UML 모델을 위한 메트릭 기술 언어 : MDL4UML

(Metrics Description Language for UML Model : MDL4UML)

김 태 연 [†] 박 진 욱 [††]

(TaeYeon Kim) (Jin-Uk Park)

채 흥 석 [†††]

(Heung Seok Chae)

요 약 객체지향 모델의 제약을 표현하는 OCL의 용도를 확장하여 UML 모델에 적용할 메트릭을 기술하는 언어로 사용하는 연구가 다양하게 진행되었다. 그러나 OCL로 메트릭을 기술하면 복잡한 OCL 문장으로 인하여 메트릭의 의미를 이해하는 데에 많은 어려움이 있다. 본 논문에서는 OCL의 기본 요소를 추상화시킨 새로운 메트릭 기술 언어(MDL4UML)를 정의하였다. MDL4UML은 OCL의 기본 요소를 추상화함으로써 OCL을 이용하여 메트릭을 기술하는 메트릭 디자이너가 이해하기 쉽고 간략하게 메트릭을 기술할 수 있는 장점이 있다.

키워드 : 메트릭, OCL, UML

Abstract Much research has been conducted to describe metrics for UML models by extending OCL that was proposed to define structural constraints. However, metrics descriptions in OCL are complex, so they are very difficult to understand. This paper defines MDL 4UML by abstracting the conventional OCL. By abstracting OCL constructs, the MDL4UML can produce simpler descriptions of metrics, which can help metrics designer to understand and describe metrics.

Key words : metirc, OCL, UML

1. 서 론

1997년 Object Management Group(OMG)에서 객체지향 분석 디자인을 위한 표준 언어로 UML을 결정한 이후로 UML을 이용한 소프트웨어 설계가 폭넓게 이용되고 있다. UML 표준안에는 객체지향 모델의 제약을 표현할 수 있는 Object Constraint Language(OCL)가 정의되어 있다. OCL은 UML 다이어그램에서 특정 요소에 대한 제약사항을 불변(invariant), 선행조건(pre-condition), 후행조건(post-condition)을 이용하여 나타낼 수 있다[1].

객체지향 모델의 제약을 표현하는 OCL의 용도를 확장하여 UML 모델에 적용할 메트릭을 기술하는 언어로 사용하는 연구가 다양하게 진행되었다[2,3]. OCL을 메트릭 기술 언어로 사용하는 방법은 OCL의 다양한 연산을 활용하여 UML 모델로부터 필요로 하는 정보를 얻는 방법이다.

그러나 OCL은 처음부터 메트릭을 기술하기 위해 만들어진 언어가 아니므로 OCL로 메트릭을 기술하기 위해서는 OCL과 UML 모델에 대한 높은 이해가 필요하며 OCL을 이용하여 메트릭을 표현하였을 경우 복잡한 OCL 문장으로 인하여 메트릭의 의미를 이해하는 데에 많은 어려움이 있다. 특히, 다양한 UML 다이어그램을 이용하여 표현하는 메트릭에 대한 기술이 복잡하다.

따라서, 본 논문에서는 OCL의 기본 요소를 추상화시킨 새로운 메트릭 기술 언어(MDL4UML)를 정의하였다. MDL4UML은 복잡한 UML 모델간의 연결을 자동으로 결정해주므로 메트릭을 기술하는 메트릭 디자이너는 복잡한 UML 모델에 관한 깊은 이해가 없이도 UML 다이어그램에 적용 가능한 메트릭을 쉽게 작성할 수 있다. 또, OCL의 기본 요소를 추상화함으로써 OCL을 이용하여 메트릭을 기술하는 메트릭 디자이너는 이해하기 쉽고 간략하게 원하는 메트릭을 기술할 수 있는 장점이 있다.

2. OCL을 이용한 메트릭 표현

본 절에서는 OCL을 이용하여 UML 모델에 적용 가능한 객체지향 메트릭을 표현하는 예를 보인다. 측정의 대상으로 그림 1의 클래스 다이어그램을 사용한다. OCL의 일관된 가독성을 위하여 context는 Package를 공통적으로 사용한다. 먼저, Number of Methods(NM) 메트릭을 표현해 본다[4]. NM 메트릭은 클래스 다이어그램에 있는 모든 클래스의 메소드의 수를 나타낸다.

· 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-(C1090-0801-0032))
· 이 논문은 제34회 추계학술대회에서 'UML 모델을 위한 메트릭 기술 언어 : MDL4UML'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 부산대학교 컴퓨터공학과
tykim@pusan.ac.kr
†† 정 회 원 : 네오플 A-Shock 스튜디오
jinuki79@neople.co.kr
††† 정 회 원 : 부산대학교 컴퓨터공학과 교수
hschae@pusan.ac.kr

논문접수 : 2008년 1월 8일
심사완료 : 2008년 4월 5일

NM 메트릭의 OCL 표현
self.packagedElement->select(p:PackageableElement|p.oclIsTypeOf(Class)=true).oclAsType(Class)->iterate(c:Class;res:Integer=0|res+c.ownedOperation->size())

NM 메트릭을 OCL로 표현하여 대상 다이그램에 적용한 결과는 14이다. 그림 1의 8개의 클래스에는 총 14개의 메소드가 있음을 확인할 수 있다.

다음으로 Number of Aggregations(NAgg) 메트릭을 표현해 본다[4]. NAgg 메트릭은 클래스 다이어그램에 있는 모든 집합관계의 수를 나타낸다.

NAgg 메트릭의 OCL 표현
self.packagedElement->select(p:PackageableElement|p.oclIsKindOf(Association)=true).oclAsType(Association)->select(memberEnd.aggregation<>Sequence{uml::AggregationKind::none,uml::AggregationKind::none})->size()

NAgg 메트릭을 OCL로 표현하여 대상 다이어그램에 적용한 결과는 5이다. 그림 1에서 집합관계가 5개임을 확인할 수 있다.

마지막으로 대상 모델의 Cyclomatic Complexity (CC)을 측정해 본다[5]. CC는 프로그램의 복잡도를 측정하는 메트릭이다. 그림 2는 CC 메트릭 측정시에 클래스 다이어그램과 함께 사용될 액티비티 다이어그램이다.

CC 메트릭의 OCL 표현
self.packagedElement->select(p:PackageableElement|p.oclIsTypeOf(Class)=true).oclAsType(Class).ownedOperation->iterate(o:Operation;cc:Integer=0|cc+Activity.allInstances()->select(a:Activity|a.name=o.name)->iterate(a;oc:Integer=0|oc+a.edge->size()-a.node->select(oclIsKindOf(Action)=true)->size() + 2))

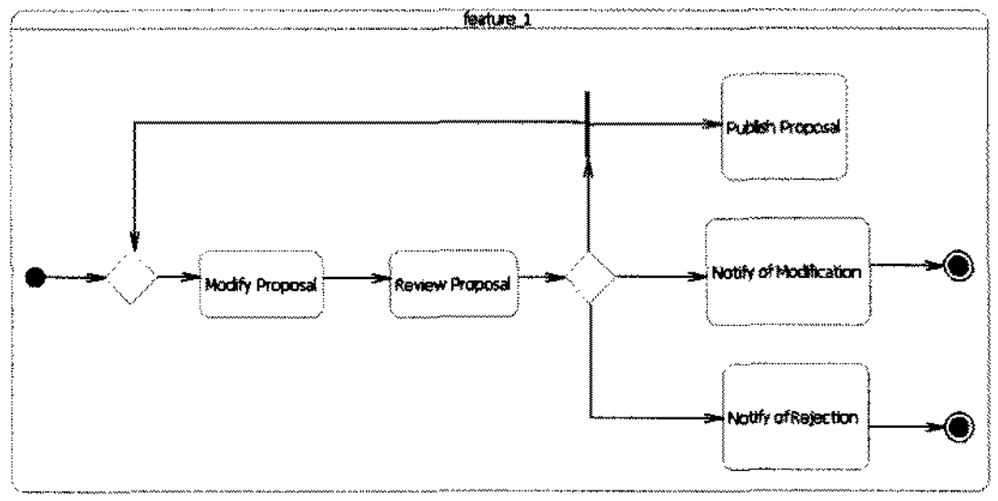

그림 2 예제 액티비티 다이어그램

CC 메트릭을 OCL로 표현하여 대상 다이어그램에 적용한 결과는 8이다. 예제 액티비티 다이어그램에서 CC가 8임을 확인할 수 있다.

3가지 메트릭을 OCL로 표현해 본 결과 3가지 메트릭 모두 OCL 표현만으로 메트릭의 내용을 이해하기가 난해하다. 그 이유를 분석해 보면 다음과 같다.

첫째, OCL의 탐색 표현식이 복잡하다. OCL에서는

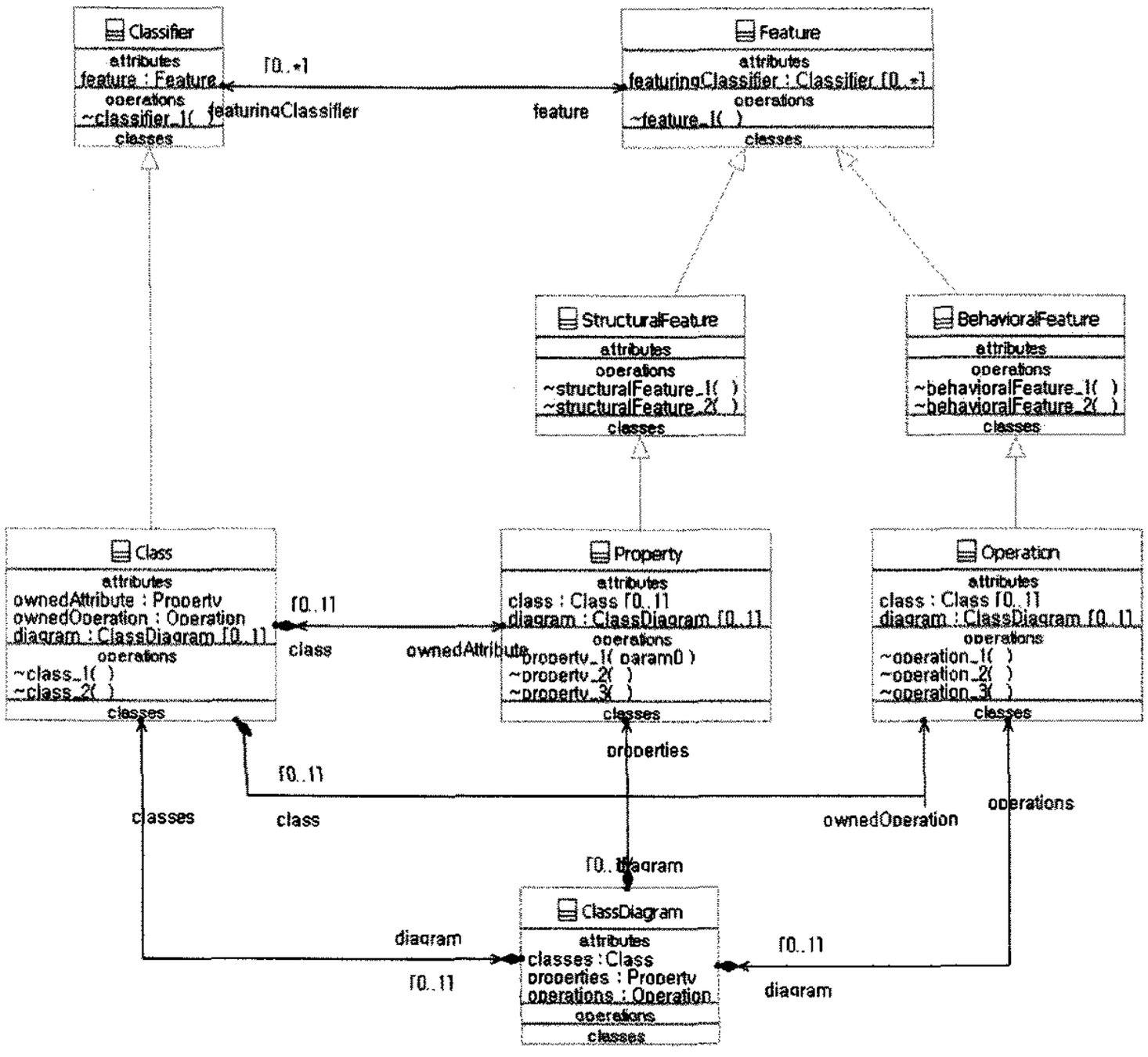

그림 1 예제 클래스 다이어그램

대상간에 탐색을 위해서 단계적인 접근이 필요하다. 그러나 UML 메타모델은 대상간에 복잡한 관계를 형성하고 있으므로 두 대상간의 접근 경로가 길면 길수록 복잡한 탐색 표현식이 불가피하다. 둘째, OCL에서 지원하는 집합함수가 단순한 기능만을 지원한다. 예를 들어, OCL에서 지원하는 sum()함수는 단순히 집합의 합을 계산하는 기능을 제공한다. 따라서 정수, 실수 집합의 경우에는 간단히 합계를 계산할 수 있으나 클래스 메소드의 집합을 가중치 3으로 합산하는 연산을 지원하지 못한다.

3. MDL4UML

MDL4UML은 METRIC과 SUBMETRIC으로 구성되어 있으며 다음과 같은 표현식으로 기술한다.

METRIC (메트릭이름) **FOR** (측정대상이름)[조건]
TYPE (타입) **VALUE** (value expression)

SUBMETRIC (메트릭이름) **FOR** (측정대상이름)[(조건)]
TYPE (타입) **VALUE** (value expression)

- 메트릭이름: 문자열로 표현되는 메트릭의 이름을 기술
- 측정대상이름: UML모델 중 대상이 되는 요소의 이름
- 조건: 측정을 원하는 대상물의 검색 조건
- 타입: 메트릭 측정 결과 형태
- value expression: 상수값, OCL표현식, 집합함수

MDL4UML의 수행 순서는 먼저 **SUBMETRIC**에 정의된 메트릭 결과값을 측정하고 그 결과를 상위 **METRIC**에서 이용하여 최종 측정값을 계산한다. MDL4UML의 이해를 돕기 위하여 MDL4UML를 이용하여 대상 클래스 내의 메소드의 개수를 측정하는 간단한 메트릭을 기술해 본다.

NOOC 메트릭의 MDL4UML 표현

```
1 METRIC NOOC FOR Class TYPE Integer
2 VALUE CNT(OP)
3 BEGIN
4  SUBMETRIC OP FOR Operation TYPE Integer
5  VALUE 1
6 END
```

MDL4UML로 기술된 메트릭의 1번 라인에 NOOC는 메트릭의 이름이고 Class는 측정대상의 이름이며 Integer는 메트릭 측정 결과가 정수형임을 나타낸다. 2번 라인의 CNT(OP)는 MDL4UML가 지원하는 집합함수 중의 하나로 SUBMETRIC 결과의 개수를 나타낸다. 3, 6번 라인의 BEGIN과 END 사이에 SUBMETRIC을 정의한다. 4번 라인의 OP는 SUBMETRIC의 이름이며 Operation은 측정대상의 이름이고 Integer는 측정 결과가 정수형임을 나타낸다. 5번 라인의 1은 SUBMETRIC 측정 대상 각각을 1의 값으로 계산한다는 선언이다.

MDL4UML은 메트릭을 기술하는 유형을 크게 4가지로 분류하였다. 첫 번째는 NOOC의 예와 같은 단순유형 경우이며 각각 METRIC과 SUBMETRIC 하나로 이루어져 있다. 단순조건유형의 경우는 단순유형의 경우와 같으나 측정 대상에 대한 조건이 추가로 기술되어 있다.

세 번째는 다중 SUBMETRIC 유형으로 동일한 레벨의 SUBMETRIC이 두개 이상 반복해서 기술되는 경우이다.

ClassSize 메트릭의 MDL4UML 표현

```
1 METRIC ClassSize FOR Class TYPE Integer
2 VALUE CNT(OC) + CNT(AC)
3 BEGIN
4  SUBMETRIC OC FOR Operation TYPE Integer
5  VALUE 1
6  SUBMETRIC AC FOR Property TYPE Integer
7  VALUE 1
8 END
```

메트릭 ClassSize는 클래스의 메소드 수를 구하는 SUBMETRIC OC와 클래스의 속성 수를 구하는 SUBMETRIC AC로 구성되어 있으며 SUBMETRIC의 결과 값의 개수들을 각각 구한 후 합산하여 최종적인 ClassSize 메트릭 측정값을 계산한다.

마지막 유형으로 는 서로 다른 대상의 SUBMETRIC을 사용하는 유형이다. 서로 다른 대상의 SUBMETRIC을 사용하는 유형의 예로 프로그램의 복잡도를 측정하는 CC메트릭을 기술하여 본다.

CC 메트릭의 MDL4UML 표현

```
1 METRIC CC FOR Class TYPE Integer
2 VALUE SUM(MW)
3 BEGIN
4  SUBMETRIC MW FOR Operation TYPE Integer
5  VALUE SUM(AC)
6  BEGIN
7   SUBMETRIC AC FOR Activity[outer.name=name] TYPE Integer
8   VALUE edge->size()-node->select(oclIsKindOf(Action)=true)->size()+2)
9  END
10 END
```

6번 라인에서 9번 라인에 SUBMETRIC이 중첩되어 사용된 것을 확인할 수 있다. 7번 라인의 outer조건은 MDL4UML에서 정의한 요소로 상위 측정대상과의 관계를 자동으로 찾아주게 된다.

MDL4UML는 메트릭을 기술함에 있어 메트릭 기술 언어로서의 OCL과 비교하여 보았을 때 다음과 같은 특징을 가진다.

- SUBMETRIC을 이용한 모듈화
  복잡한 메트릭을 기술하는 경우에 SUBMETRIC을 이용하여 메트릭에 필요한 데이타를 분할하여 처리할 수 있다.
- OCL의 select() 표현식 사용 빈도 감소
  UML 대상간의 연결에 OCL의 select() 오퍼레이션을 사용함으로써 OCL 표현이 길어지고 이해도가 떨어졌다.

MDL4UML은 UML 대상간의 연결을 자동으로 찾아주므로 select() 오퍼레이션의 사용 빈도를 감소시킨다.

- 집합함수의 지원

OCL을 이용한 메트릭의 기술시 iterate() 오퍼레이션을 이용하여 대상 집합의 value expression을 합산하거나 개수를 구하였으나 MDL4UML은 집합함수를 이용하여 대상집합에서 원하는 결과를 얻을 수 있다. NM 메트릭을 MDL4UML로 기술하면 다음과 같다.

NM 메트릭의 MDL4UML 표현

```
METRIC NM FOR Model TYPE Integer VALUE CNT(OP)
BEGIN
    SUBMETRIC OP FOR Operation TYPE Integer VALUE 1
END
```

OCL로 기술한 경우와 비교해 보았을 때 측정대상 간에 연결을 위한 select() 오퍼레이션이 전혀 없으며 Operation 집합을 대상으로 CNT(OP) 집합함수를 사용함으로써 iterate() 구문 역시 나타나지 않는다. CC 메트릭을 MDL4UML로 기술한 예는 서로 다른 대상의 SUBMETRIC을 사용하는 유형에서 설명하였다. MDL 4UML을 이용하여 복잡한 메트릭을 기술하는 경우에 SUBMETRIC을 이용하여 분할하여 기술할 수 있다. CC 메트릭의 경우 먼저 클래스 내의 메소드들의 이름과 일치하는 액티비티 다이어그램의 간선과 노드를 이용하여 CC 값을 구하고 다음으로 메소드별로 결과값을 합산하여 전체 프로그램의 CC 값을 구하고 있다.

또, 측정 대상간의 연결이 자동으로 이루어지므로 OCL 표현에서 반복적으로 나타나는 select() 오퍼레이션의 뚜렷한 감소를 확인할 수 있으며 클래스 다이어그램과 액티비티 다이어그램을 이용하여 기술된 CC 메트릭의 크기 역시 24% 감소하였다.

MDL의 표현력을 검증하기 위하여 기존 연구들에서 제안된 디자인 메트릭을 MDL을 이용하여 기술해 보았다. 그 결과 class, use cases, state machine diagram을 대상으로 size, complexity, coupling, encapsulation, inheritance 유형의 메트릭 20개를 표현할 수 있었다.

4. 자동화 지원도구

MDL4UML로 기술된 메트릭을 UML모델에 실질적으로 적용해 보기 위하여 소프트웨어 측정 도구인 METOOSE를 구현하였다. 그림 3은 METOOSE의 아키텍처를 도식화 한 그림이다. METOOSE는 MDL4UML를 OCL로 변환할 수 있고 UML 모델을 대상으로 메트릭 결과를 측정하고 관리할 수 있다. METOOSE는 두개의 중요한 도구로 구성되어 있다. MDL4UML을 입력 받아 OCL 표현으로 변환해 주는 MDL2OCL과 OCL을 UML 모델에 적용하여 결과를 측정하고 관리하는 MetriUs이다.

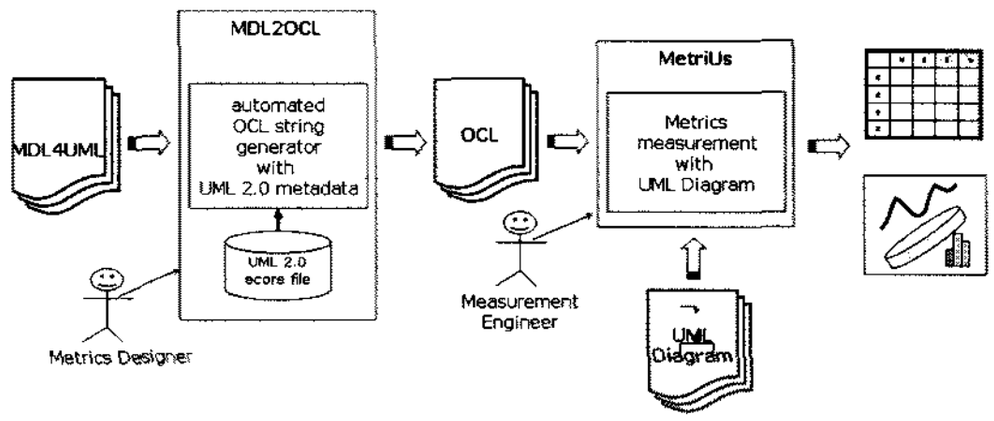

그림 3 METOOSE의 아키텍처

4.1 MetriUs

그림 4는 MetriUs의 실행화면이다. MetriUs는 총 5개의 메뉴로 구성되어 있다. Mproject는 메트릭 측정을 프로젝트별로 관리하고 프로젝트에 해당하는 UML 산출물을 업로드하는 기능을 제공한다. MetricSuite은 기술된 메트릭을 측정 유형별로 묶어서 관리하는 기능을 제공한다. Metric 메뉴는 MDL을 이용하여 메트릭을 기술하고 저장하는 기능을 제공한다. Evaluate는 UML 산출물에 메트릭을 적용하여 그 값을 계산하고 저장하는 기능을 제공한다. 마지막으로 View Result는 사용자에게 측정된 결과를 차트와 표로 조회하는 기능을 제공한다.

그림 4의 차트는 자바 Applet 1.4 버전과 1.5 버전의 클래스 다이어그램을 대상으로 5개의 메트릭을 이용하여 측정한 결과를 보여주고 있다. 빨간색은 1.4버전의 산출물을 대상으로 측정한 결과이고 푸른색은 1.5 버전의 산출물을 대상으로 측정한 결과이다.

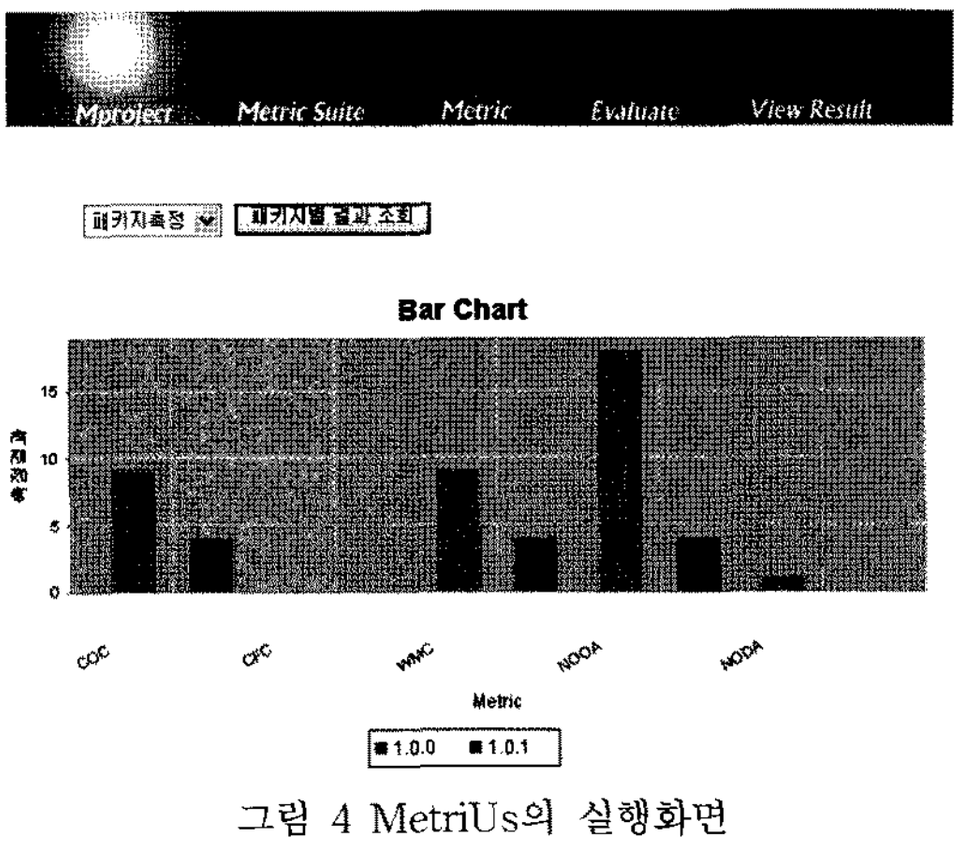

그림 4 MetriUs의 실행화면

4.2 MDL2OCL

MDL2OCL은 XML로 기술된 MDL4UML을 입력받아 DOM Parser를 이용하여 분석하고 분석된 정보를 바탕으로 복잡한 UML 메타모델 내에서 메트릭 측정에 필요한 요소들의 연결정보를 추가하여 OCL 표현을 생성한다.

MDL의 각 요소별 OCL 매핑룰을 NOPOC 메트릭 예제와 함께 설명한다.

- MDL의 **METRIC** 문장의 측정대상 이름은 OCL의 context으로 변환된다. 예를 들어 **METRIC** 문장은 "context"로 변환된다.
- MDL의 **METRIC** 문장의 메트릭 이름은 invariant의 이름으로 변환된다. 예를 들어 NOPOC 이름은 "inv NOPOC"으로 변환된다.
- **METRIC**과 **SUBMETRIC** 측정대상간의 관계는 UML 메타모델을 참조하여 OCL의 탐색 표현식이나 select 문장으로 변환된다. 두 대상간의 관계가 연관, 포함관계인 경우에는 "." + role name으로 변환되며 일반화 관계는 "시작대상->select(oclIsTypeof(종료대상))"으로 변환된다. 예를 들어 NOPOC 메트릭의 측정대상은 Class와 Operation이다. 그림 5는 UML 메타모델에서 Class와 Operaion의 관계가 포함관계임을 보여준다. 따라서 OCL의 탐색 표현식은 "self.ownedOperation"이 된다.

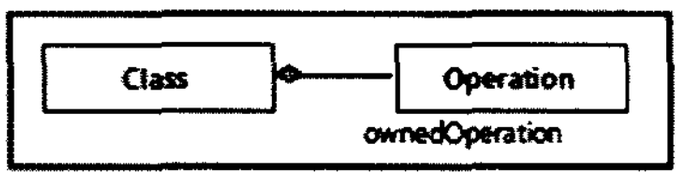

그림 5 UML 메타모델에서 Class와 Operaion의 관계

측정대상의 조건은 OCL의 select("+조건 +")"으로 변환된다. 예를 들어 NOPOC의 SUBMETRIC은 Operation에 조건이 있으므로 "->select(visibility = VisibilityKind::public)"으로 변환된다.

집합함수 중 CNT()는 OCL의 size()함수로 SUM()은 OCL의 iterate() 문장에 대응된다. 예를 들어, NOPOC에서는 CNT(OP)가 "->size()" 문장으로 변환된다.

5. 관련 연구

본 장에서는 메트릭 기술 언어로 OCL을 사용한 연구와 OCL을 지원하는 도구에 관하여 살펴본다. Baroni와 McQuillan은 UML 모델에 대한 메트릭 표현으로 OCL을 사용하여 다양한 메트릭을 기술하였다[2,3]. 하지만 OCL로 메트릭을 기술하거나 기술된 메트릭의 이해를 돕기 위한 고려가 없다.

마지막으로 메트릭 기술 언어를 제안한 연구에 관하여 살펴보자. [6]의 연구는 UML 모델을 대상으로 메트릭 적용 프레임웍을 설계하고 메트릭 기술 언어 및 기본 라이브러리를 구현하였다. 그러나 소스 코드 기반의 프레임웍이다. [7]에서는 XML파일 형식으로 메트릭을 기술한 후 SDMetrics 도구를 이용하여 메트릭을 측정할 수 있다. 그러나 UML 메타모델의 수정하여 사용하고 있으므로 표현력에 대한 검증이 필요하다. [8]의 연구는 객체지향 디자인 모델(ODEM)을 제시하고 정형 인식자를 사용하여 메트릭을 기술하였다. 그러나 메트릭 기술자가 형식언어를 사용하기 위해서는 높은 수학적 이해가 바탕이 되어야 한다.

6. 결론 및 향후 연구

OCL을 이용하여 UML 모델의 메트릭를 기술하는 경우에 UML 모델에 관한 높은 이해력이 필요하였고 OCL 표현의 복잡성을 유발하였다. 메트릭을 기술하기 위한 언어인 MDL4UML을 이용하면 UML 모델간의 복잡한 연결관계를 자동으로 찾아서 OCL 표현으로 변경할 수 있으며 OCL 표현을 추상화하여 메트릭을 기술함으로써 메트릭 디자이너는 이해하기 쉽고 간편하게 메트릭을 작성할 수있을 것으로 예상된다. 그리고 순환구조 메트릭을 표현하지 못하는 문제를 해결하기 위하여 MDL의 표현력을 높이는 연구가 향후 필요하다.

참 고 문 헌

[1] J. Warner and A. Kleppe, The Object Constraint Language : Getting Your Models Ready for MDA 2nd Ed, Addsion-Wesley Professional, 2003.

[2] J.A McQuillan and J. F. Power. "Experiences of using the Dagstuhl Middle Metamodel for defining software metrics," Proceeding of the 4th International Symposium on Principles and Practice of Programming in Java, pp. 194-198, 2006.

[3] A. L. Baroni, S. Braz, and F. B. Abreu. "Using OCL to formalize object-oriented design metrics definitions," In ECOOP'02 Workshop on Quantitative Approaches in OO Software Engineering, Lecture Notes in Computer Science:Springer-Verlog, 2002.

[4] Genero, Marcela and Manso et al., "Building measure-based prediction models for UML class diagram maintainability," Empirical Software Engineering, Vol.12, No.5, pp.517-549, 2007.

[5] McCabe, Thomas J. & Watson, Arthur H. "Software Complexity" CrossTalk: The Journal of Defense Software Engineering, Vol.7, No.12, pp.5-9, 1994.

[6] Alikacem, E. H., Sahroui, H., "Generic Metric Extraction Framework," In Proceedings of the 16th International Workshop on Software Measurement and Metrik Kongress, 2006.

[7] SDMetrics, SDMetrics Metric Language, "http://www.sdmetrics.com/CustoMetrics.html".

[8] R. Reißing. "Towards a model for object-oriented design measurement," International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, 2001.