제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

OCL을 이용한 UML Diagram의 일관성 및 정확성 검증방법

하일규*, 강병욱**
* **영남대학교 컴퓨터공학과
e-mail:ilkyuha@yumail.ac.kr,bukang@yu.ac.kr

A Technique on verifying the consistency and correctness of Class Diagram with OCL

Il-Kyu Ha*, Byong-Ug Kang**
* **Dept of Computer Engineering, Yeung-Nam University

요 약

본 연구는 UML(Unified Modeling Language) 표준에 의해 작성된 객체지향 다이어그램의 일관성과 정확성을 검증하는 방법에 관한 연구이다. 일관성은 하나의 요구사항으로 표현된 여러 가지 UML 다이어그램이 통일된 의미로 표현되었는가를 나타내는 성질이고, 정확성은 사용자가 작성한 다이어그램이 UML 표준에 적합하게 작성이 되었는가를 나타내는 성질이다. 정확성을 검증하기 위해서는 각 UML 표준에 따라 각 다이어그램 별 메타모델을 작성할 필요가 있으며, 일관성을 검증하기 위해서는 관계를 가지는 다이어그램들의 관계요소를 파악할 필요가 있다. 메타모델을 유도한 후에는 각 메타모델을 기초로 일관성 및 정확성 검증 규칙을 유도해 내고 검증의 자동화를 위하여 정형적 명세로 표현한다. 본 연구에서는 다이어그램별로 작성한 메타모델에서 규칙을 유도하고 그러한 규칙을 OCL(Object Constraint Language) 형태로 표현하여 검증하는 방법을 제안한다.

1. 서론

기존의 객체지향 소프트웨어 개발 기법들을 통합하여 객체지향 모델링 언어의 통합표준으로 제안되고 있는 UML(Unified Modeling Language)[2][4]을 이용한 객체지향 개발방법이 급속하게 확산되고 있다. UML은 시스템 개발 대상을 관점(view)에 따라 상세하게 표현할 수 있다는 장점을 가지기는 하지만, 반면에 UML로 작성된 다이어그램에 대한 정확함은 보장되지 않는다는 문제점을 가지고 있다. 따라서 초기 모델링 단계에서부터 모델에 대한 정확함의 검증을 통해 오류를 최소화하는 것이 중요하다. 일관성(consistency)은 하나의 사용자 요구사항으로부터 유도된 9가지 UML 다이어그램이 형태는 비록 다르더라도 그 의미하는 바는 같은지를 파악하는 성질이다. 또 정확성(correctness)은 사용자 요구사항을 기반으로 작성된 UML 다이어그램이 UML의 표준에 적합하게 작성이 되었는지를 파악하는 성질이다[12].

일관성과 정확성을 검증하기 위해서는 각 다이어그램의 표준모델과 다이어그램간의 관계를 파악할 필요가 있으며 메타모델이 다이어그램간의 관계와 다이어그램 자체의 표준모델을 적절하게 표현하여 줌으로 메타모델(Meta Model)[19]을 구성하는 작업이 필요하다. 메타모델은 각 다이어그램의 프로토타입과 같은 것으로 각 다이어그램에 대해서 작성될 수도 있고 다이어그램간의 관계를 표현할 수도 있다. 일반적으로 메타모델은 class 다이어그램 표기법을 이용하여 작성하게 된다. 구성된 메타모델을 기초로 일관성과 정확성을 검증하기 위한 검증규칙을 유도한다. 유도된 규칙은 텍스트 형태로 우선 정의되며 이를 자동화 검증도구에 이용할 수 있도록 하기 위하여 정형적인 명세로 표현한다. 정형적 명세의 한 방법으로서 본 연구에서는 UML specification[2]의 일부분인 OCL(Object Constraint Language)[1]을 사용한다. OCL로 표현된 규칙을 기존의 OCL 처리 TOOL을 이용하여 검증한다.

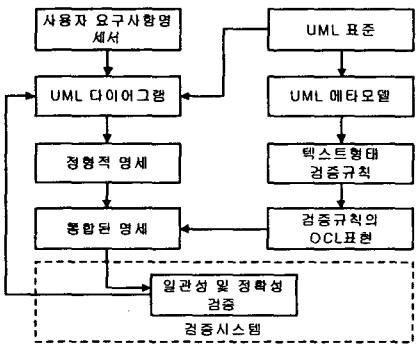
2. 관련 연구

OCL과 같은 제약언어를 이용한 검증방법은 객체지향모델이 정확하게 작성되어야함을 나타내는 제한조건을 사용하여 모델의 정확함을 검증하는 방법이다. 이 방법은 객체모델을 고유한 명세로 표현하고, 일관성 또는 완전성과 같은 제약조건을 특정한 제약언어로 표현한다. 제약조건은 검증시스템의 검증알고리즘으로 사용하고, 명세된 객체모델을 입력으로 하여 검증하는 방법을 취한다. [13]에서는 제약언어로서 MCL(Model Constraint Language)을 이용하고, [5][7]에서는 OCL을 이용하고 있다.

제약언어를 이용한 검증방법은 객체모델을 고유한 명세로 표현하기 위한 표현규칙이 요구되며, 검증규칙을 표현하기 위한 제약언어가 요구된다. 제약언어로서 OCL을 이용하는 검증방법은 UML 표준을 반영한 제약언어를 사용함으로써 검증규칙의 표현이 모델에 적합하며, [7]과 같이 표준에서 제시한 문법을 기초로 자동화시스템을 설계할 수 있다는 장점을 가진다.

3. OCL을 이용한 검증방법의 전체구조

일관성과 정확성 검증은 UML 표준으로부터 유도된 UML 메타모델을 작성하고, 메타모델간의 관계를 파악하여 텍스트 형태의 검증 규칙을 유도하고 이를 시스템에서 인식할 수 있는 형태의 언어로 변형한다. 본 연구에서는 검증 규칙을 OCL로 표현하여 자동화 검증시스템의 입력으로 사용하고자한다. 사용자가 작성한 다이어그램은 특유한 형태의 명세로 정형화 할 필요가 있다. 본 연구에서는 USE TOOL에서 제시한 방법으로 다이어그램을 명세하고 위에서 유도한 OCL 형태의 검증 규칙과 함께 자동화 시스템의 입력으로 사용한다. 일관성과 정확성 검증을 통하여 오류가 난 부분은 작성된 다이어그램에 반영되어야 하며 이는 수동적으로 행해져야 한다. 전체적인 검증절차는 [그림 1]과 같다.

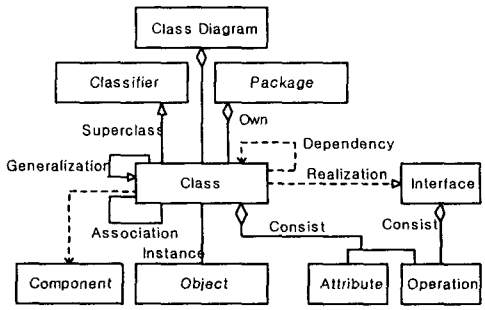


[그림 1] OCL을 이용한 검증방법의 전체구조

4. 메타모델의 유도

UML Semantics에는 Behavioral Elements, Model Management, Foundation Package 로 나누어 구성요소별 메타모델을 제시하고 있다. 여기서 제시하는 메타모델은 UML 구성요소의 정확한 의미를 전달하기 위한 모델이며 사용자가 실제 다이어그램을 그릴 때 사용하는 Class Diagram 등 9가지 각 다이어그램에 대해서는 제시되어 있지 않다. 따라서 UML 표준을 이용하여 사용자가 다이어그램을 그릴 때 구성요소별로 UML Semantics를 따라 확인해보는 작업은 상당히 불편한 일이다. 따라서 [12]에서는 모델링 작업에서 구성요소간 관계와 의미를 쉽게 파악할 수 있는 다이어그램별 메타모델을 작성하고 있다. [그림 2]는 Class Diagram의 메타모델을 설계한 것이다.

정확성 검증을 위해서는 각 다이어그램을 구성하고 있는 구성요소와 구성요소 간 관계를 중심으로 메타모델을 설계해야하고, 일관성 검증을 위해서는 다이어그램 간 대응요소를 도출하여 설계하여야 한다.



[그림 2] Class Diagram의 메타모델

5. 검증규칙의 유도

검증 규칙은 메타모델을 통해 유도한다. UML Specification에서는 각 구성요소별 Well-Formedness Rules를 제시하고 있다. 그러나 각 다이어그램별 검증규칙은 정해져 있지 않다. 따라서 본 연구에서는 UML Specification을 기초로 하여 각 다이어그램에 맞는 정확성 검증규칙을 유도하고 다이어그램간의 관계를 고려하여 일관성 검증규칙을 유도한다. 검증규칙은 모든 다이어그램에 공통적으로 적용되는 기본규칙을 기초로 각 다이어그램의 세부규칙을 도출한다. Class Diagram의 정확성 검증을 위한 규칙은 다음과 같다.

정확성 기본규칙1 : 하나의 다이어그램은 Things와 Relation으로 구분하여 각각의 구성요소만을 가진다.

■ 세부규칙1 : Class Diagram의 Things에는 class, interface, collaboration, use case, active class, package를 가진다.
■ 세부규칙2 : Class Diagram의 relation에는 dependency, association, generalization, realization관계를 가진다.
■ 세부규칙3 : 하나의 Class는 Classes, Associations, Generalizations, UseCases, Constraints, Dependency, Collaborations, Data Types, Interfaces로 구성된다.

정확성 기본규칙2 : 하나의 다이어그램 내에서 구성요소간에는 수평관계와 수직관계가 존재한다.

■ 세부규칙1 : Class Diagram에는 다음과 같은 수평관계가 존재한다.
○ Class와 Class사이에는 generalization, association, dependency 관계를 가진다.
○ Class와Interface사이에는 realization 관계를 가진다.
○ Operation과 Collaboration사이에는 realization 관계를 가진다.
■ 세부규칙2 : Class Diagram에는 다음과 같은 수직관계가 존재한다.
○ Class와 Attribute, Operation사이에는 포함(consists)관계를 가진다.
○ Class와 Package사이에는 포함관계(consists)관계를 가진다.
○ Interface와 Opreation 사이에는 포함(consists)관계를 가진다.

수평관계는 Relations으로 관계를 맺는 것을 말하며, 수직관계는 포함(consists)관계를 맺는 것을 말한다.

정확성 기본규칙3 : 하나의 다이어그램 내에서 각각의 구성요소를 포함(consists)하는 상위 구성요소 내에서는 같은 종류의 구성요소일 경우 각 구성요소의 이름은 유일해야 한다.

■ 세부규칙1 : Class Diagram의 Package 안에서 Class 이름은 유일하다.
■ 세부규칙2 : Class Diagram의 Interface 이름은 유일하다.(이상 association end)
■ 세부규칙3 : Class Diagram의 Collaboration 이름은 유일하다.
■ 세부규칙4 : Class Diagram의 package 이름은 유일하다.
■ 세부규칙5 : Class Diagram의 Class 안에서 attribute 이름은 유일하다.
■ 세부규칙6 : Class Diagram의 Class 안에서 operation 이름은 유일하다.

정확성 기본규칙4 : 각 다이어그램의 특유한 문법적 특성이 존재한다.

■ 세부규칙1 : Association에서 AssociationEnds의 이름은 유일해야한다.
■ 세부규칙2 : Class Diagram의 AssociationClass는 하나 이상의 Association에 연결시켜서는 안된다.

일관성 검증은 Diagram간의 대응관계를 중심으로 검증규칙을 유도한다. 정확성 검증규칙과 마찬가지로 기본규칙을 정의하고 다이어그램별 세부규칙을 유도한다. Class Diagram은 Object Diagram, Component Diagram, Usecase Diagram과 관계를 가진다. 일관성 검증규칙 역시 기본규칙을 기초로 하여 세부규칙을 유도한다.

일관성 기본규칙1 : 다이어그램간의 대응요소 사이에는 관계가 존재한다.

■ 세부규칙1 : Class Diagram의 Class는 Object Diagram의 Object와 instance 관계를 가진다.
■ 세부규칙2 : Class Diagram의 Class는 Component Diagram의 component와 realization 관계를 가진다.
■ 세부규칙3 : Class Diagram의 Class는 Component Diagram의 component와 dependency 관계를 가진다.
■ 세부규칙4 : Class Diagram의 operation은 Usecase Diagram과 collaboration 관계를 가진다.

6. 검증규칙의 정형명세

메타모델을 통해 유도된 텍스트 형태의 규칙을 컴퓨터 시스템이 이해할 수 있는 형태로 변환하기 위해서는 또 다른 형태로의 변환이 요구된다. 따라서 텍스트 형태의 규칙을 자동화 도구가 이해할 수 있는 형태로 변환하는 것이 필요하다. 본 연구에서는 UML의 표준제약언어로 사용되고 있는 OCL로 규칙을 표현한다. Class Diagram의 정확성 기본규칙3의 세부규칙5를 OCL로 표현하면 다음과 같다.

```
self.attribute_role_name->select (a | a.oclIsKindof
(Attribute))->forAll(p, q | p.name = q.name implies p
= q
```

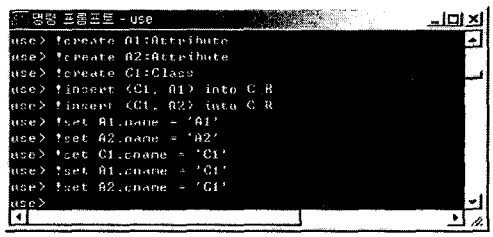
7. 검증

OCL로 표현된 세부규칙은 자동화 검증시스템을 이용하여 검증작업을 하게 된다. OCL을 입력으로 하여 문법적 오류와 제한 조건을 자동적으로 검증해주는 도구로는 IBM의 OCL parser[1], Dresden OCL Toolkit[6], TU Munich[8], ModelRun[9], USE tool[10][11]과 같은 것이 있다. IBM의 OCL parser는 OCL 표현의 문법적 분석 기능을 가진 기본적인 도구로서 다른 도구의 기초가 되며, 가장 발전된 형태의 도구가 USE로 파악된다[30].

USE는 사용자가 작성한 다이어그램을 Class 형태의 USE 명세로 변형하고, 제한자(constraints), 선후행조건(pre-/postconditions)을 부가하여 instance인 Object Diagram을 생성하면서 그 다이어그램이 정확하게 작성되었는지를 검증한다. 본 연구에서는 USE를 이용하여, 유도한 메타모델과 그것으로부터 도출한 검증규칙을 USE specification과 constraints로 작성하여 본 시스템에 적용한다. Class Diagram의 정확성 기본규칙3의 세부규칙5를 검증하기 위해 간단한 모델을 다음과 같이 구성한다.

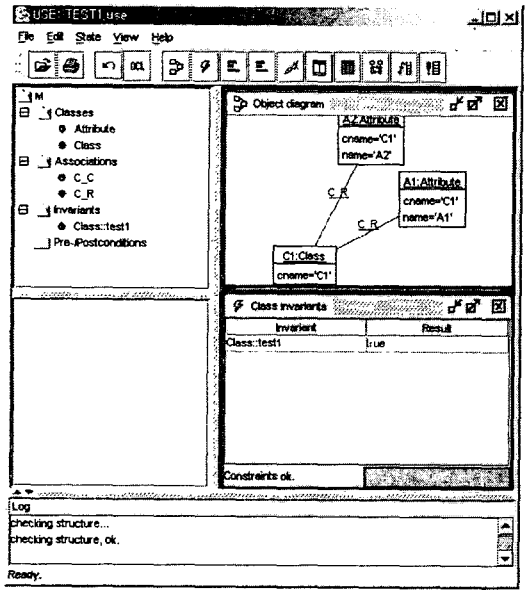
```
model M

class Class
 attributes
 cname : String;
end

class Attribute < Class
 attributes
 name : String;
end

association C_C between
 Class[*] role first
 Class[*] role second
end

association C_R between
 Class[1] role parent
 Attribute [*] role child
end

constraints
context Class
  inv test1:
self.child->select(a|a.oclIsKindOf(Attribute))->forAll(p,q|p.n
ame=q.name implies p = q)
```

이러한 명세는 parser를 통하여 Model과 OCL로 된 추상적 표현으로 변형된다. 그 다음 USE의 명령어를 이용하여 instance를 생성한다. 생성과정은 [그림 3]과 같다.



[그림 3] USE의 instance 모델 생성

Parser를 통해 생성된 Model은 [그림 4]의 왼쪽 상단부분과 같이 트리구조로 표현되고, 명령어를 통하여 생성된 메타모델의 instance는 오른쪽 상단의 다이어그램이 된다. 테스트 모델에서 제시한 두개의 제한자가 오른쪽 아래에 표시되고, 이상이 없으므로 true라는 결과가 나타난다.



[그림 4] USE의 모델 검증

USE의 다양한 기능에도 불구하고 다음과 같은 불편함을 발견하였다. 첫째, 메타모델을 검증할 수 있다고 하였지만 상당부분 USE specification으로 된 파일(*.use)을 시스템으로 load할 때 이미 Parser를 통해 정확하지 않는 다이어그램의 명세는 입력되지 않으므로 잘못된 다이어그램의 명세를 입력으로 하여 오류를 발견하는 작업을 할 수 없다. 둘째, 하나의 Model을 구성하고 그 instance를 통해 검증하는 방식으로 되어 있으므로 구성요소 자체의 관계를 표현하는 메타모델을 검증하기에는 제한점이 있다. 셋째, 메타모델을 검증하기 위해 USE에서 제시한 Core Package의 명세는 구성요소가 복잡한 관계를 맺으며 하나의 명세로 되어 있으므로 하나의 독립된 다이어그램을 검증하기에는 적합하지 않다. 넷째, 본 연구에서 제시한 일관성 검증은 두 모델을 검증하는 것인데 비해 USE는 하나의 모델을 입력으로 받아 처리하므로 적합하지 않은 면이 있다.

8. 결론 및 향후연구방향

본 연구에서는 사용자가 작성한 다이어그램의 일관성 및 정확성을 검증하기 위하여 UML 표준에 부속되어 있는 모델제약언어인 OCL을 이용하여 검증하는 방법을 제시하였고, 이에 따라 메타모델을 유도하고 그것으로부터 검증규칙을 유도하였다. 유도된 모델은 OCL로 표현되며 USE 검증TOOL을 이용하여 검증하였다. OCL은 보다 정확하고 완전한 모델을 작성하기 위해 다이어그램이 가져야 하는 제한조건(constraints)을 컴퓨터언어 형태로 표현하는 것으로 본 연구에서 제시하는 일관성과 정확성 검증조건을 표현하기에 적합한 언어라고 생각된다. OCL을 사용함으로써 얻는 이점으로는 첫째, UML 표준에 맞게 고안된 언어이기 때문에 UML구성요소와 제한조건을 정확하게 표현할 수 있다는 장점을 가진다. 둘째, OCL의 문법(grammar)이 공개되어 있으므로 새로운 도구의 개발이 용이하다는 점이다. 셋째, 기존의 복잡한 정형명세 방법에 비하여 프로그래밍 언어 형태로 구성되어 있으므로 비교적 이해하기 쉽다는 장점을 가진다.

USE는 기존의 OCL 관련 자동화 도구 중 가장 발전된 형태로 판단되지만 이미 제시한 바와 같이 몇 가지 단점이 발견되었다. 따라서 향후 연구과제로 본 연구에서 제시한 검증규칙을 모두 만족할 수 있는 자동화된 검증시스템의 개발이 필요하다.

참고문헌

[1] OMG . Object Constraint Language Specification. In OMG Unified Modeling Language Specification, Version 1.4, 2001.

[2] OMG. OMG Unified Modeling Language Specification, Version 1.4, 2001. Object Management Group, Inc., Internet: http://www.omg.org

[3] OMG, UML Semantics. In OMG Unified Modeling Language Specification, Version 1.4, 2001

[4] Grady Booch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language User Guide, Addison-Weslay, 1999

[5] Bobumila Hnatkowska, Zbigniew Huzar, Jan Magott, Consistency Checking in UML Models, 4th International conference on Information Systems Modeling ISM '01, Haradec nad Moravici, Czech Republic, 2001

[6] H. Hussmann, B. Demuth, and F. Finger. Modular architecture for a toolset supporting OCL. In A. Evans, S. Kent, and B. Selic, editors, UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings, volume 1939 of LNCS, pages 278-293. Springer, 2000.

[7] Mark Richters. A Precise Approach to Validating UML Models and OCL Constraints. PhD thesis, Universität Bremen, Logos Verlag, Berlin, BISS Monographs, No. 14, 2002.

[8] M. Wittmann. Ein Interpreter fur OCL. Diplomarbeit, Ludwig-Maximilians-Universitat Munchen, 2000.

[9] BoldSoft, Modelrun, 2000. Internet: http://www.boldsoft.com/products/modelrun/index.html

[10] M. Richters and M. Gogolla. Validating UML models and OCL Constraints. In A. Evans, S. Kent, and B.Selic, editors UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings, volume 1939 of LNCS, pages 265-277. Springer 2000.

[11] M. Richters. The USE tool: A UML-based specification environment, 2001. Internet http://www.db.informatik.uni-bremen.de/projects/USE/

[12] 하일규, 강병욱, 일관성 및 정확성 검증을 위한 구성요소와 관계로 표현된 UML 메타모델, 한국정보처리학회 봄 학술발표논문집, 2002.

[13] 김진수, 강권학, 이경환, 제약언어를 이용한 객체 모델 검증시스템, 한국정보처리학회논문지 제3권 제6호, 1996.