

역할 모델링을 이용한 객체 패턴의 추상화 방법

김정종 송호영^o 박운재 송의철
경남대학교 컴퓨터공학과
{jkkim, humanism^o}@zeus.kyungnam.ac.kr

A Study on A Method for Abstracting Object Patterns Using Role Modeling

Jungjong Kim Hoyoung Song^o Woonjai Park Euicheol Song
Dept. of Computer Engineering, Kyungnam University

요 약

대부분의 객체지향 방법론은 객체를 중심으로 한 객체 모델링을 바탕으로 소프트웨어를 개발한다. 물론, 객체 모델링 방법은 여러 가지 장점을 가지고 있지만 복잡한 문제를 가진 대규모 시스템에는 적합하지 않다. 따라서, 이런 복잡한 대규모 시스템을 객체의 패턴에 따라 간단한 모델로 분할할 필요성이 있으며 이를 위하여 역할 모델링 방법이 제안되었다.

본 논문은 객체의 패턴들을 추상화하고 복잡한 대규모 시스템을 관계의 분리를 통하여 간단한 모델로 생성할 수 있도록 객체 중심이 아닌 역할을 중심으로 한 역할 모델링 방법을 연구하였다.

1. 서 론

대부분의 객체지향 방법론은 객체를 중심으로 한 객체 모델링을 바탕으로 소프트웨어를 개발한다. 객체 모델링 방법은 시스템을 구성하고 있는 객체를 강조함으로써, 실제에 더욱 근접한 표현을 가능하게 하고 지속적인 변화에 탄력적으로 대처할 수 있게 해준다. 그러나, 이 방법은 복잡한 문제를 가진 대규모의 시스템에는 적합하지 않으며 상호작용과 협력의 표현, 상속에 의한 재사용 문제 등을 해결하는 데는 한계가 있다[1][2][3].

따라서, 본 논문은 객체들의 패턴을 추상화하고 복잡한 대규모 시스템을 관계의 분리를 통하여 간단한 모델로 생성할 수 있도록 객체 중심이 아닌 역할을 중심으로 한 역할 모델링 방법을 연구한다. 또한 관계의 분리를 통하여 생성된 각각의 역할 모델들을 합성하고 조합할 수 있는 통합 방법을 연구한다.

2. 관련연구

2.1. 객체 모델링의 문제점

객체나 클래스 개념으로 한 객체 모델링 방법으로는 재사용성과 생산성을 극대화하기에는 부족한 면이 있으며, 몇 가지 문제점을 나타내고 있다[3][4].

(1) 객체

객체의 상태 영역과 행위는 객체가 생성될 때 결정된다. 즉, 객체가 그것의 타입을 동적으로 변경하는 것이 불가능하다.

(2) 상호작용과 협력의 표현

객체들은 작업을 수행하거나 목표를 달성하기 위하여 서로 협력할 필요가 있다. 그러나, 객체 모델링 방법은 객체의 전체 집합 보다는 각 객체에 너무 초점을 둔다. 각 객체

에 초점을 맞추는 것은 혼란을 가져올 수 있고 양질의 소프트웨어를 양성할 수 없다. 객체는 하나의 추상화 단위인 어플리케이션에서, 적절한 하위집합 혹은 하위시스템을 추출하고 패키징하기가 매우 어렵다.

(3) 상속에 의한 재사용

상속에 의한 재사용은 개발자가 상속하고 오버라이드 할 대상을 결정하는 것을 어렵도록 한다. 또한 상속은 하위 클래스가 모든 상위 클래스의 특성과 행위를 얻는다는 것을 의미하기 때문에 시스템의 분해를 어렵게 한다.

(4) 복잡한 관계 구조

객체 모델링은 시스템 전체가 객체를 중심으로 결합되어 있으므로 복잡한 문제를 가진 대규모 시스템을 모델링하는 것은 쉽지가 않을 뿐만 아니라 이해하기도 어렵다. 또한, 복잡한 관계 구조로 인하여 추가적인 요구 사항에 대처가 쉽지 않다.

이런 여러 문제점들로 인해 새로운 추상화 방법으로서 객체 중심이 아닌 역할을 중심으로 한 모델링 방법을 제안한다.

2.2. 역할

클래스의 개념처럼 역할은 객체 집합의 기술이다. 그러나, 클래스와 다른 점은 클래스는 일반 특성들을 나타내는 객체 집합을 기술하지만 역할은 객체 패턴에서 같은 위치에 있는 객체 집합을 기술한다. 역할은 특정 문맥에 참여하는 개체의 행동이다. 즉, 표현된 하나의 추상개념이다. 역할은 하나의 개체에 대하여 관점에 따라 서로 다른 역할을 가질 수 있다[5][6].

또한, 역할은 객체의 속성과 오퍼레이션에 의해 행위를 기술한다. 역할에 의하여 정의된 행위는 객체가 어떻게 행동할지를 명확하게 하기 위하여 이용된다. 역할의 사용은

객체들이 변화와 확장에 잘 적응할 수 있도록 한다.

2.3. 역할 모델

역할 모델은 객체 모델에서 객체의 패턴을 추상화한 모델이며 객체 구조가 적절한 역할들을 수행함으로써 주어진 관계 영역을 완성하는 방법을 기술한다[5]. 역할 모델은 관계의 분리를 지원하고 하나의 결합 모델에서 실세계 현상의 정적·동적 속성들을 기술한다. 이 관계의 분리는 상호작용하는 객체에 대해 대규모의 복잡한 구조를 가진 시스템에서 대규모의 복잡한 현상을 많은 하위 현상으로 분할되는 것을 말하며, 각 하위 현상은 자신의 역할 모델에 의해 기술된다.

현상은 많은 협력하는 객체들에 의해 기술된다. 하위 현상은 그들의 관계 영역에 의해 명시되고, 하위 현상을 기술하는 객체들은 객체의 패턴에서 구성되며, 패턴에서 같은 위치를 가지는 모든 객체들은 역할에 의해 표현된다. 역할 모델을 생성하기 위한 기본 지침은 다음과 같다.

- ① 관계없는 객체들을 삭제하고 활동에 관여하는 객체들을 표현한다.
- ② 관계없는 측면을 삭제하고 활동 상황에 관련된 측면을 표현한다.
- ③ 관계없는 세부사항을 삭제하고 활동 상황에서 고려된 세부사항을 숨기기 위하여 객체 캡슐화 속성을 사용한다.
- ④ 객체 식별자를 일반화하고, 이들 활동을 수행하는 역할들에 대한 유사하고 전형적인 패턴에 의해 활동을 수행하는 상호작용하는 객체들의 패턴을 표현한다.

3. 역할 모델링

역할 모델링은 역할을 중심으로 객체들의 관계에 따른 새로운 추상화 방법으로서 객체가 수행하는 역할을 기반으로 모델링하는 방법이다.

3.1. 역할 모델의 표기법

다음의 표 1은 역할 모델을 구성하고 있는 요소들의 표기를 나타낸다.

표 1 역할 모델 요소의 표기

표기	설명
	역할명
	역할명
	협력하는 역할들 사이 메시지 패스를 나타낸다.
	인접한 역할이 하나의 협력자만을 알고 있다는 것을 가리키는 단일 포트를 나타낸다.
	인접한 역할이 몇몇 협력자 역할들을 가리키는 다중 포트를 나타낸다.

3.2. 역할 모델 통합

문제가 너무 크고 복잡하여 전체로서 조절이 되지 않는다면 관리할 수 있는 하위 문제들로 분할하고 역할 모델로서 그들 각각을 모델한다. 모델된 역할 모델들을 기본 모델로 하여 통합 운영에 의해 합성 혹은 조합하여 통합 모델을 생성할 수 있다. 따라서, 기본 모델들을 그대로 유지하면서 관계 영역 전체를 표현하기 위하여 새로운 통합 모델을 생성할 수 있는 통합 방법이 필요하다. 이는 복잡한 현상에 정통할 수 있고 통합으로 인해 전체 제어를 유지할 수 있도록 한다.

4. 사례연구

사례연구(여행 경비 예)에서는 회사 사람이 회사 경비로 여행을 가기 원한다고 할 때 역할 모델을 생성하는 과정을 보여준다. 먼저 회사 조직에 대한 객체 모델을 생성하고, 여행 경비 예에서 상호작용하는 객체들에 대하여 패턴을 정의하고 추상화하여 역할 모델을 생성한다. 또한, 여행 경비 모델을 확장하여 비행기 티켓 예약 모델을 생성하고, 이들 두 모델을 통합하여 새로운 통합 모델을 생성한다.

(1) 객체 모델

다음 그림 1은 회사 조직을 나타내는 객체 모델이다. 사각형은 객체이고 선은 객체들 사이 관계를 나타낸다. 화살표는 아래 시나리오를 바탕으로 한 흐름이다.

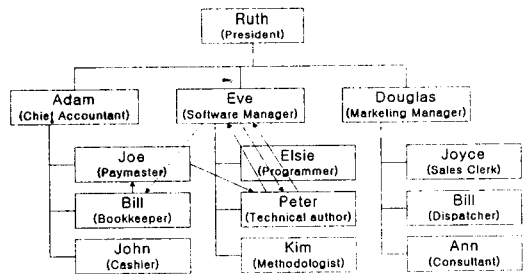


그림 1 회사 조직의 객체 모델

(2) 시나리오

Peter라는 사람이 회사 경비로 여행을 하기 원한다고 할 때 다음과 같은 처리가 이루어 질 수 있다.

- ① Peter는 그의 매니저 Eve에게 여행허가를 요청한다.
- ② Eve는 예산과 계획을 검사하고 Peter에게 여행을 허가한다.
- ③ Peter는 티켓을 구매하고 여행을 한다. 그리고 돌아와서 경비 보고서를 작성하고 Eve에게 보고서를 전달한다.
- ④ Eve는 경비 보고서를 검사하고 허가한 다음 bookkeeper인 Bill에게 허가된 경비 보고서를 전달한다.
- ⑤ Bill은 금전 장부를 정리하고 Paymaster인 Joe에게 지불해줄 것을 요구한다.
- ⑥ Joe는 Peter에게 돈을 지불한다.

(3) 역할 모델

위의 시나리오를 바탕으로 그림 1의 객체 모델에서 일정한 패턴을 정의하여 역할 모델을 생성할 수 있다. 상호작용하는 객체들은 Peter, Eve, Bill, Joe이다. 이들 객체들을 살펴보면 일정한 패턴을 유지한다고 볼 수 있다. 따라서, 이 객체 패턴을 추상화하여 Traveler, Authorizer, Bookkeeper, Payment 역할을 얻을 수 있다. 다음 그림 2는 여행 경비에 대한 역할 모델을 보여준다. 화살표는 메시지 흐름을 나타낸다. 이 메시지 흐름은 역할 모델에 포함될 수도 있고, 따로 분리하여 순차도로 나타낼 수 있다.

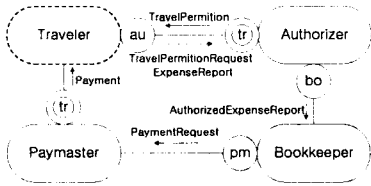


그림 2 여행 경비에 대한 역할 모델

(4) 역할 모델 통합

여행 경비 모델을 확장하여 비행기 티켓 예약 모델을 생성한다. 비행기 티켓은 예약 담당 직원에 의해 주문되고 여행사에게 직접 지불된다. 다음 그림 3의 (b)는 비행기 티켓 예약에 대한 역할 모델을 보여준다.

그림 2의 모델(그림 3(a))과 그림 3(b)의 모델을 통합하여 새로운 모델을 구성한다. 그림 3(c)는 통합 모델을 보여준다.

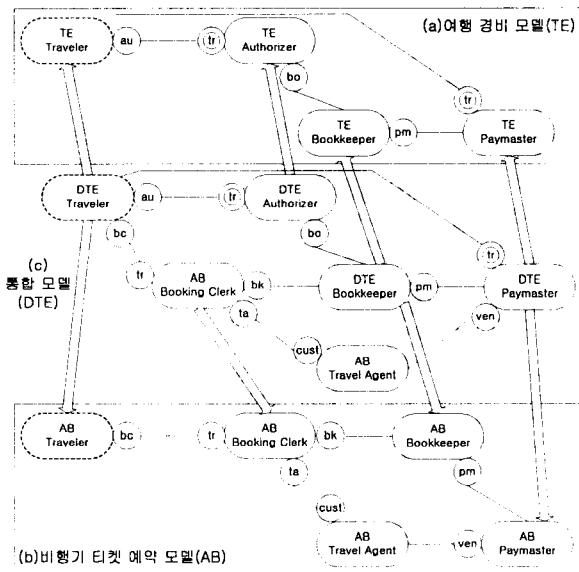


그림 3 TE와 AB의 통합 다이어그램

통합 작업의 첫 번째 단계는 TE 모델을 DTE 모델로 통합하는 것이다. 화살표는 통합 관계를 나타낸다. DTE

모델은 적어도 TE 모델의 각 역할들과 대응하는 역할을 가져야 한다.

두 번째 단계는 DTE 모델에 있는 역할 위에 AB 모델의 각 역할을 설정한다. AB 모델에서 대응하는 모델을 사상하기 위하여 DTE-BookingClerk과 DTE-TravelAgent를 추가해야 한다. 그러나, 남아있는 AB 모델의 역할들을 사상하기 위하여 존재하는 DTE 역할들을 재사용할 수 있다. DTE-Traveler, DTE-Bookkeeper, DTE-Paymaster는 두 다른 역할들을 수행하고 두 명세를 따라야 한다.

5. 결론

본 논문에서는 시스템을 개발할 때 객체를 중심으로 한 객체 모델링 방법으로 시스템을 모델링하지 않고 객체의 역할을 중심으로 한 역할 모델링 방법을 이용하여 시스템을 모델링하는데 목적을 두고 있다.

본 논문은 객체들의 패턴을 추상화하고 복잡한 대규모 시스템을 관계의 분리를 통하여 간단한 모델로 생성할 수 있도록 객체 중심이 아닌 역할을 중심으로 한 역할 모델링 방법을 연구하였다. 또한, 관계의 분리를 통하여 생성된 각각의 역할 모델들을 합성하고 조합할 수 있는 통합 방법을 연구하였다.

이 역할 모델링은 복잡한 현상에 정통할 수 있고 시스템의 이해력을 높일 수 있으며, 생성된 역할 모델들을 통합함으로써 문제 영역의 전체를 표현할 수 있어 시스템 전체의 제어를 유지할 수 있다.

향후 연구 과제로는 역할 모델 통합 방법에 대한 다양한 연구와 역할 모델에 대한 컴포넌트 설계 방법이 필요하다.

참고문헌

- [1] E. P. Andersen, "Conceptual modeling of Object: A Role Modeling Approach", PH.D Thesis, University of Oslo, 1997
- [2] Bent B. Kristensen, "Object Oriented Modeling with Role", In Proceedings of the 2nd International Conference on Object-Oriented Information Systems(OOIS'95), Dublin, Ireland, London, pp. 57-71. 1995
- [3] Liping Zhao, Elizabeth A. Kendall, "Role Modeling for Component Design", Proceedings of 33rd International Conference on Technology of Object-Oriented Languages, pp. 312-323, 2000
- [4] Ralph Depke, Gregor Engels, Jochen Malte Küster, "On the Integration of Roles in the UML", Technical Report No. 214, University of Paderborn, August, 2000
- [5] Wold Reenskaug, Trygve Reenskaug, O. A. Lehne, "Working With Object: The Ooram Software Engineering method", Prentice Hall, 1996
- [6] D. Bäumer, D. Riehle, W. Siberski, M. Wulf, "The Role Object Pattern", In Proceedings of 4th Conference on Pattern Languages of Programs, 1997