2005 한국경영과학회/대한산업공학회 춘계공동학술대회
2005 5 월 13 일~14 일, 충북대학교

UML 활용 사례연구
A Case Study on UML Utilization

김영기
충북대학교 국제경영정보시스템학부, 충북 청주시 개신동

초 록

본 연구는 국내 대형 SI 업체의 UML(Unified Modeling Language)의 도입과 활용 사례를 소개한다. 1990년대에 소프트웨어의 개발 패러다임이 객체지향 방법론(object-oriented paradigm)으로 옮겨갔고, UML은 객체지향 분석과 설계를 위한 모델링 언어로 전 세계적인 표준으로 받아들여지고 있다. 한 소프트웨어업체가 UML을 활용한다는 사실은 그 업체가 소프트웨어 모델링을 하고 있다는 것을 의미한다. 소프트웨어 모델의 기본 목적은 복잡한 소프트웨어를 추상적 수준에서 이해할 수 있게 표현하는 것이다. 소프트웨어 모델은 사용자 요구의 정확한 반영, 개발팀원간의 업무분담, 개발비의 추정, 유지보수에 유용하게 쓰일 수 있다. 본 연구는 인터뷰와 설문지를 병행하여 진행되었고, 연구의 결과는 국내 소프트웨어업계에서의 UML 활용수준의 벤치마크로 쓰일 수 있으며, 새로이 UML을 활용코자 하는 소프트웨어업체의 UML 도입전략 수립에 도움을 줄 것으로 기대한다.

1. 서 론

UML을 사용함으로써 소프트웨어의 품질을 향상시키고, 생산성을 증가 시킬 수 있다는 일반적 기대가 있다. 하지만 과연 현실적으로 현장에서 UML이 어느 정도로 활용되고 있는지에 대한 연구는 미흡하였다. 그 간의 연구는 개발 방법론, 개발 툴 및 개발 프로세스의 도입과 활용에 관한 내용이 주류를 이루고 있다 [3], [7]. UML 활용에 관한 연구로는 Brian Dobing이 OMG(Object Management Group)과 연계하여 수행하고 있는 설문조사 가 있다. 이는 UML의 기술적 활용 면에 초점을 맞추고 있는 것으로, 응답자가 자발적으로 참여하는 설문조사이다 [10].

본 연구는 국내 소프트웨어 업체에서 UML이 어떻게 도입되고 얼마나 활발히 활용되고 있는지를 가름해보고자 하는 점에서 출발하였다. 또한 개발자들이 UML의 유용성을 어느 정도 인식하고 있는가를 알아보는 것도 연구의 동기가 되었다. 이를 살펴보기 위해 본 연구에서는 국내의 대표적 대형 SI 업체(앞으로 “A사”로 칭함)의 UML 도입 과정과 활용 수준에 관련된 자료를 수집하였다. A사는 많은 수의 개발인력을 보유하고 있는 국내의 대표적 SI업체이기 때문에, A사에서의 UML 활용수준은 국내 소프트웨어업계에서 참고할 수 있는 벤치마크의 역할을 할 수 있을 것으로 보인다. 연구는 인터뷰와 설문지를 병행하였다. 인터뷰는 A사의 중견관리자들과 이루어졌으며, 주로 A사의 기술혁신추진 방향과 UML의 도입배경이 주된 내용이었다. 설문은 소프트웨어 개발경험이 있는 A사 직원 200명을 대상으로 실시되었으며, 주된 내용은 UML의 사용빈도, 소프트웨어 모델링의 중요성에 대한 인식, UML의 유용성에 대한 인식, UML의 도입동기, UML의 확산방안, UML활용의 장애요소 등이다.

본 연구의 결과는 국내 소프트웨어업계에서의 UML 활용수준의 벤치마크로 쓰일 수 있으며, 새로이 UML을 활용코자 하는 소프트웨어업체의 UML 도입전략 수립에 도움을 줄 것으로 기대한다. A사가 국내의 대표적 SI업체임을 고려해 볼 때, A사의 UML 활용수준은 국내 타 소프트웨어 개발업체가 참고할 수 있는 UML 활용수준의 기준이 될 수 있을 것이다. 또한 A사의 사례는 UML을 새로이 도입하려 활용하려는 소프트웨어업체들이 도입전략을 수립하는데 도움을 줄 수 있으며, 시행착오를 줄일 수 있을 것으로

기대한다.

2. 소프트웨어 모델링의 중요성

UML이 모델링을 위한 언어(language)인 점을 고려해볼 때, 한 소프트웨어 업체에서 UML을 사용한다는 사실이 갖는 중요한 의미는 그 업체에서 소프트웨어 개발을 위해 소프트웨어 모델링, 즉 소프트웨어 모델을 구축하고 있다는 점이다. 소프트웨어 모델링 언어가 UML뿐만 있는 것은 아니며 UML이 가장 우수한 표기법(notational system)이냐 하는 것에는 논란의 여지가 있을 수 있으나, UML 활용이 소프트웨어 모델링을 위한 것임은 명확하다. 따라서 UML을 적극적으로 활용하는 조직은 소프트웨어 모델링의 중요성을 인식하고 있는 조직이라고 할 수 있겠다.

모델은 실 세계(real world)의 객체, 현상의 간략한 표시(simplified representation) 또는 닮은꼴(analogy)이라고 정의된다 [11]. 일반적으로 모델의 목적은 세 가지, 정보 표현(information presentation), 계산(computation), 탐구와 예측(investigation and prediction) 으로 나눈다 [9]. 이중에서 소프트웨어 모델의 주된 목적은 정보 표현이라고 할 수 있겠다. 소프트웨어 모델은 구축할 소프트웨어를 추상적인 수준에서 표현하여줌으로써, 보는 이로 하여금 복잡한 소프트웨어를 이해할 수 있도록 도와주기 때문이다. 또한 소프트웨어 모델은 탐구와 예측에도 사용될 수 있다. 소프트웨어 모델에서 소스코드를 자동 또는 수동으로 생성하여 만든 프로토타입을 테스트함으로써 구축될 소프트웨어의 행태(behavior)를 분석하고 예측해 볼 수 있기 때문이다. 또한 개발 단계나 유지보수 단계에서 코드를 수정하면 이러한 변화가 소프트웨어 모델에 반영되는 리버스 엔지니어링(reverse engineering)이 가능하고, 개발 툴에서 구현되고 있다. 예를 들면, UML CASE(Computer Aided Software Engineering) 툴인 투게더(Together)는 코드를 변화시키면 클래스 다이어그램을 갱신하고, 갱신된 클래스 다이어그램들간의 일관성(consistency)의 일부를 체크한다 [1].

소프트웨어 모델은 두 가지 목적으로 사용할 수 있다. 첫째는 이미 존재하는 소프트웨어 시스템을 보여주기 위하여 모델을 사용할 수 있으며, 다른 하나는 앞으로 구축할 소프트웨어 시스템의 설계를 보여주는 설계 사양(specification)으로서 모델을 사용할 수 있다 [8]. 분석 설계 단계의 산출물인 소프트웨어 모델은 다음과 같이 활용될 수 있다.

소프트웨어 모델을 사용하여 복잡한 소프트웨어를 보다 쉽게 이해할 수 있다. 구축할 소프트웨어의 규모가 클수록 코딩이전에 보다 추상적인 수준의 형상화된 모델이 필요하다. 복잡한 시스템을 이해하기 쉬운 작은 단위로 분할하고, 작은 단위간의 개념적 연계와 유기적 상호작용을 정확히, 이해하기 쉽게 표현해주는 모델이 필요하다 [6]. 구축할 소프트웨어가 복잡하고 클수록 개발 프로젝트에 많은 인원이 투입되고, 각 개발자는 작은 부분을 맡아 개발하되, 타 개발자가 개발한 부분들과 통합해야 한다. 소프트웨어 모델을 사용하여 개발팀원간의 업무분담을 할 수 있다. 이런 경우에 모든 개발자가 설계 사양으로서의 소프트웨어 모델에 맞추어 개발을 해야 하며, 소프트웨어 모델을 정확히 이해해야 한다. 개발팀원간에 소프트웨어 모델에 대한 이해가 서로 달라서는 안되기 때문에 공통의 모델링 언어를 사용하는 것이 바람직하다.

반복적 개발 프로세스(iterative development process)를 수행하거나, 또는 개발 후 유지보수나 확장을 해야 하는 경우에, 요구 사양(requirement specification) 과 소프트웨어 모델이 확보되어 있어야 변경할 부분을 쉽게 찾아내서 고치고, 또한 변경된 결과를 다시 요구 사양과 소프트웨어 모델에 정확히 반영할 수 있다. 따라서 소프트웨어 모델은 개발뿐만 아니라 유지보수 및 확장에도 매우 필요하다. 하지만 현실적으로는 소프트웨어 모델은 일단 설계 사양으로서의 역할을 한 후에는 무시되고 버려지는 경향이 없지 않았다 [4]. 왜냐하면 코드의 변경이

모델에 제대로 반영되지 않거나 반영되더라도 소프트웨어 모델이 필요 이상으로 복잡해지곤 했기 때문이다. 하지만 코드 레벨에서의 유지보수는 오류가 축적되고 시간이 흐를수록 투입되는 비용이 증가하기 때문에 소프트웨어 모델을 사용하는 것은 불가피하다고 하겠다.

소프트웨어 구축에 필요한 공수(man-month), 인력, 예산을 체계적으로 추정하려면 구축할 소프트웨어의 모델이 필요하다. 구축할 소프트웨어의 큰 설계도라 할 수 있는 아키텍처 설계에 포함된 구성 요소 별로 견적을 해야 하며, 과거에 수행한 프로젝트들에서 축적해 온 유사 요소들에 대한 공수 및 원가 데이터를 활용함으로써 견적(estimation)의 정확도를 향상시킬 수 있다.

사용자의 요구가 설계에, 또한 구축된 소프트웨어에 정확히 반영되었는지를 확인하면서 프로젝트를 진행해야 하는데, 이를 위해서는 요구 사양에서 설계 사양으로, 설계 사양에서 소스코드로의 변환을 추적할 수 있어야 한다. 이를 위해서 설계 사양으로서의 소프트웨어 모델이 필요한 것이다.

마지막으로 중요하게 언급해야 할 점은 모델링 표기법은 하나로 통일하여 사용하여야 한다는 점이다. Brooks가 지적하였듯이 소프트웨어 개발의 공수는 의사소통의 어려움에 따라 급격히 증가한다 [2]. 이것이 시사하는 바는 모델링 표기법을 여러 가지 동시에 사용하면, 개발팀원간의 의사소통에 상당한 어려움이 있어 투입되는 공수가 급격히 증가하리라는 것은 짐작하기 어렵지 않다. UML이 de facto 표준이란 점과 더불어 고려해 볼 때, 조직 내에서 모델링 표기법을 UML로 통일하여 사용하면 의사소통이 원활해져서 투입되는 공수를 감소시킬 수 있고 이는 결국 생산성을 향상시키게 될 것이다.

3. UML의 등장배경

1990년대에 걸쳐 소프트웨어 방법론이 구조적 방법론(structured paradigm) 에서 객체지향 방법론으로 바뀌었다. 구조적 방법론은 1970년 전후에 등장한 소프트웨어 개발 방법론으로, 그 주요 내용은 종전의 goto 명령문을 많이 쓰는 프로그래밍의 방식에서 벗어나 프로그램의 제어 구조(control structure)로 순차(sequence), 선택(selection), 반복(iteration) 만을 사용하는 프로그래밍을 지향하는 것이다. 많은 기대에도 불구하고, 구조적 방법론의 문제점 들, 예를 들면 대형 소프트웨어 개발에 적합하지 못한 점, 높은 유지 보수 비용의 문제를 해결하지 못한 점 등이 인식되었다. 이러한 문제점은 구조적 방법론이 연산과 데이터를 분리하여 소프트웨어를 개발하는 방식에 있는 것으로 인식되면서, 연산과 데이터를 함께 갖는 객체를 중심으로 개발하는 방식인 객체지향 방법론이 등장하게 되었다.

UML은 세 명의 소프트웨어 방법론 전문가인 Booch, Rumbaugh, Jacobson 이 객체지향 분석과 설계를 위해 1996년에 제시한 모델링 언어로, 그 다음해에 OMG에서 표준으로 채택되었다. 그 배경을 살펴보면 다음과 같다. 1980년대에 객체에 관한 많은 연구가 진행되었고, 그 결과 객체지향 프로그래밍언어들, 예를 들면, Smalltalk 및 C++ 등이 등장하였다. 하지만 객체지향개발을 위해서는 이러한 프로그래밍언어 외에도 분석과 설계를 위한 효과적인 기법의 필요성이 인식되었다. 이에 따라 1990년 전후에 객체지향 방법론에 대한 많은 연구가 수행되었고, 그 결과로 다수의 방법들이 제시되었다. 이러한 방법들은 서로간에 차이점이 있었을 뿐만 아니라, 비슷한 점에서도 표기방식은 전혀 달라 사용자 입장에서는 매우 혼란스러웠다. 객체지향 방법의 표준화의 필요성이 인식되면서, 소위 세 명의 친구들(the Three Amigos)이라는 애칭으로 불리는 세 명의 전문가들, Booch, Rumbaugh, Jacobson이 객체지향 방법의 process로는 Rational Unified Process(RUP)를, 그리고 모델링 언어로는 UML을 제시케 되었다 [5].

UML은 소프트웨어의 모델을 명시적으로(specify), 가시적으로(visualize) 문서화(document) 하는데 사용된다. 소프트웨어

모델은 소프트웨어의 구조와 설계를 포함한다. UML의 등장배경이 객체지향개발방법의 모델링을 위한 것임을 고려해볼 때, 자연스럽게 C++, Java 등 객체지향 언어 및 개발환경에 잘 맞추어 쓸 수 있다. 하지만 이에 국한되는 것은 아니고 Fortran, COBOL과 같은 비 객체지향 소프트웨어를 개발할 때에도 활용될 수 있다.

일반적으로, UML이 가장 적합한 모델링 언어라는 주장의 주된 근거는 UML이 세계적인 표준으로 자리잡고 있다는 사실이라고 할 수 있겠다. 모델링 표기법으로는 1970년대 중반에 이미 IBM의 HIPO, Gane & Saron의 Structured Analysis & Design 등 소프트웨어 사용자 요구의 분석과 설계를 위한 표기법들이 성숙되고 있었고, 그 이후 70년대 말에서 80년대에 걸쳐 ERD(Entity-Relationship Diagram), 정보공학(IE: Information Engineering) 및 객체지향 분석 설계(OOAD: Object-Oriented Analysis & Design) 가 등장하고, 90년대에 UML이 객체지향 분석 설계를 위한 표준 표기법으로 개발되면서 ERD나 정보공학 표기법의 장점들도 일부 흡수하여 왔다. 앞에서 언급하였듯이 종래의 OOAD 표기법을 통합, 개선하였으며, UML의 창시자들인 3 Amigo의 표기법들이 실시간/임베디드 소프트웨어(real-time/embedded SW)를 당초 대상으로 하였지만, 이제는 정보시스템영역에서도 ERD나 IE 표기법에 비해 우월한 표기법이라 할 수 있다. 이유는 데이터베이스 모델링을 위한 ERD는 UML의 클래스 다이어그램(Class Diagram)으로 손색없이 대체 될 수 있다. 또한 정보공학에서는 취약했던 비즈니스 이벤트의 분석, 프로세스 모델링, 모듈/클래스/컴포넌트들간의 동적 연관구조 정의 등이 사용 케이스 다이어그램(Use Case Diagram), 액티비티 다이어그램(Activity Diagram), 협조 다이어그램(Collaboration Diagram) 및 순차 다이어그램(Sequence Diagram) 등을 통해 크게 보완되었다.

컴포넌트 기반 개발(CBD: Component Based Development) 및 SODA 플랫폼의 주류를 이루는 J2EE나 .Net을 타깃으로 한 소프트웨어 모델링 툴들이 한결같이 UML을 채택하고 있다는 점도 향후 컴포넌트 기반 개발이나 SODA를 지향하는 소프트웨어 업체의 경우 UML을 도입해야 할 이유가 된다.

지금까지 소프트웨어 모델링 표기법이 끊임없이 진화해 왔고, 앞으로도 새로운 표기법이 출현하겠지만, 앞에서 언급한 UML이 지닌 바람직한 특성들을 고려해 볼 때, 현재로는 소프트웨어 모델링 표기법으로는 UML이 가장 우선적으로 선택할만한 모델링 언어라고 할 수 있겠다.

4. 활용 사례

A사에서 UML의 활용을 확산시키기 위한 노력은 다음과 같다. 우선, UML 사용을 활성화하기 위해서 개발자들이 되도록 많은 UML 프로젝트 참여 기회를 가질 수 있도록 하고 있다. 한국어의 문법을 잘 아는 것이 한국어로 좋은 시를 쓰는 것의 필요조건은 되나, 충분조건은 아닌 것과 같이, UML 문법을 잘 안다고 해서 소프트웨어 모델링을 잘 할 수 있는 것은 아니다. A사에서는 UML 표기법이 지향하는 방법론적 원칙을 이해하고, UML을 써서 소프트웨어 모델링을 하는 방법에 대해 충분한 교육을 받고, 멘터(mentor)의 도움 하에 최소한 서너 프로젝트에서 경험을 해야 효과적인 UML 모델링을 할 능력을 갖추게 된다고 보고 있다.

A사에서는 UML의 확산 초기에 소수의 UML 전문가를 조직 내에 확보하여 방법론의 개발에, 교육에, 멘터 역할에, 설계 감리에 활용하고 있다. 이러한 TtT(Train the Trainer) 방식의 확산 방법은 소프트웨어 회사에서 새로운 프로세스, 방법론, 기술을 도입할 때 일반적으로 바람직한 접근방법이다. 소프트웨어 조직은 변화하는 기술 환경에 대처하기 위하여 상시 변화관리 체제를 갖추는 것이 필요하다. 이를 위해서는 조직 내의 인사 및 재무 시스템이 뒷받침해주어야 한다. 앞에서도 언급하였듯이 A사에서는 UML 교육을 받은 직원이 지속적으로 UML이 활용되는 프로젝트에 배정되어 UML 역량을 축적해 갈 수

있도록 노력하고 있다. 이러한 노력이 성공하기 위해서는 조직의 인사관리 시스템이 전반적으로 역량 중심의 인사 체제(competency-based HR practice)로 변화되어야 한다고 보고 있다. 최근 선진 IT 기업들이 도입하고 있는 P-CMM(People Capability Maturity Model)도 역량 중심의 인사 체제를 지향하고 있다. 초기에 양성된 UML 전문가들의 역량을 지속적으로 강화할 수 있는 방안으로 A사는 UML 전문가들을 기술지원 조직으로 편성하였고, 이 조직의 기능은 소프트웨어 모델링이다. 소프트웨어 아키텍트나 모델링 전문가들을 여러 프로젝트에 필요한 단계에 투입하는 것이 소프트웨어 개발 비용을 절감하면서 소프트웨어 품질을 향상시킬 수 있는 방안으로 보고 있다. 이러한 기술지원 조직을 효과적으로 운영하기 위해서는 이들 전문가의 용역비용을 정확히 반영할 수 있는 회계 시스템이 필요하다. 이를 위해서는 액티비티 기반 원가(Activity-Based Costing) 시스템이 도입되어야 할 것으로 보고 있다. 신기술의 도입을 위한 투자로부터의 수익을 재무적으로 측정하여야만 경영성과로 이어져 성공적 확산을 기대할 수 있다. 또한 기술지원 조직에 UML 전문가들을 모아 놓은 것만으로는 이들의 적절한 활용을 기대할 수 없기 때문에, A사에서는 사내의 표준 소프트웨어 개발 프로세스에 단계별로 아키텍트나 모델링 전문가가 수행해야 할 역할과 창출해야 할 산출물을 정의하고 표준 프로세스가 사용되도록 규범화하려는 노력을 하고 있다.

UML 사용이 실질적으로 소프트웨어 개발에 소요되는 비용과 시간을 절감하고 소프트웨어의 품질을 향상시키는 효과를 내기 위해서는 소프트웨어 모델의 재사용이 활성화되어야 한다. UML로 작성된 요구 사양, 설계 사양, 사양에 맞추어 개발된 컴포넌트들이 재사용되기 시작하여야만 상기의 효과를 얻을 수 있다. 재사용을 촉진하기 위해서는 소프트웨어 모델을 구축할 때 분석 패턴 및 설계 패턴을 적용하고, 프로젝트들의 소프트웨어 모델 산출물들을 체계적으로 정리하여 아키텍처 참조 모델(Architecture Reference Model)로 자산화하는 노력이 필요하다.

5. 결 론

UML 사용의 근본 목적은 소프트웨어 모델의 구축과 활용이라고 보겠다. 소프트웨어 모델은 복잡한 소프트웨어를 이해하기 위해 쓰이며, 또한 사용자 요구의 정확한 반영, 개발팀원간의 업무분담, 개발비의 추정, 유지보수에 유용하게 쓰일 수 있다. 본 연구는 국내 대형 소프트웨어 업체인 A사의 UML 도입과 활용 사례를 살펴보았다. A사에서는 UML 전문가를 양성하기 위한 교육프로그램 외에도 이들을 기술지원 조직에 풀로 모아, 프로젝트에 적시에 투입코자 하는 노력을 기울이고 있다. 이러한 노력이 효과를 얻기 위해서는 인사 및 재무 시스템이 지원을 해 주는 것이 필요한 것으로 보인다. 또한 실질적인 개발 비용의 절감, 소프트웨어 품질의 향상의 효과는 소프트웨어 모델과 컴포넌트의 재사용이 이루어지기 시작해야 한다. 이를 위해서는 프로젝트가 표준 프로세스에 맞추어 진행되어야 하고, 프로젝트에서 산출물로 나온 소프트웨어 모델과 컴포넌트가 체계적으로 정리되어 새로운 프로젝트에서 재사용될 수 있어야 한다. 본 논문의 내용은 인터뷰의 결과를 기반으로 하였으며, 작성되는 시점에 앞에서 언급한 설문조사가 완료되지 못하였다. 따라서 본 연구의 결과에 설문 자료가 포함되지 않았음을 밝혀두고자 한다.

6. 참고문헌

[1] Briand L. C., Y. Labiche and L. O'Sullivan, "Impact Analysis and Change Management of UML Models", *Proceedings of the International Conference on Software Maintenance (ICSM'03)*, *IEEE*, Sept. 22-26, (2003), pp. 256-265.

[2] Brooks F. P., *The Mythical Man-Month: Essays on Software Engineering*, Addison-Wesley,

New York, 1995.

[3] Diva T., N. B. Moe and E. M. Mikkelsen, "An Empirical Investigation on Factors Affecting Software Developer Acceptance and Utilization of Electronic Process Guides", *Proceedings of the 10th International Symposium on Software Metrics (METRICS'04)*, *IEEE*, (2004), pp. 220-231.

[4] Edwards C., "Model Development" *IEE Review*, Vol. 49, Issue 8, Aug., (2003), pp. 42-45.

[5] Fowler M., *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Addison-Wesley New York, 2004.

[6] Larman C., *Applying UML and patterns: an introduction to object-oriented analysis and design*, Prentice-Hall, N. J., 1998.

[7] Riemenschneider C. K., B. C. Hardgrave and F. D. Davis, "Explaining Software Developer Acceptance of Methodologies: A Comparison of Five Theoretical Models", *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, Vol. 28, No.12, Dec., (2002), pp. 1135-1145.

[8] Seidewitz, E., "What models mean" *Software, IEEE*, Vol. 20, Issue 5, Sept.-Oct., (2003), pp. 26 - 32.

[9] Stanat, D. F. and D. F. McAllister, *Discrete Mathematics in Computer Science*, Prentice Hall, N.J., 1977.

[10] Dobing B. and J. Parsons, Unified Modeling Language Usage Survey. (http://fusion.uleth.ca/crdc/uml_survey/)

[11] Capability Maturity Model Integration (CMMI) Version 1.1, Carnegie Mellon Software Engineering Institute, March 2002. (http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr011.pdf)