□신기술해설□

객체지향 소프트웨어 개발 방법론의 표준화 : UML(Unified Modeling Language)

강 문 설† 김 태 희††

◈ 목 차 ◈

1. 서 론

엔터프라이즈 솔루션에서 가장 중요한 위치를 차지하는 분야가 바로 소프트웨어 공학, 즉 소프트웨어 개발 방법론분야이다. 소프트웨어 개발 방법론은 프로젝트 수행을 어떻게 효율적으로 수행하여 고품질의 제품을 생산할 것인가 하는 관점에서 정보시스템을 도입하는 기업 뿐만 아니라 솔루션을 제공하는 업체들에게도 매우 중요한 관심사이다. 우리나라에서는 개발 방법론에 입각한 프로젝트 수행이 제대로 적용되지 못하다가 90년 이후부터 대기업과 일부 벤처 기업을 중심으로 수용되기 시작하였다. 그러나 개발 방법론은 소프트웨어 공학의 원리, 프로젝트 개발 환경 보장, 프로젝트 관리자 및 개발자들의 상당한 경험이 뒷받침되어야만 분석, 설계, 구현, 시험 및 유지보수 등의 소프트웨어 개발 생명주기 전 과정에서 정상적으로 적용될 수 있다. 그렇지 않으면 단순한 문서를 위한 방편으로 활용되는데 그칠 가능성이 높다.

1990년대 초반 이후 객체지향 방법론이 학계 및 업계에 계속 확대 연구되어 오더니 1996년을 시점으로 업계에 본격적으로 도입되고 있는 상황이다. 상대적으로 기존 구조적 방법론이나 정보공학 방법론은 매우 빠른 속도로 빛을 잃어가고 있다. 이처럼 객체지향 개발 방법론이 확산되고 있는 것은 프로그램 언어, 개발 환경, 소프트웨어(시스템) 구조 등의 인프라가 객체지향을 강력히 뒷받침 해줄 수 있도록 형성되었기 때문이다. 최종적인 소프트웨어 산출물이 객체지향 구조를 가졌다면, 그것을 개발하는 과정과 방법 역시 객체지향 개념을 적용하는 것이 가장 정확하고 빠른 방법이 아닐까 한다.

그러나 객체지향 소프트웨어 공학이 과연 얼마나 현실적으로 프로젝트의 수행 속도와 품질을 향상시켜 줄 수 있을 것인가, 얼마나 쉽고도 실용성 있게 업무에 적용될 수 있을 것인가 하는 물음에 아직까지 이렇다 할 지침서나 교육 시설, 참고 자료 조차 빈약한 현실이다. 이러한 환경에서 객체지향 개발 방법론을 적용하는 것은 시기적으로 빠르지 않을까?

결론부터 말한다면, 지금은 시기적으로 결코 이른 것이 아니며, 객체지향 소프트웨어 공학을 기반으로 프로젝트를 수행하는 조직이 먼저 엔터프라이즈 솔루션의 경쟁력을 갖게 될 것이라는 것이다. 특히, 1995년부터 기존의 다양한 객체지

† 종신회원 : 광주대학교 컴퓨터학과 교수
†† 정 회 원 : 동신대학교 컴퓨터학과 교수

향 개발 방법론에 대한 이론들이 몇가지 방향으로 통일되어, UML(Unified Modeling Language)이 1997년 11월 OMG(Object Management Group) 표준화위원회의 심사를 거쳐 객체지향 소프트웨어 개발 방법론의 표준으로 발표되었기 때문에 빠른 속도로 객체지향 개발 방법론이 정착하게 될 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 UML의 역사, UML을 이용한 소프트웨어 개발 과정, UML에서 사용하는 기호와 다이어그램 및 다이어그램을 작성하는 과정을 살펴보고, 3장에서는 UML, OMT 및 Booch 방법론을 비교 설명한다. 4장에서는 결론을 기술하였다.

2. UML(Unified Modeling Language)

UML은 Rumbaugh의 OMT 방법론, Booch의 Booch 방법론, 그리고 Jacobson의 OOSE 방법론 등을 통합하여 만든 모델링 개념의 공통 집합으로 객체지향 분석 및 설계 방법론의 표준 지정을 목표로 제안되었다. 1997년 11월 UML version 1.1이 OMG에서 표준방법론으로 결정되었으며, 벌써부터 많은 CASE 도구들이 UML을 지원하게 되었고 CASE 도구의 평가 항목에도 UML 지원 여부가 중요한 평가 항목으로 자리 잡고 있다.

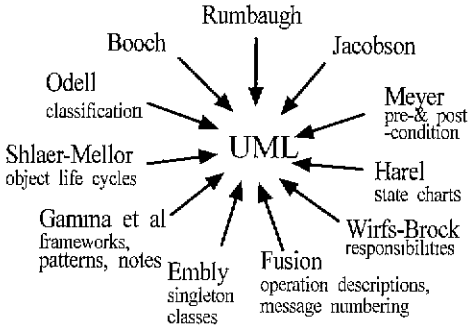
2.1 UML의 역사

객체지향 기술의 발전은 그 동안 객체지향 프로그래밍 언어를 축으로 하는 사람들과 방법론을 축으로 하는 사람들의 양대 산맥에 의해 이루어져 왔다. 최초의 객체지향 기술의 발달이 프로그래밍 언어를 중심으로 이루어졌다면, 최근에 이르러 객체지향 기술의 발달은 주로 분석과 설계 방법론 분야에서 이루어지고 있다. 다수의 사람들이 이 분야 발전에 많은 공헌을 했을 뿐만 아니라 과거 전통적인 방법론으로 명성을 떨쳤던 상당수의 사람들, 예를 들면 구조적 방법론으로 명성을 떨쳤던 Edward Yourdon이나 정보공학 방법론으로 널리 알려진 James Martin 등도 이제는 객체지향 기술의 발전에 일익을 담당하고 있다.

현재까지 개발된 객체지향 방법론들 중에서 OMT, Booch 및 OOSE 방법론이 현재 가장 인기있으며, 널리 사용되고 있는 객체지향 소프트웨어 개발 방법론이다. 각 방법론은 각각의 장단점을 가지고 있다. 즉, OMT는 분석 단계가 강력하지만, 설계 단계는 분석 단계에 비하여 미흡한 특징을 가지고 있고, Booch '91은 설계 단계만 지원되고, 분석 단계는 거의 지원되지 않는다. 그리고 OOSE는 행위 분석을 강력하게 지원하고 있지만, 다른 단계는 거의 지원하지 못하는 특징을 가지고 있다. Booch는 Rumbaugh와 Jacobson 등 다른 사람들이 주장한 방법론들로부터 좋은 분석 기법을 채택하여 Booch '91을 Booch '93으로 개선시켰으며, Rumbaugh는 Booch의 훌륭한 설계 기법을 채택하여 OMT-1을 OMT-2로 개선하였다. 방법론들이 많은 발전을 추구하고 하나로 집중하기 시작하였으나 아직 자신들의 유일한 표기법을 사용하고 있었다. 서로 다른 표기법의 사용은 하나의 기호가 다른 사람들에게 서로 다른 의미로 인식되기 때문에 현장에서 많은 혼란을 초래하였다. 예를 들어, 검정색으로 채워진 동그라미(●)는 OMT 방법론에서는 다양성(multiplicity)을 나타내지만, Booch 방법론에서는 집단화를 나타낸다.

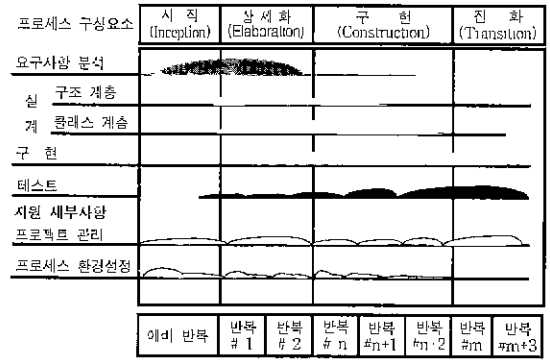
서로 다른 표기법으로부터 나타나는 혼란을 제거하기 위하여 통합된 모델링 언어(UML)의 적용에 관심을 가지게 되었다. UML은 객체지향 시스템에서 개발되는 부품들의 명세, 시각화 및 문서화시키기 위하여 사용되는 하나의 모델링 언어이다. UML은 Booch, OMT 및 OOSE의 표기법 뿐만아니라 (그림 1)에 소개된 다른 방

법론들로부터 가장 좋은 아이디어를 통일시켜 표현하였다. 이들 객체지향 방법들을 사용하여 표기법을 통일시키므로써 UML은 객체지향 분석과 설계 영역에서 표준을 위한 기초를 제공하였다.



(그림 1) UML에서 참조한 방법론들

UML은 의미 모델(semantic models), 구문 표기(syntactic notaions) 및 다이어그램(diagrams) 등의 분석과 설계 부품들의 표준화에 많은 노력을 투자하였다. 그 결과로서 1995년 10월에 UML 0.8이 공식 발표되었고, 발표된 UML 0.8에 대한 피드백과 Jacobson의 OOSE를 통합하여 1996년 7월에 버전 0.9, 1996년 10월에 버전 0.91이 발표되었다. 그리고 UML 1.0을 1997년 7월에 객체지향 소프트웨어 개발 방법론의 표준 방법론으로 채택하기 위하여 OMG(Object Management Group)에 제출하였고, 1997년 9월에는 UML 1.1로 개선하여 OMG에 제출하였다. 1997년 11월에 OMG에서는 UML 1.1을 객체지향 소프트웨어 개발을 위한 표준 방법론으로 결정하여 발표하였다.

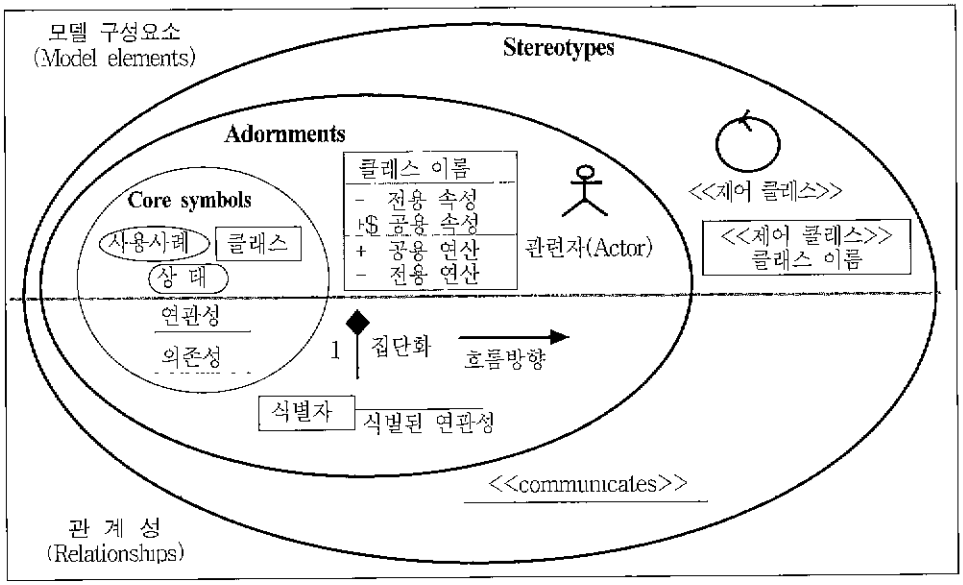


(그림 2) UML의 점진·반복적 개발 과정

2.2 UML를 이용한 개발 과정

개발 방법론은 다이어그램 작성 도구와 개발 과정으로 이루어져 있다. 그림만 그린다고 설계가 되는 것이 아니고, 방법과 순서에 맞추어 도구를 사용해야 하는 것이다. 이 순서는 매우 유기적으로 연결되어 있어서 앞 단계가 있어야 뒷 단계를 계속할 수 있는 경우가 대부분이다. 그렇지 않은 경우에는 산출물의 품질이 매우 떨어지게 마련이다. UML에서 개발 과정에 대한 명확한 정의가 비록 정립되지 않았다고 하더라도 소프트웨어 시스템 개발에 필수적인 것이다. UML 1.0에서 발표한 개발 과정은 (그림 2)와 같다.

이 개발 과정에서 "시작(inception)" 또는 "예비 반복(preliminary iteration)"으로 표현된 부분은 개괄적인 모델을 구축하는 단계이고, 나머지 단계들은 점진·반복적 개발에 해당한다. 상세화(elaboration) 단계에서 수행되는 반복 작업들, 즉 반복 #1, 반복 #2, … 들에서는 기술적으로 볼 때 가장 실패 위험이 큰 사용 사례(use case)들이 선정 된다. 이들에 대한 조기 개발을 시도함으로써 시스템 개발에 수반되는 위험 요소들을 조기에 검증할 수 있다. UML의 개발 과정에서는 개괄 모델의 구축과 점진적 개발이라는 분류와 함께 또 다른 관점에서 전체 공정을 "시작 → 상세화 → 구현 → 진화"라는 4 단계로 분류하고 있다.



(그림 3) UML에서 사용되는 표기 기호(UML notational symbols)

UML의 개발 과정은 소프트웨어가 프로젝트의 마지막 단계에서 하나의 최종 산출물로 개발되는 것이 아니라 여러 개의 부분 산출물로 개발되는 점진·반복적 개발 과정이다. 구현 단계는 많은 반복으로 이루어지며, 각각의 반복에서는 프로젝트의 부분 요구사항들을 만족시키기 위하여 시험되고 통합된 생산성과 품질 중심의 소프트웨어를 구축한다. 또한 각각의 반복은 분석, 설계, 구현 및 시험 등의 일반적인 생명주기의 모든 단계를 포함한다.

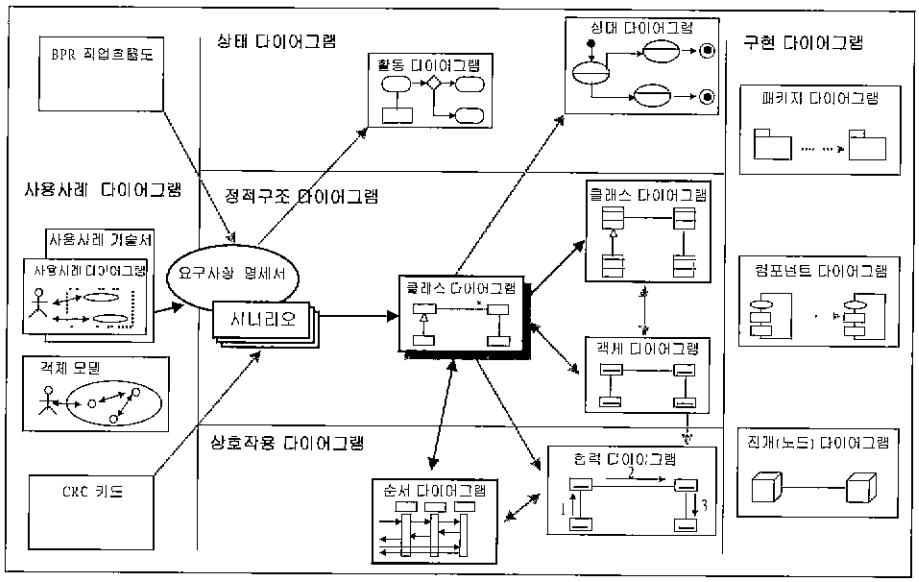
2.3 UML의 기호

UML 다이어그램들을 고찰하기 이전에 다이어그램들에서 사용되는 기호들(symbols)의 특성을 살펴보는 것이 다이어그램들을 이해하는데 도움이 될 것이다. UML 기호들은 핵심 기호(core symbols; first class symbols), 핵심 기호들을 확장시킨 장식 기호(adornment symbols), 매우 특수화된 기호인 스테레오타입(stereotypes) 등의 세 가지 기본 유형으로 구성된다. 그리고 핵심 기호들은 두 개의 집단, 즉 모델링 기호와 관계성 기호들로 분류한다.

UML 기호들 사이의 관계는 (그림 3)과 같이 나타낼 수 있다. 클래스(class), 상태(state) 및 사용사례(use case)는 핵심 모델 요소들이고, 연관성(association)과 의존성(dependency)은 핵심 관계성을 나타낸다. 클래스의 속성들과 연산들에 관련된 정보를 클래스에 추가하여 클래스를 완성한다. 특별한 유형의 클래스를 사용하려면 OOSE의 제어 클래스와 개체 클래스와 같은 스테레오타입을 이용한다. 비슷한 방법으로 직선(—)은 연관성을 나타내며, 직선에 다이아몬드(◆)가 연결된 연관성은 집단화(aggregation)를 나타낸다. 직선에 표시된 숫자는 관계성에 포함되는 인스탄스들의 개수를 의미한다.

2.4 UML의 다이어그램

UML에서 사용하고 있는 다이어그램들의 5가지 다른 범주는 동일한 문제를 기술하는 5가지의 다른 방법을 제공한다. 각 범주에서 다이어그램을

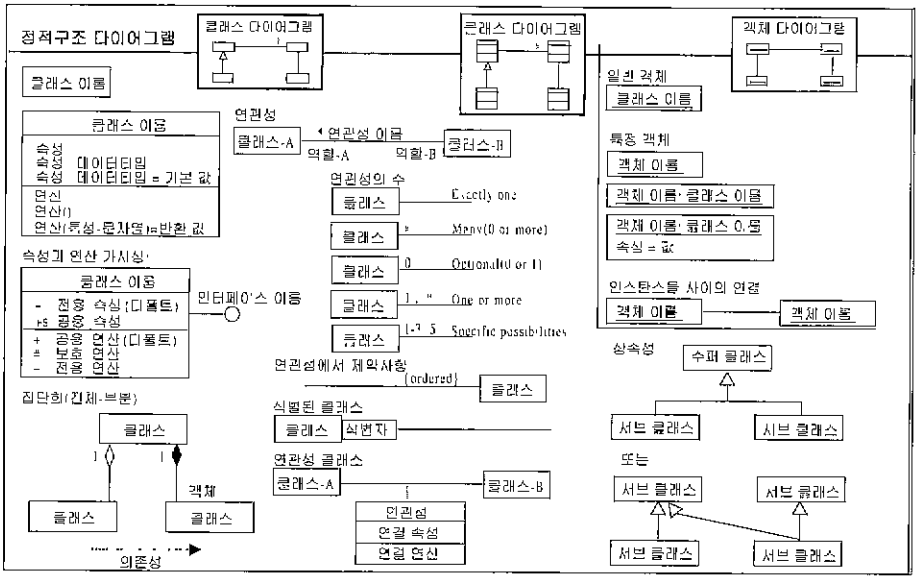


(그림 4) UML 다이어그램들 간의 관계

작성함으로써 하나의 문제를 다른 관점에서 분석할 수 있다. UML 다이어그램들은 사용 사례 다이어그램, 정적 구조 다이어그램, 상호작용 다이어그램, 상태 다이어그램 및 구현 다이어그램으로 분류할 수 있으며, 이 UML 다이어그램들 간의 관계는 (그림 4)와 같다. UML을 이용한 분석과 설계의 완전한 문서화 집합은 개발된 모든 UML 다이어그램들로 구성된다.

UML에서 사용하는 여러 가지 다이어그램들 사이의 관계를 나타내는 한 가지 방법을 (그림 4)와 같이 표현하였다. 다이어그램 작성은 문제의 구조를 식별하고 시나리오를 작성하기 위해 BPR, 사용 사례 또는 CRC 카드를 이용함으로써 시작된다고 가정한다. 또한, 최종 목표 시스템이 무엇을 수행하는 가를 나타내기 위해 요구사항 기술서를 개발할 것이라고 가정한다.

일단 요구사항 기술서가 작성되면, (그림 4)는 UML 다이어그램들이 어플리케이션을 분석, 설계 및 구현하기 위해 사용된다는 것을 나타낸다. 진한 화살표는 주요 개발 경로를 나타낸다. 대부분의 개발자들은 최소한 하나의 사용 사례 다이어그램과 여러 개의 시나리오를 생성하며, 또한, 여러 개의 순서 다이어그램과 클래스 다이어그램을 생성한다. 순서 다이어그램과 클래스 다이어그램은 분석과 설계 작업의 핵심 부분으로 어플리케이션을 보다 정확하게 이해하기 위해 더욱 상세하게 작성해야 한다. 다른 다이어그램들은 빈번하게 사용되지 않는다. 각각의 다이어그램들은 특별한 기능을 가지고 있으며, 개발하는 어플리케이션의 유형에 따라 사용되는 경향이 있다. 다시말하면, 어떤 특정 다이어그램을 사용하기 위한 순서가 정해져 있는 것은 아니다. 큰 규모의 시스템 개발 작업에서는 종종 백여 개의 다이어그램들을 생성하고, 여러 가지의 모든 다이어그램 유형을 사용한다. 작은 시스템은 두 개 또는 세 개의 클래스 다이어그램, 두 개 또는 세 개의 순서 다이어그램, 그리고 다른 다이어그램들로 해결할 수 있다. 물론 대부분의 어플리케이션 개발 작업은 이들 다이어그램을 이용하여 해결할 수 있다.



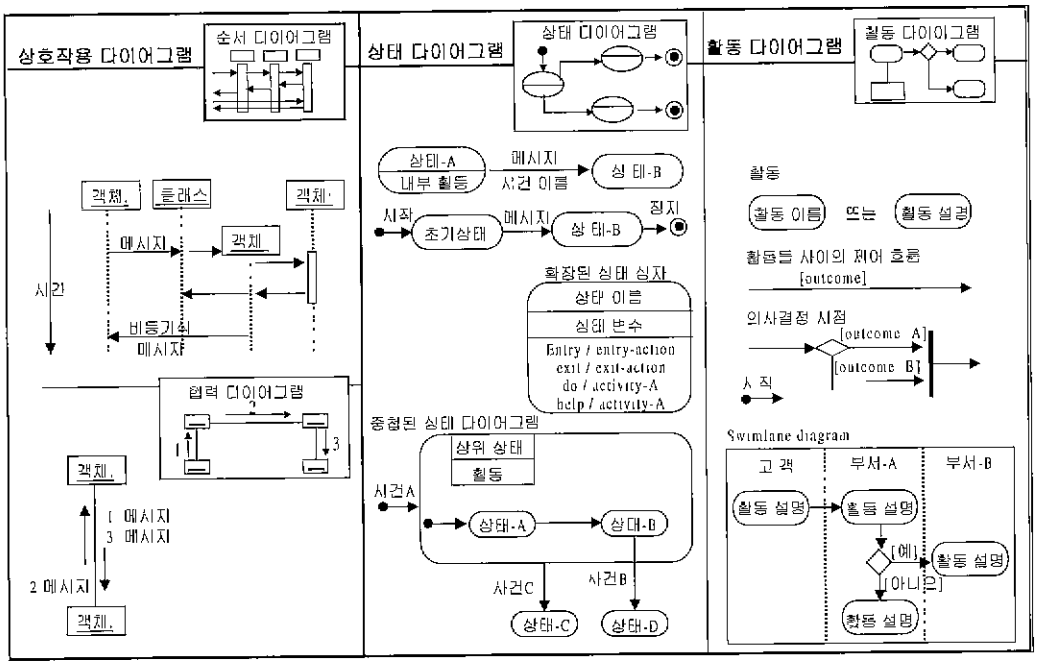
(그림 5) 정적 구조도 다이어그램의 작성

2.5 UML 다이어그램의 작성 과정

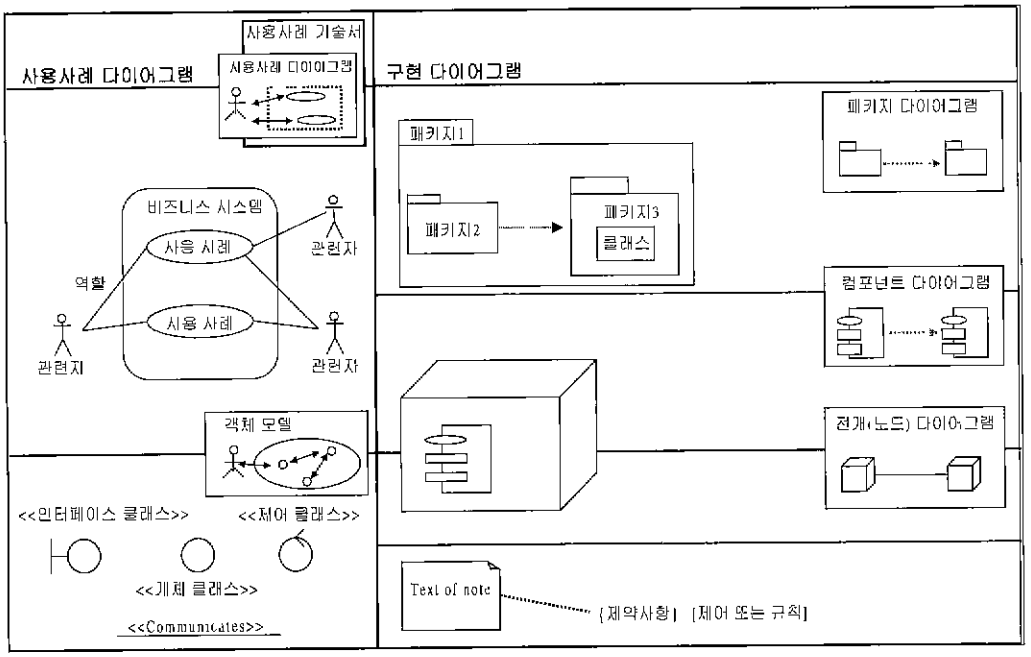
정적 구조 다이어그램(static structure diagrams)은 UML에서 사용되는 2가지의 핵심 다이어그램인 클래스 다이어그램과 객체 다이어그램으로 구성된다. 클래스 다이어그램은 시스템에 포함된 객체들의 유형(클래스)들과 그 클래스들 사이에 존재하는 다양한 종류의 정적 관계성을 표현한다. 또한, 클래스 다이어그램은 한 클래스의 속성과 연산을 나타내고 객체들을 연결하는 방법에 적용하는 제약 사항을 표현한다. 정적 구조 다이어그램의 작성 과정을 (그림 5)와 같이 도식화하였다.

상호작용 다이어그램에서 수직선은 객체들의 생명선을 표현하고, 수직 박스는 객체들의 활동 상태를 포함한다. 화살표는 한 인스탄스에서 다른 인스탄스로 메시지의 전송을 표현하고, 하나의 사건은 하나의 메시지가 도착할 때 일어난다. 협력 다이어그램은 객체 다이어그램과 순서 다이어그램을 결합시킨 것이다. 화살표와 숫자는 메시지가 하나의 시나리오 순서에 따라 한 객체에서 다른 객체로 전송되는 순서를 나타낸다. 상태 다이어그램에서 모든 상태는 하나의 단일 클래스를 표현하고, 메시지를 처리함에 따라 값이 변경되는 방법을 나타낸다. 메시지는 한 상태에서 다른 상태로 변경하기 위해 하나의 객체에서 발생하고, 사건은 객체가 실질적으로 변경될 때 발생한다. 활동 다이어그램은 활동들이 사건 혹은 의사결정에 의해 연결되는 방법을 나타내고, 프로세스들 혹은 의사결정 패턴들로 표현한다. 그리고 활동 다이어그램은 사용 사례에서 일어난 것이 무엇인가를 정의하거나 한 클래스내에서 어떤 의사결정이 일어나는가를 보여주기 위해 작업 흐름 분석에 사용한다. (그림 6)은 상호작용 다이어그램, 협력 다이어그램, 상태 다이어그램 및 활동 다이어그램의 작성 과정을 도식화한 것이다.

(그림 7)은 사용 사례 다이어그램과 구현 다이어그램의 작성 과정을 도식화한 것이다. 사용 사례는 시스템이 조작해야 하는 일반적인 처리 과정들을 정의하고, 사용 사례 기술서는 일반적인 시나리오들을 정의한다. 사용 사례 다이어그램에서 왼쪽의 관련자(actor)는 시스템이 오른쪽의 관련



(그림 6) 상호작용, 상태 및 활동 다이어그램의 작성



(그림 7) 사용 사례 및 구현 다이어그램의 작성

자들에 의해 표현되는 동안에 관련되는 사람들을 나타낸다. 패키지 다이어그램은 어플리케이션에서 클래스의 논리적인 모듈화를 나타낸다. 패키지는 클래스, 인터페이스 또는 다른 패키지를 포함하며, 어떤 깊이(depth)에 따라 배열할 수 있다. 점선 화살표는 패키지들 사이의 종속성을 표현한다. 컴포넌트 다이어그램은 실제 코드를 모듈화시켜 작성하는 방법을 나타낸다. 컴포넌트는 클래스, 인터페이스 혹은 절차적 코드의 모듈(기존 어플리케이션)을 포함하며, 컴포넌트가 물리적 플랫홈에서 무엇을 하는가를 나타내기 위해 전개 다이어그램과 결합된다. 전개 다이어그램은 어플리케이션에 사용되는 하드웨어와 네트워크를 나타낸다. 플랫홈 또는 노드는 컴포넌트나 클래스들을 포함한다.

3. UML, Booch 및 OMT 방법론의 비교

현재 객체지향 방법론은 Rumbaugh의 OMT 방법론이 주도를 하고 있으며, Booch의 Booch 방법론, Jacobson의 OOSE 등 40여개의 다른 방법론들도 나름대로 영역을 가지고 활용되고 있으나 표준 방법론으로 등장한 UML과 객체지향 방법론의 지속적인 발전을 통하여 계속적인 시장 변화가 예상된다. 현재 객체지향 소프트웨어 개발 방법론을 이용하는 소프트웨어 개발자들의 약 60% 이상이 OMT 또는 Booch 방법론의 표기법을 사용하고, 나머지 소프트웨어 개발자들은 CRC 카드를 사용한 책임기반 설계(Responsibility Driven Design), 사용 사례(use cases)와 이상적인 객체 다이어그램(ideal object diagrams)을 이용한 Jacobson의

〈표 1〉 UML, Booch 및 OMT의 비교

표기 항목		UML	OMT	Booch
클래스 다이어그램·객체 다이어그램	클 래 스	Solid rectangle	solid rectangle	dashed cloud
	클 래 스 사각형 표시	Name Aattributes Operations	Name Attributes Operations	Name All members
	속 성	Name:type=inital value	Name, type, inital value	Name, type, inital value
	연 산	Name(argument list): result type	Full signature	Full signature
	가 시 성	+ public # protected - private null unspecified	+ public # protected - private	null public I protected II private III implementation
	매개변수화된 클 래 스	Class rectangle with a small dashed rectangle overlying the right coner with the list of template arguments in attribute format	Solid rectangle with object name: class	Solid cloud
	객 체	Solid rectangle (*with object or class name* underlined)	Solid rectangle (with object name: class)	Solid cloud
	객체 속성	Name = value	Name = value	Name = value
	인스탄스화	Dashed line from object to class	Dashed arrow from object to class	Dashed arrow from object to class
	상 속 성	Line from subclass to superclass with open triangle pointing to superclass	Tree from triangle fanning out to subclass	Sloid arrow from subclass to superclass

Objectory 방법론 등의 다양한 표기법들을 사용한다.

OMT나 Booch 방법론의 표기법에 친숙한 소프트웨어 개발자들이 객체지향 소프트웨어 개발을 위한 표준 방법론인 UML에서 제공하는 유사한 표기법을 쉽게 사용할 수 있도록 <표 1>과 같이 UML, OMT 및 Booch 방법론의 표기법을 서로 비교하여 나타내었다. <표 1>에는 각 표기법들에 대한 자세한 사항들은 나타내지 않았지만, 대부분의 어플리케이션에서 적용할 수 있는 중요한 특징들을 중심으로 비교하였다.

4. 결 론

최근의 컴퓨팅 환경이 메인프레임 환경에서 클라이언트-서버 환경, 인트라넷 환경으로 변화되고, 비즈니스 업무 분야에서도 조직의 구조가 기능중심의 계층적 구조에서 절차와 결과 중심의 팀 단위 구조로 변화가 신속하게 일어나고 있기 때문에 소프트웨어 개발 과정에서 보다 유연한 개발 방법론이 필요하게 되었다. 즉, 클라이언트-서버 방법론, 패키지 중심의 방법론, 객체지향 방법론 등이 기존의 구조적 방법론 및 정보공학 방법론과 함께 사용되고 있다.

이들 방법론들은 재사용 구조를 기반으로 한 방법론으로 계속 발전하고 있으며, 앞으로는 특정 방법론의 사용보다는 프로젝트의 성격과 규모에 적합한 방법론의 선정을 통한 개발이 중요하게 부각될 전망이다. 따라서, 다양한 형태의 방법론에 대한 이해가 필요하며, 적절한 컨설팅이나 아웃소싱을 활용한 방법론의 적용이 필요하고, 소프트웨어 부품에 대한 단순한 재사용에서 탈피하여 지식 경영등을 통한 기업 차원에서 효율적인 재사용 전략이 절실히 요구된다.

참고문헌

[1] G. Booch, J. Rumbaugh, and I. Jacobson, Unified Modeling Language User Giude, Addison-Wesley, Inc., 1997.

[2] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language for Object-Oriented Development, Documentation Set, Version 1.0, Rational Software(www.rational.com), 1996.

[3] J. Rumbaugh, I. Jacobson, and G. Booch, Unified Modeling Language Reference Manual, Addison-Wesley, Inc., 1997.

[4] L. Martin, Succeeding with the Booch and OMT Methods : A Practical Approach, Addison-Wesley, Inc., 1996.

[5] M. Fowler, UML Distilled : Applying the Standard Object Modeling Language, Addison-Wesley, Inc., 1997.

[6] P. Harmon and M. Watson, Understangding UML : The Developer's Guide with a Web-based Application in Java, Morgan Kaufmann Publishers, Inc., 1998.

[7] R.C. Lee and W.M. Tepfenhart, UML and C++: A Practical Guide to Object-Oriented Development, Prentice-Hall, Inc., 1997.

[8] T. Quatrani, Visual Modeling with Rational Rose and UML, Addison-Wesley, Inc., 1998.

[9] 서봉원, 소프트웨어 공학과 객체지향 개발방법론, 열린마당(KCC정보통신 사보) 통권 18호, pp.26~31. March 1997.

강 문 설

1986년 전남대학교 계산통계학과 (이학사)
1989년 전남대학교 대학원 전산통계학과(이학석사)
1994년 전남대학교 대학원 전산통계학과(이학박사)

1983년-1985년 12월 제1군수지원단 전산실
1989년-1994년 8월 전남대학교 전산학과 조교 및 강사
1994년-현재 광주대학교 컴퓨터학과 조교수
관심분야 : 소프트웨공학(재사용, 역공학, 재공학), 객체지향시스템, 컴포넌트기반 소프트웨어 개발

김 태 희

1991년 동신대학교 전자계산학과 (공학사)
1993년 전남대학교 대학원 전산통계학과(이학석사)
1996년 전남대학교 대학원 전산통계학과(박사수료)

1993년-1997년 동신대학교 컴퓨터학과 시간강사
1997년-현재 동신대학교 컴퓨터학과 전임강사
관심분야 : 소프트웨어공학, 객체지향시스템