지능정보연구 제15권 제3호 2009년 9월(pp.31~51)

MDD 프로세스 효과성 측정을 위한 사례 연구

문성욱
서울시립대학교 경영학부
(swmoon2@netsgo.com)

홍사능
서울시립대학교 경영학부
(snhong@uos.ac.kr)

정보시스템 개발 프로세스와 기법에 대한 연구는 1960년대 이후 지속적으로 이뤄져 왔으나 실제 개발 현장에 성공적으로 적용되는 사례는 적은 경우에 불과하다. 최근 기술 발전과 더불어 모델 기반의 정보시스템 개발 기법인 MDD(Model-Driven Development)가 많은 관심을 모으고 있다. MDD의 발전과 더불어 그 효과를 검증하기 위해 다양한 연구가 이뤄지고 있으나, 대부분은 사례연구를 통해 얻어진 교훈이나, 제한된 측정 데이터로 분석된 효과를 제시하고 있다. 본 연구에서는 MDE(Model-Driven Engineering)의 기술현황 및 주요 이슈를 알아보고, MDE 도입을 검토하는 조직을 위해 효과성 측정을 위한 기존 연구 결과를 기술적, 조직적 측면에서 정리하였다. 또한, MDD를 실제 정보시스템 개발 프로세스에 적용한 사례연구를 통해 새로운 측정지표로 정량적, 정성적 효과성을 측정하였다. 더불어, MDE 도입 시 주요 고려사항과 사례연구를 통해 얻은 교훈을 기술적, 조직적 차원에서 제시한다.

논문접수일 : 2009년 06월 29일 논문수정일 : 2009년 07월 09일 게재확정일 : 2009년 07월 15일 교신저자 : 홍사능

1. 서론

불확실성과 도전으로 가득한 경영환경에서 생존해야 하는 기업에게 정보의 수집과 활용은 선택이 아닌 필수적인 역량이며, 기업들은 이를 위해 정보시스템 개발과 운영에 지속적으로 투자하고 있다(Laudon and Laudon, 2006). 빠른 변화, 복잡한 환경 그리고 치열한 경쟁으로 인하여 정보기술의 효과적인 활용이 없이는 위기를 기회로 전환하여 경쟁우위를 확보하는 것이 불가능하게 되었기 때문이다. 그러나, 정보시스템의 도입과 개발에는 많은 위험과 어려움이 도사리고 있으며, 정보기술이 복잡해지고 정보시스템의 규모가 커지면서 이러한 경향은 더욱 심화되고 있다. 정보시스템 프로젝트에 따르는 위험을 효과적으로 관리하고 작업자의 생산성을 높이기 위해 개발 프로세스와 생명주기에 대한 연구가 1960년대 이후 지속적으로 이뤄져 왔으나(Boehm, 2003), 여전히 많은 개발 프로젝트에서는 1960년대에 개발된 순차적 개발 프로세스를 답습하고 있으며(Selic, 2003), 성공적인 정보시스템 개발 프로젝트의 비율은 실망스러운 수준이다(The Standish Group, 1995). 이러한 상황을 개선하기 위해서 소프트웨어 공학 분야에서는 새로운 개발 방법론, 기술, 절차 및 도구를 끊임없이 연구해 왔으나, 연구 결과의 실용적인 효과에 대해서는 의문이 제기되고 있다(Baskerville et al., 1992; Fitzgerald, 1998). 여기에는 다양한 원인이 있겠으나, 대다수의 소프트웨어 개발 프로젝트에서 개발 프로세스의 체계적인 정착과 반복이 이뤄지지 않아, 하나의 우수사례를 다른 프로젝트에도

성공적으로 재현할 확률이 낮은 것이 중요한 원인의 하나로 지적된다. CMMI[1]는 프로세스의 표준을 정립하고 지속적으로 개선하려는 노력을 통해 이러한 문제를 해결함으로써 조직의 프로젝트 수행 역량을 체계적으로 높여가려는 시도이다(Chrissis et al., 2006). 새로운 프로세스 개발이나 적용 절차의 표준화에 더하여, 개별 조직이 가진 문화적 특성이 개발 프로세스 적용에 미치는 영향에 대한 연구가 최근에 이뤄지고 있어서, 정보시스템의 기술적 문제와 더불어 조직적 측면에서의 이해를 더해가고 있다(Iivari and Huisman, 2007).

MDE(Model-Driven Engineering)는 새로운 프로세스의 고안, 표준화, 조직의 특성에 대한 연구와 다르게 개발 산출물에 대한 근본적인 개념 전환을 통해 정보시스템 개발을 획기적으로 개선하고자 한다. 전통적인 개발 프로젝트에서는 프로그램 언어로 작성된 소스코드에 중점을 두는 데 비해, MDE는 비즈니스와 기술적 해결방안을 표현하는 다양한 모델에 중점을 두고, 비즈니스와 기술적 요소를 분리하여 표현함으로써, 프로젝트의 이해가능성을 높일 뿐만 아니라 모델의 생명주기를 연장시키고자 한다. 비즈니스와 분리된 기술적 해결방안의 설계를 자동화된 일련의 모델 변환 과정으로 대체하여 개발 생산성과 품질을 높이려는 것이 MDE의 핵심 목표이다.

지금까지 MDE의 개념과 기술에 관한 연구는 많이 진전되었으나, MDE 효과에 관한 연구는 매우 빈약하다. MDE 도구를 개발한 업체나 컨설팅 업체의 적용 사례들을(Bertrand, 2003) 제외하고는, MDE 기반의 도구와 프로세스를 적용한 실험이나 사례를 통해 MDE의 효과를 검증한 연구가 많지 않으며, 또한 MDE의 생산성 효과를 정량적으로 분석한 단편적인 연구만 있었을 뿐(김학인, 최오훈, 2004; 윤정모, 김치호, 2008), 기술과 조직을 통합해서 조망하여 MDE 효과를 실제 데이터를 제시하며 분석한 연구는 거의 없다.

이에 본 연구는 전통적인 개발 프로세스에 MDE 기술을 도입하였을 때 기대되는 기술적, 조직적 측면의 효과성에 대한 기존의 연구 현황을 알아보고, 실제 정보시스템 개발 프로젝트에 MDE를 적용한 사례 연구를 통해 생산성 및 품질에 미치는 MDE 효과를 분석한다. 또한, MDE의 도입을 추진하는 조직이 고려해야 할 사항을 논의한다.

먼저, 제 2장에서 전형적인 소프트웨어 개발 프로세스의 개략적인 구성을 살펴보고, 그것이 갖고 있는 모델링의 한계점을 논한 후, MDE 전반의 주요 개념과 이슈사항을 알아본다. 특히, 전형적인 개발 프로세스에 MDE 기술을 적용할 때, 변화되는 프로세스를 기술한다. 제 3장에서는 MDE 효과 측정을 위한 기존 연구 현황을 정리하고, 제 4장에서는 MDE 효과를 측정한 사례연구를 통해 MDE를 전형적인 개발 프로세스에 적용하기 위한 접근방법과 MDE 효과 분석을 위한 접근방법을 기술한다. 이어서 제 5장에서는 연구 시사점과 교훈을, 마지막으로 제 6장에서는 결론 및 향후 연구방향에 대하여 논의한다.

2. MDE와 개발 프로세스에 관한 연구

2.1 전형적인 정보시스템 개발 프로세스

소프트웨어 공학 분야에서는 개발 프로젝트의 실패 가능성을 줄이고, 개발의 효율성을 높이기 위해서 1960년대 초반부터 프로세스에 대한 연구를 시작하였다. 연구의 대부분은 과거 경험을 바탕으로 반복할 가치가 있는 우수사례를 통합, 정리하여 새로운 프로세스를 정립하는데 중점을 두었다. 그간에 다양한 기술 패러다임에 기반한 다양한 개발

1) CMMI : Capability Maturity Model Integration.

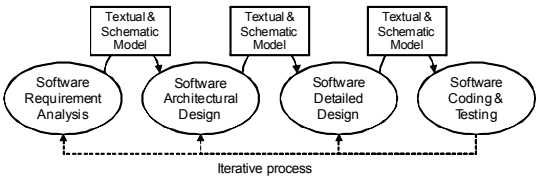
<표 1> ISO/IEC 12207-소프트웨어 개발 활동

활동	설명
Requirements Analysis	소프트웨어 요구사항을 개발하는 활동으로, 요구사항 식별, 명세 등의 작업 수행
Architectural Design	소프트웨어 요구사항을 상위 수준의 소프트웨어 기본설계로 전환하는 작업 수행
Detailed Design	소프트웨어 기본설계를 하위 구성요소로 분할하고, 각 요소의 세부 내용을 설계하는 작업 수행
Coding and Testing	소프트웨어 상세설계를 소프트웨어 소스코드로 전환하고 시험하는 작업 수행
Integration	구현 및 단위 테스트된 소프트웨어 부품을 통합하고 평가하는 작업 수행
Qualification Testing	사용자 요구사항 수준에서 테스트 케이스와 절차를 이용하여 소프트웨어를 평가하는 작업 수행

프로세스가 등장하였지만(Boehm, 2003), 본 연구에서는 ISO/IEC 12207 소프트웨어 생명주기 프로세스로(ISO/IEC, 1995) 전형적인 개발 프로세스의 모델을 살펴본다. ISO/IEC 12207는 소프트웨어의 전체 생명주기를 지원하는 국제표준으로써, 지원, 조직 및 프로세스로 구성되며, 프로세스 중 13개의 활동으로 수행되는 개발 부분이 소프트웨어 개발 프로세스에 해당한다. 이중에서 개발을 위한 6개의 핵심 활동을 정리하면 <표 1>과 같다.

소프트웨어 개발의 핵심이 되는 처음 네 개 활동의 작업 흐름을 다이어그램으로 나타내면 <그림 1>과 같다.

<그림 1>은 개발 프로세스의 흐름에 따라 세 개의 직사각형으로 표기된 글과 도형으로 작성된 모델이 네 개의 타원형으로 표기된 개발활동을 매개함을 보여준다. 아래의 점선은 하위 활동 수행결과의 확인 및 검증에 따라 이전 작업의 수정 또는 보완을 위해 순환되는 흐름을 나타낸다. 순환 흐름은 적용하는 개발 프로세스에서 공식적으로 정의되



<그림 1> 전형적인 소프트웨어 개발 프로세스

지 않는 경우도 있기 때문에 점선으로 표시하였다.

2.1.1 전통적인 정보시스템 모델링의 한계

소스코드 기반의 전통적인 개발 프로세스에서 모델은 문제 영역을 분석하고 해결책을 찾는 과정에 문서 위주의 산출물을 만들기 위한 용도로 사용되었다. CASE(Computer-Aided Software Engineering)는 이를 위한 소프트웨어 공학 연구의 한 결과로, 모델을 이용하여 산출물 생성과정을 최대한 자동화함으로써 소프트웨어 개발자의 부담을 줄여주려는 것이다. UML과 같은 표준화된 모델링 언어의 출현으로 소프트웨어 아키텍처와 기술 플랫폼의 관점을 부여한 모델을 프로그래밍 언어로 변환할 수 있는 가능성은 열렸지만, 여전히 대다수의 개발자들은 모델을 소스코드를 생성하기 위한 중간 산출물이 아니라, 단지 소스코드를 작성하기 위한 참조 자료로 사용하고 있다(Brown et al., 2005).

정보시스템 개발에 있어서 모델의 역할과 모델링 기법은 다음과 같은 한계점을 가지고 있다.

2.1.1.1 문제-구현의 간격(Problem-implementation gap)

해결하고자 하는 문제 영역을 표현한 모델에서 소스코드 위주로 구성된 모델로 해결방안을 구현하고자 할 때 소프트웨어 개발자들은 문제와 구현의 추상화 수준에 큰 차이가 있음을 알게 된다. 복잡한 시스템을 개발하는 경우에, 문제-구현의 간

격의 대부분을 수작업으로 극복하려면 상당한 수준의 우발적(accidental) 복잡성을 유발하게 된다. 만일, 새로운 기술을 적용하여 이러한 격차를 좁히려 한다면, 반대급부로 시스템이 더 복잡해지는 결과를 초래할 수 있어, 결국 문제-구현 사이의 간격이 다시 벌어지게 되는 어려움에 봉착하게 된다(France and Rumpe, 2007).

2.1.1.2 모델링 언어(기법)의 의미론(semantics)의 부족

소프트웨어 시스템 개발은 문제 영역의 모델에서 문제해결 영역의 모델로 변환 과정을 거치면서 점차 구체화 되는데, 이 과정에서 모델을 표현하는 언어가 엄정하게 정의 되어 있어야 개발 참여자들이 명확하게 의사를 소통할 수 있다. 그러나, 아직도 많은 경우 비정형화된 언어를 사용하거나, UML과 같은 모델링 언어를 사용하지만 의미론(semantics)을 제대로 적용하지 못해 모델 구성요소의 의미에 대한 엄정성(rigorousness)을 확보하지 못하고 있다. Baker et al.(2005)은 사례연구를 통해 현장에서도 일반화된 의미론의 규약이나, 명시적인 의미론이 잘 정의되지 않는 문제를 지적하였다.

2.1.1.3 환경 변화에 취약한 모델로 인한 생산성 저하

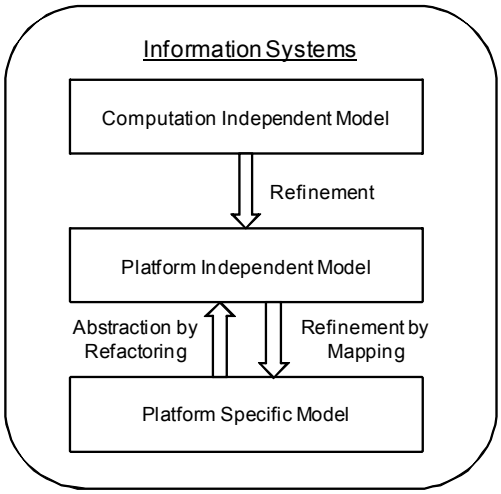
급변하는 경영환경과 치열한 경쟁, 다양한 최신 정보기술의 빈번한 등장과 점차 짧아지는 기술 생명주기 등은 경영지원을 위한 정보시스템이 환경변화에 즉응해야 하는 당위성을 설명하기 위해 자주 언급되는 현상들이다. 그러나 대부분의 정보시스템 개발 프로젝트에 적용되는 전형적인 개발 프로세스와 산출물로는 개발이나 운영 과정에서 발생하는 변화에 대응하기 어렵기 때문에 대개의 경우 최종 산출물인 프로그램 소스코드만을 수정하여 대응하고 있다. 이로 인해 개발 과정에서 생산되는 단계별 모델의 중요성이 저평가되고 충실하게 만들어지지 못하고 있다. 또한, 모델간의 불일치가 발생할 뿐만 아니라 조직 내부 혹은 유사 분야의 정보시스템 개발을 위해 모델이 재사용 되는 사례는 극히 드문 실정이다. 이는 개발 산출물인 모델이 변화에 매우 취약하기 때문인데, Atkinson and Kühne(2003)는 산출물의 수정을 필요로 하여 생산성을 저하시키는 변경요인을 참여인력, 요구사항, 개발환경 및 운영환경의 네 가지로 분류하였다.

2.2 Model-Driven Engineering(MDE)

2.2.1 MDE 개요

전통적인 정보시스템 개발 프로세스에서 나타나는 모델의 한계를 극복하고, 소프트웨어 전체 생명주기에서 모델의 역할을 강화하기 위해 연구되기 시작한 MDE는 소프트웨어 시스템 개발을 위해 최초 추상화된 모델을 만들고, 이 모델로부터 체계적인 변환과정을 거쳐 최종적으로 구현된 소프트웨어를 생산하는 소프트웨어 개발 기법을 말한다(France and Rumpe, 2007). 여기서 추상화된 모델이란 구체적인 구현 기술이나 알고리즘 개념이 배제된 특정 영역의 업무 모델을 말하며, 이러한 추상화된 모델을 소프트웨어 시스템으로 구현하는 변환은 한번에 일어나는 것이 아니라 여러 단계에 나뉘어 이뤄진다. 각 단계의 모델 변환의 결과로 추상화된 모델과 구현 모델 사이에 하나 이상의 중간 단계 모델이 생성된다(Kent, 2002).

MDE 연구는 모델의 중요성을 유지하기 위하여 모델을 효과적으로 표현할 수 있는 개념과 도구를 개발하고, 모델 간의 변환을 최대한 자동화할 수 있는 기술을 개발하는 것을 핵심으로 한다.



<그림 2> MDA 모델 유형과 상호관계(OMG, 2001)

2.2.2 Model-Driven Architecture (MDA)

MDA는 2001년 OMG (Object Management Group)가 제정한 기술 표준을 중심으로 하는 MDE 프레임워크이다(France and Rumpe, 2007). 이는 OMG의 분산환경 소프트웨어 시스템 통합 및 상호운용성을 위한 기술 표준을 발전시킨 모델 주도적인 개발을 위한 새로운 소프트웨어 아키텍처로써, 소프트웨어 개발에 있어 모델링이 개발 프로세스 전반을 주도하는 핵심 활동임을 강조한다(Kleppe et al., 2003).

2.2.2.1 MDA 모델 유형과 상호관계

MDA에서는 정보시스템 추상화 수준에 따라 모델 유형을 CIM(Computation Independent Model), PIM(Platform Independent Model), PSM(Platform Specific Model)로 분류한다(OMG, 2001; Frankel, 2003). <그림 2>는 세 종류의 모형 사이에 존재하는 관계를 보여준다. CIM, PIM, PSM 사이에는 정제(refinement) 관계가 있다. 즉, CIM의 업무 모델에 전산처리 상세 요소를 추가한 모델이 PIM이고, PIM에 특정 플랫폼에 대한 상세 요소를 추가한 모델이 PSM이다.

PIM에서 PSM으로 정제는 대응(mapping) 관계로 정의한다. 반대로, PSM에서 PIM으로의 추상화(abstraction)는 리팩토링(refactoring)에 의해 이뤄진다. 그러나, CIM에서 PIM으로의 정제는 CIM의 모든 요소가 PIM의 요소로 대응되지 않는 것이 보통이다. 만일 서로 다른 두 시스템의 PIM과 PSM이 동일한 메타모델을 기반으로 하고 있다면, PIM에서 PSM으로의 매핑 규칙은 공유될 수 있으며, 두 시스템의 PIM과 PSM은 상호호환성을 가져 통합이 좀 더 용이하게 된다.

2.2.2.2 모델 변환 (Model Transformation)

모델 변환은 정보시스템 개발을 위한 MDA의 핵심 개념이다. 모델 변환은 추상화된 모델에 특정 관점의 시스템 상세 정보를 추가하여 좀 더 구체적인 모델을 만드는 것을 의미한다. MDA에서는 정보시스템 개발 프로세스를 일련의 모델 정제로 간주할 수 있는데, 이때 모델 변환은 개발 프로세스의 핵심이 된다(Brown et al., 2005).

2.2.2.3 UML(Unified Modeling Language)

MDA 핵심 기술 표준의 하나인 UML은 모델링을 위한 선언적(declarative) 언어로써 모델을 작성하기 위한 모형 요소의 표기법(notation), 의미론(semantics), 제약조건(constraints) 등을 제공한다. MDA의 유형별 모델을 작성함에 있어서 UML 모델링 요소에 스테레오 타입을 이용하여 새로운 속성이나 관계를 추가하거나 수정하여 메타모델을 확장하는 프로파일(profiles)을 정의하여 사용할 수 있다. 그러나 프로파일은 새로운 모델링 요소를 추가하거나 기존의 요소를 삭제할 수는 없다(France et al., 2006).

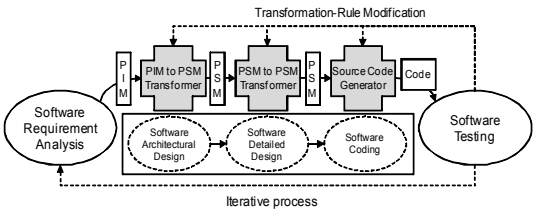
2.2.2.4 MOF(Meta Object Facility)

또 다른 MDA 핵심 기술 표준의 하나인 MOF는 UML, SPEM[2]과 같은 메타모델을 정의하기 위한 메타 언어 체계로, 모델링 언어의 필수 요소, 문법, 구조를 정의한다(OMG, 2001; France et al., 2006). 이를 통해 도메인에 특화된 새로운 개념, 즉 새로운 메타모델 체계를 정의하여 UML 프로파일의 한계를 극복한 모델링 확장 메커니즘 역할을 한다. 또한, MOF는 메타데이터를 관리할 수 있는 기반을 제공함으로써, 메타언어를 이용해 정의된 모델의 교환 및 접근을 용이하게 하여 모델 혹은 모델링 도구의 상호호환성을 높여준다(Frankel, 2003; Mellor et al., 2004). 예를 들어 XMI (XML Metadata Interchange)는 MOF 기반 메타언어로부터 XML 스키마를 도출하고, 모델을 XML 문서로 변환하는 규칙을 정의한 것으로, 메타데이터 관리 체계가 서로 다른 모델링 도구들 사이에 존재하는 모델의 상호호환 문제를 해결할 수 있도록 한다.

2.2.3 MDD(Model-Driven Development)

MDD는 모델 기반의 소프트웨어 개발방법론으로 MDA처럼 MDE 개념을 구현한 기술 표준, 도구 및 절차를 소프트웨어 개발에 적용하는 것이다. MDA 핵심기술 요소인 PIM 모델링 기술, 모델 변환 기술, MOF 기반의 메타모델링 언어체계와 모델 저장소 기술 등이 MDD에 적용될 수 있다. MDD는 수작업 위주의 전통적인 개발 프로세스를 연속적인 모델 변환 프로세스로 전환시켜 개발 과정을 단순화시킴으로써, 개발 작업의 중심이 프로그램 코딩에서 모델 작성과 변환으로 이동하게 한다.

2) SPEM(Software Process Engineering Metamodel) : 소프트웨어 프로세스를 정의하기 위한 메타모델



<그림 3> MDD를 도입한 개발 프로세스

2.2.4 MDD 기반 정보시스템 개발 프로세스 모델링

제 2.1에서 정의한 전형적인 개발 프로세스에 MDD를 적용하면 <그림 3>와 같이 수정된다. 여기에서 설계 단계에 해당하는 Software Architectural Design과 Software Detail Design, 그리고 Software Coding 활동은 PIM to PSM, PSM to PSM 모델 변환으로 대체됨을 나타내기 위해 점선의 타원으로 표시하였다. 생성된 PSM 모델은 소스코드 생성기(source code generator)를 통해 실행 가능한 소스코드로 변환되어 코딩 단계가 대체되고 곧바로 테스트 단계로 넘어가게 된다.

MDD를 적용하면 순환 흐름에 큰 변화가 발생한다. 기존 프로세스에서는 Coding and Testing 활동에서 상위 각 단계로 순환되나, MDD를 도입할 경우 <그림 3>에서 보듯이 모델 변환 활동으로 순환되는 흐름은 사라지고, 변환기의 변환규칙에 수정이 가해지는 순환 흐름이 생긴다.

2.2.5 MDE 주요 이슈

다른 혁신적인 기술과 마찬가지로, 개발 프로세스의 혁신을 추구하는 MDE를 성공적으로 도입하기 위해서도 해결해야 과제가 있다. MDE가 제시하는 기술적 효과뿐만 아니라, 조직적인 차원에서 MDE 도입이 미치는 효과를 측정할 수 있어야만 MDE 도입의 실용적 근거를 입증할 수 있을 것이다. 새로운 정보시스템 개발 패러다임인 MDE가 안

고 있는 주요한 기술적, 조직적 이슈는 다음과 같다.

2.2.5.1 기술적 이슈

기술적 이슈는 현재 MDE 연구에서 주로 다뤄지는 주제들로써, MDE의 도입 목표인 생산성 향상, 품질제고, 모델 생명주기 연장을 달성하기 위해서 해결해야 할 기술적 과제들에 해당한다.

● 모델링 언어의 효과성

비즈니스 문제를 분석하는데 적합한 추상화 수준의 모델의 생성과 처리뿐만 아니라, 엄밀한 모델 분석을 효과적으로 지원하기 위해 모델링 언어에 요구되는 기술적 과제이다. France and Rumpe(2007)는 모델링 언어가 다뤄야 할 추상화와 정형화(formalization) 문제에 있어 UML과 같은 범용 모델링 언어와 특정 도메인에 특화된 모델링 언어의 특징을 비교하였다. 또한, Staron et al.(2004)은 MDE를 적용할 때 모델 확장과 변환을 위한 모델링 언어의 효과적인 사용을 결정하는 요인을 제시하였다.

● 모델 조작과 관리

모델의 정의, 분석, 변환, 추적성 유지, 일관성 유지, 형상관리 등 모델의 처리와 관리에 관계된 제반 문제로, 예를 들어 자동 변환으로 생성된 구현 모델과 수작업으로 추가된 소스코드의 동기화와 같이 추상화 수준이 낮은 모델에 변형이 일어났을 때 상위 수준의 모델과의 동기화, 추적, 형상관리 등의 조작과 관리에 관한 과제이다(France and Rumpe, 2007).

● 지원 도구의 성능 및 통합

대규모 시스템의 모델 작성 및 변환을 다루는 MDE 도구의 성능 및 효율성과, 연속적인 모델 변환과 자동화된 개발 과정을 지원할 수 있는 통합환경에 관한 과제이다(Baker et al., 2005).

2.2.5.2 조직적 이슈

조직적 이슈는 MDE 뿐만 아니라 모든 신기술을 도입하려는 조직이 당면할 수 있는 다양한 이슈에 해당한다.

● 조직성과에 대한 영향

MDE 기술 도입을 통해 얻게 되는 가치인 생산성, 품질 등의 향상이 IT 조직성과에 미치는 영향에 대한 연구는 찾아보기 힘들다(Mohagheghi and Dehlen, 2008). MDD를 방법론으로 적용했을 때 나타나는 혁신성과 제공 가치 측면의 측정 지표 연구는 있으나(Fujita and Zualkernan, 2008), 이 역시 MDE 기술도입이 제안하는 가치가 조직성과 달성에 미치는 영향을 다루고 있지는 못하므로 MDE의 조직성과에 대한 인과 모델과 개념 설계를 위한 탐색적 연구가 필요하다.

● 신기술을 도입할 때 조직이 인지하는 효과

MDE와 같은 새로운 소프트웨어 개발 기술을 도입할 때 예상되는 효과를 실제로 조직 구성원이 인지하는가에 대한 문제이다. 더불어, 정보기술 조직의 관리자와 기술자 사이에 존재하는 효과에 대한 인식의 차이에 대한 연구와, 조직문화의 유형과 인지된 효과와 연관관계에 대한 해외 연구가 있으나(Huisman and Iivari, 2006; Iivari and Huisman, 2007) 국내의 조직 유형과 문화가 해외에서 연구된 경우와 다를 수 있으므로 국내 MDE 기술 도입 실태 및 현장 자료를 기반으로 하는 실증연구가 필요하다.

● 기술도입 수준과 기술인력 양성

신기술에 대한 저항을 최소화하면서 기술 도입

의 효과를 극대화하려면 무엇보다 현재 사용하고 있는 기술 현황을 정확히 파악하여야 한다. MDE 기술을 도입하려는 조직이 보유한 기술적 역량(도구, 절차, 전문인력)을 진단하고, 그 결과에 따라 도입하려는 기술수준을 결정해야 한다. 또한 필요한 기술인력을 지속적으로 양성하기 위해 내부 프로세스 개선 전략이 수립되어야 한다. 이러한 이슈를 다루기 위해서 조직 내부에 MDE 기술자문위원회를 운영하거나 모델링 기술수준 체계를 수립하는 방안과(Baker et al., 2005), MDE 성숙도 모델을 제안한 사례가 있다(Rios et al., 2006).

3. MDE 효과에 관한 연구 현황

MDE를 성공적으로 도입하려면 조직목표와 부합하는 측정 지표를 설정하고 지표의 측정 방안을 마련해야 하며, 조직에 관련된 이슈를 파악하여 적절하게 대처해야 한다.

MDE 효과를 검증하기 위한 노력은 지속되고 있으나, 효과에 대한 정의가 서로 다르고, MDE를 도입하려는 조직의 목적이 서로 달라 연구의 방향과 결과를 일관성 있게 비교하지 못하고 있다. 주된 연구의 방향은 MDE 관련 기술과 도구의 발전으로 실제 적용사례가 다수 등장하면서 이로부터 얻어진 데이터를 기반으로 MDE 효과를 분석하는 것이다.

3.1 생산성 제고

MDE의 도입으로 기대되는 효과 중에서 생산성 개선에 대한 측정이 가장 많이 시도되고 있다. 생산성 개선 효과는 MDE를 적용한 경우와 그렇지 않은 경우에 동일한 규모의 프로젝트를 수행하는데 필요한 자원의 양을 비교하여 산정한다. Middleware Company의 2003년 MDA 생산성 측정을 위한 사례연구보고서에 따르면, 두 개의 서로 다른 팀으로 하여금 동일한 시스템을 개발하도록 하여 투입된 인적 자원을 측정한 결과, MDE 팀이 다른 팀에 비해 약 35%의 높은 생산성을 보였다 (Middleware Company, 2003). 그러나 이 연구에서는 MDE 팀이 사용한 도구의 개발에 투입된 자원을 고려하지 않았고 또 자동 생성된 소스코드의 성능 및 유지보수 단계에서의 생산성을 감안하지 않았다(Mohagheghi and Dehlen, 2008). MODELWARE의 2006년 연구는 생산성 향상이 있었다는 사례와 생산성 향상이 없거나 오히려 떨어졌다는 사례를 함께 보고하고 있다. 주목할 것은 생산성이 괄목할 만큼 향상되었다고 보고한 사례에서는 MDE 도구 개발 비용을 고려하지 않았으며, 생산성이 없었다고 보고한 사례에서는 MDE 도구의 낮은 성숙도와 사용용이성이 생산성을 저해하는 주된 요인으로 지적되었다. 그러나 보고서의 결론에서는 MDE 기술 도입이 생산성을 향상시킨다는 가설에 대한 유의한 결과를 얻지 못했으나, 유지보수 작업에 관해서는 평균 21~24%의 생산성 향상이 있었다고 보고하였다. 이는 유지보수가 소프트웨어 생명주기 전체에서 가장 큰 비중을 차지하고 있음을 감안하면, MDE 도입이 생산성 향상에 긍정적인 효과가 있음을 시사한다(MODELWARE, 2006).

Mohagheghi and Dehlen(2008)은 MDE의 산업계 적용 사례에 대한 문헌연구를 통해 생산성 향상의 요인으로 소스코드 자동생성, 모델 기반 시뮬레이션 및 테스트, 테스트 자동생성, 결함 회피, 설계 및 테스트의 재사용 등을 들고 있다. 또한 많은 경우에 생산성 측정을 위한 지표의 정의가 적절하지 않거나, 정량적 데이터를 포함하지 않고 있으며, 아직은 MDE의 확장성(scalability)을 평가하기 위한 대규모 MDE 적용 사례가 부족하여 더 많은

사례연구가 필요함을 지적하였다.

3.2 품질 개선

품질은 생산성과 함께 MDE의 효과를 대표하는 지표로서 많은 연구에서 MDE 도입으로 얻을 수 있는 이점으로 품질향상을 들고 있다. 사례연구를 통해 주로 주장되는 품질 측면의 이점은 소프트웨어 제품 결함의 획기적인 감소이다. Motorola의 경우 MDE 도입으로 인해 소스코드 검사(inspection) 속도가 기존의 시간당 100line에서 300line 혹은 많게는 1,000line으로 3배 이상의 향상이 있었으며, 결함이 발생한 시점과 가까운 시점에서 발견되는 경향이 있음을 보고하였다(Weigert and Weil, 2006). 이것은 Boehm의 연구에서 알려진 바와 같이 요구사항 단계에서 발견된 결함을 처리하는 비용이 1이라면 이후 단계에서 발견된 결함의 처리비용은 급격히 증가하여 유지보수 단계에서는 무려 40~1,000배까지 증가하는(Boehm, 1981) 것을 감안하면, 결함의 조기 발견은 품질개선 뿐만 아니라 생산성 향상에도 크게 기여한다. 이에 주목하여 Baker et al.(2005)은 inspection 속도를 3배 개선하면 생명주기 전체로 볼 때 결함 제거를 위해 필요한 시간이 30~70배까지 차이가 날 수 있다고 주장하였다. 특히, 소스코드가 아니라 모델을 중심으로 하는 MDE는 비즈니스 수준으로 추상화된 모델의 시뮬레이션으로 완전성을 검증하여 개발되는 정보시스템의 품질을 더욱 향상시킬 수 있다(MODELWARE, 2006; Weigert and Weil, 2006).

그러나, 대부분의 사례연구는 정량적 연구 데이터를 제시하지 않고 있으며, 품질 향상의 수치도 대략적인 값으로 제시되고 있어, 일관성 있는 데이터의 기준이 부족하고 연구 결과에 대한 평가도 대체로 주관적이라고 판단된다. 생산성 지표와 마찬가지로 품질 지표의 정의가 명확하지 않은 것도 기존 연구의 한계이다(Mohagheghi and Dehlen, 2008).

3.3 MDE 기반의 개발방법론

소프트웨어의 규모와 복잡성이 커지고 수요가 증가함에 따라 소프트웨어 개발은 산업화의 과정을 거치고 있다. 이에 따라, CASE 도구 등을 통한 수작업 대체와 지원, 개발작업의 수평적, 수직적 분화, 그리고 소프트웨어 제품의 표준화와 품질관리를 통한 공식화가 이루어지고 있다. 앞의 제 2.2.4절에서 기술한 MDD 기반의 정보시스템 개발 프로세스는 산업화의 실례를 보여주는 것으로, 기존 수작업으로 연계되던 개발 단계를 자동화된 모델 변환 과정으로 대체한다. 새로운 개발방법론을 도입하고자 할 때 생산성과 품질에 대한 개선효과를 도입의 판단기준으로 제시하는 것이 일반적이나, 조직적인 이슈를 고려하지 않은 방법론의 적용은 실패하거나 제한적인 효과를 얻는데 그치게 된다(Yourdon, 1999).

Huisman and Iivari는 개발방법론이 미치는 효과를 다음과 같은 지원과 영향에 대해 참여자가 체감하는 정도로 측정할 것을 제안 하였다(Huisman and Iivari, 2006):

- 개발 작업의 지원
- 관리와 통제의 지원
- 인지와 협업의 지원
- 개발될 시스템의 품질에 미치는 영향
- 생산성과 프로세스의 품질에 미치는 영향

그러므로, MDE의 효과를 제대로 파악하려면 생산성과 품질에 대한 개선효과와 더하여 개발방법론으로써 미친 영향을 조직적 관점에서 설정된 지표로 측정해야 한다.

3.4 MDE 적용의 성공요인

성공적인 MDE 도입을 결정하는 요인을 파악하면 MDE의 효과를 제고하는 방안에 관한 규범적 연구가 가능할 것이나, 아직까지 이러한 결정요인에 관한 탐색적 연구가 부족하다. 특히, 소프트웨어 개발방법론의 적용 효과에 대한 연구는 통제된 실험과 사례연구의 재현에 제약조건이 많아서 일반적인 이론을 정립하기 어렵다.

3.4.1 기술적 측면의 결정 요인

도입의 성공여부를 결정하는 요인에 대한 연구의 대부분은 탐색적 사례연구로 수행되었다. Staron et al.(2004)은 기술 측면의 결정 요인을 두 개의 MDE 적용 사례의 연구를 통해 모델링 프로파일과 모델 변환의 정의(definition)와 사용(use)이라는 두 개의 그룹으로 분류하였다

첫 번째 그룹은 MDE 기술 도입을 위한 도구 개발자가 도입하려는 기술과 프로세스를 엄정하면서도 이해하기 쉽게 정의하여, 개발자들이 표준이나 프로세스를 준수하는데 있어 어려움을 최소화하고 개발작업에 유용하도록 하는 요인들이다. 반면에, 두 번째 그룹은 새로 만들어진 프로세스를 조직에 성공적으로 도입하여 원활하게 사용되도록 하는 요인으로써, MDE가 도입되기 이전의 상태와 비교할 수 있다.

그러나, Staron et al.(2004)의 연구는 두 개의 사례에 의존하고 있어서 일반화에 제약이 있으며, 제시된 결정 요인을 이용한 효과를 측정할 수 있는 모델의 설계와 모델의 실질적 검증을 위한 연구가 필요하다.

3.4.2 조직적 측면의 결정 요인

개발방법론과 같은 새로운 기술 도입의 효과를 알고자 할 때, 단순히 기술적 요인만이 아니라 조직적 결정 요인에 대해서도 함께 연구해야 한다. 하지만, 이 부분에 대한 연구가 부족하며, 더욱이 MDE 자체만으로 조직적 결정 요인을 탐구한 연구를 찾아볼 수 없다.

Fitzgerald(1998)는 개발방법론이 조직과 개발 프로세스에 미치는 효과를 파악하기 위해서 개발방법론을 도입한 조직과 그렇지 않은 조직에 있어서 개발방법론에 대한 인식과 정보시스템 개발에 미치는 영향을 분석하였으나, 당시에는 개발방법론이 정보시스템 개발에 기여하는 바가 적은 것으로 판단하였고 따라서 개발방법론이 미치는 효과를 결정하는 요인을 찾는데 실패하였다.

관리자와 개발자는 조직 문화의 유형에 따라 개발방법론의 효과의 방향과 정도를 그룹별로 서로 다르게 인식한다(Iivari and Huisman, 2007). 이는 개발방법론 도입을 고려하는 조직의 관리자는 조직 문화를 고려하여 도입 기술과 교육 대상을 선정해야 함을 시사한다.

4. 사례 연구

사례는 국내 증권 관련 업체의 정보시스템의 재개발 프로젝트이다. 대상 시스템은 10여 명의 주가지수 부서 담당자들이 주가지수를 설계 및 개발하고, 상장사의 주가에 변동을 야기할 요인이 발생하면 이를 주가지수에 반영되도록 조치하는 업무에 사용하고 있다.

프로젝트는 클라이언트-서버 방식에서 J2EE와 웹을 기반으로 하는 오픈 시스템으로 전환하기 위해서 폭포수형 프로세스와 컴포넌트 기반 개발(CBD) 기법을 채택하였다. 최초 프로젝트 계획 수립 단계에서는 MDD 도입을 고려하지 않았으나, 개발 프로세스 맞춤 작업과 분석과 설계의 모델링

<표 2> 사례 프로젝트 주요 특징

항목	내용
시스템 규모	약 100k LOC(Lines of Code : 프로그램 크기 측정단위)
개발 기간	9개월
총 투입 공수	66M/M(Man-month : 1개월 작업량)(8명)
개발 투입 공수	41M/M(5명)
분석/설계 기간	3.5개월
코딩/단위테스트 기간	2.5개월
통합테스트 및 이행	3.0개월
개발 프로세스	폭포수형 프로세스 컴포넌트 기반 개발
기술 플랫폼	J2EE, 웹, X-Internet
분석/설계 도구	UML 2.0(IBM RSA)

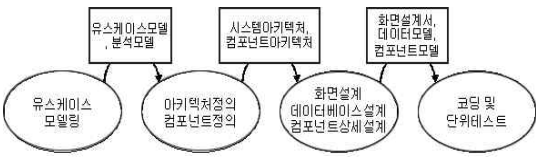
도구 선정 과정에서 MDD 기법 도입을 결정하였다. 프로젝트의 주요 특징을 정리하면 <표 2>와 같다.

프로젝트는 외주에 의한 개발이며 개발자는 모두 폭포수형 프로세스는 경험하였으나, CBD는 경험하지 못하였다. 그럼에도 불구하고 CBD 방법론을 채택한 이유는 발주사의 정보전략 계획에 따른 발주 시의 제약사항이었기 때문이다. 이에 CBD 수행 경험이 부족한 주사업자는 외부 CBD 컨설턴트를 영입하여 사업을 수주하였으며, 컨설턴트는 프로젝트 착수계획, 개발 프로세스 조정, 방법론 교육, 산출물 확인과 검증 작업에 참여하였다. 개발자는 각 단계 수행 직전에 과업 수행을 위한 CBD 기술교육을 받아 기법을 숙지한 후 개별 과업을 수행하였다.

연구자는 CBD 컨설턴트로 프로젝트 착수부터 코딩 및 단위테스트 단계까지 참여하였다. 생산성 측정을 위해 코딩 단계에서 구현된 소스코드의 LOC를 주 3회 측정하여 코드의 단순 증가분 뿐만 아니라, 측정구간의 코드 형상을 추적하여 삭제하고 새로 추가한 코드 분량도 고려하였다. 또한 품질 측면의 영향과 방법론으로써 MDD에 대한 효과를 파악하기 위해서 관리자와 개발자들에 대한 심층 면접을 수행하였다.

<표 3> ISO/IEC 12207와 사례 개발 프로세스의 비교

ISO/IEC 12207	사례 개발 프로세스
Requirements Analysis	유스케이스 모델링
Architectural Design	아키텍처 정의, 컴포넌트 정의
Detailed Design	화면설계, 데이터설계, 컴포넌트 상세설계
Coding and Testing	코딩 및 단위테스트
Integration	통합 테스트
Qualification Testing	인수 테스트



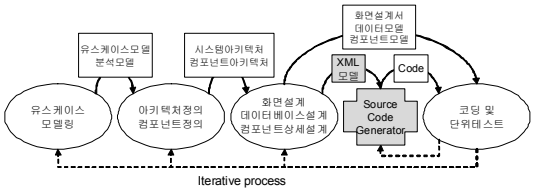
<그림 4> MDD 도입 이전 개발 프로세스

4.1 MDD 기반 개발 프로세스 적용

사례 프로젝트 주사업자는 자체 CBD 방법론을 보유하고 있었으며, 이를 ISO/IEC 12207 개발 프로세스에 대응시키면 <표 3>과 같다.

<그림 4>는 <그림 1>에 MDD 도입 이전의 사례 개발 프로세스의 개발활동을 대응하여 나타낸 것이다. 그림에서 보듯이 사례 프로젝트는 반복 프로세스를 공식적으로 적용하지 않았다.

시스템 분석 단계에서 CBD 컨설턴트는 프레임워크 기반의 패턴화된 시스템 아키텍처를 분석하여 시스템 개발 생산성과 품질 향상을 위해 MDD 기법 도입을 검토하였다. 시스템 아키텍처, 지원도구에 대한 기술지원 가능성과 일정을 고려하여 화



<그림 5> MDD를 도입한 개발 프로세스

면과 데이터를 제외한 컴포넌트의 설계에서 소스코드를 생성하는 단계에 한정하여 MDD 기법을 적용하기로 결정하였다. 화면 개발의 경우는 X-Internet 도구를 이용하여 WYSWYG[3] 방식으로 기본적인 소스코드 생성이 가능하고. 데이터베이스 개발의 경우, 전용 모델링 도구를 통해 대상 DBMS에 물리 데이터베이스를 생성할 수 있기 때문에 범위에서 제외되었고, 컴포넌트 개발에는 UML 2.0 CASE 도구를 이용하여 MDD를 적용하기로 하였다. MDD 도입으로 실행 가능한 소스코드를 완벽하게 생성하려면 정형화 기법(formal method)을 이용하여 특정 플랫폼에 종속되지 않으면서도, 프로그램의 의미(semantics)가 명시적으로 표현된 PIM을 설계해야 한다. 그러나, Baker et al.(2005)이 지적한 것처럼 MDD 도입 시 작업자의 MDD 프로세스 및 요소 기술에 대한 경험 부족으로 인해 컴포넌트 개발 프로세스를 <그림 3>과 같이 PIM에서 소스코드까지 생성되는 모든 과정을 자동화하지는 못하였다. <그림 5>는 사례 연구에 적용된 개발 프로세스를 나타낸다.

<그림 5>의 수정된 프로세스는 <그림 4>의 MDD 도입 이전 개발 프로세스에서 상세설계와 코딩 및 단위테스트 활동 사이에 소스코드 생성기를 이용한 자동화 단계를 추가하여 프로그램 작성의 생산성과 품질을 향상시키는데 중점을 두고 있다. 소스코드 생성을 위해서 EMF(Eclipse Modeling Framework)의 JET(Java Emitter Template) 기술을 도입하였다. JET는 소스코드 생성 도구로, JSP와 유사한 문법을 사용하여 생성하려는 소스코드 템플릿을 쉽게 작성할 수 있다(Popma, 2003). 소스코드 생성기로 소스코드를 자동생성하기 위해 필요한 입력 자료는 XML 설계 모델과 생성될 소스코드 템플릿이다. XML 모델은 설계된 컴포넌트와 관계로 표현되는 기능 측면의 컴포넌트 아키텍처 정보로, 소스코드 템플릿은 애플리케이션의 아키텍처 정보로 구성되며, 이 두 정보를 결합하여 개별 컴포넌트의 소스코드를 생성한다. 템플릿 개발자는 시스템 아키텍처를 구성하는 요소의 표준 소스코드를 강제하거나, 반복적인 소스코드 영역을 재사용하도록 할 수 있다. 이렇게 생성된 소스코드는 완전한 업무 로직을 표현한 것은 아니므로, 개발자가 설계 산출물을 참조하여 추가로 필요한 업무 로직을 구현하였다.

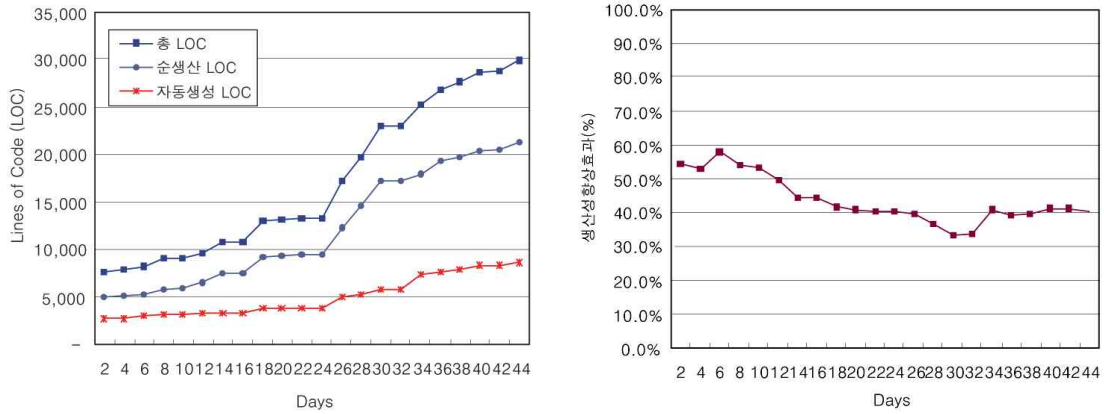
4.2 사례 연구 결과

MDD 적용 효과는 일반적으로 기대되는 생산성 제고와 품질 개선 측면에서 분석하였다.

4.2.1 생산성 향상

컴포넌트 개발에 한정하여 적용한 MDD 프로세스가 전체 개발 생산성에 미친 효과를 측정하였다. 먼저, 화면 개발과 컴포넌트 개발의 비중을 파악하기 위하여, 동일한 기능을 가진 프로그램 개발에 필요한 소스코드의 크기(LOC)를 컨설팅 통계자료를 (QSM, 2005) 이용하여 비교한 결과, 약 4 : 6 정도가 적절한 것으로 파악되었다. 이는 전체 프로그래밍 작업량의 60%가 컴포넌트 개발에 투입된 것을 의미한다. 또한, 개발자는 일일 평균 9.7시간으로 작업하여 21% 정도의 초과근무가 있는 것으

3) WYSWYG : What You See What You Get.



<그림 6> MDD 프로세스 생산성 효과

로 파악되었다. 이는 계획된 개발 인력의 21%에 해당하는 추가 인력이 투입된 것과 동일한 효과이다.

측정된 LOC 자료를 최초 소스코드 생성기로 생성한 소스코드 LOC와 비교하여, 자동 생성된 소스코드의 비율과 생산성에 미친 영향을 분석하였다. <그림 6>은 소스코드 LOC와 MDD 프로세스의 생산성 효과를 시계열로 나타낸 것이다.

<그림 6>을 보면 자동으로 생성된 프로그램에 최초의 추가 프로그램 작성이 시작되는 시점에는 자동 생성된 소스코드의 비중이 높아 생산성에 기여하는 바가 컸으나, 개발이 진행되면서 자동 생성된 소스코드의 상대적 비중이 낮아지고, 생산성 기여도도 떨어지는 것을 알 수 있다. 이는 요구사항 변경, 설계 변경, 디버깅 등에 의해 작업자가 프로그램을 작성하는 분량이 증가하게 되면서 자동 생성된 소스코드의 상대적 비중이 낮아지고, 생산성 기여도도 떨어지고 있음을 의미한다. 그러나 생산성 효과의 시계열에 나타난 것처럼 일정 기간(개발 시작 후 34일)이 경과한 후에는 생산성 기여도가 일정하게 나타났다. 코딩 완료시점의 생산성 효과는 약 40% 정도이며, 작업량으로 환산하면 약 2.4M/M(전체 코딩 작업량의 24%)의 절감효과가 있는 것으로 나타났다.

4.2.2 품질 개선

자동으로 생성된 소스코드는 개발자가 텍스트와 그림으로 표현된 설계 모델을 참조하여 작성한 소스코드와 비교할 때, 명명규칙, 코딩 표준과 같은 각종 표준을 일관성 있게 준수하였으며, 프로그램의 가독성이 높아지고 운영단계에는 유지보수가 용이할 것으로 기대할 수 있었다. 또한, 프로토타입 결과물이나 다른 프로그램의 단순 복사로 인해 유입되는 불필요한 로직이나, 오타를 예방함으로써, 수작업으로 인한 오류의 유입을 줄일 수 있었다.

이렇게 불필요한 소스코드 유입 방지를 통한 경량화와 오류의 원천적인 제거는 추가의 투입공수 절감을 의미한다. 소프트웨어 규모 산정을 위한 COCOMO 모델에 따르면 소프트웨어 개발에 필요한 노력(effort)은 식 (1)과 같이 소프트웨어 규모의 멱승에 비례한다(Royce, 1998).

$$Effort = 3.2\ EAF\ (Size)^{p} \qquad (1)$$

여기서, Effort는 투입공수(M/M), EAF는 15가지의 조절 요인(factor), Size는 개발규모(LOC)를 의미한다. 마지막으로 P는 개발 프로세스의 규모의 경제

를 특징짓는 지수 값으로, 재작업과 폐기작업, 관료적 지연 및 과도한 의사소통 비용 등의 불필요한 작업과 기타 프로세스의 효율성에 의해 결정된다. 문헌상 1.05~1.2의 값을 가지며 당연히 효율적인 프로세스일수록 P 값이 적어진다. 식 (1)을 통해 알 수 있듯이 개발규모를 줄임으로써 투입공수를 절감하는 효과도 있지만, 프로세스의 품질(효율성)을 높임으로써 더 큰 투입공수의 절감 효과를 얻을 수 있다.

일반적으로 개발자의 숙련도는 생산성과 품질에 영향을 크게 미친다(Royce, 1998). 소스코드를 자동으로 생성하여 숙련도에 따른 개발자간 편차를 제거함으로써 소스코드 품질을 상향 평준화하고, 소스코드의 검사(inspection) 작업을 생략하거나 단순화하여 품질검토와 검증을 위한 작업량을 줄일 수 있었다.

주사업자의 설계 관행은 시스템 아키텍처를 설계에 반영하지 않아서 설계 모델과 소스코드 사이의 추적성이 결여되었다. 그러나, 소스코드 자동생성 단계를 도입함으로써, 설계자는 아키텍처를 감안한 모델을 설계하고, 아키텍처 모델이 소스코드 생성기에 변환규칙으로 구현되어 설계 모델과 소스코드 간의 추적성이 확보되는 효과가 있었다.

5. MDE 도입에 관한 고려사항과 교훈

본 장에서는 MDE를 도입하기 전에 고려해야 할 사항을 논의하고 사례연구에서 얻은 교훈을 토의 한다.

5.1 MDE를 도입할 때 고려사항

5.1.1 MDE 지원도구

MDE 지원도구는 생산성에 커다란 영향을 미친다. 연구자들이 제시한 MDE 지원도구를 선택할 때 고려해야 할 사항을 정리하면 <표 4>와 같다.

<표 4> MDE 지원도구 선택시 고려사항

고려 사항	관련 연구
모델의 실행가능성 (executability)	Selic, 2003; Staron, 2006
모델 확장성 (scalability)	Selic, 2003; Baker et al., 2005
모델링 언어의 확장성	Staron et al., 2004; Baker et al., 2005
변환된 모델간의 추적성	Staron et al., 2004
기존 모델의 마이그레이션	Baker et al., 2005
생성된 소스코드의 성능	Selic, 2003; Baker et al., 2005
다른 도구 및 기존 환경과의 통합	Selic, 2003; Staron et al., 2004; Baker et al., 2005; Shirtz et al., 2007

모델의 실행가능성은 MDE 도구가 모델의 시뮬레이션과 테스트를 수행할 수 있는 기능과 환경을 제공해서 소스코드가 아닌 모델의 실행으로 구축되는 정보시스템의 요구사항 만족 여부를 검증할 수 있어야 함을 의미한다. 모델의 확장성은 대규모의 정보시스템을 모델링 하더라도 도구의 성능이 보장되어야 함을 의미한다. 반면 모델링 언어의 확장성은 다양한 도메인의 정보시스템 모델을 작성하기 위해 모델링 언어의 문법 및 의미에 대한 확장 메커니즘이 요구됨을 의미한다. MDE가 일련의 모델 변환을 통한 개발 프로세스인 만큼, 도구를 이용하여 모델 요소의 변화과정을 추적할 수 있어야 한다. 또한, MDE 도입 이전에 보유한 모델을 도구가 지원하는 모델로 변환하여 유지보수 및 개선을 위해 MDE 도입이 가능해야 한다. 모델 변환과정의 최종 단계에 생성된 소스코드의 성능은 MDE의 생산성 효과에 당위성을 부여하기 위해 반드시 만족할만한 수준이어야 한다. 마지막으로 MDE 도구가 다른 개발 도구와 기존 시스템 환경에 통합이 어려우면 MDE 효과를 얻기 위한 초기

<표 5> MDE 기술지식 고려사항

고려사항	관련 연구
모델 및 모델링 언어 지식	Baker et al., 2005; Staron, 2006
모델링 도구 지식	Baker et al., 2005; Staron, 2006; Shirtz et al., 2007
모델 변환 지식	Baker et al., 2005; Staron, 2006
프로파일 및 메타 언어 체계 지식	Staron et al., 2004
소프트웨어 아키텍처 지식	Baker et al., 2005
MDD 프로세스 지식	Baker et al., 2005; Staron, 2006; Shirtz et al., 2007

투자가 과도하여 오히려 역효과를 보일 것이다.

5.1.2 MDE 기술지식

MDD 도입에 있어 지원도구뿐만 아니라, 필요한 관련 MDE 기술지식을 고려해야 한다. 조직에서 도입하려는 MDE 수준에 따라 차이는 있겠지만, 기존 연구에서 제시된 MDE 기술지식에 대한 고려사항은 <표 5>와 같다.

MDE는 모델 주도적 개발방법인 만큼 모델 생성, 실행, 변환 및 테스트 등의 모델과 모델링 언어에 대한 지식이 요구된다. 또한, 수작업이 아닌 도구를 통한 연속적인 모델링 작업이므로, 모델링 도구를 능숙하게 다룰 수 있는 기술적 지식이 필요하다. 특히 모델 변환을 설계하는 기술자는 모델링 언어의 확장 체계에 대한 지식을 갖고 있어야 하며, 여기에는 특정 도메인에 특화된 프로파일 작성 지식과 모델링 언어의 확장을 위한 메타언어에 대한 지식이 포함된다. 더불어 모델 변환은 대상 소프트웨어 아키텍처와 변환 단계에 적용될 아키텍처 스타일에 대한 이해를 필요로 한다. 마지막으로, MDD 프로세스를 이해하고, 성공적인 적용을 위해 프로세스를 조정할 수 있는 지식도 필요하다.

<표 6> MDE 도입 시 조직적 관점에서의 고려사항

고려사항	관련 연구
기술조직의 문화	Baker et al., 2005 Iivari and Huisman, 2007
기술조직 구성	Baker et al., 2005; Shirtz et al., 2007
기술인력 양성 및 교육	Baker et al., 2005; Staron, 2006; Shirtz et al., 2007
기술도입 목표 및 전략	Baker et al., 2005; Staron, 2006; Shirtz et al., 2007
기술도입 성과 측정	Shirtz et al., 2007

5.1.3 조직적 관점

MDE를 도입할 때, 기술적 측면 이상으로 조직적 측면의 고려사항이 중요하다. 필요한 기술지식과 도구를 모두 갖추었다 할지라도 조직 차원에서 준비가 되어 있지 않으면 MDE 도입은 무의미한 도전이 될 수 있다. 많은 사례연구에서도 이점을 강조하고 있으며, 주요한 조직적 차원의 고려사항을 살펴보면 <표 6>과 같다.

조직의 문화를 고려하지 않은 MDE와 같은 신기술 도입은 성공을 보장받을 수 없다. Iivari and Huisman(2007)의 개발방법론 도입 효과에 조직문화가 미치는 영향에 대한 연구는 개발방법론으로써의 MDE 도입에 시사하는 바가 크다. MDD 프로세스는 기존의 개발 프로세스와는 많은 차이가 있는 만큼, 기술조직의 변화를 초래한다. 새로운 조직구조의 설계와 변화관리가 요구된다. 기술인력의 교육과 양성은 혁신을 꾀하는 조직에 반드시 필요하며, 만일 내부에서 주도할 수 있는 기술인력이 부족할 경우 외부의 지원과 인력충원이 있어야 한다. 위에서 열거한 항목을 모두 고려하였다 할지라도, 조직의 목표에 일치(aligned)하지 않는 기술도입은 오히려 조직의 목표 달성을 저해할 수 있으므로, 기술도입의 목적 및 목표와 이에 따른 전략적 접근방법이 사전에 모색되어야 한다. 많은 기존 사례연구

에서 MDE의 점진적, 단계적 도입을 권고하고 있다. 이는, 조직의 현재 상태를 분석하여 수립한 기술 도입 목적과 목표를 단계별로 달성할 수 있는 도입 전략을 수립해야 함을 의미한다. 마지막으로 이러한 전략적 기술도입의 조직적 성과를 측정할 수 있어야 한다. 도입 목표에 대한 기술도입의 효과를 측정할 수 있는 지표를 설정하고, 지속적으로 측정하여 혁신적인 기술을 도입하는데 따르는 시행착오를 최소화해야 한다.

5.2 연구사례의 교훈

최근 MDE 도구는 빠르게 개선되고 있으며, 개별 MDA 표준을 많은 부분 준수하고 있으나, 모델간 변환기술과 설계 모델에 의미(semantics) 구현을 위한 도구 지원은 아직은 부족하다. 특히, 변환기술은 대부분 제공업체 전유의 기술이 사용되고 있다. 따라서 이 연구에서는 개발 프로세스에 MDD를 전면적으로 적용하기 이전에 도구가 현재 지원할 수 있는 수준에서 MDD 기법을 적용하는 단계적 도입을 권고한다. 사례에서는 완벽한 지원이 가능한 MDA 표준 기술을 우선적으로 적용하였다.

- 교훈 1 : MDE 기술은 개발 프로세스의 가능한 단계부터 점증적으로 적용한다.
- 교훈 2 : MDE를 지원하는 도구를 선정할 때, 전유의(proprietary) 기술 기반의 도구보다, MDE 기술 표준을 지원하는 도구를 선정한다.

본 사례 프로젝트 참여자 모두 MDD를 경험해보지 못하였다. 또한, 프로젝트 조직 구성이 동일한 제품이나 서비스를 지속적으로 개발하거나, 동일 사업을 장기간 수행하기 위한 것이 아니었기 때문에, 체계적 제도를 통한 기술 습득이나 경험 축적을 우선적으로 고려할 수는 없었다. 이에, MDE 기술 및 프로세스를 교육하고 이끌어나갈 컨설턴트를 영입하였다.

- 교훈 3 : MDE를 처음 적용하거나, 외주로 정보시스템을 개발하는 경우 MDE 기술자문 조직이나 인력을 참여시켜야 한다.

사례 프로젝트의 경우 적용하는 기술에 대한 충분한 사전 교육이 불가능한 상황이었으므로, 컨설턴트는 각 개발단계 수행 직전에 JIT(Just-In-Time) 방식의 기술 교육을 실시하여 MDD 프로세스와 지원도구를 처음 경험함에도 불구하고 작업자들이 적극적으로 참여하고 빠른 기간에 적응할 수 있었다. 그러나, 관리자의 경우 개발 활동을 직접 수행하지 않기 때문에, 프로세스에 대한 이해도가 작업자에 비해 떨어졌으며, 과거에 경험한 다른 기술과의 비교를 통해 개념적으로 신기술을 이해하려고 했기 때문에 용어의 명확한 정의가 필요하였다. 즉, 프로젝트 초기에 새로운 기술을 이해하는 데 필요한 용어를 정의하여 공유할 것을 제안한다.

- 교훈 4 : MDE 기술에 대한 조직의 저항을 줄이기 위해 사례 중심의 교육이 적시에 제공되어야 한다.
- 교훈 5 : 성공적인 MDE 도입을 위해 MDE 용어 및 개념의 이해와 공유가 중요하다.

5.2.1 조직 문화에 따른 새로운 기술 도입에 대한 인식

Iivari and Huisman(2007)은 위계적인 문화를 가진 조직의 개발자와 관리자는 개발 방법론이 생산에 도움이 되는 기술이라고 인식하는 반면, 합리적인 문화를 가진 조직의 관리자는 개발 방법론이 오히려 방해가 되는 것으로 인식한다고 하였다.

연구사례의 대상 조직은 발주업체의 정보시스템 개발과 운영을 대부분 담당하고 있어, 발주업체에 종속적인 사업구조를 가지고 있으며, 개발 프로세스를 포함한 발주업체의 기술적 요구사항의 대부분을 수용하고 있어, 안정 지향적이고 내부 환경에 초점을 맞춘, 매우 위계적인 조직 문화를 가지고 있다. 사례에서는 Iivari and Huisman(2007)의 연구결과에서와 같이 개발자의 경우 개발 프로세스에 긍정적인 생각을 보였으며, 관리자는 용어정의의 혼선으로 개발자에 비해 이해 정도가 낮았지만, 새로운 개발 프로세스를 비교적 긍정적으로 수용하였다.

- 교훈 6 : MDD 프로세스는 적용대상 조직의 문화를 고려하여 절차와 기법을 조정할 수 있어야 한다.

6. 결론 및 향후 과제

이 연구에서는 MDE 기술을 실제 개발 프로젝트에 적용하여 MDE 도입 효과를 분석하고, MDE를 도입할 때 고려해야 할 사항과 사례연구를 통해 얻은 교훈을 논하였다. 이론적인 MDD 프로세스는 업무 지식을 중심으로 분석/설계한 PIM으로 PSM과 소스코드를 자동 생성하는 것이지만, 본 연구에서는 설계 모델까지는 기존 모델링 프로세스를 적용하고 소스코드만을 MDE 도구로 자동 생성한 후 MDE 도입 효과를 정량적, 정성적으로 분석하였다. 정량적인 효과는 작업량의 축소로 생산성이 향상되는 것으로 나타났으며, 정성적으로는 시스템 산출물의 전반적인 품질이 향상되는 것으로 나타났다. 또한, MDE를 적용하는 과정에 나타난 몇 가지 교훈을 제시하였다.

본 연구는 단일 사례를 통해 측정지표를 도출하고 성과는 측정하였으나 MDE를 도입하지 않았을 경우와 비교하지 못했다. 그러나, 대부분의 현장연구의 경우 동일한 한계를 가지고 있으며, 이는 실험연구를 통해 기술 도입 이전과 이후의 비교가 가능할 것이다. 둘째, 정보시스템의 규모를 자동으로 생성된 소스코드의 양과 수작업을 통해 생성된 양으로 생산성에 대한 MDE 효과를 측정하였기 때문에, CASE 도구로 자동 생성한 소스코드의 양으로 측정한 규모의 타당성에 의문을 제기할 수 있다. 하지만, 연구에 적용한 소스코드 자동생성은 아키텍처 검증을 통해 얻어진 표준 소스코드를 자동 생성된 소스코드의 템플릿으로 사용함으로써, 생성된 소스코드의 양적 팽창 및 성능의 문제는 대부분 해결되었다. 셋째, 전체 개발 프로세스에서 MDE 기술이 소스코드 생성 부분에 한정하여 적용되었기 때문에 전반적인 MDE 도입 효과를 예측하기 어렵다. 넷째, 사례연구를 위한 정보시스템의 규모가 작아서 MDE 도입의 타당성을 현장전문가에게 설득하기에는 호소력이 부족하였다. 마지막으로 다양한 효과를 측정하기 위한 데이터 수집이 불가능했다는 점에서 생산성 효과 측정의 편의(偏倚)가 존재할 수 있다.

위에서 열거한 이 연구의 한계를 극복하기 위해서는 MDE 도입 효과에 영향을 주는 요인을 탐색하기 위한 실험연구 및 사례연구가 요구되며, 이를 통해 MDE 효과 측정을 위한 모델 설계와 실증 연구가 이뤄져야 할 것이다.

참고문헌

김학인, 최오훈, “MDA 기반 모델 변환 기법을 이용한 컴포넌트 생산성 향상에 대한 사례연구”, *한국정보과학회 2004년도 가을 학술대회*, 한국정보과학회 (2004), 463~465.

윤정모, 김치호, "MDA 기반 학사관리 프로세스 유효성 분석", *정보처리학회논문지*, 15권 2호 (2008), 187~196.

Atkinson, C. and T. Kühne, "Model-Driven Development : A Metamodeling Foundation", *IEEE Software*, Vol.20, No.5(2003), 36~41.

Baker, P., S. Loh, and F. Weil, "Model-Driven Engineering in a Large Industrial Context - Motorola Case Study", *Lecture Notes in Computer Science*, Vol.37, No.13(2005), 476~491.

Baskerville, R., J. Travis, and D. Truex, "Systems without Method: The Impact of New Technologies on Information Systems Development Projects", *IFIP WG8.2 Working Conference on the Impact of Computer Supported Technologies in Information Systems Development*, Vol.A-8(1992), 241~269.

Bertrand, D., *Model-Driven Architecture Case Study*, CGI Group, 2003.

Boehm, B., "A View of 20th and 21st Century Software Engineering", *Proceedings of ICSE'06*, (2003), 12~29.

Boehm, B., *Software engineering economics*, Prentice Hall, 1981.

Brown, A. W., J. Conallen, and D. Tropeano, "Introduction : Models, Modeling, and Model-Driven Architecture (MDA)", *Model-Driven Software Development*, Springer(2005), 1~16.

Chrissis, M. B., M. Konrad and S. Shrum, *CMMI : Guidelines for Process Integration and Product Improvement*, Addison-Wesley, 2006.

Fitzgerald, B., "An Empirical Investigation into the Adoption of Systems Development Methodologies", *Information and Management*, Vol.34, No.6(1998), 317~328.

France, R. and B. Rumpe, "Model-driven Development of Complex Software : A Research Roadmap", *International Conference on Software Engineering, 2007 Future of Software Engineering, IEEE Computer Society*(2007), 37~54.

France, R. B., S. Ghosh, and T. Dinh-Trong, "Model-Driven Development Using UML 2.0 : Promises and Pitfalls", *IEEE Computer*, Vol.39, No.2(2006), 59~66.

Frankel, D. S., *Model Driven Architecture-Applying MDA to Enterprise Computing*, Wiley, 2003.

Fujita, H. and I. Zualkernan, "Evaluating Software Development Methodologies Based on their Practices and Promises", *New Trends in Software Methodologies, Tools and Techniques*(2008), 14.

Huisman, M. and J. Iivari, "Deployment of systems development methodologies : Perceptual congruence between IS managers and systems developers", *Information and management*, Vol.43, No.1(2006), 29~49.

Iivari, J. and M. Huisman, "The Relationship Between Organizational Culture And The Deployment of Systems Development Methodologies", *MIS Quarterly*, Vol.31, No.1(2007), 35~58.

ISO/IEC, *ISO/IEC 12207 : 1995, Standard for Information Technology-Software life cycle processes*, 1995.

Kent, S., "Model Driven Engineering", *Lecture Notes in Computer Science*, 2335(2002), 286~298.

Kleppe, A., J. Warmer, and W. Bast, *MDA Explained : The Practice and Promise of the Model Driven Architecture*, Addison Wesley, 2003.

Laudon, K. and J. Laudon, *Management Information Systems: Managing the Digital Firm*, Pearson Prentice Hall, New Jersey, 2006.

Mellor, S. J., K. Scott, A. Uhl, and D. Weise, *MDA*

Distilled: Principles of Model-Driven Architecture, Addison Wesley, 2004.

Middleware Company, "Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA)-Approach : A Productivity Analysis", *TMC Research Report*, 2003.

MODELWARE, *Industrial ROI, Assessment, and Feedback-Master Document. Revision*, Vol.2, No.2, 2006.

Mohagheghi, P. and V. Dehlen, "Where Is the Proof?-A Review of Experiences from Applying MDE in Industry", *Lecture Notes in Computer Science*, 5095(2008), 432~444.

OMG, *Model Driven Architecture (MDA)*, 2001.

Popma, R., *JET Tutorial Part 1 (Introduction to JET)* from http://www.eclipse.org/articles/Article-JET/jet_tutorial1.html, 2003.

QSM, *QSM Function Point Programming Languages Table*, from http://www.qsm.com/FPGearing.html, 2005.

Rios, E., T. Bozheva, A. Bediaga, and N. Guilloreau, "MDD Maturity Model : A Roadmap for Introducing Model-Driven Development", *Lecture Notes in Computer Science*, 4066(2006), 78~89.

Royce, W., *Software Project Management-A Unified Framework*, Addison-Wesley, 1998.

Selic, B., "The Pragmatics of Model-Driven Development", *IEEE Software*, Vol.20, No.5(2003), 19~25.

Shirtz, D., M. Kazakov, and Y. Shaham-Gafni, "Adopting Model Driven Development in a Large Financial Organization", *Lecture Notes in Computer Science*, 4530(2007), 172~183.

Staron, M., "Adopting Model Driven Software Development in Industry-A Case Study at Two Companies", *Lecture Notes in Computer Science,* 4199(2006), 57-72.

Staron, M., L. Kuzniarz and L. Wallin, "Case study on a process of industrial MDA realization: determinants of effectiveness", *Nordic Journal of Computing*, Vol.11, No.3(2004), 254~278.

The Standish Group, *CHAOS : THE STANDISH GROUP REPORT*, 1995.

Weigert, T. and F. Weil, "Practical experiences in using model-driven engineering to develop trustworthy computing systems", *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2006.

Yourdon, E., *Death March: The complete software developer's guide to surviving mission impossible projects*, Prentice Hall PTR Upper Saddle River, New Jersey, 1999.

Abstract

Effectiveness of Model-Driven Development Process : Case Study

Sungwook Moon · Saneung Hong

Research on how to develop information systems efficiently and effectively since early 1960s has resulted in many techniques, methods and methodologies. Only a few of them, however, have been successfully practiced in the field. Model-Driven Development(MDD) is an innovative approach emphasizing the central role of model for development activities, attracting many practitioners' attention as well as researchers'. As MDD matures, many researchers have been trying to establish the evidence of its effectiveness. But many of them only suggest lessons learned or report limited evidence of effectiveness based on isolated case studies. This paper reports the state of the art of Model-Driven Engineering(MDE) and its major issues. We reviewed a number of papers and collected the conceptual definitions of MDE effectiveness from the technological and organizational perspectives. A case study in which MDD technology was adopted has been performed in order to measure the effectiveness of MDD quantitatively and qualitatively. This paper also analyzes and summarizes key considerations and lessons learned for IT organizations to adopt MDE successfully from the case study.

Key Words : MDE, MDA, MDD, Software Development Process Effectiveness, Case Study

* School of Business Administration, University of Seoul

저 자 소 개

문성욱

1996년 연세대학교와 1998년 한국과학기술원에서 각각 공학학사와 공학석사 학위를 취득하였다. 현재는 동양시스템즈㈜ 기술표준화팀에 재직중이면서 서울시립대학교 경영학과에서 MIS 전공으로 박사과정을 이수중이다. 다수의 정보시스템 개발 프로젝트 관리 및 컴포넌트 기반 개발(CBD) 컨설팅을 수행하였다. 관심분야는 컴포넌트 기반 개발(CBD), 모델주도적 개발(MDD) 및 소프트웨어 프로세스 자동화이다.

홍사능

서울시립대학교 농업경영학과를 졸업하고 University of Texas at Austin에서 경영학박사 학위를 받았다. 대학을 졸업한 후에 한국산업은행에서 근무하였으며 귀국 후에 한국통신기술 신재무팀장, 금융감독원 정보관리국장을 역임하였다. 한국정보시스템학회와 한국경영교육학회 부회장으로 활동하였으며 서울시립대학교 경영학부 교수로 재직하고 있다. Decision Support Systems 등에 논문을 게재하였으며 주요 관심분야는 정보시스템 개발, 정보기술 관리, 프로젝트 관리이다.