UML과 OCL에 바탕을 둔 객체지향 소프트웨어를 위한 측정 프레임웍의 설계 및 구현

박진욱* · 채흥석**

1. 서 론

최근 컴퓨터 분야만이 아니라 다양한 분야에서 소프트웨어가 사용되는 비중이 점점 증가함에 따라 소프트웨어 개발은 급격하게 복잡해지고, 대규모화, 다양화되고 있다. 이로 인해 단지 기능만 동작하는 소프트웨어가 아니라 좀 더 나은 품질의 소프트웨어에 대한 수요가 증가하고 있다. 이와 같이 소프트웨어 품질에 대한 관심이 커지면서 품질을 예측하고 평가하는 것이 중요해 졌으며 이를 위한 소프트웨어 측정에 대한 중요성이 점점 부각되고 있다[1].

이러한 소프트웨어 측정은 다양한 소프트웨어 산출물에 대해 메트릭을 적용하고 그 결과를 산출한다. 하지만 측정의 대상이 되는 산출물의 대상이 매우 다양하다. 요구사항 단계의 산출물이나 설계단계의 산출물, 또는 구현단계의 산출물들이 모두 소프트웨어 측정의 대상이 된다. 이렇게 다양한 산출물을 표현하기 위한 방법이 필요하다. 또한 소프트웨어 개발 방법이 발전하면서 소프트웨어 메트릭도 매우 다양해져 왔다. 객체지향 개념 이전에는 소스코드의 라인수나, 전체 함수의 수등의 메트릭만이 존재하였지만 객체지향 개념이 나타나면서 클래스와 관련된 다양한 메트릭들이 새롭게 제시되었다. 그리고 컴포넌트 개념이 제시되면서 컴포넌트 관련 메트릭들도 새롭게 제시되었다. 즉, 앞으로도 다양한 개념들이 제시될 것이며 그에 따라 새로운 메트릭들이 소프트웨어 측정을 위해 필요하게 될 것이다. 그러므로 메트릭을 명시적으로 기술할 수 있는 메트릭 기술언어가 필요하다. 이러한 메트릭 기술언어의 필요에 의해 기존에 명시적인 메트릭 기술을 위한 언어들이 연구 되었지만 메트릭 기술자들이 메트릭 기술언어를 새롭게 익혀야 하거나 도구의 지원이 미비하였다. 그러므로 사용성이 좋은 메트릭 기술언어가 필요하다[2].

본 논문에서는 다양한 산출물을 표현하기 위한 방법으로 UML(Unified Modeling Language)을 사용하고 명시적인 메트릭 기술언어로서 OCL (Object Constraint Language)을 사용하며 메트릭 기술의 사용성을 증대시키기 위해 VisualOCL을 사용하는 객체지향 소프트웨어 측정 프레임웍을 설계하고 구현한다.

본 논문의 구성은 다음과 같다. 2장에서는 논문의 연구배경으로서 UML과 OCL에 대해서 설명한다. 3장에서는 객체지향 소프트웨어 측정 프레임웍을 설명하며 4장에서는 프레임웍의 개발 / 적

※ 교신저자(Corresponding Author) : 채흥석, 주소 : 부산시 금정구 장전동 산 30번지(605-735), 전화 : 051)510-3940, FAX : 051)517-5740, E-mail : hschae@pusan.ac.kr
* 부산대학교 컴퓨터공학과 석사과정 (E-mail : jwpark@pusan.ac.kr)
** 부산대학교 정보컴퓨터공학과 교수

용사례를 살펴 본다. 5장에서는 관련 연구들을 살펴보고 마지막으로 6장에서 결론 및 향후연구방향을 기술한다.

2. 연구 배경

본 장에서는 논문의 연구배경으로서 UML, OCL 에 대해서 알아 본다. 본 논문에서 UML은 프레임웍에서 다양한 산출물을 표현하기 위해 사용되며 OCL은 메트릭 기술을 위해 사용된다.

2.1 UML

UML은 소프트웨어 개발 전반에서 소프트웨어를 표현하기 위한 수단으로 사용되는 모델링 언어이다[3]. 1997년에 1.0이 발표되었으며 최근에 2.1 버전이 발표 되었다. UML은 현재 13개의 다이어그램을 표현할 수 있으며 각 다이어그램은 시스템의 특정 부분을 모델링한다.

UML은 다양한 소프트웨어 개발 방법론에서 각 과정의 산출물을 나타내기 위한 수단으로 사용된다. UP(Unified Process)[4]계열의 개발 방법론에서는 각 과정의 산출물을 UML을 사용하여 기술하도록 하고 있으며 XP(Extreme Programming)[5]에서도 UML을 사용하는 기민한 모델링 방법에 대한 필요성을 언급하고 있다.

또한 UML은 다양한 용도로 확장할 수 있는 확장 메커니즘을 제공한다. 그 중에서 가장 널리 쓰이는 것은 '스테레오타입'이며 특정 용도에 맞게 다양한 스테레오 타입을 정의하여 그룹지은 것을 'Profile'이라고 한다. 즉, 기본 UML 다이어그램이 제공하지 않는 내용을 모델링 하기 위해서 Profile을 활용할 수 있다. 예를 들어, 요구사항 단계의 산출물을 지원하는 다이어그램은 UML이 명확하게 제공하지 않지만 요구사항을 위해 정의된 Profile을 통해 UML을 확장하여 사용함으로써 UML로도 요구사항 단계의 산출물을 명확하게 표현할 수 있다. 이미 UML 2.0에서는 J2EE/EJB 컴포넌트 Profile, COM 컴포넌트 Profile, .NET 컴포넌트 Profile등을 이미 정의하고 있다[6].

2.2 OCL

OCL은 UML 모델에 제약을 표현하는 정형적 언어이다. UML 다이어그램에서 특정 요소에 대한 제약사항을 invariant, pre-condition, post-condition을 이용하여 나타낼 수 있으며 기존 모델에서 누락된 요소나 함수를 정의할 수도 있다[7].

그림 1은 OCL의 예를 적용하기 위한 UML 클래스 다이어그램의 예이다. 그림의 클래스 다이어그램은 사람, 자동차, 회사간의 관계에 대해서 표현하고 있다. 그림의 다이어그램 내용을 요약하면 다음과 같다. 사람은 하나의 차를 가지며 차는 하나의 엔진을 가진다. 사람은 두 명의 부모와 여러 명의 자식을 둘 수 있으며 배우자를 가지거나 가지지 않는다. 회사는 여러명의 사람을 고용할 수 있으며 한 명의 관리자를 가진다. 한 명의 관리자는 여러 회사의 관리자를 병행할 수 있다.

그림 1의 클래스 다이어그램에 적용가능한 OCL 예는 다음과 같다.

1. 자식의 수

```
context Person : self.child->size()
```

2. maxPower가 10이상인 차를 가진 사람 수

```
context Person inv :
Person.allInstances()->select(p:Person |
p.car.engine.maxPower <= 10)->size()
```

먼저 자식의 수는 Person 클래스를 컨텍스트로 하고 자신과 자식 관계에 있는 클래스의 수를 알

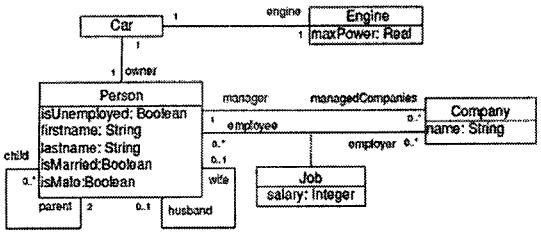

그림 1. 클래스 다이어그램의 예

려준다. 두 번째는 자동차 엔진의 파워가 10이상인 차를 가진 사람의 수이다. 이 경우는 모든 사람에 대해서 각 사람의 자동차의 엔진의 파워가 10 이상인 사람의 집합만 선택한 후 해당 집합의 수를 알려준다. 표는 OCL 문장의 형식을 보여준다.

3. 측정 프레임웍

본 장에서는 다양한 산출물과 메트릭 기술을 지원하기 위한 측정 프레임웍에 대해서 설명한다. 먼저 측정 프레임웍을 두 개의 개념으로 분리하고 다음으로 두 개념의 사용을 위해 실제 구현되는 프레임웍을 제시한다.

3.1 측정 프레임웍의 구성요소

본 논문에서는 소프트웨어 측정 프레임웍에서 다양한 산출물과 새로운 메트릭에 대한 확장성을 제공하기 위해서 크게 측정 프레임웍을 산출물 표현과 메트릭 기술이라는 두 가지 개념으로 나눈다. 그림 2는 측정 프레임웍이 산출물 표현과 메트릭 기술의 두 가지 요소로 구성되는 것을 보여준다.

그림에서 산출물 표현은 소프트웨어 측정의 대상이 되는 산출물을 나타내는 표현을 말한다. 즉, 다양한 산출물을 표현하기 위한 방법이다. 요구사항 단계의 산출물, 설계단계의 산출물등 다양한 종류의 산출물이 있으며 이러한 산출물을 표현하

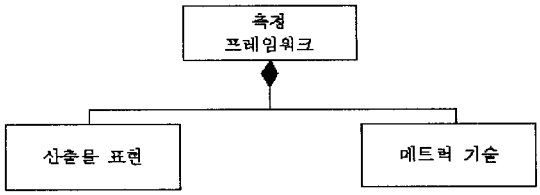

그림 2. 측정 프레임웍의 구성

기 위한 다양한 방법들이 있다. 하지만 소프트웨어를 측정하는 조직에 따라서 산출물 표현이 서로 상이한 문제가 있다. 산출물 표현을 위한 다양한 방법으로는 데이터베이스를 활용하는 방법, UML 다이어그램을 사용하는 방법, XMI 파일 형식을 사용하는 방법등이 있다.

본 논문에서는 UML을 사용하여 산출물을 표현하며 실제 측정에는 XMI 파일을 사용한다. UML의 Profile 기술을 사용하여 다양한 Profile을 정의하고 적용함으로써 다양한 산출물에 대한 지원이 가능하다.

소프트웨어 개발 방법이 발전하면서 새로운 패러다임들이 제시되었고 그 때마다 새로운 메트릭이 제시되었다. 예를 들어 객체지향 개념이 나오기 이전에는 라인의 수, 함수의 수, 함수의 라인수 등이 메트릭이 되었지만 객체지향 개념이 제시된 이후에는 클래스와 관련된 메트릭들이 새롭게 추가되었다. 이후 컴포넌트 개념이 제시되면서 컴포넌트와 관련된 메트릭 역시 새롭게 제시되었다. 즉, 앞으로 소프트웨어 개발에 새로운 개념이 제시되거나 추가적인 개념이 제시되면 그에 맞는 새로운 메트릭을 필요로 하게 된다. 이러한 메트릭을 명시적으로 기술하기 위해서는 메트릭 기술언어가 필요하며 이에 대해 기존에 다양한 연구가 진행되었다. 하지만 이러한 소프트웨어 측정에 사용되는 메트릭 기술언어에 대한 표준이 없으므로 조직마다 다른 메트릭 기술언어를 사용하였다. 기존에 연구된 메트릭 기술언어로는 SQL과 OCL

그리고 XQuery를 사용하는 방법이 있다.

본 논문에 메트릭 기술언어로 OCL을 사용하여 메트릭을 정의한다. OCL은 UML로 기술된 모든 산출물에 대해 기술할 수 있으므로 새로운 메트릭 정의에 적당하다.

3.2 측정 프레임웍의 개념적인 아키텍처

본 논문에서 제시하는 측정 프레임웍의 개념적인 아키텍처는 그림 3과 같다.

측정 프레임웍을 사용해서 소프트웨어 산출물을 측정하는 과정은 크게 세 단계로 나누어 볼 수 있다. 각 단계는 다음과 같다.

1) 메트릭 기술 / 입력 단계

메트릭 기술자가 메트릭 기술언어를 사용해서 메트릭을 기술하고 그 내용을 메트릭 기술 저장소에 저장하는 단계

2) 대상 산출물 입력 단계

측정 엔지니어가 대상 산출물의 정보를 측정 도구에 입력하고 그 내용을 대상 산출물 저장소에 저장하는 단계

3) 측정 단계

측정 엔지니어가 대상 산출물의 정보를 측정 도구에 입력하고 그 내용을 대상 산출물 저장소에 저장하는 단계

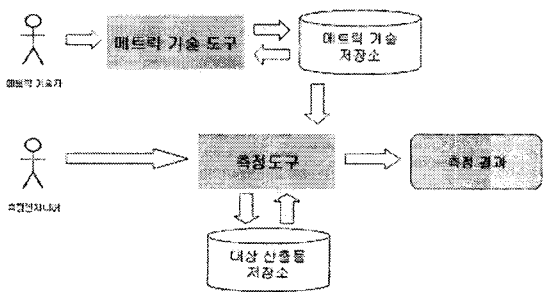

그림 3. 측정 프레임웍 아키텍처

즉, 메트릭 기술자가 메트릭을 기술하고 메트릭 기술 저장소에 저장한다. 측정 엔지니어는 산출물 정보를 대상 산출물 저장소에 저장하고 메트릭 기술 저장소의 메트릭 기술정보와 대상 산출물 정보를 이용하여 산출물을 측정한다.

3.3 프레임웍의 확장성

본 논문의 프레임웍에서는 산출물의 종류에 대한 확장성을 위해서 UML을 사용하고 새로운 메트릭 기술에 대한 확장성을 위해서 OCL을 사용한다. 산출물은 UML의 Profile과 다양한 종류의 다이어그램을 사용하여 다양한 종류의 산출물을 표현하는 것이 가능하다. 하지만 OCL이 새로운 메트릭에 대한 확장성 제공을 위해 충분한지에 대한 고려가 필요하다.

OCL의 문법에서 새로운 메트릭 기술에 대한 확장 여부에 대해서 알아볼 수 있다. OCL 문장은 Context부분과 Body부분으로 나누어 볼 수 있다. Context 부분은 Body 문장을 적용할 대상을 명시하는 부분이며 Body 부분은 실제로 해당 Context를 대상으로 OCL 기술자가 원하는 내용을 기술하는 부분이다. 그림 4는 OCL 문장에서 Context 부분과 Body 부분의 예를 보여준다.

그림 4의 예는 Person 이라는 Context에 대해서 Body 부분의 결과를 받아오는 OCL 문장이다. 여기서 OCL은 Context를 지정하는 것이 가능하므로 OCL의 대상이 되는 모델의 모든 요소를 Context로 지정할 수 있다. 그리고 Body 부분에서 해당 Context의 내부 요소와 해당 Context와

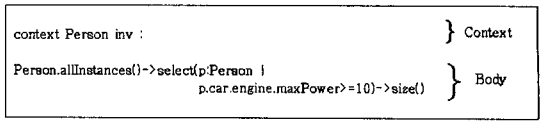

그림 4. OCL 문장 구조

연관이 있는 요소들에 대한 접근이 가능하므로 모델의 모든 곳에 대한 접근이 가능하다. Context를 설정하는 부분은 SQL에서 테이블을 선택하는 것과 매우 유사하다.

그리고 OCL이 모델에 대한 질의 언어로서 충분한 지에 대한 고려가 필요하다. 기존에 질의 언어의 기본적인 오퍼레이션에 대한 연구가 있었으며 Union, Difference, Select, Product, Project의 5가지의 기본 오퍼레이션이 정의되었다[8]. Union은 두 집합의 합집합 연산이며 Difference는 차집합 연산이다. Select는 한 집합에서 특정 조건을 만족하는 집합을 추출하는 연산이며 Product는 두 집합의 곱, Project는 한 집합에서 선택한 열의 정보만을 선택하여 돌려주는 연산이다. OCL 1.x 에서는 이 5가지 기본 오퍼레이션중 3가지만 만족하지만 2.0 부터 5가지 기본 오퍼레이션을 모두 만족한다[9]. 5가지 기본 오퍼레이션에 대응하는 OCL의 요소는 표 1과 같다.

질의 언어의 기본적인 오퍼레이션 5가지는 모두 직접/간접적으로 OCL에서 모두 표현이 가능하다. Product 연산을 제외한 4가지 연산은 바로 대응되는 OCL 연산을 가진다. Product 연산의 경우 두 집합을 파라메터로 받아서 필요한 요소를 선택하여 다시 집합으로 받아와야 하므로 def, let in, collect 연산을 이용하여 구현된다. 즉, 질의 언어의 기본 연산 5가지를 OCL이 모두 제공하므로 OCL은 메트릭 기술에 충분하다.

표 1. 질의 언어의 기본 오퍼레이션에 대응하는 OCL 요소

기본 Operation	OCL 요소
Union	union 연산
Difference	minus 연산
Select	select 연산
Product	def, let in, coolect 이용
Project	collect 연산

3.4 메트릭 기술도구의 사용성

다양한 메트릭 기술언어들이 연구되었고 그 중에서 UML 산출물을 대상으로 가장 손쉽게 메트릭을 표현할 수 있는 언어로서 OCL을 선택하였다. 하지만 OCL을 메트릭 기술언어로 사용하여 메트릭을 기술하는 일과 OCL로 기술된 메트릭을 이해하는 일 또한 복잡하고 어려운 일이다. 본 논문에서는 OCL로 메트릭을 기술하는 것과 기술된 메트릭의 이해를 돕기 위해서 OCL의 시각화인 VisualOCL을 사용한다.

OCL은 텍스트 기반의 언어이므로 표현하고자 하는 내용을 모두 텍스트로만 표현한다. 그리고 특정 요소에 대한 정보는 OCL 문장에서 표현되지 못하므로 OCL로 메트릭을 기술 할 때에 어려움이 있다. 그에 반해 VisualOCL은 OCL의 내용을 몇 가지의 주요 표기법을 이용하여 나타내며 OCL 문장이 나타내지 못하는 부분까지 시각화하여 보여주므로 좀 더 쉽게 이해할 수 있다. 표 2는 VisualOCL의 가장 기본적인 표기법을 보여준다.

VisualOCL은 표 2의 4가지 표기법을 기본적으로 사용하여 표현한다. 표 2의 4가지 표기법 중 두 번째 표기법은 OCL에서는 직접적으로 표기되지 않는 부분으로 해당 객체가 집합임을 나타낸다. 그림 5는 메트릭을 표현하는 OCL코드와 VisualOCL 다이어그램의 예를 보여준다.

표 2. VisualOCL의 표기법

표기법	설 명
객체:타입	타입 클래스의 객체 하나를 의미
:타입	타입 클래스의 객체 집합을 의미
———	Navigation 방향의 연관을 의미
[ ]	집합 연산에 필요한 정보를 표현

(a) 메트릭 설명 : 부모가 없는 클래스들에 대해서 추상 함수가 아닌 함수의 수

(b) OCL 표현

```
context Class inv :
let
    parentCount : Integer = self.generalization.general->size()
in
    self.ownedOperation->select( o:Operation | (parentCount=0) and
                                 (o.isAbstract = false) )->size()
```

(c) VisualOCL 표현

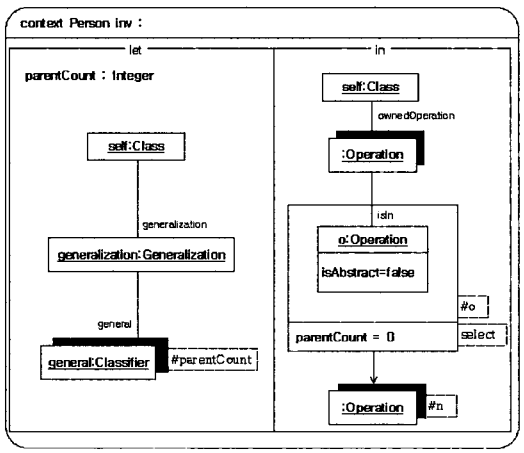

그림 5. 메트릭 표현

그림 5의 메트릭은 상속을 받지 않는 클래스들에 대해서 추상 함수가 아닌 함수의 수를 나타낸다. (a)는 자연어로 기술한 메트릭이며 (b)는 같은 의미의 OCL 문장, (c)는 역시 같은 의미를 가지는 VisualOCL 다이어그램이다. OCL 문장과 VisualOCL 다이어그램에서 가장 차이가 나타나는 것은 집합에 대한 표현과 각 요소들의 타입에 대한 것이다. (b)의 문장에는 generalization과 general의 타입이 Generalization과 Classifier인 것이 표현되지 않는다. 그리고 self의 ownedOperation이 집합이라는 사실도 명시적으로 표현되지 않는다. 즉, 같은 의미를 가지는 OCL 문장보다 VisualOCL 다이어그램이 좀더 명확한 정보를 가지며 시각화 되었기 때문에 좀 더 이해하기 쉽다.

4. 프레임웍의 구현

본 절에서는 3절에서 설명한 측정 프레임웍의 구현을 보여준다. 구현된 프레임웍의 아키텍처를 설명하고, 구현된 프레임웍이 가지는 확장성과 사용성 측면의 장점을 실제 예를 들어 설명한다.

4.1 플랫폼

3장에서 설명한 측정 프레임웍의 개념적인 아키텍처는 메트릭 기술도구와 측정 도구의 두 가지 도구를 포함한다. 메트릭 기술도구는 메트릭을 정의하고 기술하여 메트릭 기술 저장소에 저장하는 도구이며 측정 도구는 산출물 표현을 저장하고 저장된 메트릭 기술을 산출물 표현에 적용하고 결과를 산출하는 도구이다. 그림 6과 그림 7은 메트릭 기술 도구와 측정 도구의 플랫폼을 보여준다.

메트릭 기술도구와 측정도구는 모두 자바기반의 이클립스 플러그인으로 개발되었으며 메트릭 기술도구는 EMF(Eclipse Modeling Framework)와 GEF(Graphical Editor Framework)기반의 VisualOCL 플러그인을 확장하여 개발되었다. 측정 도구는 EMF와 OCL 파싱/평가 라이브러리 중의 하나인 Kent OCL Library를 사용하여 개발되었다. 그림 8은 메트릭 기술도구의 실행 화면을 보여준다. 그림 8은 메트릭 기술도구의 실행화면으로 VisualOCL 도구로 그린 다이어그램의 정보를 이용하여 메트릭 정보를 추가하는 화면을 보여준다.

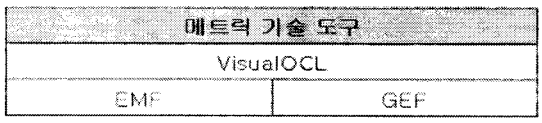

그림 6. 메트릭 기술 도구의 플랫폼

그림 7. 측정 도구의 플랫폼

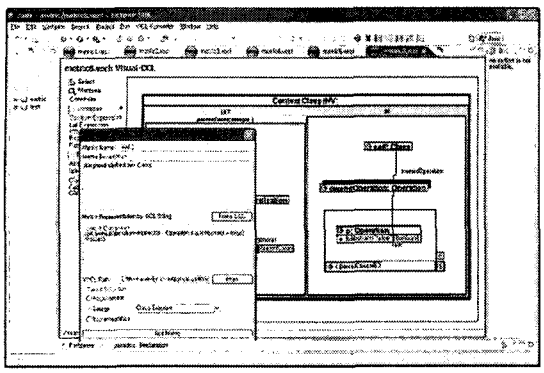

그림 8. 메트릭 기술 도구 실행화면

4.3 도구의 확장성

본 논문에서 제시하는 측정 프레임웍은 다양한 종류의 산출물에 대해서 메트릭을 적용할 수 있으며 새로운 메트릭에 대한 기술이 가능하다. 구현된 프레임웍은 다양한 종류의 산출물을 추가할 수 있으며 새롭게 메트릭을 추가하며 추가된 메트릭을 산출물에 적용할 수 있다.

확장성에 대한 설명으로 요구사항 단계의 산출물로 사용할 수 있는 유스케이스 다이어그램과 분석/설계 단계의 산출물로 사용할 수 있는 클래스 다이어그램을 예로 들고 각각의 종류에 맞는 메트릭을 기술하고 적용하는 과정을 보여준다. 그림 9는 그 중에서 먼저 요구사항 단계의 산출물로 사용할 수 있는 유스케이스 다이어그램의 예를 보여준다[3].

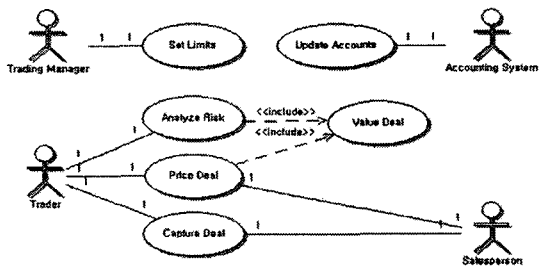

그림 9. 유스케이스 다이어그램의 예

그림 9의 유스케이스 다이어그램은 판매자와 구매자사이의 가격 협상을 위한 시스템의 요구사항 단계의 산출물이다. 이 유스케이스 다이어그램에 다음 표 3과 같은 메트릭을 적용할 수 있다.

표 3의 메트릭은 요구사항 단계의 유스케이스 다이어그램 산출물을 대상으로 적용 가능한 메트릭으로 모든 유스케이스의 수를 나타낸다. 표 4는 그림 9의 유스케이스 다이어그램에 표 3의 메트릭을 적용한 결과의 출력이다.

메트릭 적용결과는 적용 메트릭의 이름, 대상 컨텍스트의 이름이 먼저 출력되고 실제 컨텍스트의 객체마다 객체정보와 측정 결과를 차례로 출력한다.

이와 같이 요구사항 단계의 산출물과 분석/설계 단계의 산출물에 대해서 메트릭을 적용하는 것을 확인할 수 있다. 마지막으로 표 5는 현재 측정 도구에 구현된 메트릭을 보여준다. 현재는 10

표 3. 메트릭 기술의 예

항 목	내 용
메트릭 이름	NUM
메트릭 적용대상 이름	요구사항 단계의 유스케이스 다이어그램
메트릭 설명	Number of the UseCases in a Model, 모델 내부의 유스케이스의 수
메트릭 정의	context Model inv : self.packagedElement->select (p:PackageableElement | p.oclIsTypeOf(UseCase) = true)->size()
VisualOCL 파일 경로	c:/wmc_metric.vocl

표 4. NUM 메트릭 적용 결과

NUM
Model
uml.impl.ModelImpl(name:testUSE)
6

표 5. 측정 도구가 포함하는 메트릭

단계	다이어그램	메트릭 이름	메트릭 설명
요구사항	유스케이스	NAM	모델 내의 액터의 수 (Number of the Actors in a Model)
		NUM	모델 내의 유스케이스 수 (Number of the UseCases in a Model)
분석 / 설계	패키지	NPM	모델 내의 패키지 수 (Number of the Packages in a Model)
	클래스	NCM	모델 내의 클래스 수 (Number of the Classes in a Model)
		NASM	모델 내의 연관 관계 수 (Number of the ASsociations in a Model)
		NIM	모델 내의 상속 관계 수 (Number of the Inheritance relations in a Model)
		NATC	클래스 내부 속성(Attribute) 수 (Number of the Attributes in a Class)
		NOPC	클래스 내부 함수(Operation) 수 (Number of the OPeration in a Class)
		NASC	클래스에 연결된 연관관계 수 (Number of the ASsociations linke to a Class)
		DIT	클래스 상속의 깊이 (Depth of Inheritance Tree)

개의 메트릭이 저장되어 있으며 필요에 따라 새로운 메트릭을 기술하여 추가할 수 있다.

5. 관련 연구

본 절에서는 본 연구의 관련연구들에 대해서 살펴 본다. UML 모델을 편집하는 대표적인 도구 중에서 메트릭 적용이 가능한 도구들이 있다. IBM의 Rational Rose[10]와 Borland의 Together [11]가 대표적인 UML 모델 편집 도구이다.

Rose의 경우는 Rose가 생성하는 모델 형식에 대한 메트릭 적용이 가능한 지원 모듈이 연구되었다. Rose의 모델을 지원하는 모듈 중 Fast\ &Serious[12]는 Rose의 프로젝트에 대한 정보를 자동으로 추출하여 각 모델에 대한 연산을 수행한다. Rose 모델을 지원하는 다른 모듈인 UMP (UML Metrics Producer)[2]는 BasicScript 언어를 사용하여 개발되었다. UMP는 소프트웨어 생명 주기의 초기에 상품의 다양한 특징을 예측하기 위해서 사용하는 소프트웨어 메트릭의 집합을 계산한다. 하지만 Rose에서 사용 가능한 이 두 가지 모듈은 오직 Rose로 생성된 UML 다이어그램에만 사용이 가능하며 새로운 메트릭을 적용할 수 없다.

Together는 최신 버전에서 OCL을 이용하여 메트릭을 기술하고 기술된 메트릭을 모델에 적용하는 것을 지원한다. 하지만 Together는 OCL을 그대로 사용하였기 때문에 일반 사용자가 메트릭을 기술하거나 기술된 메트릭의 이해에 어려움이 있다.

Objecteering/Metrics는 Objecteering[13]으로 생성된 UML 모델에 대해 메트릭을 적용하는 UML 모델 편집도구의 모듈이다. 하지만 Rose의 모듈과 마찬가지로 Objecteering/Metrics는 사용자가 새로운 메트릭을 정의할 수 없다.

XMI의 파일을 사용하는 Metrics[14]는 UML 모델의 XMI 파일에 객체지향 메트릭을 적용하는 자바 어플리케이션이다. 그러나, Metrics는 특정 다이어그램에 대한 측정을 지원하지 않으며 사용자 정의 메트릭을 지원하지 않는다.

SDMetrics[15]는 액티비티 다이어그램과 유스케이스 다이어그램을 포함하는 UML 다이어그램의 측정을 지원하는 도구이다. 게다가 같은 모델에 대한 다른 버전사이의 차이에 대한 측정과 사용자 정의 메트릭 기술도 지원한다. 하지만 SDMetrics는 새로운 메트릭의 정의를 위해 자신들이 정의한 메트릭 기술 언어를 사용하므로 새로운 언어를 익혀야 하는 부담이 있으며 자신들이 정의한 메트릭 기술언어로 모든 메트릭의 표현 가능 여부에 대한 검증이 부족하다.

6. 결론 및 향후 연구

앞에서 객체지향 소프트웨어를 측정하기 위한 프레임웍을 제시하고 구현하였다. 본 논문에서 제시하는 소프트웨어 측정 프레임웍은 산출물 표현과 메트릭 기술로 구분할 수 있으며 다양한 산출물과 새로운 메트릭 기술에 대한 확장성을 제공하기 위해 산출물 표현은 UML을, 메트릭 기술은 OCL을 사용하였다. 구현된 측정 프레임웍의 장점은 다음과 같다.

- 대상 산출물 종류에 대한 확장성

소프트웨어 각 개발 단계의 산출물과 다른 종류의 다이어그램에 대한 측정을 지원한다.

- 새로운 메트릭의 추가에 대한 확장성

기존에 제공되는 메트릭 이외에도 모델의 컨텍스트를 지정하고 메트릭의 내용을 표현하는 OCL 문장을 작성함으로써 새로운 메트릭을 정의할 수 있다.

- 메트릭 기술언어의 사용성 증대

OCL로 메트릭을 기술하거나 OCL로 기술된 메트릭을 이해하려 할 때에 OCL의 시각화인 VisualOCL을 사용할 수 있다.

현재 구현된 측정 프레임웍은 몇 가지 개선사항이 존재한다. 이에 따른 향후연구로 UML 편집도구와 측정 도구의 통합, VisualOCL 도구 보완, 측정 도구가 지원하는 UML 다이어그램 추가, 프로젝트 단위의 측정을 수행할 예정이다.

참 고 문 헌

[1] H. Zuse, A Framework of Software Engineering Measurement, Walter de Gruyter, 1998.

[2] H. Kim, C. Boldyreff, "Developing Software Metrics Applicable to UML Models," In 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, 2002.

[3] M. Fowler, UML Distilled : A Brief Guide to the Standard Object Modeling Language 3rd Edition, Addison-Wesley Professional, 2003.

[4] P. Kruchten, The Rational Unified Process : An Intorduction 2nd Edition, Addison-Wesley Professtional, 2000.

[5] K. Beck, Extreme Programming Explained : Embrace Change 2nd Edition, Addison-Wesley Professional, 2004.

[6] OMG, UML 2.0 Superstructure Specification, http://www.uml.org.

[7] J. Warner and A. Kleppe, The Object Con-

straint Language : Getting Your Models Ready for MDA 2nd Edition, Addsion-Wesley Professional, 2003.
[8] E. F. Code, Relational Completeness of Database sub-language, In Database System, 1972.
[9] J. Chimiak-Opoka and C. Lenz, "Use of OCL in a Model Assessment Framework : An Exprience Report," In Proceedings 9th International Conference on Model-Driven Engineering Languages and Systems, pp. 53-67, 2006.
[10] IBM, IBM Rose,. http://www-128.ibm.com/developerworks/rational/prodcuts/rose.
[11] Borland, Together, http://www.borland.com/us/products/together/index.html.
[12] M. Carbone and G. Satucci, "Fast & Serious : A UML Based Metric for Effort Estimation," In 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, 2002.
[13] Objecteering, Objecteering/Metrics, http://www.objecteering.com
[14] T. Paternson, "Object-Oriented Software Design Metrics from XMI," In Heriot Watt Universit MSc Dissertation, 2002.
[15] SDMetrics, http://www.sdmetrics.com.

박 진 욱

• 2005년 부산대학교 컴퓨터공학과(공학사)
• 2005년~현재 부산대학교 컴퓨터공학과 석사 재학중
• 관심분야 : 소프트웨어 측정 자동화, 소프트웨어 아키텍처

채 흥 석

• 1994년 서울대학교 원자핵공학과(공학사)
• 1996년 KAIST 전산학 석사
• 2000년 KAIST 전산학 박사
• 2000년~2003년 동양시스템즈 선임연구원
• 2003년~2004년 KAIST 초빙 교수
• 2004년~현재 부산대학교 정보컴퓨터 공학과 교수
• 관심분야 : 객체지향 방법론, 소프트웨어 테스팅, 소프트웨어 메트릭, 소프트웨어 유지보수, 미들웨어 설계