모델기반아키텍처(MDA) 환경에서의 레거시 시스템 통합 전략 설계

김 희 숙*, 이 서 정**, 박 재 년*
*숙명여자대학교 컴퓨터과학과, **한국해양대학교 IT공학부

Design of Integration Technology for Legacy System on Model-Driven Architecture

HeeSook Kim*, SeoJeong Lee**, JaeNyun Park*
*Dept. of Computer Science, Sookmyung Women's University
**Division of Information Technology, Korea Maritime University
E-mail: *{hskim11, jnpark}@sookmyung.ac.kr, **sjlee@bada.hhu.ac.kr

요 약

기존의 레거시 시스템은 개발과정에 있어서 이미 많은 시간과 경험과 지식을 갖고 있기 때문에 조직의 핵심 가치를 지닌 재사용 자산으로 활용할 수 있는 의미있는 자산이다. 그러므로 레거시 시스템에서 사용해 왔던 가치있는 자산들을 웹을 기반으로 한 e-비즈니스 환경에 적용시키고, 다양한 플랫폼에서 사용하기 위한 필요성이 증대되었다. 변화하는 환경에 따라 레거시 시스템을 현대화(Modernization)하는 방법 중의 하나로 현재 시스템에서 운용되고 있는 것을 OMG에서 제안한 모델기반아키텍처(MDA)를 사용하여 여러 플랫폼을 쉽게 지원하고 모델기반의 통합을 하고자 한다. MDA 환경을 기반으로 기존의 레거시 시스템을 활용하기 위하여 기존의 소프트웨어를 직접 수정 없이 재사용하거나 PSM에서 PIM 변환을 하기 위하여 래퍼(wrapper를 사용한다. 본 논문에서는 이러한 래퍼의 사용에 따른 문제점을 분석하고, 요구에 따라 모델에 대한 접근방식을 다르게 사용하여 시스템의 개선상황에 유연하게 대처할 수 있도록 래퍼를 상황에 따라 적절하게 사용하는 혼합(hybrid) 방식을 적용한 개선전략을 제안하고자 한다.

1. 서론

오늘날의 조직은 변화하는 시장 상황에 보다 효과적으로 대응하기 위하여 이미 기존에 잘 사용되고 있는 비즈니스 모델과 그와 연관된 엔터프라이즈 정보 시스템을 재설계(re-engineer)해야 할뿐만 아니라, 빠른 속도로 성장중인 웹 기반의 e-비즈니스 환경에도 들어서야 한다.[1]. 엔터프라이즈 정보 시스템은 대개 규모가 크고, 이기종, 분산 환경이고, 진화하고, 임무결정적(mission-critical) 시스템이다[1].

레거시(Legacy) 시스템은 조직의 핵심 가치를 지닌 재사용 자산으로 활용할 수 있는 의미있는 자산이다. 이는 개발과정에 많은 시간과 축적된 경험과 지식을 투자했고 입증된 안정성과 신뢰성을 갖고 있다[2]. 기존의 레거시 시스템은 대부분 같은 플랫폼하에서 개발되었기 때문에 웹 기반, 분산 환경, 이기종 아키텍처, 컴포넌트 기반 등의 요즘 기술 시스템에는 잘 맞지 않는다. 그러므로 새로운 프로젝트를 추진하거나 기존의 시스템을 분산 환경으로 변경시킬

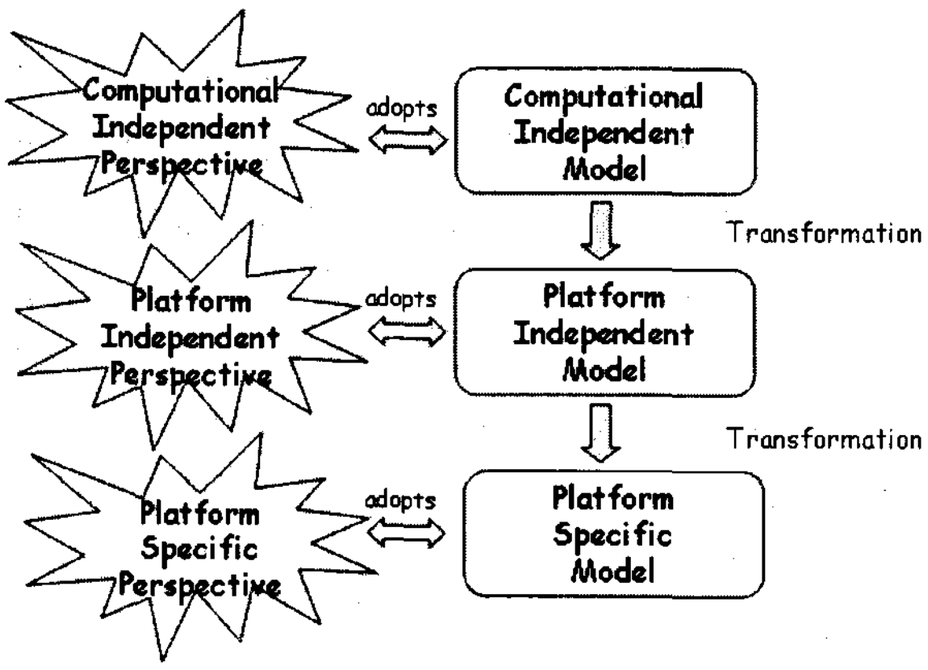
때 기존의 레거시 시스템을 충분히 재사용할 수 있음에도 불구하고 다시 새로운 시스템을 구축하는 경우가 많았다[2]. 따라서 이기종의 컴포넌트 개발 플랫폼 및 다양한 구현 기술의 발달로 이들간의 상호 운용성 및 통합을 지원하기 위하여 OMG에서 채택한 모델기반아키텍처(MDA: Model-Driven Architecture) 환경하에서 기존의 레거시 시스템에 대한 통합 전략 방법을 제안하고자 한다.

대부분의 소프트웨어 공학 접근 방법은 레거시 시스템을 통합하기 위하여 데이터 위주로 다루는데 비즈니스 프로세스의 지속적인 변화는 다루지 못하고 있다. 본 논문에서는 요구에 따라 모델에 대한 접근방식을 다르게 사용하여 시스템의 개선상황에 유연하게 대처할 수 있도록 기존 모델의 문제점을 해결 및 보완할 수 있는 전략을 제안하고자 한다.

2. 연구배경

2.1 모델 기반 아키텍처(MDA)

OMG는 객체 기술에 대한 사용을 증진시키고 표준안을 만들기 위해 제안한 컨소시엄으로써 OMA(Object Management Architecture)를 제안하였다. OMG에서 객체지향 소프트웨어 설계를 위해 다양하게 제시되었던 모델링 시스템을 표준화해 객체지향 모델링 언어 표준으로 UML을 제안하였다. 그리고 모델기반 아키텍처(MDA: Model Driven Architecture)는 OMG에서 UML과 OMA를 결합해 만들어낸 소프트웨어 아키텍처이다[3].



[그림 1] 모델기반 아키텍처(MDA)

MDA는 세가지 관점으로 명세하며 [그림 1]에 나타나있다[1]. 도메인을 명세하기 위한 CIM(Computation Independent Model), 플랫폼에 독립적인 모델인 PIM(Platform Independent Model), 플랫폼에 종속적인 모델인 PSM(Platform Dependent management) 세가지 관점으로 시스템을 명세한다[3][4].

MDA는 플랫폼 독립 모델(PIM: Platform Independent Model)과 플랫폼 종속 모델(PSM: Platform Specific Model)의 변환(transformation)과 적응(adopts)을 통해 여러 플랫폼을 쉽게 지원하고 코드 작성에 소요되는 시간을 줄일 수 있다[1].

MDA는 소프트웨어 개발 프레임워크로서 소프트웨어 개발 프로세스에서 모델을 강조한다. 모델은 다른 모델로 변환하므로 소프트웨어 개발의 자동화를 지원할 수 있다.

MDA는 모델의 자동화와 변환을 통해 여러 플랫폼을 쉽게 지원하고 플랫폼 독립성이 가능하므로 어떤 기능을 갖는 소프트웨어 부품이 다양한 플랫폼을 기반으로 상세 설계될 수 있고, 한번 작성된 모델은 비즈니스 핵심 자산(Core Assets)으로써 유사한 목적을 갖는 다른 시스템에 쉽게 이식할 수 있다. 이런 특징을 통해, 전체 모델의 재사용성이 높아질 수 있다.

MDA에서는 PIM과 PSM을 분리했기 때문에 PIM을 변경하지 않고도 기술 플랫폼의 변화나 요구사항 변화에 빨리 대처할 수 있다. 플랫폼 의존적인 시스템은 기존의 아키텍처가 새로운 요구사항을 만족시키지 못하는 경우에 많이 발생한다. MDA를 이용하면 기능과 아키텍처를 분리해서 정의하기 때문에 아키텍처의 변화가 있더라도 변환(trasformation)과정을 거쳐 구현과정으로 쉽게 진행할 수 있다[3][5].

MDA는 하나의 응용 프로그램을 목적으로 개발하던 기존의 방식을 벗어나, 특정업종에 적합한 아키텍처를 미리 정의해 놓고 이후 개발로 전환되는 방식을 제공한다. 즉, 기존의 코딩이나 디자인보다는 아키텍처 설계에 중심을 두게 되어 사용자 주도적인 방법으로 인정되고 있으며 향후 전 세계 산업계 전반에서 사용될 전망이다.

모델변환의 요구사항은 PIM에 적용해야 하는 경우도 있고, PSM에 적용해야 하는 경우도 있

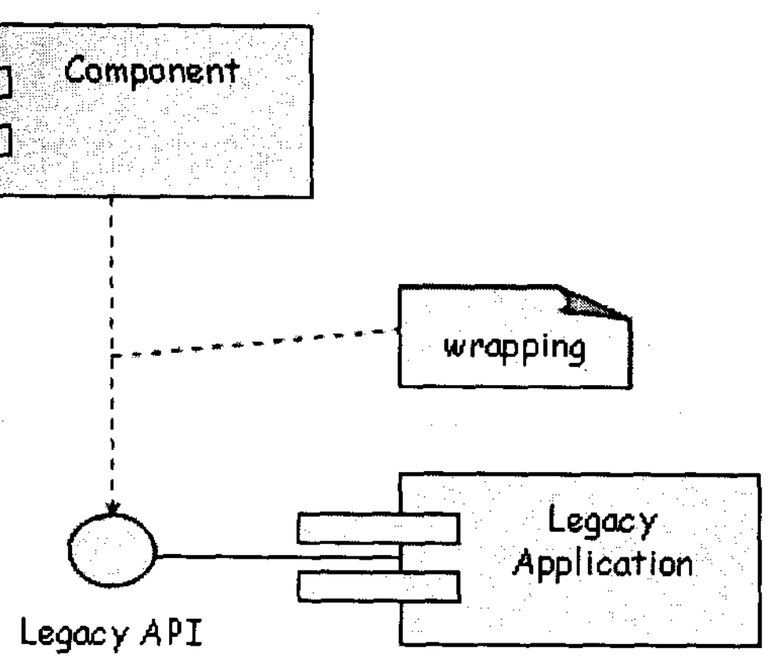
다. 두 경우 모두 연결된 PIM 또는 PSM에 변경 사항을 반영해야 한다. 즉, PIM사이의 변환, PSM사이의 변환, PIM에서 PSM으로의 변환 그리고 PSM에서 PIM으로의 변환이 상황에 따라 적절히 응용되어야 한다. 매핑하고 변환하는 과정을 통합이라고도 한다.

2.2 래퍼(Wrapper)

컴포넌트의 래퍼는 컴포넌트를 둘러싸고 있는 형태의 또 다른 컴포넌트나 클래스라고 말할 수 있다. 이러한 래퍼를 통하여 기존의 컴포넌트를 직접 수정 없이 재사용 하거나 기능의 변경, 성능의 개선 등을 이룰 수 있다[7].

컴포넌트 상호간의 호출시 컴포넌트는 컴포넌트를 직접 호출하는 것이 아니라 컴포넌트를 감싸고 있는 래퍼에게 호출을 요청하게 되고 래퍼는 컴포넌트를 대신하여 호출하고자 하는 컴포넌트의 래퍼에게 호출을 요청한다. 호출을 받은 컴포넌트의 래퍼는 실제의 컴포넌트에게 수행을 요청하고 이에 대한 응답을 받아 요청한 컴포넌트의 래퍼에게 이러한 결과를 되돌려 주며 래퍼는 호출을 요청한 컴포넌트가 인식할 수 있는 결과로 다시 가공하여 컨포넌트에게 결과를 알려주게 된다. 이러한 형태로써 컴포넌트 상호간의 이질적인 요소를 없애 기존의 컴포넌트와 새로운 컴포넌트간의 통합 시스템을 구성할 수 있는 것이다[8].

래퍼를 사용하여 코드의 수정 없이 기존 소프트웨어를 컴포넌트하여 사용할 수 있으며, 기존의 시스템을 래핑하기 위해서는 컴포넌트와 기존의 레거시 응용프로그램 사이에 레거시 API(Legacy API)가 있어 데이터와 기능의 불일치를 정제하는 역할을 한다. 다음[그림 2] 에 래퍼 프로세스를 나타내고 있다[7].



[그림 2] 래퍼 프로세스

래퍼를 사용한 응용프로그램은 인터페이스를 활용하여 다른 코드들이 자신의 서비스 기능을 호출하도록 허용하며 이를 네트워크 상의 모든 시스템에 제공한다. 컴포넌트는 그 내부 구조가 클라이언트에게는 감추어져 있는 객체이기 때문에 이러한 객체들은 레거시 시스템과의 통합을 위한 근간을 제공할 수 있다.

3. 레거시 시스템 통합 전략

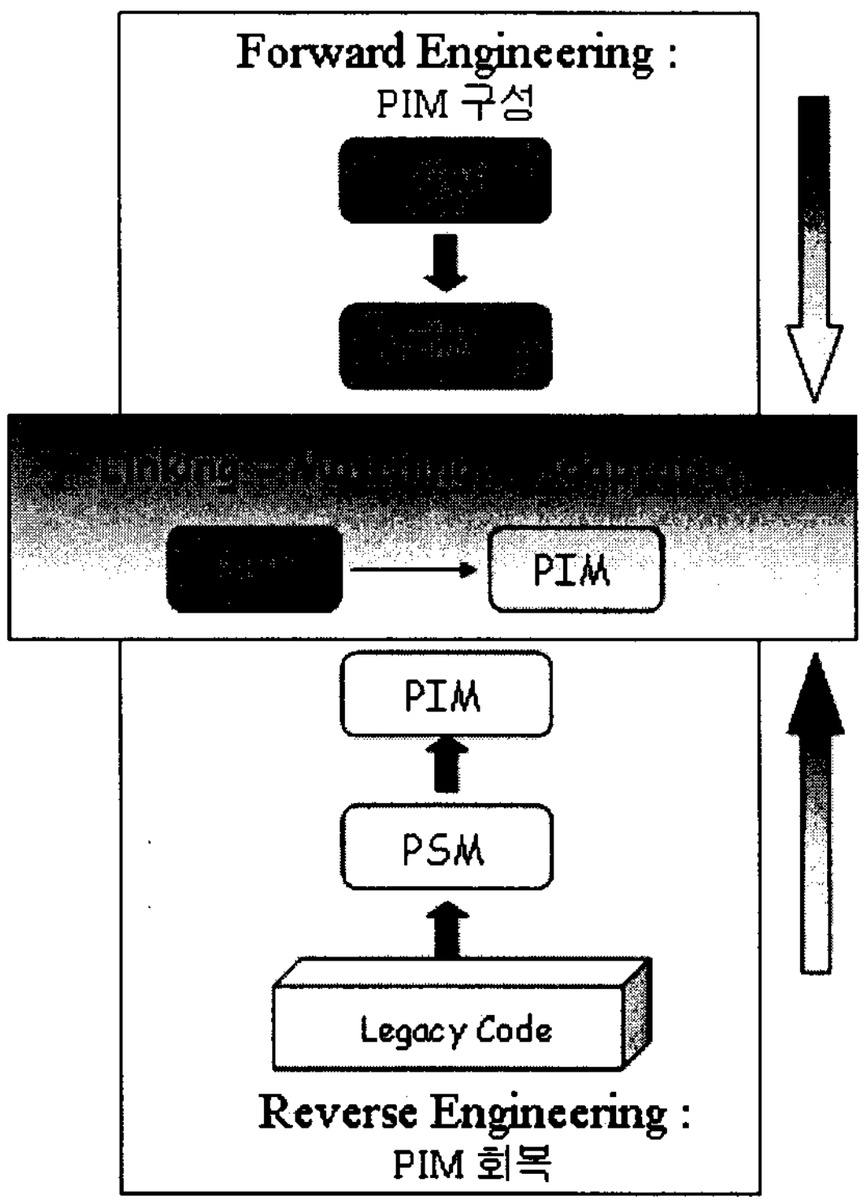
3.1 모델기반아키텍처에 방법론 적용

본 논문은 변화하는 환경에 따라 기존 레거시 시스템을 현대화(Modernization)하는 방법 중의 하나로 현재 시스템에서 운용되고 있는 것을 MDA 컴포넌트로 래핑(wrapping) 하는 방법을 사용하고자 한다.

레거시 시스템에서는 이미 존재하는 시스템에 대한 깊은 이해가 중요하다. 현재 시스템의 문제점뿐만 아니라 각 컴포넌트들과의 관계, 설계상의 문제점들을 일일이 파악하고 있어야 한다. 또한 새로 추가하는 기능이 존재하는 시스템의 구조와 조화를 이루도록 컴포넌트 인터페이스를 잘 설계하여야 한다.

이러한 사항을 적용한 방법론으로 순공학(Forward engineering)과 역공학(Reverse engineering)의 조합이 가능하다. [그림 3]에 나타나 있는 링킹 단계는 순공학과 역공학의 결과물을 비교하여 새로운 모델을 만드는 과정이다[1]. 이 방법론을 인터페이스 명세(Specification)에 적용시킬 수 있을 것이다. 인터페이스는 엔터프라이즈 모델과 레거시 시스템으로부터 상대적으로 저가의 비용으로 얻어질 수 있다. 순공학 단계에서는 CIM, 즉 도메인 명세와 유사하고 PIM으로 변환되어 진다. CIM은 엔터프라이즈 모델을 구성하고 비즈니스 프로세스와 컴포넌트 집합으로 표현되고 이 모델은 PIM으로 대응된다. 역공학 단계에서는 이와 반대로 레거시 데이터에 기초한 프로시저를 포함한 레거시 코드 수행문을 가지는 PSM에서 시작하여 추상적인 PIM으로 변환된다.

이와 같은 방법론에서는 래퍼를 사용하여 역공학을 통해 PSM에서 PIM을 변환시키는 방법을 사용하는데, 실제 구현사례에 있어서는 적절한 래퍼의 사용이 성능 및 레거시 시스템의 재사용도에 중요하게 작용될 것이다. 본 논문에



[그림 3] Forward Engineering과 Reverse Engineering의 조합

서는 이러한 래퍼의 사용에 따른 문제점을 분석하고, 래퍼를 상황에 따라 적절하게 사용하는 혼합(hybrid) 방식을 적용한 개선전략을 제안하고자 한다.

3.2 문제점 분석

(1) 과도한 인터페이스 구현

기존 컴포넌트는 그대로 두고 래퍼를 구현하는 작업은 기존의 컴포넌트를 쉽게 조절하여 새로운 컴포넌트를 생성하는 것 보다 경우에 따라서는 더욱 복잡해 질 수 있다. 오히려 인터페이스 구현작업만 늘어날 수 있다.

(2) 전체 응용소프트웨어 크기의 증가

래퍼를 이용한 새로운 모델의 구축은 전체 응용소프트웨어의 크기를 증가시키고, 이는 최근 문제가 되고 있는 탑재의 부담(overhead of loading)에서 좋은 결과를 도출하기 어려울 수 있다.

(3) 래핑대상 컴포넌트의 크기의 정의

레가시 시스템에 래퍼를 적용해 새로운 모델을 만들기 위해서는 래핑을 적용할 모델 또는 컴포넌트의 크기(granularity)에 대한 정확한 정의가 필요하다.

(4) 래퍼의 재사용성

새로운 요구에 따라 설계된 인터페이스는 기능과 개발환경 등에 대한 다양한 요소를 갖는 래퍼가 될 것이다. 이를 기존의 제안대로 구현한다면, 하나의 래퍼 내의 기능과 환경 등에 대한 개선요구가 모두 포함될 것이다. 이는 다시 발생할 수 있는 시스템의 개선상황에 유연하게 대처하기 어렵다. 이는 시스템의 재사용성에 문제를 초래할 수 있다.

3.3 개선전략 제안

앞에서 분석한 기존 래퍼모델의 문제점을 해결 및 보완할 수 있는 전략을 제안한다. 요구에 따라 작업의 방법을 다르게 적용하는 모델 기반 방식이다.

(1) 모델 대상 수준 결정

래퍼 방식의 문제점 중 하나는 대상 모델의 크기에 대한 정의가 없다는 점이다. 일반적으로 모델기반 방식에서는 컴포넌트를 대상으로 하는데 그 컴포넌트의 크기에 따라서 래퍼의 크기가 달라질 수 있고, 이에 따라 래퍼의 재사용성도 달라질 수 있다.

(2) 모델 대체

새로운 시스템을 위한 요구사항을 분석한 결과, 기존 모델의 기능을 개선하는 것으로 판단되면, 새로운 컴포넌트를 생성하여 기존 컴포넌트를 대체한다. 기존 컴포넌트의 기능을 변경하는 것을 의미한다.

(3) 모델 추가

기존 모델에 새로운 기능이 추가 되는 것으로 분석되었다면, 새로운 컴포넌트와 이에 대한 인터페이스를 이용해 기능을 추가할 수 있다.

(4) 모델 통합

모델의 합병은 둘 이상의 모델을 하나의 모델로 통합하는 요구사항에 대응하기 위한 전략이

다. 래퍼를 적용할 수 있는 상황이지만, 모델의 대상 컴포넌트를 찾고, 모델 내부 또는 외부 컴포넌트 사이의 연관성(독립성, 결합성)을 분석해서 효율적인 래핑을 시도해야 한다.

4. 결론 및 향후 연구방향

대부분의 역공학 접근방식에서는 레거시 시스템을 통합하기 위하여 데이터 위주로 하기 때문에 비즈니스 프로세스의 지속적인 변화는 다루지 못하고 있다. 그래서 본 논문에서는 모델기반을 사용하여 레거시시스템의 연계에도 컴포넌트 기본 원칙이 적용되어 통합 작업을 용이하게 하도록 한다. 또한 순공학 접근방식에서는 기존의 시스템을 다 고치려 하는 경향이 있기 때문에 많은 비용과 시간이 소요된다. 그러므로 새로운 비즈니스 컴포넌트를 설계하고 기존의 시스템과 통합하기 전에 레거시 시스템에서 재사용될 수 것과 실제 수정될 수 있는 부분을 점검하여야 한다.

그러기 위해서는 기존의 자산을 적극적으로 재사용하기 위해서 잘 정의된 인터페이스를 통해 독립적인 단위가 되도록 만들어져야 하며, 이러한 인터페이스로 재정의 됨으로써, 레거시 응용프로그램은 하나의 큰 규모의 컴포넌트로 다뤄질 수 있으며, 또한 여러 컴포넌트로 새롭게 구성되어 질 수도 있다.

레거시 시스템에 래핑 기술을 적용함으로써 새로운 환경에서 재사용이 가능해짐에 따라 레거시 시스템도 기업의 주요 자산(Core Assets)이라는 마인드가 확산되고, 유지보수의 중요성이 더욱 더 커질 것이다.

래퍼를 사용하는 것과 사용하지 않는 것에 따른 상황이 서로 다르게 작용할 수가 있을 것이다. 본 논문에서는 이러한 래퍼의 사용에 따른 문제점을 분석하고, 요구에 따라 모델에 대한 접근방식을 다르게 사용하여 시스템의 개선 상황에 유연하게 대처할 수 있도록 래퍼를 상황에 따라 적절하게 사용하는 혼합(hybrid) 방식을 적용한 개선전략을 제안하였다.

앞으로의 향후 과제로는 이러한 혼합방식의 래퍼를 구체화시키기 위해 여러 가지 요소들을 식별화(identify)하고 구현하는 작업이 필요하다. 또한 컴포넌트간의 분야가 서로 다른 경우 어떠한 방식으로 래퍼를 융합할 수 있는지에 대한 전략이 필요하고, 모델기반아키텍처에서 모델 내부간의 통합을 위해 모델간의 융합에 대한 연구가 이루어져야 할 것이다.

[참 고 문 헌]

[1] Willem-Jan van den Heuvel, "Matching and Adaptation: Core Techniques for MDA-(ADM)-driven Integration of new Business Applications with Wrapped Legacy System", http://www.cis.uab.edu/EDOC-MELS/Papers/Willem-Jan.pdf

[2] ADM Task Force, "Why do we need standards for the modernization of legacy systems", http://adm.omg.org/legacy/ADM_whitepaper.pdf

[3] Joaquin Miller and Jishnu Mukerji et al, "MDA Guide Version 1.0.1 Technical Report" Object Management Grou p, June 2003, http://www.omg.org/docs/omg/03-06-01.pdf

[4] Stephen J.Mellor, Scott Kendall, Axel Uhl and Dirk Weise, "MDA Distilled", Addison-Wesley, 2004

[5] Anneke Kleppe, Jos Warmer and Wim Bast, "MDA Explained: The Model Driven Architecture-Practice and Promise", Addison-Wesley, 2003

[6] Ph. Rhiran, T.Risch, C.costilla, J.Henrard, Th.Kabisch, J.Petrini and W-J.van den Heuvel, "Report on the Workshop on Wrapper Techniques for Legacy Data Systems", SIGMOD Record, Vol.34, No.3, Sept. 2005

[7] Paul Asman, "Legacy Wrapping", Federal Reserve Bank of New York, November 2000.

[8] 송호진, 최은만, "컴포넌트 테스트를 위한 래퍼의 자동 생성에 관한 연구", 정보과학회논문지 제 32권 제 8호, 2005.8