# OCL을 이용한 상태 기계의 메타모델링

구자철, 권기현
경기대학교 정보과학부
{jachul, khkwon}@kyonggi.ac.kr

# State Machine Meta-Modeling Using OCL

Ja-Chul Koo, Gi-Hwon Kwon
Dept. of Computer Science, Kyonggi University

**요 약**

정형 언어 OCL은 UML의 한 부분으로써 클래스 다이어그램과 상태 기계의 조건을 표현하는데 많은 장점을 가지고 있다. 그렇지만, OCL을 이용하여 상태 기계를 메타모델링 하는데는 적지 않은 한계가 있다. UML/OCL은 모델의 정적인 구조를 강조하지만, 시스템이나 서브 시스템의 시간과 환경에 대한 반응을 고려는 미흡하다. 또한, 시스템의 변화에 있어 시그널은 오퍼레이션과 유사하지만 상태 기계에서만 다루어진다. 결국, 시스템의 변화에 대한 고려는 상태 기계를 통하여 보다 명확하게 명세 될 수 있으며, 현재 OCL의 확장으로써 이러한 문제를 해결 할 수 있다.

## 1. 서 론

객체지향 분석/설계를 위한 방법론은 많은 언어와 표기법이 지원 되고 있으며, 이중에서 UML은 산업 분야와 학술 분야에서 가장 널리 사용되는 모델링 언어이다. 또한, 다양한 분야에 적용이 되고 있으며, 좋은 품질의 소프트웨어를 생산하기 위한 모델 작성을 지원한다.

최근 UML의 한 부분으로써 객체 제약 언어(Object Constraint Language)를 제시 했다[1]. OCL은 특히 개별적인 객체의 측면을 표현할 수 있도록 아주 세부적인 규칙을 개발자로 하여금 생성하게 할 때 특히 유용하다. 오늘날 많은 소프트웨어 프로젝트가 비즈니스 모델을 위해 작성된 고유하고 복잡한 규칙이 필요로 함에 따라 OCL은 객체 개발의 중요한 부분으로 빠르게 자리잡아 가고 있다. 특히 클래스 다이어그램과 행위적 다이어그램(상태 기계)의 조건에 초점을 맞추고 있다. 그렇지만, OCL은 현재 상태 지향적 메타모델링 중 동적 행위를 표현하는데 있어 지원이 미약하다. 결국, 상태 지향적 UML 메타모델링의 지원은 부적합하다. 따라서, 상태지향적 메타모델링을 위해 OCL의 확장을 고려해야 한다. 본 연구에서는 활성 상태들의 집합인 형상[2]을 이용한 상태 기계의 형상 변화에 초점을 맞추어 OCL의 명세를 위한 연구이다.

2장에서는 OCL구조의 확장, 3장에서는 상태와 형상 그리고 형상의 변화 관하여 설명하고, 4장에서는 결론 및 향후 연구 방향에 대하여 설명한다.

## 2. OCL 확장

OMG에서 제공하는 OCL 1.4[1]에서는 여러 가지 사항이 문제로 제기 되고 있으며, 이에 대한 개선 연구가 있었다. 특히 메타 모델 구조에 있어 [3][4]의 연구는 현재 OCL과 크게 차이점을 보이지 않고 문제를 해결 하는 메타 모델 구조를 보이고 있다. 본 연구에서도 [3][4]의 OCL 메타 모델을 이용하여 상태 지향적 메타 모델의 표현을 강화 하였다. 최상위 구조의 루트 클래스는 OclType이고, OCL 타입의 OclType을 표현한다. 기존 UML 스펙의 문제점으로 나타났던 타입 일치 문제를 해결하기 위해 conform 관계를 통하여 subType과superType의 규칙 이름이 부여 되었다[3]. 그리고, OclType의 인스턴스 식별을 위해 속성 name을 이용한다. 메타 모델에서 주의할 사항은 OclType은 자신의 인스턴스도 아니고, OclAny의 서브 타입도 아니라는 것이다. subtype 관계는 oclType 클래스가 아닌, oclType의 인스턴스에 적용이 된다.

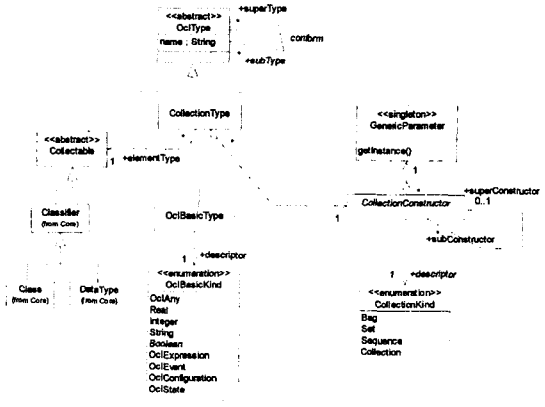



그림 1 OCL 메타 모델 확장

UML 열거형은 계층 구조의 부분이 될 수 없기 때문에, CollectionConstructor는 직접적으로 유한 집합 {Collection, Set, Bag, Sequence}를 표현 할 수 없다. 이러한 문제를 해결하기 위한 방법으로 열거형 메타 클래스 CollectionKind를 이용하여 분리 시킨다. 이와 같은 기법

을 적용한 것이 OclBasicType이다.

### 3. 상태와 형상

#### 3.1 상태

현재 OCL은 상태 기계의 상태를 표현하기 위하여 타입 OclState를 지원한다. 그렇지만, OclState는 단일 상태를 표현하는데 적합하지만, 상태 기계의 상태들을 커버하기에는 적절하지 못하다. 상태 기계에는 오직 하나의 상태만이 활성화 될 수 있는 Or 상태와 동시에 활성화 될 수 있는 And 상태, 그리고 더 이상 하위 상태를 갖지 못하는 Basic 상태로 구분이 된다. 이러한 핵심 상태의 표현에 대한 메타 표현을 위하여 OCL을 확장 해야 한다. 그 이외의 History, Synch, Final 등의 수도 상태들은 고려 하지 않고, 핵심 상태 표현에 초점을 맞춘다. 상태 기계의 메타 모델을 위한 OclState의 확장을 위하여 다음의 새로운 속성과 오퍼레이션을 이용한다.[1)]

```
DataType <<enumeration>> StateType {
        orstate,
        andstate,
        basic
}

OclBasicType OclState <super> OclAny {
        stateType      :       StateType;
        isRoot()       :       Boolean;
        parentState()  :       OclState;
        subState()     :       Set(OclState);
        isActive()     :       Boolean;
        notActive()    :       Boolean;
        anySubState()  :       OclState;
}
```

상태와 관련된 상태 기계의 메타 모델을 정형적으로 나타내고, 오퍼레이션의 의미에 대하여 설명한다. 상태의 전체 집합을 $\Sigma$라 한다면 아래와 같다.

$$\Sigma := \Sigma_{orsate} \cup \Sigma_{andstate} \cup \Sigma_{basic}$$

상태 기계는 특성상 상태 구조를 펼쳐 놓으면 트리 구조와 같다. 이러한 특성을 이용하여 먼저 상태간의 계층구조를 표현하는 관계를 정의한다. 상태 계층구조를 나타내는 함수 $\Upsilon$는 다음과 같다.

$$\Upsilon : \Sigma \rightarrow 2^{\Sigma}$$

1) 본 상태 기계의 정형적 의미는 [2]의 연구를 기반으로 작성되었다.

트리 구조로 표현된 상태도는 다음과 같은 특성을 만족해야 한다. 단 $\Sigma_{root}$는 최상위 상태라 가정한다.

(1) 최상위 상태는 오직 하나이고, 그 타입은 orstate이다.

$\mathbf{dom(\Upsilon) \setminus \bigcup ran(\Upsilon) = \Sigma_{root} \wedge \Sigma_{root}.stateType = orstate}$

(2) 각 상태의 계층에서 사이클이 없어야 한다.

$\forall s: \bigcup ran(\Upsilon)\ (\exists s': OclState\ (\forall s'': OclState\ s' \not\in \Upsilon(s'')))$

(3)부모 상태는 오직 하나이어야 한다.

$\forall s: \bigcup ran(\Upsilon)\ (\exists !anc: dom(\Upsilon)\ s' \in \Upsilon(anc))$

열거형 속성 stateType은 하나의 상태가 orstate, andstate, basic 중 오직 하나의 타입이어야 한다. 오퍼레이션 isRoot는 오직 하나의 상태만이 참이고, 나머지 하위 상태들은 거짓값을 갖는다. 부모 상태를 위한 parentState는 오직 하나의 부모만 갖을 수 있다. subState는 트리 구조의 상태도에서 Set 타입의 하위 상태들의 집합을 갖는다. 현재 상태가 활성화 되었다면 isActive는 참이되고, 그렇지 않다면 거짓이다.

#### 3.2 형상

현재 OCL에서 상태에 관한 정보를 제공하기 위하여 불린 오퍼레이션 oclInState()을 제공한다. 이는 복잡한 상태도의 의미를 나타내는데 불충분하다. 따라서, 좀더 확장된 의미의 오퍼레이션을 제공해야 한다. 특히, 상태 기계에서 활성 상태의 집합은 더욱 중요시 된다. 이러한 활성 상태의 집합을 형상이라 한다.

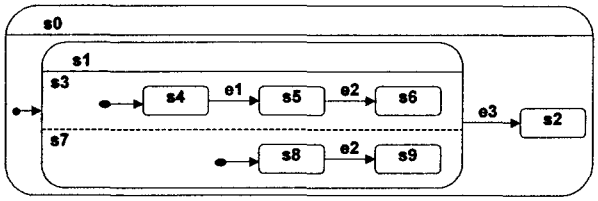



그림 2 상태 기계의 예

그림 2에서 s0가 최상위 상태이고, 초기 형상은 s1::s3::s4와 s1::s7::s8이다. 즉, Set{s1::s3::s4}은 초기 형상에서 불완전하나 s0.oclInState(s1::s3::s4)은 참이된다. 단지 oclInState만을 지원하기 때문에 현재는 좀더 명확한 형상의 의미를 나타내기 위해서 다음과 같이 표현 할 수 있다.

```
s0.oclInState(s1::s3::s4) and
s0.oclInState(s1::s7::s8)
```

복잡한 상태도의 경우 이러한 기술은 매우 복잡하고, 표현하는데 불편하다. 따라서 형상의 효과적 표현을 위하여, 추가된 기본 타입 OclConfiguration과 오퍼레이션

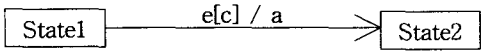
config():Set(OclConfiguration)을 이용한다[4]. 기존의 오퍼레이션을 사용하지 않고, 새로운 오퍼레이션을 이용하여 초기 형상을 표현하면 다음과 같다.

```
context s0 inv:
self.config->includes(c:OclConfiguration |
                c=Set{s1::s3::s4,s1::s7::s8}
```

위의 명세는 Set{s1::s3::s4,s1::s7::s8}이 s0에 대하여 정당한 형상인지 검사를 하게 된다.

### 3.3 형상의 변화

UML과 OCL은 모델의 정적인 구조를 강조하지만 시스템이나 서브시스템의 시간과 환경에 대한 반응을 고려해야 한다. 시스템의 변화를 유발하는 시그널은 오퍼레이션과 유사하지만 UML 모델중 유일하게 상태 기계에서만 다루어 진다. 따라서, OCL에서 시그널에 대한 고려를 해야 한다. 이러한 입력신호를 이벤트라 한다. 상태들은 이벤트의 영향을 받아 전이가 발생된다. 일반적인 전이 조건은 다음과 같다.




화살표는 전이를 나타내며, e는 이벤트, c는 가드 조건, a는 액션을 나타낸다. 만약 e가 발생하고 c가 참이라면 a의 액션을 발생하며 전이된다. 세가지 e, c, a는 모두 선택적으로 사용이 가능하다.

상태에서 상태로의 변화는 형상의 변화를 유발한다. 따라서, 이벤트에의한 형상의 변화를 명세가 필요하다. 이벤트에 관련된 새로운 타입 OclEvent을 이용한다.

```
OclBasicType OclEvent <super> OclAny {
      isActive()    :      Boolean;
      notActive()   :      Boolean;
}
```

이벤트는 그 발생 여부에 관심이 있다. 또한, 한 단계 동안만 유효하다. 이벤트 저장과 관련된 이벤트 큐를 사용할 수 있지만, 본 연구에서는 이벤트 큐에 관한 고려는 하지 않고, 이벤트가 발생되면 한 단계 동안만 유효하다고 가정한다.

그림 2에서 이벤트 초기 형상에서 e1의 발생으로 Set{s1::s3::s4,s1::s7::s8}에서 Set{s1::s3::s5,s1::s7::s8}으로 변화된다. 이벤트의 발생은 명시적인 상태의 변화를 발생하지만, 암시적으로 상태의 집합이 변화된다. 이러한 형상의 변화는 상태의 변화보다 많은 정보를 포함하고 있으며, 시스템 변화에 중요하다. 이러한 내용을 OCL로 나타내면 아래와 같다.

```
context e1 inv:
s0.config=Set{s1::s3::s4,s1::s7::s8}
implies
s0.config=Set{s1::s3::s5,s1::s7::s8}
```

전이 발생시 가드 조건 c가 포함된다면 다음과 같은 내용이 추가된다.

```
inv:    s0.config=Set{s1::s3::s4,s1::s7::s8}
and c implies
s0.config=Set{s1::s3::s4,s1::s7::s8}
```

아직 이벤트의 종류에 따르는 명세는 고려 되지 않았다. 외부 이벤트/내부 이벤트, 동기적 이벤트/비동기적 이벤트 등의 구별에 따르는 보완적 명세 방법이 필요하다.

## 4. 결론 및 향후 연구 방향

본 연구는 UML의 메타 모델링중 OCL을 이용하여 상태 기계를 표현하는데 초점을 두었다. 그 중에서도 시스템의 변화에 따르는 명세를 지원하기 위해, OCL을 이용하여 형상을 표현하였다. 또한 상태 기계의 구문과 의미를 이용하여 기존 OCL의 구문과 유사한 구조를 나타내었다. UML의 모델과 메타 모델을 이용한 분석/설계는 높은 품질의 소프트웨어 생산과 기존의 시각적 언어의 단점을 보완할 수 있다.

앞으로의 연구 과제는 보다 상태 기계의 의미에 충실한 OCL의 확장을 꼽을수 있을 것이다. 상태의 변화를 정의하는 단계 의미에 관한 연구는 성공적으로 진행이 되고 있으며, 정형적 기술을 통한 구문과 의미를 잘 정의하고 있다. 따라서, 이러한 결과를 OCL에 접목 시킨다면 강력한 메타 모델 언어가 될 것이다. 또한, 아직까지 지원이 미흡한 OCL 도구의 개발이 목적이라 할 수 있다. 현재 상업용/교육용 OCL 도구의 개발은 초기 단계이다. 따라서, 사용자의 편의성과 보다 잘 정의된 구문과 의미를 표현 할 수 있는 OCL 도구는 UML의 다이어그램 도구 개발이상으로 그 의미가 있다고 할 수 있다. 이러한 연구는 좋은 도전이 될 수 있을 것이다.

## 참고 문헌

[1] OMG. UML Unified Modeling Language Specification, Version 1.4, URL: http://www.omg.org
[2] D. Harel, A. Naamad, "The STATEMATE semantics of Statecharts," ACM Transactions on Software Engineering and Methodology, Vol.5, No.4, pp.293-333, 1996
[3] Thomas Baar, Reiner Hahnle, "An Integrated Metamodel for OCL Types", Proc. OOPSLA 2000, Workshop Refactoring the UML: In Search of the Core, Minneapolis, Minnesota, USA, 2000.
[4] S. Flake, W. Mueller, "An OCL Extension for Real-Time Constraints", In T. Clark and J. Warmer (eds.) 'Advances in Object Modelling with the OCL'. Springer Verlag, December 2001