모델링 언어의 표준으로 자리잡는
Unified Modeling Language (UML)

인하대학교 전자계산공학과

심우호*, 유원희**

문명이 발전됨에 따라 소프트웨어 산업이 보다 전문화되는 것과 동시에 복잡성이 증가하게 되었다. 우리는 소프트웨어 위기(Software Crisis)를 통해 소프트웨어 개발의 어려움을 알게 되었고, 이는 곧 소프트웨어 개발에 있어 프로그래밍 위주의 개발이 아닌 충분한 분석과 설계의 중요성을 알았다. 모델링 언어는 바로 프로그래밍을 위한 언어가 아닌 분석과 설계를 위한 언어인 셈이다.

모델링 언어의 일종인 UML은 일반 프로그램 언어와는 다른 목적을 위해 설계된 언어이다. UML은 소프트웨어 중심의 시스템을 명세화, 구성, 가시화, 그리고 문서화하는 모델링 언어이다. 좀더 확장한다면, 비즈니스 모델과 같은 소프트웨어 시스템이 아닌 것에도 사용될 수 있는 언어이다. 이는 UML이 마치 큰 빌딩을 세우는데 있어 설계도면이 그려진 청사진에 비유될 수 있다. UML은 객체지향적 분석 및 설계의 대가인 Booch, Rumbaugh[OMT], Jacobson[OOSE]에 의해 기본이 완성되고, 이후 계속 HP, IBM, Microsoft, Oracle의 참여로 현재 UML1.1(97년 11월)이 나왔다.

모델링 언어의 일종인 UML을 소개하기 앞서 모델링 언어의 이해를 돕기 위해 모델링 언어의 중요한 요소에 대해 설명하겠다.

- ▶ 모델 요소(Model element) — 기본적인 모델링 개념과 의미
- ▶ 표기법(Notation) — 모델 요소에 대한 가시화된 표현
- ▶ 지침서(Guideline) — 모델링에 대한 용어 및 형식에 대한 지침

UML은 일반 프로그램 언어와 같이 표현력 있는 문법과 풍부한 의미를 갖고 있다. UML의 구조적 특징은 계층화된 구조와 패키지에 의해 논리적으로 조직되어 있다는 점이다. 이에 대해서는 아래에서 자세히 다루겠다.

1. UML의 개념

UML은 영문 의미로 통합된 모델링 언어(Unified Modeling Language)이다. 여기서 '통합'되었다는 의미는 다음의 모델링 개념들을 모두 포함하고 이들을 적절히 사용했다는 것이다.

- ▶ 데이터 모델링 개념(Data Modeling concepts)

---

* 학생회원

**정회원

▶ 비즈니스 모델링(Business Modeling)

▶ 객체 모델링(Object Modeling)

▶ 컴포넌트 모델링(Component Modeling)

데이터 모델링은 데이터베이스 구축을 위해 데이터와 데이터간의 상호관계를 도식한 모델로써, 일반적으로 개체 관계 다이어그램(entity relationship diagrams)으로 표현된다. 비즈니스 모델링은 조직 경영의 관점에서 일처리 흐름(work flow)을 도식화하는 것으로, 효율적인 업무처리를 도모하기 위함이다. 객체 모델링은 객체지향적 분석 및 설계에서 요구되어진 실세계의 객체를 구현하고자 하는 객체로 다시 모델링하는 것이고, 컴포넌트 모델링은 소프트웨어의 실행 관점에서 레고 블릭과 같은 실행 모듈들의 모델링이다.

2. UML의 구조

UML의 구조는 4-계층의 메타모델 구조로 이루어졌으며, UML의 메타모델은 객체 모델 표현의 완전한 의미를 정의한다. 메타모델 계층은 상대적으로 복잡하여, 논리적 패키지로 분리되었는데, 이는 UML의 모듈성을 증가시키고 표현의 유연성을 향상시키다.

UML의 4-계층 메타모델 구조는 복잡한 실세계 모델의 요구에 대하여 정확한 의미를 정의하는 기본적인 구조를 갖고 있다. 이러한 4-계층 메타모델 구조의 접근 방법에는 다음과 같은 이점이 있다. 첫째, 연속적인 메타계층에 반복적으로 적용하여 핵심 구조를 파악할 수 있다. 둘째, 앞으로의 UML 메타모델의 확장을 위한 기초적인 구조를 제공한다. 셋째, 4-계층 메타모델링 구조에 기초한 다른 표준과 제휴할 수 있는 기초적인 구조를 제공한다.

일반적으로 받아들여지는 메타모델링에 대한 개념적인 프레임웍은 다음의 4-계층에 기초하는데, 이는 OMG(Object Management Group)의 MOF(Meta Object Facility)를 기초로 구성되었다.

계층	설명	예제
meta-metamodel	메타모델링 구조의 기초로써, 메타모델을 명세화하는 언어를 정의 한다.	MetaClass, MetaAttributes, MetaOperation
metamodel	메타-메타모델의 인스턴스로써, 모델을 명세화하는 언어를 정의 한다.	Class, Attribute, Operation, Component
model	메타모델의 인스턴스로써, 객체의 영역을 기술하기 위한 언어를 정의한다.	StockShare, askPrice, sellLimitOrder, StockQuoteServer
user object (user data)	모델의 인스턴스로써, 특정 객체의 영역을 정의한다.	<Acm_Software_Share_98789>, 654.56, sell_limit_order, <Stock_Quote_Svr_32123>

[표 1] 4-계층 메타모델링 구조

메타-메타모델은 메타모델보다 높은 추상화 단계로 모델을 정의하고 보다 간단하게 기술된다. 메타-메타모델은 여러 개의 메타모델을 정의함과 동시에 각각의 메타모델에 여러

개의 메타-메타모델이 있을 수 있다. 명시적으로 메타모델에 메타-메타모델이 지정이 되어 있지 않다면, 묵시적으로 모든 메타모델에 메타-메타모델이 연합되었다.

UML의 또 다른 구조적인 특징은 바로 패키지이다. 패키지 안에는 추상화된 문법, WFR, 의미와 같은 모델링 요소가 있다. UML 메타모델은 대략 90개의 메타클래스(metaclasses)와 100개 이상의 메타관계(metaassocaiations) 그리고, 50여 개의 stereotype으로 구성되어 있는데 이러한 메타모델의 복잡성은 패키지로 관리되어진다. 또하 이러한 패키지 그룹은 다른 패키지에 있는 메타클래스와의 결합력(cohesion)을 보여준다.

3. UML의 정형성

UML의 문법은 문자와 함께 그래픽으로 이루어져 있는데 이는 추상화된 문법(abstract syntax)에서 표기법을 얻을 수 있다. UML에서 추상화된 문법은 메타클래스의 구성 및 서로간의 관계를 보여주는 다이어그램으로 표현된다.

언어에 대한 정적 의미(static semantics)는 언어의 요소(또는 토큰)이 서로 조합되어 어떠한 의미를 가지는가를 정의하는데, 이러한 언어의 정적 의미는 WFR(Well Formedness Rules)에 의한 구성에 의해서만 기술된다. 동적 의미(dynamic semantics)는 다시 정적 의미를 통해 정의 될 수 있다. UML에서 WFR은 메타모델에서 정의된 속성(attribute)과 연합(association)에 대한 제한을 규정한다.

UML의 추상화된 문법과 의미를 기술하기 위해서는 여러 언어 - OCL(Object Constraint Language), 자연어(Precise natural language) - 를 조합해서 사용한다. UML에서 OCL은 WFR을 표현하는데 이용된다. OCL에서 가독성을 향상시키기 위해 다음과 같은 표기 규칙이 있다. 'self'는 문맥상 변하지 않는 같은 메타클레스의 참조에서 생략될 수 있고, 배열과 같은 집합 객체에서 각 요소에 대한 반복적인 의미의 표기는 생략될 수 있다.

그리고 UML에서 의미를 정의하기 위해 자연어를 이용하는데, 예를 들면 영어에서 "X provides the ability to ..." and "X is a Y." 등이 사용된다.

4. UML의 표준화

객체지향적 소프트웨어 개발에 있어 문제점 중에 하나로 지적되었던 것이 바로 정해진 표준 방법론이 없었다는 것이다. 그러나 1994년에 Grady Booch와 Jim Rumbaugh의 Rational Software에서 기존의 다양한 방법론을 통합하기 시작 하여, 1995년 Ivar Jacobson의 OOSE 방법론을 합쳐 현재의 UML이 나왔다. UML 1.1은 1997년 9월 객체지향 모델링 그룹의 모임인 OMG에게 받아들여지고, 지금까지 많은 회사와 단체에서 UML을 객체지향적 모델링 언어의 표준으로 받아들이고 있다.

나름대로 UML을 쉽게 이해시키기 위해 썼던 글이지만, 다소 난해한 내용이 있으리라 본다. 또한 UML을 좀더 이해하려면 객체지향적 소프트웨어 공학에 대하여 이해한 뒤, UML을 이용하여 실제로 분석 및 설계를 해야 될 것이다. 기회가 된다면 실제 UML을 이용하여 소프트웨어의 분석과 설계과정에 관한 글을 쓰고자 한다.

참고 문헌

[1] _, "UML Sumary Version 1.1", Rational Software Corparation, 1997
http://www.rational.com/uml/html/summary/

[2] _, "UML Semantics Version 1.1", Rational Software Corparation, 1997
http://www.rational.com/uml/html/semantics/