USE 모델검증 효율화를 위한 명령어 생성기

하일규*, 강병욱**
*영남대학교 컴퓨터공학과
**영남대학교 컴퓨터공학과
e-mail:gildong@somewhere.sck.ac.kr

A USE command Generator for the efficiency of model verifications

Il-Kyu Ha*, Byong-uk Kang**
*Dept. of Computer Engineering, Yeung-nam University
**Dept. of Computer Engineering, Yeung-nam University

요 약

USE는 OCL(Object Constraint Language)로 작성된 UML 다이어그램의 제약조건을 검증해볼 수 있는 가장 뛰어난 도구이다. USE는 다이어그램에 적용되는 제약조건을 미리 저장해두고 검증 다이어그램을 명령어 형태로 입력받아 정확성 또는 일관성을 검증하는데 사용된다. 본 연구에서는 이러한 검증과정의 복잡함을 줄이기 위하여, UML 모델링 시에 검증 대상이 되는 다이어그램의 USE 명령어를 생성하는 효율적인 도구를 설계하고 구현한다.

1. 서론

OCL(Object Constraint Language)[1]은 UML (Unified Modeling Language) 다이어그램의 제약조건을 명세하기 위한 표준 제한언어이다. OCL은 UML 다이어그램이 준수하여야 하는 정확성 규칙을 표현을 하는데도 사용된다. OCL로 표현된 정확성 규칙은 이를 해석하는 몇 가지 도구에 입력되어 다양한 형태로 다이어그램의 정확성을 검증해준다. 현재 가장 많이 사용되고 있는 OCL 해석도구는 USE tool[2]이다. USE는 OCL 해석도구 중 가장 발전된 형태로 파악되지만, 정확성 검증을 위해 검증 대상 다이어그램을 USE 명세라는 정형적 형태로 표현하여 입력하여야 하는 불편함이 있다. 따라서 본 연구에서는 USE의 검증 절차를 보다 간소화하기 위하여, UML 모델링 시에 다이어그램으로부터 USE 명령어를 직접 생성하는 아이디어를 제공하고, 간단한 모델링 도구를 구현하여 명령어 생성을 검증한다.

2. OCL 관련도구

OCL을 입력으로 받아들여 문법적 오류와 제한조건을 자동적으로 검증해주는 도구로는 IBM의 OCL parser[3], Dresden OCL Toolkit[4], TU Munich[5], ModelRun[6], USE tool과 같은 것이 있다. 이 도구들은 OCL 자체의 문법적 정확성을 체크하는 도구가 주를 이루고 있으며, 다이어그램의 정확성 규칙을 표현한 OCL을 입력받아 다이어그램의 정확성을 검증하는 도구는 소수에 불과하다. <표 1>은 OCL 관련 도구의 기능을 비교 분석한 것이다.

IBM의 OCL 파서는 OCL의 문법적 정확성을 분석할 수 있는 도구로서 가장 먼저 개발된 도구이다. 이 도구는 OCL 명세의 문법분석 기능 및 타입체크 기능을 가지고 있으며 다른 도구 발전의 기초가 되고 있다. 현재는 Klasse Objecten에서 관리되고 있다. Dresden OCL Toolkit은 OCL의 컴파일 기능을 가지는 것으로 OCL에 관한 문법 체크 기능이 UML 모델링 도구인 Argo/UML에 삽입되어 사용되고 있다. TU Munich은 OCL 메타모델을 기반으로 OCL의 문법을 체크하고 불변조건을 검증하는 기능을 가지고 있다. ModelRun은 상업적인 도구로서 다이어그램의 스냅샷에 대하여 불변조건을 검증해주는 기능을 가진다. USE는 보다 발전된 형태의 도구로서 문법적 오류뿐만 아니라 OCL의 type을 체크하는 기

능, 스냅샷에 대한 불변조건, 선후행조건을 검증하는 기능을 가지고 있다. 따라서 USE는 OCL 관련 도구 중 보다 발전된 형태로 파악된다[7].

<표 1> OCL 관련도구의 기능비교

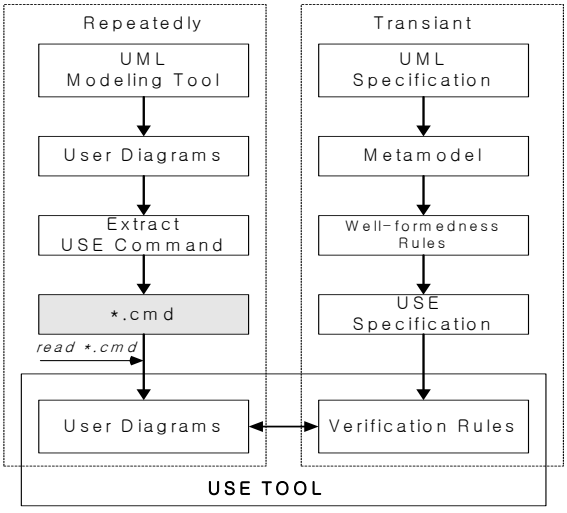
Function	TOOL				
	IBM Parser	Dresden OCL Toolkit	TU Munich	Model Run	USE
syntactical analysis	•	•	•	•	•
type checking	–	•	•	•	•
logical consistency checking	–	–	–	–	–
dynamic invariant validation	–	–	•	•	•
dynamic pre-/post-condition validation	–	–	–	–	•
test automation	–	–	–	–	•
code verification and synthesis	–	–	–	–	–

3. USE 도구의 분석

USE는 OCL을 이용한 다이어그램의 검증에 사용되는 도구로서, 관련 도구 중 가장 발전된 형태로 파악이 되지만 다음과 같은 몇 가지 문제점을 발견할 수 있다. 첫째, 검증을 위해 USE 명세라는 정형적 형태로 표현해야 하는 불편함이 있다. 이러한 문제점은 현실에서 아주 중요한 문제이다. 다이어그램의 정확성을 검증하기 위해 작성한 다이어그램을 또 다른 형태로 변형해야하는 작업은 굉장히 번거로운 일이다. 따라서 UML 도구와 연계하여 사용자가 작성한 다이어그램을 변환과정 없이 검증할 수 있도록 하는 것이 필요하다. 둘째, 명세방법이 제한적이다. 즉 사용자 다이어그램이나 메타모델을 클래스 다이어그램 형태의 USE 명세로 표현하고, 이를 검증시스템에 입력하고 인스턴스 다이어그램을 생성하여 검증하는 구조로 되어 있다. 셋째, 단독 다이어그램의 정확성 검증 기능은 구현할 수 있으나 몇 개 다이어그램간의 일관성을 검증하기 위한 환경은 제공하지 못한다. 일관성이란 다이어그램간의 상관요소의 존재여부 등을 검증하는 것이므로 몇 개 다이어그램을 수용하여 검증할 수 있는 환경이 요구된다.

이와 같은 문제점 중 가장 중요하다고 판단되는 문제는 첫 번째 문제이다. 모델링 도구를 이용하여 작성한 다이어그램의 정확함을 검증하기 위해서는 다시 다이어그램 검증도구에 다이어그램을 입력하여 검증하여야 한다. 즉 검증을 위하여 번거로운 변환과정을 거쳐야 한다. USE도 검증하고자 하는 다이어그램을 USE 명령어를 통해 생성하도록 하고 있다. 따라서 USE의 검증절차를 보다 간소화하기 위한 방안으로 다이어그램을 모델링할 때 다이어그램으로부터 USE 명령어를 직접 생성하는 도구가 필요하다. 상용화되어 있는 UML 모델링 도구의 모델 검증기능은 아직 미비한 수준이고, 특히 OCL언어의 해석은 모두 불가능하다. OCL은 사용자 수준의 검증규칙을 유연성 있게 정하여 모델을 제한할 수 있고, UML 구성요소에 밀착된 표기법을 가지는 등 다양한 장점을 가지므로, OCL 해석도구와 UML 모델링 도구의 결합이 요구된다.

USE 도구에서 사용되는 명령은 그림 5-8과 같은 create, insert, set과 같은 명령어 이외에도 write, read와 같은 명령어가 있다. read 명령은 외부에서 작성한 '*.cmd' 형태의 USE 명령어 파일을 USE로 읽어들이는 명령이고, write 명령은 USE에서 실행한 USE 명령어를 하나의 파일로 저장하는 명령이다. (그림 1)은 USE 명령어의 생성과 이용과정을 나타낸 것이다.



(그림 1) USE 명령어의 생성과 이용과정

오른쪽 부분은 메타모델과 검증규칙을 통합명세로 표현하여 검증시스템에 입력하는 과정을 나타내는 것으로서 일시적 작업에 해당된다. 즉 메타모델과 검증규칙에 대한 USE 명세는 한번만 작성하여 검증도구에 입력해 두면 계속해서 사용할 수 있다. 반면에 왼쪽은 모델링 도구에서 다이어그램을 생성하고 이에 대응하는 USE 명령어를 추출하여 검증모델을 생성하는 과정으로서 반복적 작업에 해당된다. 즉 모델링

한 다이어그램을 검증할 때마다 해야하는 작업이다. 따라서 본 연구에서는 모델링 도구에서 설계된 다이어그램을 USE 명령어를 통해 시스템에 입력하는 반복적 작업과정의 번거로움을 줄이고자 모델링 도구에서 직접 명령어를 생성하는 도구를 구현한다.

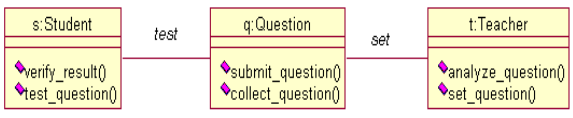
4. USE 명령어 생성기의 설계

명령어 생성기능을 지원하는 모델링 도구를 구현하기 위해서는 UML의 구성요소와 USE 명령어간의 매핑 관계를 파악할 필요가 있다. USE 명령어는 create, insert, set과 같은 명령어가 있다. create는 인스턴스 모델의 구성요소를 생성하는 명령어이고, insert는 구성요소 사이의 관계를 설정하는 명령어이며, set 명령어는 구성요소의 속성에 값을 설정하여 주는 명령어이다. 9개의 UML 다이어그램 중 Object Diagram의 각 구성요소와 그에 대응하는 USE 명령어를 살펴보면 <표 2>와 같다.

<표 2> Object Diagram요소와 USE 명령어의 대응

UML element	USE command
Object Diagram	!create diagram_name : Diagram
Object	!create object_name : Object !insert(diagram_name,object_name)into association !set Object.object_name = 'name'
Operation	!create operation_name : Operation !insert(object_name,operation_name)into association !set verify_result.operation_name='verify_result'
Association	!create link_name : Link !insert(object_name, link_name)into association !set link_name.startobject_name='name' !set link_name.endobject_name='name'

<표 2>와 같은 대응관계를 이용하여 (그림 2)와 같은 Object Diagram에 해당하는 USE 명령어를 생성해보면 <표 3>과 같다.



(그림 2) Object Diagram의 예

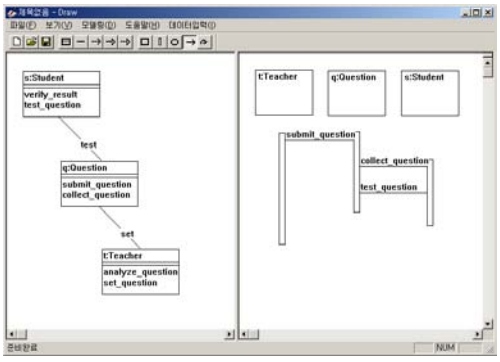
검증하고자 하는 다이어그램은 명령어를 통하여 클래스 다이어그램 형태의 인스턴스 모델이 형성된다. 인스턴스 모델의 최상위에는 Object Diagram을 나타내는 클래스가 생성되고, 이어서 Object, Operation, Association 순서로 해당되는 명령어가 생성된다. 각 Object는 create, insert, set 명령어를 생성하고, 다른 요소와의 관계를 맺어주며, 내부 값을 설정한다. Operation과 Association도 마찬가지로 해당하는 create, insert, set 명령어를 생성한다.

<표 3> (그림 2)에 대응하는 USE 명령어의 생성

UML element	element name	USE command
Object Diagram		!create OB1:ObjectDiagramElement
Object	s:Student	!create s:Object !insert (OB1, s) into OB_OBJ !set s.object_name = 's'
Operation	verify_result	!create verify_result:Operation !insert (s, verify_result) into OBJ_OPE !set verify_result.operation_name='verify_result'
Operation	test_question	!create test_question:Operation !insert (s, test_question) into OBJ_OPE !set test_question.operation_name='test_question'
Association	test	!create L1:Link !insert (OB1, L1) into OB_LIN !set L1.startobject_name='s' !set L1.endobject_name='q'

5. USE 명령어 생성기의 구현

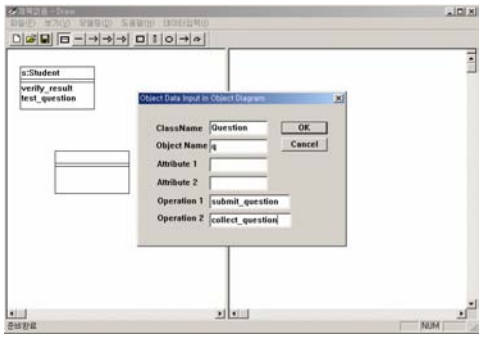
구현한 USE 명령어 생성기는 (그림 3)과 같다. 생성기는 간단한 모델링 기능을 가지고 있어서, Object Diagram과 Sequence Diagram을 모델링 할 수 있다. 화면의 왼쪽에는 Object Diagram을 모델링하게 되고, 화면의 오른쪽에는 Sequence Diagram을 모델링한다.



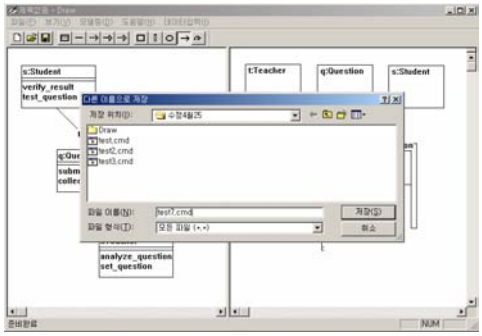
(그림 3) USE 명령어 생성기

(그림 4)는 Object Diagram의 모델링 과정을 나타낸 것이고, (그림 5)는 다이어그램으로부터 USE

명령어를 저장하는 과정을 나타낸 것이다.

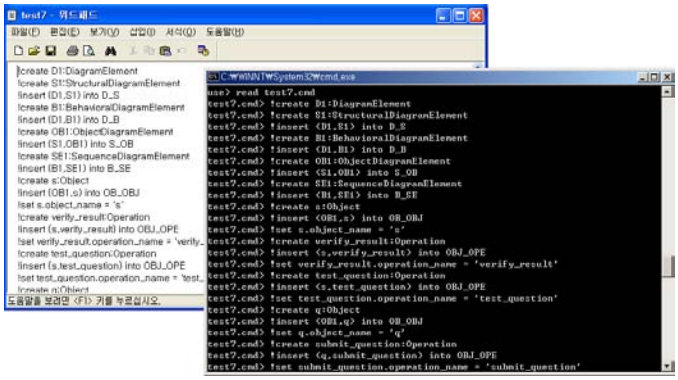


(그림 4) 객체 다이어그램의 모델링

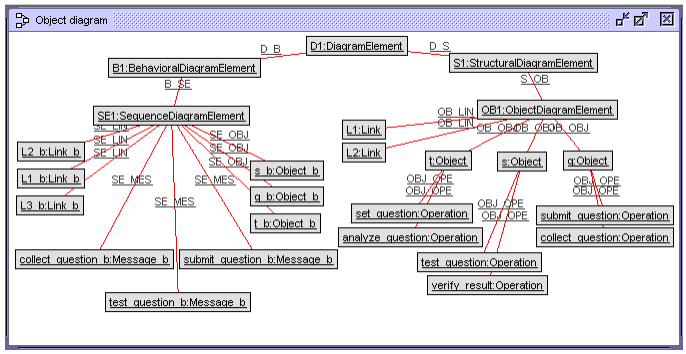


(그림 5) 생성한 다이어그램의 USE 명령어 형태로의 저장

(그림 5)와 같은 모델링을 통해서 생성한 명령어는 (그림 6)과 같다. 생성된 명령어는 'cmd' 확장자를 가진 명령어 파일로 저장된다. '*.cmd' 형태의 파일로 생성된 명령어는 USE 도구가 실행되는 '/bin' 폴더로 옮겨져 read 명령을 통해 시스템에 일괄적으로 입력된다. (그림 6)은 생성된 명령어를 USE 내부에서 일괄적으로 실행하여 입력하는 과정을 나타낸 것이다. 명령어를 통해 생성된 검증 대상이 되는 인스턴스 다이어그램은 (그림 7)과 같다.



(그림 6) 모델링 결과로 생성된 명령어와 그 실행



(그림 7) 명령어를 통해 생성된 검증 인스턴스 모델

6. 결론

USE 도구는 OCL로 작성된 제한규칙을 검증하기 위한 도구이다. 이 도구는 현재 OCL을 해석할 수 있는 가장 뛰어난 도구로 평가되기는 하지만 모델링도구로 작성한 UML 다이어그램의 정확성을 검증하기 위해서는 작성된 다이어그램을 또 다른 형태의 명세로 전환해야하는 불편함이 있다. 따라서 본 연구에서는 이러한 불편함을 덜어주기 위해 모델링 도구와 검증도구를 결합하여 모델링 도구로부터 바로 검증 모델을 표현하는 USE 명령어를 생성하는 도구를 설계하고 구현하였다. 향후 연구과제로 보다 세련된 도구로 발전시켜 9개 다이어그램을 모두 지원할 수 있도록 하는 것이 필요하다.

참고문헌

[1] OMG . Object Constraint Language Specification, In OMG Unified Modeling Language Specification, Version 1.4, http://www.omg.org, 2001.

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

[3] IBM, OCL Parser, ver. o.3, http://www-3.ibm.com/software/ad/library/standards/ocl.html, 1998.

[4] H. Hussmann, B. Demuth, F. Finger. "Modular architecture for a toolset supporting OCL," Proc. of UML2000, vol. 1939 of LNCS, pp. 278-293, October 2000.

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

[6] BoldSoft, Modelrun, http://www.boldsoft.com/product /modelrun/index.html, 2000.

[7] M. Richters, A Precise Approach to Validating UML Models and OCL Constraints, Ph.D. thesis, University Bremen, 2002.