 방식이 있을 수 있다. 하나는 원래의 계약과 변경된 계약간의 타입을 모두 비교하여 검사를 하는 것이다. 다른 하나는 미리 각 계약 재사용시 생길 수 있는 문제를 기본 컴포넌트 계약 재사용 타입간에 비교하여 표를 작성하고, 이를 자동적으로 감지하는 방법이다. 후자를 위해서 [표3]은 제안된 계약 재사용 타입간에 발생하는 문제 중 인터페이스 충돌 문제에 대한 표이다.

	operation add	operation remove	operation refine	operation separate
operation add	오퍼레이션 이름 충돌	-	-	-
operation remove	-	-	-	-
operation refine	-	-	operation간의 invocation충돌	-
operation separate	-	-	-	완전히 분리됨

[표3] 계약 오퍼레이션 타입 중 인터페이스 충돌의 종류

5. 결론

본 연구에서는 개선된 컴포넌트 인터페이스 명세로써 컴포넌트 계약 시각화 방법을 제시하였다. 이를 통해 UML Notation을 사용하여 컴포넌트 자체의 인터페이스뿐만 아니라 컴포넌트간의 관계를 쉽게 알 수 있고, 컴포넌트간의 상호작용 및 내부 특성을 쉽게 알 수 있다 즉, UML의 확장 메커니즘의 하나인 stereotype을 이용하여 기본 계약 재사용 타입을 정의하였고, 컴포넌트들간의 계약을 package로 나타내었다. 그리고 기본 컴포넌트 계약 재사용 타입을 정의하여 컴포넌트 개선 및 합성시에 발생할 수 있는 문제점을 미리 감지할 수 있는 방법을 제공하였다.

그러나 기본 계약 재사용 타입만으로는 발생하는 모든 문제점을 감지하기는 불가능할 것이다. 그러므로 기본 계약 재사용 타입의 확장을 위한 연구가 필요하다.

참고 문헌

[1] Cuno Pfister, Clemens Szyperski, Why Objects Are Not Enough, Proceedings, First International Component Users Conference, 1996

[2] Clemens Szyperski, "Component Software", Addison Wesley, 1997

[3] Patrick Steyaert, Carine Lucas, Kim Mens, Theo D'Hondt, Reuse Contracts Managing the Evolution of Reusable Assets, ACM SIGPLAN Notices, Vol.31, No 10, pp 268-285, October 1996

[4] Helm, R., Holland, I., Gangopadhyay, D, Contracts' Specifying Behavioral Compositions in Object-Oriented System, Proceeding of Joint ECOOP/OOPSLA '90 Conference, pp. 169-180, ACM Press, 1990

[5] Lamping, J., Typing the Specialisation Interface", Proceedings of OOPSLA '93, Conference on Object-Oriented Programming, Systems, Languages and Applications, pp 201-215, ACM Press, 1993

[6] Grady Booch, Ivar Jacobson, James Rumbaugh, The Unified Modeling Language User Guide, ADDISON-WESLY, 3rd, 1999

[7] 이경환, 객체모델링방법론, 홍릉대학교출판국, 1995

[8] 이칭훈, 컴포넌트 합성의 호환성 체크 및 일의성 검증을 위한 정형적명세 방법, 박사학위 논문, 1998

[9] 이건희 외3인, 컴포넌트 재사용을 위한 지원 방법에 관한 연구, 정보과학회, '99 봄 학술발표논문집(A), pp 539-541, 1999