설계 의미를 사용한 설계내역의 검색 방법

○배명남†, 최완†, 양재동†

† 한국전자통신연구원

† 전북대학교 컴퓨터과학과

Retrieval Method of Sofrware Contents using Design Semantics

M. N. Bae, W. Choi, and J. D. Yang

† Electronics and Telecommunications Research Institute

† Dept. Computer Science, Chonbuk Nat'l Univ.

요 약

사용자가 저장소내의 특정 내역을 파악하기 위해서는 내역에 대해 다양한 형식과 의미 표현이 필요하다. 이를 위해, 사용자가 사용할 수 있는 기본 연산인 추출 및 관계 함수들을 제공하고 있다. 그러나, 사용자가 부품 저장소내에 내역의 저장 구조를 파악하고, 복잡하게 형식화 되어있는 내역간의 관계에 대한 요구를 추출 및 관계 함수들의 조합으로 명세하는 것은 매우 어렵다.

따라서, 본 논문에서는 설계 정보 자체와 그 들간의 의미 정보들에 대해, 추상화된 인식 모델을 제공하고, 이 모델위에서 내역의 의미 표현에 적합하고, 사용자에게 친숙한 질의어를 제안한다.

1. 서론

객체지향 개발 방법론을 사용한 소프트웨어 개발은 잘 정의된 객체지향 특성들을 사용하여 프로그램 구조를 보다 단순화하면서도 강력한 표현력을 가진 프로그램의 각성을 돕는다 또한, 최종 생산물인 프로그램 코드 이외에, 구현 이전 단계까지 각 설계정보에 대해 잘 정제된 고수준의 연관 정보들을 함께 제공하고 있다. 현재, 이를 지원하는 CASE 시스템들이 안정된 틀 위에서 정립되어 가고 있고, 이 위에서 설계 내역의 분석 및 재사용 측면의 쟁점으로, 각각의 정보들을 구조화하고 이들이 가진 유용한 정보를 활용한 여러 검색 수단에 대한 방안이 고려되고 있다

소프트웨어의 분석 및 재사용을 지원하기 위해 많은 연구들이 여러 측면에서 수행되어 왔지만, 그 중에서도 소프트웨어 부품들을 체계적으로 분류하여 관리하고, 사용자가 적합한 부품을 검색하는 과정은 매우 중요하다. 여기에서, 부품의 검색 방법은 등록된 부품의 분류 방식에 매우 의존적인데, 주로 사용하는 방법으로는 색인에 의한 방법[1], 패싯 방법[2], 열거형 방법 등이 있다. 그러나, 여러 방식 중에서 열거형 방법은 실제로 부품들이 등록되기 전에 미리 모든 응용 영역을 세밀하게 예측하여 분할하여야 하기 때문에, 동적인 영역의 수정 · 분할에 제한적이며[3], 색인과 패싯에 의한 방법 역시, 부품 관리자가 부품의 분류를 위해 사용하는 패싯 값과 일반 사용자가 패싯 질의에 표현하는 패싯 값 사이의 의미적 불일치성을 해결할 수 없다는 한계를 가지고 있다

객체지향 개발 환경은 개발자가 목적 시스템의 구축에 필요한 여러 개념적 도구들을 제공하고 있다. 개발자들은 이 도구들을 통해, 보다 강한 표현력과 검증된 소프트웨어 생산물을 작성할 수 있다. 고안된 소프트웨어 생산물은 소프트웨어 개발 과정에서 고려한 매우 풍부하고 다양한 의미들을 내포하고 있으며, 이들은 하나의 독립된 단위로 구분되어 명세된다. 또한 시스템은 각 내역들을 서로 연관시키고, 참조할 수 있는 수단을 제공하고 있다

이와 같이, 개발 환경에 있어 작업의 단위가 매우 다양화되었다. 즉, 기존의 개발 환경에서의 작업 대상은 함수 레벨이었으나, 객체지향 방법에서는 설계단계의 단위 클래스 혹은 요구분석 단계의 spec. 까지 매우 다양화하고 있다. 따라서, 분석 및 재사용 측면에서 사용자가 시스템이 가진 다양한 분석 정보를 바탕으로 한 부품의 분류 및 검색을 위한 요구도 점차 대두되고 있다.

본 논문에서는 이와 같이 개발자에 의해 작성되고, 시스템에 의해 파악된 다양한 의미 정보를 바탕으로, 사용자의 요구에 적합한 소프트웨어의 검색 방법에 대해 설명한다 본 논문의 2장에서는 연구의 배경과 관련 내용을, 3장에서는 분석 환경내의 각종 데이터와 표현 방식과 내역의 검색을 위한 질의에 대해 설명하고, 4장에서는 결론 및 향후 과제에 대하여 기술한다

2. 관련연구

객체지향 개발 방법론은 실세계를 "상호 작용하는 객체들의 집합"으로 인식한다. 따라서, 목적 시스템을 크게 객체 및 그 구조를 정의하는 정적 관계 부분과 객체들간의 상호작용을 정의하는 동적 행위 부분으로 파악한다 파악된 내역은 객체도, 동적도와 같은 그래픽화된 설계정보들로 작성된다. 예를 들어, 그림 1의 'Movie Box' 객체도는 'Control', 'Motor' 클래스와 이들과 집성화 관계가 있는 'Movie Box' 클래스를 보이고 있다. 또한, 'Control' 클래스의 두 메

소드는 두 상태간의 변화를 유발하는 행위로서 명세되고 있으며, 그 결과 상태 변화에 따라 'Motor' 클래스의 'speed' 속성 값이 변경되고 있음을 알 수 있다

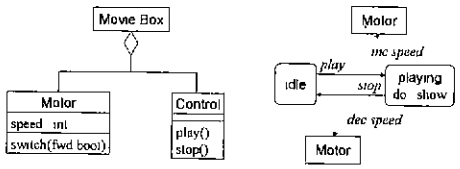

그림 1. 'Movie Box' 객체도와 'Control' 동적도

이외에도, 각종 사건(event)에 대한 흐름과 대상을 명세한 사건추적도(Event Trace Diagram) 등과 함께 명세된다. 또한, 명세된 'Motor' 액터(Actor)의 'speed'는 그림 1에서 보인 'Movie Box' 객체도에서 명세되었음을 알 수 있다. 이와 같이, 객체 모델링의 관점에서 한 객체도가 이와 연관된 동적도 들과 밀접하게 관련되어 있음을 알 수 있다

3. 설계내역의 검색

본 논문에서는 이미 고안된 바 있는 설계내역 저장소가 가진 시맨틱을 사용한 질의 방법에 대해 설명한다

3.1 설계정보 저장소

소프트웨어의 설계 내역은 재사용을 위해 미리 정의된 설계정보 데이터베이스[6]내에 유지된다. 이 데이터베이스는 설계 내역 자체에 대한 저장뿐만 아니라 여러 설계 내역들간의 관계(예를 들어, 'Movie Box' 객체도와 'Control' 클래스의 동적도 간의)도 같이 저장되어 관리된다. 그림 2는 'Movie Box' 객체도에 대한 저장 방식을 예로 보인 것이다.

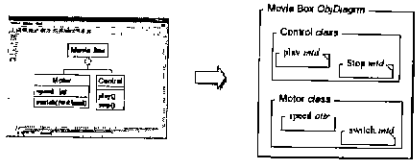

그림 2. 'Movie Box' 객체도의 저장

참조 내역은 자체의 고유한 의미를 표현하며, 다른 내역과 연관성을 가진다. 각각 단위 시스템내에서 유일한 항해 식별자 N에 의해 구분된다 모든 참조 내역 i_k, $i_l \in RC$의 항해 식별자 n_k, $n_l \in N$에 대해, $i_k \neq i_l$이면 $n_k \neq n_l$이고, 만일 $i_k = i_l$이라면 $n_k = n_l$을 항상 만족한다.

설계정보 데이터베이스는 클래스(객체도, 클래스, 속성 등)에서 명시된 특성을 가진 내역을 유도하기 위한 추출 연산 F_P를 가진다. C를 추출된 내역 집합이라 할 때, 추출 연산 F_P는 다음과 같이 정의된다.

$$F_P : D \rightarrow C$$

F는 다수의 virtual 함수로 구현되었으며, P는 추출 연산 F가 적용될 클래스 명이다 속성 D는 설계 정보를 식별하는 값이다. 예를 들어, $F_{객체도}$(Movie Box)=C_1인 실계 정보 집합 C_1은 'Movie Box'을 구성하는 설계 정보이며, 모두 그림 2에서 보인 'Movie Box' 객체를 구성하는 인스턴스들 집합이다. 다시, $C_1.F_{클래스}(*)=C_2$에서, $C_2 \subseteq C_1$이고, $s \in C_2$인 s는 'Movie Box' 객체도 일부 설계 정보로 '클래스'의 인스턴스이며, s의 이름∈{'Movie Box', 'Control', 'Motor'}이다. 따라서, 'Movie Box' 객체도의 'Control' 클래스는 $F_{객체}$(Movie Box).$F_{클래스}$(Control)로 유도된다.

그런데, 설계정보 데이터베이스내에는 이러한 내역의 구조적인 정보이외에도 각 내역들간의 의미적인 연관성을 파악하고 있다. 우리는 이러한 의미적인 연관성을 표현하는 참조 내역들간의 관계 R을 내역간의 관계명과 내역을 명시하는 항해 식별자를 갖는 세요소 튜플로 표현한다 현재 시스템이 파악하는 연관성의 종류는 객체도에서 코드와 같이 다음 개발 단계 내역으로의 발전을 나타내는 '진화' 관계, 코드내 메소드 선언부에서 정의부로의 '정의' 관계 등이 있다. 이때, 항해 식별자의 정의에 따라 모든 내역은 자신만의 유일한 항해 식별자를 갖는다 따라서, 두 내역간의 관계 R를 내역의 항해 식별자를 사용하여 정의한다.

참조가능한 내역들의 관계 트리플 $t = < r, n, n' >$
여기서, $n, n' \in N$이고,
$r \in R' = (\text{'진화'}, \text{'정의'}, \ldots)$
R'는 n, n'로 구분된 내역 c, c'∈ RC간에 가능한 관계

여기에서 관계는 참조 가능한 내역들간에 참조하는 관점을 의미한다. 예를 들어, 그림 1에서 'Control'클래스내 'play' 내역($=i_\gamma$)은 동적도내의 전이 'play' 내역($=i_\theta$)과 다른 뷰 관점을 가져, 관계 트리플 $t_{\gamma\theta}$는 <진화, n_γ, n_θ>로 표현될 수 있다. 개발 환경을 지원하는 여러 도구는 클래스(class)나 상태(state), 액터(actor)와 같이 참조 가능한 내역을 새로이 생성할 때, 내역의 항해 식별자를 얻고 다른 내역과의 관계를 파악한다.

요약하여, 부품은 고안된 관점에 따라 여러 형태로 보여지며, 보다 세밀한 여러 정보들의 조합으로 구성되는데, 이 부품의 내용을 저장하기 위한 저장 스키마를 다음과 같이 추상화하여 볼 수 있다. 즉, 한 부품을 나타내는 부품 인스턴스는 '요구분석', '분석/설계', '코드'의 인스턴스들의 조합이며, 이 중에서 '객체도'를 예로 든다면, 객체도는 다시 보다 세부적인 정보를 나타내는 '클래스'의 인스턴스들로 이루어진다.

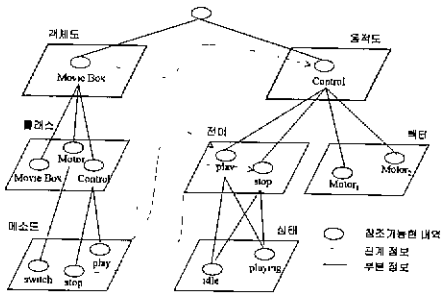

그림 3 설계정보 저장소

이와 같이, 일련의 시스템에서 파악한 의미 정보들을 설계 정보의 세부 내역을 명세하는 부분 관계와 동일하게 표현함으로써, 우리의 모델에서는 관계 정보를 추출 함수의 조합으로도 해석하고 파악할 수 있다. 이를 위해, 사용자가 요구한 관계 정보를 실제 설계 정보 저장 스키마내의 연산으로 변환하기 위한 과정이 추가로 필요하다.

3.2 일반적인 형태의 항해 질의

개발자가 저장소 내에서 특정 내역을 파악하기 위해서는 내역에 대한 다양한 요구가 발생된다. 그러나, 개발자가 각 부품내의 설계 정보들, 설계 정보내의 세부 내역들이 부품 저장소에 어떤 스키마 내에, 어떤 방식으로 저장되어 있는가에 대해 인식하기는 쉽지 않다. 또한, 이러한 다양한 요구들을 기본 연산인 추출 및 관계 함수들의 조합으로 명세하는 것 역시 쉽지는 않다. 예를 들어, '동적도 내의 모든 액터(actor)의 속성들을 추출하라'는 사용자 요구가 주어졌다고 하자. 이 요구의 결과를 얻기 위해서는 동적도를 저장한 스키마의 내역과 액터의 속성을 추출하기 위한 스키마(객체도 내의 클래스의 저장 스키마 내에 정의된)에 대한 구조를 알고 이를 사용한 질의가 필요하다.

고려된 질의어는 일반 데이터베이스 사용자들에게 친숙한 표준 SQL에 항해 모델의 특성을 반영하기 위한 각종 경로(path) 연산자를 사용하도록 확장하였다. 이들은 최종적으로 추출 및 관계 함수로 변환되어 수행된다.

질의의 예로서, 개발자는 액터의 메소드를 추출하기 위한 사용자 요구에 대한 질의를 다음과 같이 간단하게 명세할 수 있다

```
select M.메소드
from 객체도.클래스 M
where M.*.액터.name='motor'
```

이 질의에서, '*(zero or more)'는 항해 모델 내에서 여러 경로(부분, 항해 관계)로 대치될 수 있다는 표현이다 따라서, 이 질의는 여러 개의 질의로 변환될 것이고, 그 각각에 대해 적합성을 평가한 후 실행될 것이다. 만일, 이 질의가 그림 3에 적용되었다면, 다음과 같이 변환된 질의는 적합하게 평가되어 실행되고, 결과를 얻을 수 있을 것이다.

```
select M.메소드
from 객체도.클래스@M.관계(*) N
where N.name like 'motor%'
```

이 질의에서 '@(assign)'는 현재까지의 경로(부품 객체도.클래스)를 경로 변수 M에 할당하는 할당 연산자이다. 변환된 질의에서 새로이 추가된 경로는 항해 모델내의 추적 가능한 경로 중의 하나이며, 모델이 인식하고 있는 가능한 경로 중에 하나가 확정된 것이다. 따라서 모델 내에 '클래스'에서 '액터'까지의 또 다른 경로가 있을 경우 해당 경로로 확장되어 다시 실행되고, 그 결과는 결합(union)된다.

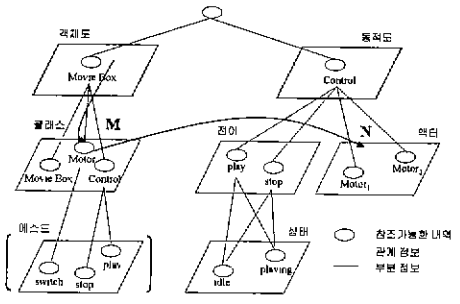

그림 4 질의의 적용

이 질의는 실제로 다음 그림 5와 같이 일련의 추출 함수들로 변환되어 실행된다 그림 5에서는 객체도 내에 포함된 모든 클래스에 대해, 클래스가 동적도 내의 액터로 사용되고 있으면 즉, 이 클래스에서 액터로의 항해 정보가 존재한다면, 해당 클래스의 속성이 질의의 결과임을 보이고 있다.

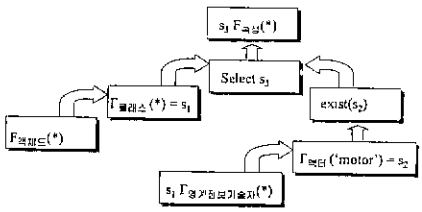

그림 4. 추출 함수를 이용한 질의 표현과 실행

이와 같이, 항해 모델을 기반으로 한 사용자 질의는 모두 일련의 추출 함수들의 조합으로 재구성되어 실행된다 이때, 질의의 결과로 얻어진 결과는 다양한 형태의 설계 정보이다. 즉, 객체도내의 '클래스' 이미지일 수 도 있으며, 코드내의 '텍스트'일 수도 있다. 따라서, 검색된 결과를 그 형태에 따라 구조화하고 시각화하기 위한 도구가 필요하다.

4. 결론 및 향후 연구과제

본 논문에서는 설계 정보 자체와 그 들간의 의미 정보들에 대해, 추상화된 인식 모델을 제공하고, 이 모델위에서 내역의 의미 표현에 적합하고, 사용자에게 친숙한 질의어를 제안하였다.

이를 사용한 부품의 검색 방법은 개발 환경내에 내역이 가진 여러 의미적인 정보를 파악하고 이를 활용한 검색이 가능하기 때문에, 소프트웨어 분석 및 재사용을 과정에서 소프트웨어 부품의 구분 및 적용에 많은 도움을 줄 수 있다.

향후 연구 과제로, 보다 세밀한 프로그램 코드의 참조 및 분석을 위한 추가의 수단이 필요하고, 저장소가 비정형화된 여러 형태의 분석 정보를 다루기 위한 추가의 고려가 필요하다.

참고문헌

[1] W. B Frakes and T. P. Pole, "Proteus: A Software Reuse Library System that Supports Multiple Representaion Methods," SIGIR Forum, Vol. 24, No. 3, pp. 43-55, 1990

[2] J M.Morel and J. Faget, "The REBOOT environment," 2nd International Workshop on Software Reusability : Advances in Software Reuse, pp. 80-88, 1993.

[3] R Rada and B. K. Martin, "Augmenting Thesauri for Information Systems," ACM Transaction on Office Information System, Vol. 5, No.4, pp. 378-392, 1987.

[4] R. H. Bourdeau and Betty H. C. Cheng, "A Formal Semantics for Object Model Diagrams," IEEE Trans on Software Engineering, Oct. 1995.

[5] J. Rumbaugh et al., Object-Oriented Modeling and Design, Prentice Hall, 1991.

[6] 배명남, 김훈희, 양재동, 최완, "객체 모델링을 지원하는 의미 기반 설계 도구의 개발", 한국정보과학회 논문지(C), Vol. 3, No. 3, pp. 248-261, 1997.