재사용을 통한 객체 모델링 지원 기법

김 정아†

관동대학교 컴퓨터 교육과

요 약

윈도우 프로그래밍과 인터넷 프로그래밍의 수요가 증대함에 따라 객체 지향 프로그래밍 언어에 대한 교육과 객체 지향 소프트웨어 개발에 관한 교육의 중요성이 높아가고 있다. 그러나, 새로운 분야의 개발 기법을 익힌다는 것은 쉬운 일이 아니다. 본 논문에서는 소프트웨어 재사용의 개념과 기법을 객체 모델링 교육에 접목하려고 노력하였다. 즉, 객체 모델링 단계에서 이전의 경험을 재사용할 수 있는 환경을 통해 객체 모델 구축 기법을 효과적으로 학습하도록 지원하고자 한다. 이를 위하여 학습과정에서 질의와 라이브러리에 저장된 컴포넌트에 대한 유사,일치성(Aanalogy)을 판단하여 라이브러리의 모델과 패턴을 재사용할 수 있는 방법을 제안하였다. 이로써 이미 잘 정의된 모델의 이해를 통해 교육 과정의 효과를 증대할 수 있을 것으로 기대한다. 또한 유추 기법(Analogy reasoning)을 활용하므로써 단순한 키워드에 의한 재사용 라이브러리 검색 보다는 보다 폭넓은 범위의 대상 검색이 가능하도록 지원한다.

Object Modeling Supporting Technique By Reuse

Jeong Ah Kim†

Kwandong University, Computer Education Department

Abstract

As window programming and internet programming are more required, requirement of the training on the object-oriented programming and the object oriented software development are growing. But, it is not easy to learn new brand methodologies or techniques. In this paper, we tried to apply software reuse to object modeling education for effective learning of new programming and modeling method. In this paper, we present analogical matching techniques for the reuse of object models and patterns in object modeling education. Analogy-based matching is better than keyword-based retrieval for model reuse. Reuse can help to reduce the learning curve of object modeling. Also, by applying analogical reasoning, the performance of retrieval is better than keyword-based retrieval.

1. 서 론

소프트웨어의 재사용은 이전의 개발 경험을 새로운 소프트웨어 개발 과정에 재적용 하는 것으로 소프트웨어 개발 환경 및 관리 과정에서 생산성 향상에 기여할 수 있을 뿐만 아니라, 새로운 소프트웨어 개발 기법을 배우는 초보자에게도 중요한 수단이 된다[4,10].

최근, 객체지향 방법론이 활성화됨에 따라 객체지향 방법을 적용하여 소프트웨어를 개발하고자 하는 시도가 증가하고 있다[1,2,3]. 그러나 객체 모델링 경험이 없는 개발자들에게는 자신에게 주어진 문제에서 적절한 객체를 식별하고 정확한 속성과 행위를 부여하는 것이 쉽지 않다[2,3]. 유사한 분야에서 이미 작성된 모델을 참조할 수 있다면, 문제에 대한 개념을 쉽게 이해할 수 있고 이를 바탕으로 새로운 모델을 구축할 수 있을 것이다. 즉, 새로운 모델링 기법에도 재사용이 적용 될 필요가 있다.

† 정회원: 관동대학교 컴퓨터교육과 교수
논문접수: 2001년 11월 1일, 심사완료: 2002년 1월 3일

코드 재사용에는 코드의 기능적 특성을 표현하는 키워드를 중심으로 한 키워드 기반 매칭 기법이 적용될 수 있지만, 모델은 분석가에 따라 동일한 개념이라도 서로 다른 용어로 모델링 할 수 있고, 서로 다른 개념을 응용 영역에 따라 같은 용어로 모델링 할 수도 있기 때문에 코드 재사용 기법과는 달라져야 한다.

본 논문에서는 객체 모델링 학습에 유추 기법을[4,6] 바탕으로한 유추적 매칭(analogical matching)을 적용하여 재사용에 기반한 학습 환경을 제안하고자 한다. 모델링의 재사용은 어떤 추상화 관점에서 구축된 모델의 재사용인가에 따라서 모델의 유사성을 판단하는 기준이 달라질 수 있을 것이다. 본 논문에서는 요구 분석 단계에서 도출되는 객체 모델과 상세 설계 이전에 작성되는 개략적 객체 모델을 대상으로 한다. 그 이유는 현재 객체 모델링을 지원하기 위한 개념으로 제시되고 있는 미리 잘 정의된 패턴이 분석 패턴과 설계 패턴의 개념으로 제시되고 있기 때문이다. 즉 본 연구에서 모델링 과정에서 유사, 일치성을 판단할 기본 저장소의 객체 모델 부품으로 이미 입증된 분석 및 설계 패턴을 사용하기 위해서이다. 다양한 추상화 수준의 결과에 대한 유추 기법은 기존의 코드 재사용에 적용한 유사도(similarity) 측정 방식과는 차이가 있다. 유사도란 두 대상의 특성이나 본질들 사이에 공통되는 성질이 존재함을 의미하지만, 유사 일치성(analogy)이란 두 대상들간에 어떠한 관점에서든 서로 일치하는 구조나 대응하는 성질이 존재함을 의미한다[4,6].

본 논문에서는 유추기법에 기반한 객체 모델과 패턴(pattern)의 재사용 환경을 제안한다. 이를 위해 객체 모델에 대한 라이브러리 표현 기법을 정의하고, analogical matching을 통한 검색 기법을 정의하고 이를 기반으로 한 검색 엔진을 개발하므로써 객체 모델 학습자가 재사용을 통해 모델링을 학습하는 환경을 구축하였다.

2장에서는 재사용에 의한 학습 모델을 제시하고 3장에서는 analogical matching 기법을 기반으로 한 패턴의 구문적 매칭 기법과 의미적 매칭 기법을 정의한다. 4장에서는 구축한 재사용 라이브러리로부터 주어진 상황에 적합한 모델을 검색하는 매칭 에이전트(agent) 구현에 대해 기술하고 이에 대한 평가를 했다. 5장에서는 결론과 향후 연구 방향에 관하여 기술하였다.

2. 재사용을 통한 객체 모델링 학습 모델

객체 모델링의 중심은 해결할 문제로부터 객체와 이들 간의 관련성을 추출하는 것이다. 객체 모델링은 크게 문제의 정적 특성을 정의하는 객체 모델, 동적 특성을 정의하는 행위 모델로 구분된다. 객체 모델링 관련 교육 중에 학습자가 격게 되는 가장 어려운 점은 문제로부터 객체를 식별하는 과정이다. 그 이유는 지금까지 적용한 개발 방법론이 대부분은 기능 중심이거나 최근에는 정보 공학 기법 중심으로 프로세스와 엔터티가 구분된 엔터티 중심의 모델링 기법이였기 때문이다. 즉, 자신들이 구축한 모델이 객체 지향 개념을 따르고 있는지 아닌지에 대한 확신을 갖지 못하는 것이다.

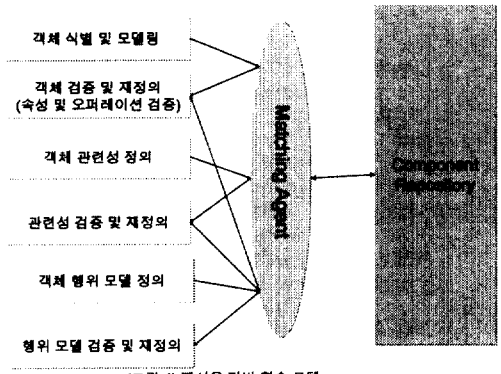
재사용을 객체 모델 학습에 반영하려는 목표는 다음과 같다.

1) 학습 과정에서 이미 작성된 검증받은 모델을 바탕으로 객체 추출과 관련성 추출이 제대로 되었는지 확인한다.

2) 이미 잘 설계된 모델을 참조 하므로써 객체 지향 개념을 잘 반영한 모델 구축 기법을 익힌다.

객체 모델링 학습에 재사용을 적용할 수 있는 근거는, 학습 과정에서 사용하는 객체 모델링의 사례 및 예는 거의 일정한 범위에서 이루어지기 때문에 재사용 라이브러리에 대한 카탈로그 작성이 가능하며, 검색의 영역이 방대한 것이 아니므로 matching agent의 검색 시간도 실시간으로 이루어 질 수 있다. 즉, 모델링 과정에서 즉각적인 권고 모델을 보여 줄 수 있게 되기 때문이다. 물론 모델의 검색을 위해서 본 논문에서는 일반적인 키워드 매칭이 아닌 Analogical matching을 사용한다.

<그림 1>에 재사용에 기반한 객체 모델링 학습 모델을 제시한다. 기본적으로 객체 모델링의 과정을 거치면서 각각의 모델-객체 모델(객체, 관련성), 행위 모델-을 정의하면 Matching agent 에 의해 Component Respository에 저장된 객체 모델 및 패턴과 비교후 유사한 것을 찾아서 대안을 제시한다. 객체 모델링 후 agent는 유사한 객체 및 속성을 제시하고, 관련성 모델링 후 agent는 잘 정의된 객체 모델과 패턴을 제시하게 된다. 이로써 학습자는 객체와 관련성을 재정의하게 된다. 행위 모델을 작성하면 Agent는 유사한 상태 모델을 제시하면서 상태 모델에 따른 객체 모델의 변형을 권고하게 되고 이에 따라 학습자는 자신의 문제의 의미에 따라 객체 모델을 수정한다.



(그림 1) 재사용 기반 학습 모델
(Fig.1) Learning Model based on Reuse

이러한 과정에서 문제의 유형에 적합한 객체 모델링 기법을 익히게 된다.

3. Analogical Matching 기법

모델은 개념의 재사용이기 때문에, 모델의 재사용에 있어서는 키워드 매칭 방식의 검색은 부적합하다. 본 장에서는 모델의 재사용을 지원하는 Analogical matching 기법을 제안한다.

3.1 모델의 라이브러리 표현 기법

재사용 대상에 대한 유사 일치성을 평가한다는 것은 두 대상의 정적 특성과 동적 특성의 유사도를 평가하는 것이다. 객체의 단일 구조에 대해서는 객체의 정적 구조를 시그네쳐(signature)로 정의하고 시그네쳐에 기술된 인터페이스를 기준으로 하여 동적인 특성을 객체 명세(specification)로 정의하였다. 패턴에 대해서는 패턴을 이루고 있는 객체들에 대한 관련성을 중심으로 패턴 시그네쳐와 패턴 명세로 정의하였다.

3.1.1 객체 시그네쳐

객체 모델을 구성하는 최소 단위인 객체를 재사용 라이브러리에 저장하고 이에 대해 analogical matching을 수행하기 위해서는 (그림 2)와 같은 특성으로 객체를 표현한다.

```
Object = (ObjectID, ObjectName, set-of-Attributes,
          set-of-Operations,  ObjectKind)
ObjectKind = resource | process | history | logic
          | role | interaction
Attribute = (AttributeName, AttributeType,
   AttributeKey, AttributeKind, AttributeStructure)
AttributeType = data type of attribute
AttributeKey = identifier | non-identifier
AttributeKind = basic | derived | acquired
AttributeStructure = simple | composite
Operation = (OperationName, ResultType,
          set-of-Arguments)
Argument = (ArgumentName, ArgumentType,
          ArgumentStructure)
ResultType = ArgumentType = data type
ArgumentStructure = simple | composite | object
```

(그림 2) 객체 시그네쳐 표현
(Fig. 2) Representation of Object Signature

ObjectID는 재사용 라이브러리에 등록된 객체에 대한 고유한 식별자로서 라이브러리 내에서 식별자로 생성되는 번호이며, ObjectName은 객체 모델에서 부여한 객체의 이름이다. set-of-Attributes는 객체가 갖는 정적 속성들의 집합으로 속성들의 이름만 기술한다. set-of-Operations는 객체의 서비스를 나타내는 인터페이스들의 집합이다. ObjectKind는 객체의 유형을 정의하는 것으로 객체가 미리 정의한 유형들 중 어느 부류에 속하는지를 표현한 것으로 설계 라이브러리에 영역에 대한 정보나 일반적 정보를 기술하는 부분에 Is-a lattice로 표현되며, 계속적으로 추가할 수 있다.

3.1.2 객체 명세

객체 명세 부분에서는 한 객체가 가지고 있는 상태와 행위 개념을 표현하므로써 객체의 구문적 특성에다 각 인터페이스의 의미를 추가하여 정의한다. 이는 객체에 대한 동적 모델링에 해당하는 부분이다.

(1) 상태 도메인에 대한 표현

객체의 행위 중 어느 한 단계를 나타내는 상태는 객체에 대해 구문적으로 정의된 속성들의 조합으로 표현할 수 있으며, (그림 3)과 같이 정의한다.

```
State = (set-of-StateDomains, set-of-Invariants,
          set-of-StateConstraints)
StateDomain = AbstractState | Predicate
AbstractState = abstract concept of state
Predicate = concrete predicate of state
Invariant = always predicate
StateConstraint = (AttributeName, Relation,
          SourceAttribute)
```

(그림 3) 상태 도메인 표현
(Fig. 3) Representation of State Domain

상태 도메인은 어플리케이션에서 객체가 갖는 의미를 정의하는 중요한 요소이다.

(2) 행위에 대한 표현

객체의 행위는 객체의 상태와 함께 객체의 중요한 동적 의미를 제공하며, 어떤 객체가 이벤트에 의해 한 상태로부터 다른 상태로 전이되기까지를 의미한다. 이것을 (그림 4)와 같이 정의하였다.

```
Behavior = (EventName, Pre-State, Post-State,
            set-of-UpdateAttributes,
            set-of-BehaviorConstraints)
Pre-State = StateDomain
Post-State = StateDomain
UpdateAttribute = AttributeName
BehaviorConstraint = (EventName, Relation,
                      EventName)
```

(그림 4) 행위 표현

(Fig. 4) Representation of Behavior

행위는 행위를 유발하는 이벤트의 EventName과 행위를 수행하기 위해 만족해야 하는 선행조건인 Pre-State와 행위를 수행한 후 형성되는 Post-State로 정의한다.

3.1.3 패턴 시그네쳐

객체보다 큰 규모의 재사용 단위인 패턴은 객체들간의 관련성 구조를 포함하고 있다[5]. 패턴의 구문적 표현은 패턴 시그네쳐로 정의하고 의미적 표현은 패턴 명세로 정의하였다. 패턴 시그네쳐는 (그림 5)와 같이 패턴을 구성하는 객체들간의 관련성 정보를 중심으로 표현한다.

```
Pattern = set-of-Relationships
Relationship = Inherit | Aggregate | Associate
Inherit = (SuperClassName, set-of-SubClasses,
           Discriminator)
SuperClassName = SubClass = ObjectName in
object
           signature
Discriminator = abstract concept
Aggregate = (WholeClassName, Multiplicity,
           set-of-PartClasses)
PartClass = (PartClassName, Multiplicity)
WholeClassName = PartClassName = ObjectName in
           object signature
Associate = (AssociateName, (LhsClassName,
           Multiplicity, role),
            (RhsClassName, Multiplicity, role))
LhsClassName = RhsClassName = ObjectName in
           object signature
Multiplicity = 0 | 1 | n | * | Range
Range = integer '-' integer
n : integer
```

(그림 5) 패턴 시그네쳐 표현

(Fig. 5) Representation of Pattern Signature

3.1.4 패턴 명세

패턴 명세는 패턴이 가지고 있는 의미를 패턴을 구성하는 객체들간의 협력 관계로 표현한 것이며 (그림 6)과 같다. 패턴의 활용과 의미에 대한 정보를 표현하기 위해 주로 패턴을 구성하는 객체들간의 행위적 특성을 정의하는데 주력하였다. Trigger 관계는 패턴을 구성하는 객체들 사이의 메시지 송수신 관계를 정의한 것이다.

```
Trigger = (SourceObject, Operation,
           TargetObject, TargetOperation)
SourceObject = TargetObject = ObjectName in
           object signature
Operation ∈ Operation(SourceObject)
TargetOperation ∈ Operation(TargetObject)
```

(그림 6) 패턴 명세 표현

(Fig. 6) Representation of Pattern Specification

3.2. Analogical Matching 함수

본 절에서는 대상들간의 유사 일치성을 규명하는데 필요한 논리를 정의한다. 이에 대한 일반적인 analogical matching을 (정의 1)과 같이 정의한다.

(정의 1) 질의(Q)와 컴포넌트(C)간의 Analogical Matching 함수 AMatch(Q,C)

$$amatch(Q_{signature}, C_{signature}) \vee amatch(Q_{specification}, C_{specification}) \vee amatch(Q_{pattern}, C_{pattern}) \vee amatch(Q_{framework}, C_{framework}) \blacksquare$$

이것은 질의 모델과 컴포넌트 모델간에 객체 수준의 시그네쳐가 유사하거나 명세가 유사하던지, 패턴 수준의 인터페이스가 유사하거나 패턴의 상호 작용이 유사하다면 두 대상간에 유사 일치성이 존재하는 것으로 판단한 것이다.

(정의 2) 기본 매칭 함수

(1) Type Equality $T_1 \equiv_e T_2$

① T_1과 T_2의 이름이 같을 경우

② T_1과 T_2의 구문적 성격으로 자료형이 같을 경우

③ 두 타입 $T_1(e_1,e_2,...e_n)$, $T_2(m_1,m_2,...,m_m)$가 모두 복합 구조인 경우, $\forall e_i, \exists m_j \cdot e_i \equiv_e m_j$, (단, $1 \le i \le n$, $1 \le j \le m$)

④ 하나의 타입 $T_1(e_1,e_2,...e_n)$이 복합 구조인 경우 $\exists e_i \cdot e_i \equiv_e T_2$ (단, $1 \le i \le n$)

(2) Renaming $E \equiv_r E'$

① $R(E) \equiv_r E'$ or

② Is-a(E, E') ∈ DesignLibrary

(3) Type Equivalence $T_1 \equiv_q T_2$

① $T_1 \equiv_r T_2$

② 두 타입 $T_1(e_1,e_2,...e_n)$, $T_2(m_1,m_2,...,m_m)$가 모두 복합 타입일 경우, $\forall e_i, \exists m_j \cdot e_i \equiv_r m_j$,(단, $1 \le i \le n$, $1 \le j \le m$)

③ 하나의 타입 $T_1(e_1,e_2,...e_n)$이 복합 타입인 경우 $\exists e_i \cdot e_i \equiv_r T_2$, (단, $1 \le i \le n$)

(4) Matching by reOrdering $O1 \equiv_o O2$ $\forall e_i \in O_1, \exists m_j \in O_2 \cdot e_i \equiv_e m_j$, (단, $1 \le i \le n$, $1 \le j \le m$)

O : Parameter-Set | Attribute-Set | Operation-Set

e_i , m_j : ParameterName | AttributeName | OperationName ■

(정의 2)의 (1)은 매칭 하고자 하는 두 대상의 타입이 일치하는지를 판단하기 위한 기준을 정의한 것이며, (2)는 매칭 하고자 하는 두 대상의 이름이나 자료형이 같지 않더라도 동질성을 인정할 수 있는가를 확인한다. 또한 (정의 2)의 (3)에서는 매칭 하고자 하는 두 대상의 타입이 완전히 일치하지는 않더라도 의미적 동질성을 가질 수 있는 기준을 정의했으며, (4)에서는 매칭 하고자 하는 대상이 파라미터, 속성 이름, 오퍼레이션 이름인 경우, 이들이 모두 집합의 개념으로 표현되기 때문에 기술된 순서와 상관없이 매칭 시킬 수 있다는 것을 정의했다.

3.2.1 객체 시그네쳐 매칭

객체의 유형이 완전히 일치하거나 설계 라이브러리에 정의된 도메인 정보를 바탕으로 동질의 유형을 갖는 것으로 판단되면 유사하다고 간주할 수 있다. 또한 정의한 속성들간에 일치성이나 동질성이 발견되거나 오퍼레이션들간의 특성이 동질이면 두 객체는 유사한 것으로 간주한다.

(정의 3) 객체 질의 대 객체 컴포넌트간의 부분 매칭 함수

$Partial\_amatch(OQ_{interface}, OC_{interface})$

① ($\forall attribute_c \in set\text{-}of\text{-}Attributes(OC)$, $\exists attribute_q \in set\text{-}of\text{-}Attributes(OQ)$ $\cdot attribute_q \equiv_e attribute_c \vee attribute_q \equiv_q attribute_c$) 또는

② ($\forall operation_c \in set\text{-}of\text{-}Operations(OC)$, $\exists operation_q \in set\text{-}of\text{-}Operations(OQ)$ $. operation_q \equiv_e operation_c \vee operation_q \equiv_q operation_c$) ■

(정의 3)의 경우는 컴포넌트에 표현된 모든 것이 질의의 요소에 대응될 수 있으나, 질의에 표현된 요소들 중 일부는 컴포넌트에 표현되어 있지 않을 수도 있는 경우이다.

3.2.2 객체 명세 매칭

객체들간의 행위적 유사성을 판단하기 위한 객체 명세 매칭은 객체의 상태 도메인에 대한 일치 정도와 행위의 일치성을 판단하는 것이다. 또한 객체의 속성에 대한 제약 조건이 존재할 경우 이 조건에 대한 일치성도 고려해야 한다.

(정의 4) 행위 동질성에 의한 행위 매칭 함수

$amatch(OQ_{behavior}, OC_{behavior})$

if $Pre\text{-}State(operation_q) \equiv_e Pre\text{-}State(operation_c)$ or $Pre\text{-}State(operation_q) \equiv_q Pre\text{-}State(operation_c)$

then $Post\text{-}State(operation_q) \equiv_e Post\text{-}State(operation_c)$ or $Post\text{-}State(operation_q) \equiv_q Post\text{-}State(operation_c)$

(정의 4)는 서로 유사한 상태에서 유사한 오퍼레이션에 대해 반응하는 방식이 유사함을 의미한다. 이는 대부분의 객체 모델이 추상화된 상태 개념(Full, Waiting for something등)으로 표현되지만, 구체적 조건식(x.time < 100 ∧ y.interval > 10)으로 표현되는 경우도 있다. 이와 같이 구체적 조건식으로 표현되는 경우는 기존의 명세 재사용에서 이루어진 기법을 적용할 수 있다. 객체의 전체 동적 특성간의 매칭은 (정의 5)로 정의하였다.

(정의 5) Behavior Graph BG(V, E)

$V = \{v_i \mid v_i \text{ is AbstractState}\}$

$E = \{(u, v, l) \mid (u, v \in V) \wedge (l \text{ is EventName})\}$

객체의 동적 특성을 나타내는 행위는 방향성 그래프로 표현할 수 있다. (정의 5)에서 행위에 표현되는 상태는 그래프의 노드로 표현하고, 한 상태로부터 다른

상태로의 전이는 방향성 연결선(edge)으로 표현한다. 이벤트 이름은 연결선의 레이블로 표현된다.

객체의 행위를 그래프로 표현함에 따라 객체 행위에 대한 analogical matching은 그래프의 동형 사상(Homomorphism)과 합동 사상(Isomorphism)에 대한 개념[5]을 적용할 수 있다. 객체 명세의 완전 매칭은 객체의 행위 유형이 완전히 일치하는 경우로 질의에 정의된 각 상태 전이 규칙이 컴포넌트의 상태 전이 규칙과 대응될 수 있다는 것을 의미한다. 그러나 이러한 이형사상에 의한 매칭은 너무 제한적이기 때문에 동형 사상에 의한 부분 매칭을 제안한다.

(정의 6) 객체 명세의 부분 매칭 함수 Partial_amatch(BGQ, BGC)

①$\exists$(u,v,l)$\in$BGQ · $\exists$(f(u), f(v), f(l))$\in$BGC,
(단, f : E $\rightarrow$ E*)
②BGQ′에 대해 Exact_amatch(BGQ′, BGC)가 존재
③BGC′에 대해 Exact_amatch(BGQ, BGC′)가 존재
// BGQ′(V′,E′) : BGQ를 확장한 그래프
V′ = V $\cup${v_i},
E′ = E $\cup$ {(v_j,v_i,l′), (v_i,v_k,l″)}
(단, v_j,$v_k$$\in$V, $v_i$$\notin$V, ($v_j$,$v_k$,l)$\in$E,
(v_j,v_i,l′),(v_i,v_k,l″)$\notin$E′,
l′,l″ : 새로 정의할 수 있는 이벤트)
BGC′(V′,E′) : BGC를 확장한 그래프
V′ = V $\cup${v_i}
E′ = E $\cup$ {(v_j,v_i,l′), (v_i,v_k,l″)}
(단, v_j,$v_k$$\in$V, $v_i$$\notin$V, ($v_j$,$v_k$,l)$\in$E,
(v_j,v_i,l′),(v_i,v_k,l″)$\notin$E′,
l′,l″ : 새로 정의할 수 있는 이벤트)■

(정의 6)의 ①은 기본적으로 질의에 존재하는 행위 유형중 하나라도 컴포넌트에서 발견할 수 있다면 재사용 후보 모델로 간주할 수 있음을 의미한다.

3.2.3 패턴 시그네쳐 매칭

패턴은 하나의 클래스로만 구성된 것이 아니라 클래스들간의 관련성을 포함하고 있기 때문에 패턴에 대한 매칭을 수행하는 것은 보다 복잡하다. 이를테면 어떤 모델에서는 하나의 클래스에 속성과 오퍼레이션으로 정의된 것이 다른 모델에서는 서로 다른 클래스로 구분되고 이들간의 관련성으로 정의할 수도 있기 때문이다. 이런 경우라면 두 모델을 서로 유사한 것으로 판단해야 한다. 이를 위해서 먼저 패턴을 구성하는 관련성에 대한 의미를 바탕으로 매칭을 정의해야 한다. 이는 질의 객체 모델에 정의된 관련성이 컴포넌트 객체 모델에는 정의되어 있지 않지만, 상속이나 집합 관계를 통해 하나의 클래스에 모두 다 정의되어 있을 수도 있기 때문이다.

패턴에 대한 그래프 표현은 (정의 7)과 같다. 그래프의 정점은 객체를 나타내며, 연결선의 꼬리 쪽은 상속 관계의 SuperClass, 집합 관계의 WholeClass 또는 연결 관계의 LhsClass를 나타내고, 머리 쪽은 상속 관계의 SubClass, 집합 관계의 PartClass 또는 연결 관계의 RhsClass를 나타낸다. 또한 관련성은 연결선의 레이블로 표현한다.

(정의 7) Pattern Graph PG(V, E)
V = {x | x is object}
E = {(u, v, r) | (u is SuperClass | WholeClass | LhsClass) $\wedge$ (v is SubClass | PartClass | RhsClass) $\wedge$ (r is Inherit | Aggregate | Associate)} ■

질의에 정의된 모든 객체와 관련성이 컴포넌트 패턴에 대응되고, 반대로 컴포넌트에 표현된 모든 객체와 관련성이 질의 패턴에 대응되는 경우로 패턴의 완전한 매칭이 이루어지는 경우로 정의하였다. 그러나 이는 질의로 작성된 패턴이 완전하게 작성된 경우에만 가능하므로 너무 제약적이다.

(정의 8) 패턴의 구조에 대한 부분 매칭 함수 Partial_amatch(PGQ, PGC)

for each element in PGQ, PGC
$\exists$r $\in$ E, $\exists$transformation function h
from V onto V′ · (u,v,r) $\in$ E
iff (h(u), h(v), r)) $\in$ E′
end for

(정의 8)은 완전한 매칭을 좀 완화하여 질의에 정의된 모든 객체와 관련성 중 부분적으로 컴포넌트 패턴에 대응되는 경우를 정의한 것이다. 동형사상의 정의에 따라 매칭 하면 이형사상의 정의에 의한 완전 매칭보다 많은 패턴을 재사용 후보로 검색할 수 있다.

3.2.4 패턴 명세 매칭

패턴 명세 매칭의 경우는 객체들간의 관련성 구조가 정립된 상황에서 관련성에 따른 객체들간의 상호 작용 방식이 일치하는지를 판단하는 것이다.

패턴 명세의 경우도 패턴 시그네쳐 매칭의 경우와 마찬가지로 패턴을 그래프로 변환한 후 그래프 사상에 의해 매칭을 판단한다. (정의 9)는 패턴 명세를 그래프로 정의한 것이다. 여기에서 그래프의 정점은 객체를 나타내는데, 연결선의 꼬리 쪽은 ClientObject를 나타내고, 머리 쪽은 ServerObject를 나타낸다. 또한 Request는 연결선의 레이블로 표현한다.

(정의 9) Pattern Specification Graph PSG(V, E)

V = {x | x is object}
E = {(u, v, r) | (u is ClientObject) ∧ (v is ServerObject) ∧ (r is Request)} ■

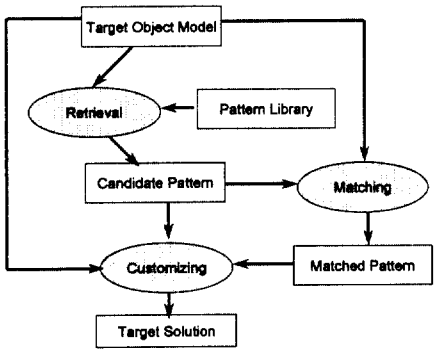
학습자가 문제에 대한 객체 모델을 작성하면 라이브러리에 정의된 객체와 패턴 모델과 제안한 analogical matching 함수 및 정의에 의해 유사한 모델을 제시하여, 작성된 모델에서 개선할 점이나 추가할 사항을 파악하게 된다. 이러한 개념을 실제 학습에 활용하기 위해서는 모델을 작성하고, 라이브러리를 검색하는 시스템의 지원이 필수적이다.

4. 재사용을 통한 객체 모델링의 학습 환경

analogical matching에 의해 모델의 재사용을 통합 객체 모델링을 학습 할 수 있도록 정의된 라이브러리 구조와 매칭 함수를 적용하여 질의를 만족하는 컴포넌트를 찾을 수 있도록 학습 환경을 구축하고 평가하였다.

4.1 학습 시스템을 위한 기본 설계

재사용을 통한 객체 모델의 학습에서 가장 중요한 핵심은 검색이다. 검색은 모델 작성 과정에서 초기에 작성한 객체 단위에서부터 점차적으로 객체들과의 관련성을 갖는 패턴으로 확대되어 간다. 따라서 검색 과정은 완전한 모델 뿐만 아니라 부분적 객체 모델도 질의로 처리할 수 있어야 한다. 재사용 라이브러리를 검색하는 개략적인 시나리오는 (그림 7)과 같다.



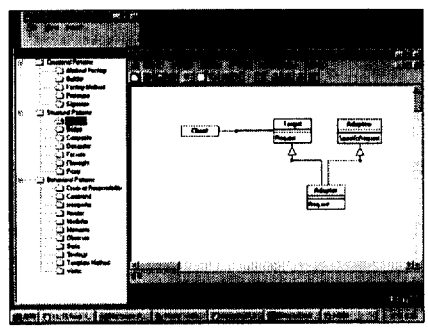
(그림 7) 재사용 시스템의 전체 흐름도
(Fig.7) Flow Diagram of Reuse System

사용자의 모델 작성 과정과 라이브러리의 검색 과정을 함께 지속적으로 모니터링 할 수 있는 analogical agent를 개발하였다.

4.2 구현 예

본 시스템은 Visual Basic 개발 환경으로 개발하였으며, 크게 모델 작성기와 모델 검색기로 구성되어 있다. 모델 검색기는 검색된 하나의 모델에 대한 사례를 살펴 볼 수 있도록 모델 이해 지원기를 포함하고 있으며, 라이브러리를 관리하는 부가적인 서브 시스템을 포함하고 있다.

컴포넌트 라이브러리에는 객체와 패턴들이 저장되어 있지만, 실제로 패턴이란 라이브러리에 저장된 객체들간의 관련성을 정의한 객체 보다 큰 추상화 단위일 뿐이다. 따라서 컴포넌트에 대한 이해를 돕기 위해 각 객체 모델이나 패턴에 대한 실제 적용 예를 제공하였다. (그림 8)은 라이브러리에 저장된 패턴의 종류에 대한 계층도와 그 중 Adapter에 대한 패턴 컴포넌트를 보여 주고 있다.



(그림 8) 라이브러리의 패턴 계층도
(Fig. 8) Hierarchy of Patterns in Library

이 패턴 계층도는 Gamma가 제안한 패턴 구조[3]를 따르고 있다. 현재 라이브러리는 Gamma의 설계 패턴[3]과 Fowler가 제안한 분석 패턴들로 구성되어 있다[2].

모델 작성기는 사용자가 클래스 다이아그램과 관련성 다이아그램을 그려서 질의로 사용할 수 있도록 지원하는 도구이다. 기본적으로는 특정 모델링 방법론에 종속적이지 않아도 되지만, 기준이 되는 다이아그램 표기법이 필요했다. 이를 위해 가장 널리 사용되는 객체 모델링 다이아그램인 UML(Unified Modeling Language)의 형태를 갖추도록 하였다[8].

모델 검색기는 작성한 질의를 바탕으로 하여 라이브

러리를 검색하고 가장 유사한 후보 컴포넌트들을 검색하는 서브 시스템이다. 완전 매칭 수준의 검색과 부분 매칭 수준의 검색을 지원하며, 객체 단위의 검색과 패턴 단위의 검색을 지정할 수도 있다. 이해 지원 시스템은 검색된 컴포넌트 모델에 대한 구체적인 내용을 확인할 수 있도록 클래스를 구성하는 관련성 정보, 클래스에 대한 자세한 속성 정보, 오퍼레이션에 대한 구체적인 정보들을 제공한다.

4.3 평가

본 논문에서는 두가지 관점으로 평가하였다. 첫 번째는 제안한 재사용 기법의 효율성을 검증하기 위하여 검색 환경에서 널리 쓰이는 평가 기법인 재현율(recall)과 정확성(precision)을 기준으로 평가하였다. 두 번째는 학습 환경으로써의 효과를 검증하였다.

4.3.1 재현율과 정확도에 의한 평가

재현율은 관련 없는 컴포넌트를 검색에서 배제시킬 수 있는 능력에 관한 평가 기준으로, 검색된 컴포넌트들 중 실제 질의에 부합되는 컴포넌트의 비율로 정의된다. 정확도란 관련된 컴포넌트의 검색 능력에 관한 평가로서, 실제 라이브러리에 저장되어 있는 질의를 만족하는 컴포넌트들 가운데 검색된 컴포넌트의 비율로 정의된다[11].

재현율 = $\frac{W}{F}$ (식 1) 정확도= $\frac{W}{R}$ (식 2)

객체 단위의 검색과 패턴 단위의 검색 각각에 대해 완전 매칭과 부분 매칭을 (식 1)과 (식 2)를 이용하여 평가한 결과는 (표 1)과 같다. 본 논문에서 실험 대상으로 한 영역은 기존의 패턴 분야에서 연구된 다양한 설계 패턴과 분석 패턴을 모델 라이브러리로 구축하여 평가 하였다.

(표 1) 재현율, 정확도에 따른 평가 결과
(Table 3) Recall, Precision Measure
(a) 재현율에 대한 평가 결과

	객체 검색	패턴 검색
완전 검색	56.3	44.2
부분 검색	75.8	60.3

(b) 정확도에 따른 평가 결과

	객체 검색	패턴 검색
완전 검색	78.6	67.2
부분 검색	66.3	58.5

재현율과 정확도에 의한 평가 기준을 본 논문에서 구축한 시스템에 적용한 결과 평균적인 기준[14]을 만족한다는 것을 알 수 있었다. 정확도와 검색 범위에 있어서는 객체 검색의 경우가 패턴 검색의 경우보다 좋다는 것을 알 수 있었다. 객체는 매칭 대상이 하나이므로 매칭 결과가 설계자의 의도에 따라 크게 좌우되지 않지만 패턴의 경우는 동일한 관련성을 정의하더라고 그 관련성을 통해 객체들간에 어떠한 상호 작용이 일어나며, 어떠한 행위가 공유될 수 있는가를 판단하는 것이 전적으로 설계자에게 달려 있기 때문이다. 패턴 매칭의 경우는 정확도가 떨어진다는 것을 알 수 있다. 이는 패턴 명세의 매칭에서 패턴의 구조를 중심으로 매칭 함수를 적용하였기 때문으로, 패턴을 구성하는 각 클래스의 특성이 해결하고자 하는 문제와 다를 수 있기 때문으로 생각된다. 그러나 부분 매칭을 통해서는 패턴의 경우도 만족할 만한 수준의 재현율을 얻을 수 있으므로, 이를 바탕으로 사용자의 수정을 거치면 문제 영역에서 요구되는 모델을 완성할 수 있다.

4.3.2 모델링 과정을 지원하는 환경으로써의 장점

모델의 재사용을 통해 객체 모델의 개선되고 학습한 적용 사례를 보이므로써 재사용이 객체 모델링에 적용될 수 있음과 그 효과를 보이고자 한다. 이를 위해 서로 다른 두 어플리케이션을 사례로 선정하였다. 하나는 도서관의 논문 복사 서비스에 관한 문제이고, 다른 하나는 부품 판매상의 주문/배달 관리 시스템에 관한 문제이다. 본 논문의 실험에서는 부품 판매상의 주문/배달 관리 모델을 이미 라이브러리에 저장해 두었다. 이 두 문제는 서로 다른 어플리케이션이고, 기능적으로도 유사하지 않다. 도서관이 가지고 있는 서비스나 정보의 속성과 물류 창고가 가지고 있는 서비스와 정보의 속성들 사이에는 유사성이 없기 때문이다. 그러나 도서관이 도서를 가지고 있듯이 물류 창고도 물품을 가지고 있으며, 도서관에서 책을 빌려갈 고객이 있듯이 물류 창고에서 물건을 받아갈 고객이 있는 등 구조적으로는 대응성이 존재한다. 이러한 관점에서 보면 도서관과 물류 창고는 유사하다고 할 수 있다. 이를 위해 설계 라이브러리에 무엇인가를 대여하는 도메인에 대한 사전 정보를 저장해 두었다. 도서관 문제에 대한 개략적 모델을 (그림 9-b)와 같이 작성하였을 때 우리는 라이브러리에 저장된 (그림 9-a)의 모델을 검색할 수 있었고, 이를 바탕으로 도서관 문제에 대한 모델을 (그림 10)과 같이 개선하였다. 즉, 새로운 문제인 판매 어플리케이션에 관한 모델을 작성할 때 Requester의 개념을 세분화하거나 기관과의 연결 속성으로 Order를 식별하거나 장부의 개념을 식별하거나

대상과 요구자간의 서비스 관계를 식별하는 등의 모델링 정보를 재사용한 것이다. 이처럼 이미 구축된 모델이 기능적으로 유사성을 갖는 것은 아니지만 문제의 구조나 문제에 내포된 객체들의 행위적 관점에서 유사성이 있다면, 기존의 경험과 모델을 재사용 하는 것이 객체 모델링의 학습 과정에서도 필요하다.

5. 결 론

모델은 문제에 대한 추상적인 개념을 보여 주는 것으로 해당 도메인에 대한 지식과 문제 해결 관점을 보여주므로써 새로 개발하고자 하는 문제에 대한 경험이 부족한 개발자들에게 해당 영역의 문제를 이해할 수 있도록 지원할 수 있다. 또한 모델링 과정에서 기존의 모델을 참조하므로써 이전의 모델링 경험을 재사용 하게 되고 따라서 사전에 모델이 잘못 구축되지 않도록 도와줄 수도 있다.

본 연구에서는 새로운 문제에 대한 객체 모델을 작성할 때 객체를 추출하는 과정과 객체들간의 관련성을 정립하는 과정에서 이전의 경험을 재사용할 수 있는 환경을 제공하기 위해 질의와 컴포넌트에 대한 유사 일치성을 판단하여 라이브러리의 모델과 패턴을 재사용할 수 있는 방법을 제안하였다.

모델을 재사용 하기 위해 객체 모델의 최소 단위인 하나의 객체와 객체들간의 관련성을 포함하고 있는 패턴으로 재사용의 대상을 구분하였고 각각을 저장하는 데 필요한 표현 기법을 정립하였다. 모델의 재사용을 위한 기법은 코드의 재사용 기법과는 달라서 구성요소들간의 완전한 일치를 통해서만 유사도를 판단할 수는 없으므로, analogical matching 기법을 적용했다. 이를 위해 필요한 정보 구조와 매칭 판단에 필요한 함수를 재사용 대상별로 정의하였다. 또한 모델 작성 과정 중 사용자에게 부합되는 모델을 제공할 수 있는 환경을 조성하기 위해 analogical agent를 설계하였다. 개발자들은 당면한 문제에서 처음으로 식별한 객체를 중심으로 부분적 모델을 작성하고 이를 바탕으로 라이브러리에서 유사 객체를 검색할 수 있으며, 이로부터 객체를 식별하고 객체의 특성을 정의하는데 필요한 정보를 제공받을 수 있다. 이러한 과정을 반복하여 객체들을 식별하고 식별한 객체들간의 관련성을 정의하여 라이브러리에서 패턴 단위의 검색을 할 수 있다. 이렇게 검색된 유사한 패턴을 통해 유사한 문제들간의 모델 재사용을 지원할 수 있게 된다.

그러나 이와 같은 모델의 재사용은 개발자들 마다의 창의성에 기반한 모델의 표현력과 문제를 바라보는 시각과 중요한 요소에 따라 달라 질 수 있다는 모델의 유연성을 떨어뜨린다는 단점이 있을 수 있다. 재사용으로 얻어지는 모델 작성의 효율성을 유지하면서 모델 표현력의 증대를 위한 연구가 계속 되어야 하며, 모델의 재사용을 통해 얻어지는 교육의 효과에 대한 명확한 검증이 필요할 것으로 보인다.

참 고 문 헌

[1] Peter Deutsch, "Design reuse and framework in the Smalltalk-80 system," In Ted J. Biggerstaff and Alan J. Perlis, Editors, Software Reusability (II), 57-72, ACM Press, 1989

[2] Martin Fowler, Analysis Patterns, Reusable Object Models, Addison-Wesley, 1997

[3] Erich Gamma, et. al., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995

[4] Gedre Gentner, "The mechanisms of analogical learning," In Bruce G. Buchanan and David C. Wilkins, editors, Readings in Knowledge Acquisition and Learning, Morgan Kaufmann Publishers, 673-694, 1993

[5] Seymour Lipschutz, Discrete Mathematics, McGraw-Hill,

[6] Neil Arthur McDougall Maiden, Analogical Specification Reuse During Requirements Analysis, Ph.D. thesis, City University, London, July 1992

[7] Bruce W. Porter, et. al., "Concept Learning and Heuristic Classification in Weak-Theory Domains," Artificial Intelligence, 45, 229-263, 1990

[8] Martin Fowler, Scott, UML Distilled, Prentice-Hall, 1999

[10] R. Alan Whitehurst, Systemic Software Reuse Through Analogical Reasoning, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1995

[11] G. Salton and M.J.Mcgill, Introduction to Modern Information Retrieval, McGraw-Hill, 1983

김 정 아

1988 중앙대학교 전자계산학과 (이학사)
1990 중앙대학교 전자계산학과 (공학석사)
1994 중앙대학교 컴퓨터공학과 (공학박사)
1994-1995 중앙대학교 기술과학 연구소 (Post-Doc)
1995-1996 (주) 아블렉스 부설 기술연구소 소장
1996-현재 관동대학교 컴퓨터 교육과 부교수
관심분야: 컴퓨터교육, 소프트웨어 재사용, 소프트웨어 공학 및 소프트웨어 개발 방법론
E-Mail: clara@mail.kwandong.ac.kr