# MDA기반의 분산 컴포넌트 성능측정도구 설계

황길승° 이긍해
한국항공대학교 컴퓨터공학과
{avi94°, khlee}@mail.hangkong.ac.kr

# The design of a MDA based performance measurement tool for distributed components

Kil-Seung Hwang° Keung-Hae Lee
Dept. of Computer Engineering, Hankuk Aviation University

요 약

본 논문은 OMG의 MDA(Model Driven Architecture)를 기반으로 한 분산 컴포넌트의 성능 측정도구의 개발모델에 대하여 설명한다. 이 모델을 이용하면 컴포넌트 모델과 미들웨어 프레임워크에 중립적인 성능 측정기를 개발할 수 있으며 동일 내지는 유사기능을 수행하는 컴포넌트들의 비교와 검증에 필요한 객관적인 성능 측정 결과를 얻을 수 있다.

## 1. 서 론

CBD(Component Based Development)는 특정 기능별로 모듈화된 소프트웨어 컴포넌트를 조립하여 소프트웨어를 개발하는 방법이다. 이 방법은 컴포넌트의 재사용성을 극대화하여 소프트웨어의 개발에 드는 비용을 줄여주고 생산성을 향상시킬 수 있다. [1][2]

컴포넌트 기반 소프트웨어 개발에서 컴포넌트의 성능은 개발된 소프트웨어의 성능에 영향을 준다. 그러나 표준화된 성능측정방법의 부재와 상호 호환되지 않는 다양한 분산 컴포넌트 모델이 존재하는 현재의 분산 컴퓨팅 환경에서 특정 컴포넌트 모델에 종속되지 않으면서 객관적이고 정확한 방법으로 컴포넌트의 성능을 검증하는 것은 매우 어려운 일이다. 그리고 컴포넌트 시장이 커질수록 같은 기능의 다양한 컴포넌트들의 성능을 비교할 수 있는 방법의 필요성은 증가할 것이 분명하다. 현재 관련 연구들은 특정 컴포넌트 모델에 종속된 성능검증이나 서로다른 컴포넌트 모델 사이에서 측정된 상호운용되지 못하는 객관적이지 못한 성능측정 결과 도출방법에 제한되고 있는 현실이다. [3][4][5]

본 논문에서는 이러한 필요성을 충족하고 문제점을 해결하기 위한 방법으로 특정 분산 컴포넌트 모델에 종속되지 않고 컴포넌트의 성능을 측정하고 검증할 수 있는 성능측정 도구의 개발 모델을 제시한다.

## 2. 관련연구 및 기반기술

### 2.1 관련연구

Baskar 등[3]은 CORBA 플랫폼에서 Visibroker의 Interceptor를 이용하여 컴포넌트 lifecycle에 대한 이벤트를 데이터화 하고 이를 이용하여 성능을 파악하는, 배치된 컴포넌트에 영향을 주지않는 Non-intrusive한 모델을 제시하였다. 이 모델은 이벤트를 공유하는 방법을 제시함으로써 의미가 있지만 Visibroker를 사용하는 CORBA System에서만 적용할 수 있다는 단점을 가지고 있다.

Adrian Mos와 John Murphy는[4] 대상이 되는 EJB와 같은 JNDI 이름을 가진 proxy EJB의 배치를 통해 Response time이나 Execution time의 데이터를 추출하는 방법을 사용하였다. Client와 대상이 되는 EJB사이에 proxy의 역할을 하는 EJB를 자동생성하여 배치함으로써 Client와 EJB사이의 Response와 Request에 대한 이벤트들의 정보를 가로챌 수 있다. 이 방법은 *proxy* EJB를 만드는 등의 성능측정을 위한 번거로운 단계들이 존재하고 proxy EJB의 존재가 기존 EJB의 성능에 영향을 주기 때문에 Non-intrusive하지 못하다.

Kazi 등[6]은 RMI로 호출되는 원격의 자바 분산객체를 JVMPI[7]를 이용하여 프로파일링하는 도구를 제안하였다. 이 논문에서는 Java SDK에서 제공하는 프로파일링 API를 사용하여 원격지의 JVM상의 런타임 객체들을 프로파일링하고 그 결과를 성능데이터로 하여 원격객체의 성능을 모니터링 하였다. JVMPI에서 원격객체의 RMI호출을 프로파일링하는 것은 불가능하므로 이 논문에서는 JVM을 modify하여 RMI 프로파일링을 지원하는 JVM을 만드는 방법을 사용하였다. 이 방법은 현재의 분산 컴포넌트 플랫폼에 적용하기에는 적당하지 못하다.

### 2.2 Model Driven Architecture

분산 컴포넌트를 위한 다양한 미들웨어의 등장은 소프트웨어의 개발 및 유지보수에 많은 비용낭비를 초래하였고 다른 미들웨어를 기반으로 하는 소프트웨어와의상호운용성을 떨어뜨렸다. 이러한 문제를 해결하기 위하여 2000년 말에 OMG(Object Management Group)에서 이기종의 플랫폼간의 상호운용성을 보장하고 개발과 유지보수에 드는 비용을 줄이는 새로운 소프트웨어 개발 패러다임인 MDA(Model Driven Architecture)를 제안하였다.

MDA는 모델과 구현을 분리함으로써같은 도메인의 표준화된 소프트웨어 모델을 생성할 수 있다. 그리고 이기종의 플랫폼을 기반으로 하는 소프트웨어나 저장소를 쉽게 통합할 수 있는 방법을 제공한다.[8][9]

MDA의 개발단계는 크게 세가지 단계로 나눌 수 있다. 첫번째 단계에서는 비즈니스 모델링에 따른 소프트웨어의 구조, 관계, 기능, 제약조건등을 정형화한 PIM(Platform Independent Model)을 만든다. 이는 UML(Unified Modeling Language)을 기반으로 하며 OCL(Object Constratint Language)과 Action Language로 표현된다. 두번째 단계에서는 만들어진 PIM에 미들웨어 기술에 종속적인 요소들을 추가하여 PSM(Platform Specific Model)을 생성한다. 이때 PIM에서 PSM으로의 매핑을 위해서 특정 기술의 UML Profile을 이용할 수 있다. 마지막 단계는 특정 미들웨어에 종속되어 상세하게 모델링된 PSM으로 코드를 생성하는 단계이다.

MDA 개발단계를 도식화 하면 다음과 같다.

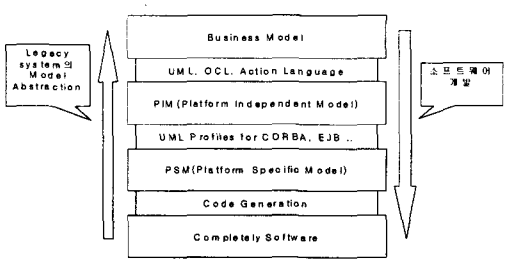


그림 1. MDA 기반 개발 프로세스

### 2.3 분산 컴포넌트 아키텍쳐

CBD기반 개발방법은 소프트웨어의 기능을 구성할 때 고유한 기능을 가지는 소프트웨어 컴포넌트를 사용한다. 원격지의 컴포넌트에 접근하여 컴포넌트의 기능을 호출하기 위해서 플랫폼에 공통적인 표준화된 아키텍쳐를 가진다. 아키텍쳐의 개략적인 그림은 다음과 같다.

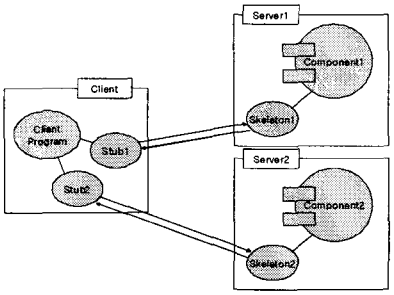


그림 2. 분산객체 아키텍쳐

분산된 각각의 컴포넌트는 두 가지 필수 구성요소를 가진다. Operation의 시그니쳐를 가지는 컴포넌트 인터페이스와 클라이언트 사이의 통신을 담당하는 proxy이다. 컴포넌트 모델마다 각각 다른 명칭을 사용하고 있지만 클라이언트와 서버 컴포넌트와의 통신 방법은 동일하다. [10][11]

### 2.4 분산 컴포넌트의 성능측정

분산 컴포넌트의 성능측정을 위해서는 성능측정 요소를 정의하는 것이 중요하다. 성능측정 요소는 컴포넌트의 성능을 표현하는 데이터 타입으로써 본 논문에서는 네가지 성능측정 요소를 정의한다.[1][5]

1. 메소드별 응답시간 (Method Response Time) : 컴포넌트의 단위기능의 응답시간.
2. 컴포넌트 응답시간 (Component Response Time) : 컴포넌트 인스턴스 생성시간을 포함한 컴포넌트 내의 모든 메소드의 응답시간의 합.
3. CPU 사용률 (CPU Usage Rate) : 컴포넌트가 호출됨으로써 생기는 CPU사용률의 변화량.
4. 메모리 사용률(Memory Usage Rate) : 컴포넌트의 메소드를 사용할 때의 메모리 사용률의 변화량.

이들 네가지 성능측정 요소는 컴포넌트의 사용시 소비되는 가장 대표적인 네가지 리소스의 변화를 의미한다. 그러므로 이들을 적용하여 컴포넌트에 대한 성능데이터를 추출한다면 추출된 성능데이터는 컴포넌트의 객관적인 성능을 나타내는 데이터로써 활용될 수 있다.

그림 2의 분산객체 아키텍쳐를 기반으로 하여 분산 컴포넌트의 성능을 측정하는 성능측정기의 구조는 그림 3과 같다.

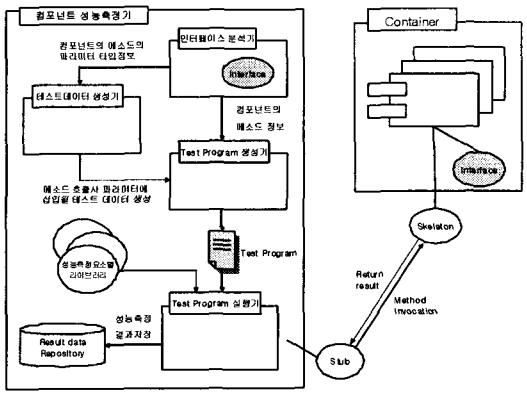


그림 3. 컴포넌트 성능측정 아키텍쳐

분산환경에서 배치된 컴포넌트에 접근하여 컴포넌트의 Operation을 호출하기 위해서는 각각의 분산 컴포넌트 모델들이 정의하는 명세를 준수하여야 한다. 분산 컴포넌트 서버에 접근하기 위한 통신 프로토콜, 컴포넌트의 객체로의 바인딩을 위한 절차 등은 각각의 컴포넌트 모델마다 표준화된 아키텍쳐로 정의하고 있다. 하지만 추상적인 단계에서의 컴포넌트로의 접근은 특정 미들웨어에 상관없이 동일하다.

## 3. 성능측정기 개발 프로세스

본 논문에서 제안하는 성능측정기의 개발방법은 Unified Software Development Process[12]와 MDA를 기반으로 한다. 개발방법은 다음과 같은 과정을 가진다.

1. 개발에 필요한 요구사항을 수집하고 명세화.
2. 요구사항 명세를 UseCase별로 분류하고 분석하여 Requirement Analysis Model(RAM)을 작성.
3. RAM을 상세화, 구조화하여 Platform Independent Model(PIM)을 생성.
4. PIM을 기초로 개발될 소프트웨어의 특성에 맞는 기술종속적인 요소를 더하여 Platform Specific Model(PSM)을 생성.
5. 생성된 PSM에서 수동 또는 자동으로 코드를 생성하여 소프트웨어를 완성.

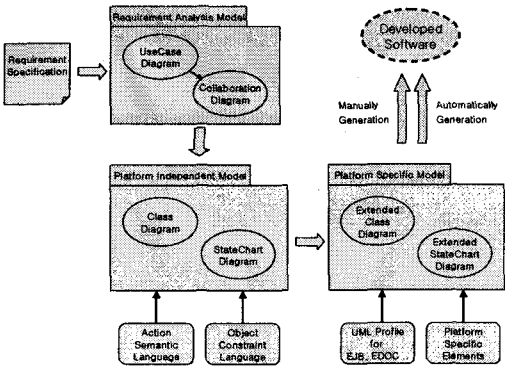


그림 4. 성능측정기 개발 프로세스

성능측정기 개발 프로세스에서는 생성된 모델을 다양한 컴포넌트 모델로 적용하기 위해서 모델링 단계를 플랫폼 독립모델과 종속모델로 구분하였다.[8][13] 이 구분은 성능측정 대상이 되는 모든 컴포넌트 모델에 적용가능한 모델과 특정 컴포넌트 모델에 종속적인 상세모델로 구분함으로써 상호운용성과 확장성을 보장할 수 있다.

### 3.1 성능측정기의 요구사항 분석모델

성능측정기를 설계하기 위해서 성능측정기에 요구되는 요구사항들과 대략적인 프로세스를 구조화하여 사용자의 관점에서 본 전체 시스템의 그림을 그리는 것은 중요하다. 성능측정기에 대한 요구사항은 그 후의 모델링이나 구현에 그대로 반영되며 정확하고 체계적인 모델링 및 구현을 위해서는 정확한 요구사항 분석이 필요하기 때문이다.

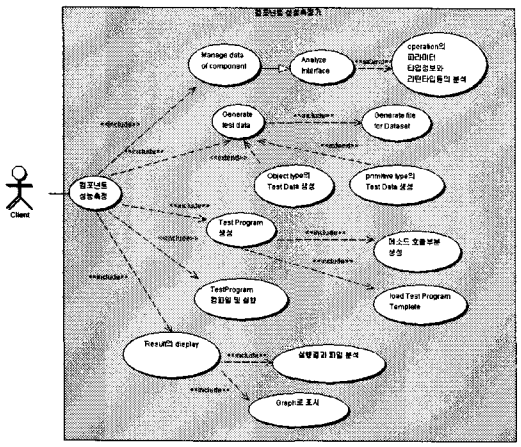


그림 5. 컴포넌트 성능측정기의 UseCase Diagram

비즈니스 모델링 단계에서 컴포넌트 성능측정기에 요구되는 조건들은 다음과 같은 몇가지로 정리된다.

1. 컴포넌트의 성능측정 수행방법이 정의되어야 한다.
2. 성능측정에 요구되는 데이터의 수집방법이 정의되어야 한다.
3. 성능측정 요소들에 대해 정의되어 있어야 한다.
4. 성능측정 요소별 성능측정값 수집방법이 정의되어 있어야 한다.
5. 성능측정결과가 어떻게 표시될 것인지 정의되어 있어야 한다.

### 3.2 성능측정기의 플랫폼 독립모델

플랫폼 독립모델(PIM)은 소프트웨어의 정적인 모습과 동적인 모습을 추상적으로 표현한다.

성능측정기의 PIM을 정의하는 것은 성능측정 대상이 되는 컴포넌트 모델에 상관없이 적용 가능한 도메인 표준모델을 제시한다는 점에서 중요하다.

PIM은 UML표기법으로 표현될 수 있다. 소프트웨어 시스템의 정적인 모습은 Class Diagram으로 표현하고 상태변화와 상태에 따른 행위들을 표현한 동적인 모습은 StateChart Diagram과 Action semantic으로 나타낸다.[14] 위의 Class diagram과 StateChart Diagram을 통해서 컴포넌트 성능측정기의 구조와 관계, 행위, 상태의 변화등을 알 수 있다.

### 3.3 성능측정기의 플랫폼 종속모델

PIM은 특정기술에 독립적으로 작성된다. PIM이 완료되면 좀 더 상세한 부분으로의 접근, 즉 플랫폼 종속적인 요소들을 첨가하여 PSM을 생성하게 된다.

컴포넌트 성능측정기의 개발에는 특정한 분산 컴포넌트 플랫폼을 요구하지 않는다. 그러므로 PSM으로의 매핑시 특별한 UML Profile을 필요로 하지 않는다. 하지만 어떤 분산 컴포넌트 플랫폼을 성능측정의 대상으로 하느냐에 따라 다른 PSM으로의 매핑 구조를 가지게 된다. 컴포넌트 성능측정기의 PIM에서 PSM으로 매핑할 때 추가되어야 하는 요소들은 다음과 같다.

- 성능측정시 사용자로부터 수집되어야 하는 데이터의 종류
- 성능측정시 성능측정 대상이 되는 컴포넌트의 인터페이스 분석방법
- 테스트 프로그램 생성시 language의 종류
- 성능측정시의 connection정보(프로토콜, 포트 등)
- 성능측정 요소별로 로딩되는 성능측정 라이브러리
- 테스트 프로그램에서 컴포넌트의 호출을 위해 생성되는 테스트

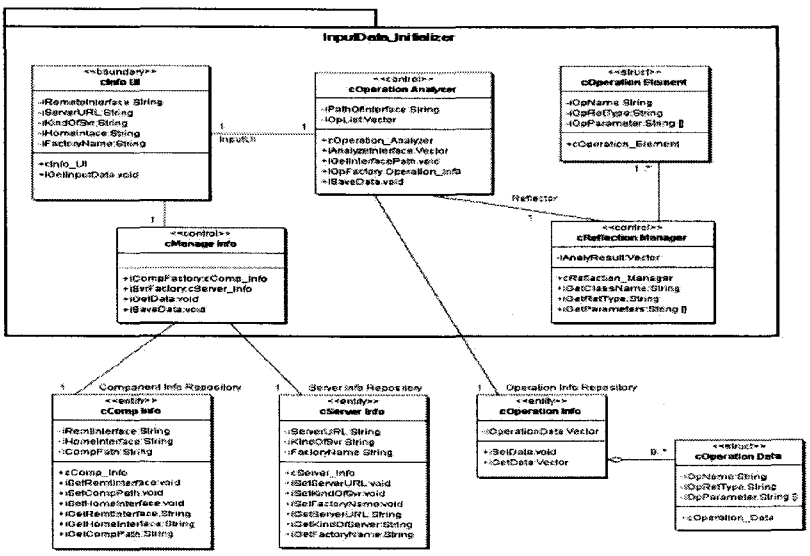


그림 6. 성능측정기의 입력데이터 초기화 부분의 PSM

데이터의 타입
- 테스트 프로그램 컴파일 및 실행시 컴파일러의 종류
- 테스트 프로그램 컴파일 및 실행시 필요한 라이브러리

성능측정기의 PIM에 위에서 얻어낸 요소들을 적용하여 상세화하면 성능측정기의 PSM을 수동으로 생성할 수 있다.

### 3.4 시스템 구현

완성된 PSM을 바탕으로 하여 코드를 생성할 수 있다. 구현된 시스템은J2EE플랫폼 기반의 EJB 컴포넌트의 성능측정을 수행한다. 그리고 성능측정기의 구현은 자바를 기반언어로 하고있다. 성능측정기의 모습은 그림 7과 같다.

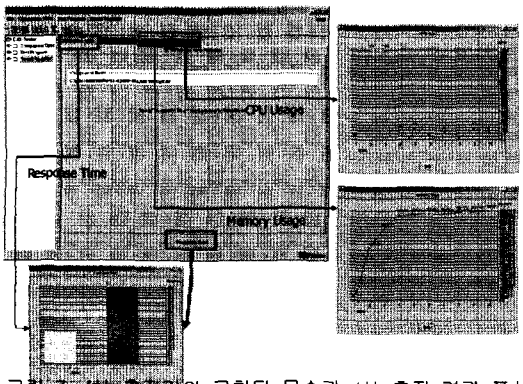


그림 7. 성능측정기의 구현된 모습과 성능측정 결과 표시

구현된 컴포넌트 성능측정기는 앞에서 정의된 네가지 성능 요소별 데이터를 도출한다.

성능측정 요소별 성능측정 결과 데이터는 같은 컴포넌트 모델상의 서로다른 컴포넌트에 대한 결과를 비교할 수 있을 뿐만 아니라 서로 다른 컴포넌트 모델상의 같은 기능을 하는 컴포넌트에 대한 성능비교도 가능하다. 이러한 성능비교가 가능한 이유는 성능측정 요소별 결과 데이터 타입이 PIM 레벨에서 이미 정의되어 PSM과 구현에 적용되기 때문이다. 이를 이용하면 서로 다른 컴포넌트 모델을 대상으로 하는 성능측정기 간의 결과데이터를 상호 운용하는 것이 가능하다. 즉 서로 다른 컴포넌트 모델 상의 컴포넌트에 대한 성능비교가 가능하게 된다.

## 4. 결론

본 논문에서는 분산환경에서 배치된 컴포넌트의 성능측정 방
법을 제안하였다. 그리고 이를 적용하여 분산 컴포넌트의 성능을 측정할 수 있는 컴포넌트 성능측정기의 플랫폼 중립적인 모델을 정의하였고 구현을 통하여 성능측정기의 모델이 플랫폼에
따라 구현되고 실행될 수 있다는 것을 증명하였다.

컴포넌트 기반 어플리케이션을 위해서는 높은 성능을 가지는 컴포넌트의 선택이 무엇보다 중요하다. 이를 위해서는 객관적인 방법으로 신뢰성 있는 성능측정결과를 도출할 수 있는 성능측정 방법과 컴포넌트 성능측정기의 필요성은 높다.

본 논문에서 제시한 분산된 컴포넌트의 성능을 측정하는 방법과 개발 방법을 이용하면 분산 컴포넌트 플랫폼이나 컴포넌트 모델에 종속됨 없이 컴포넌트의 성능을 측정하고 객관적인 결과 데이터를 도출하는 컴포넌트 성능측정기를 쉽게 구현할 수 있다. 정의된 개발방법을 통해 다양한 컴포넌트 모델이 존재하는 현재의 컴포넌트 시장에서 컴포넌트의 성능측정이라는 도메인의 표준 모델로서 활용될 수 있을 것으로 기대된다.

## 참고문헌

[1] 한국항공대학교 인터넷소프트웨어기술 연구실, "컴포넌트 성능측정 기법에 관한 연구, 연구결과 보고서", 한국전자통신연구원, 2002.

[2] M. J. Harrold, D. Liang and S. Sinha, "An Approach to Analyzing and Testing Component-Based Systems", Proc. of 1st Int ICSE Workshop on Testing Distributed Component-Based Systems, Los Angeles, CA, May 1999.

[3] B.Sridharan, B. Dasarathy andA. P. Mathur, "On Building Non-intrusive Performance Instrumentation Blocks for CORBA-based Distributed Systems", 4th IEEE International Computer Performance and Dependability Symposium, Chicago March 2000.

[4] A. Mos and J. Murphy, "Performance Monitoring of Java Component-Oriented Distributed Applications" Proc. of 9thIEEE Conference on Software, Telecommunications and Computer Networks (SoftCOM), October 9-12, 2001.

[5] 황길승, 이긍해, 권오천, 신규상, "Black Box방식의 EJB컴포넌트 성능측정", 한국정보과학회, 봄학술발표논문집 (B) pp. 382-384, 2002.

[6] I. H. Kazi, et al. "JaViz : A client/server Java profiling tool", IBM System Journal VOL 39, NO 1, 2000.

[7] Java Virtual Machine Profiler Interface (JVMPI), http://www.javasoft.com/products/jdk/1.2/docs/guide/jvmpi/jvmpi.html.

[8] Richard Soley and OMG Staff Strategy Group, "Model Driven Architecture", Draft3.2, http://doc.omg.org/omg/2000-11-05,2000.

[9] Richard Soley, "Model Driven Architecture : An Introduction", http://www.omg.org/mda/presentations.htm

[10] Rahim Adatia, et al. *Professional EJB*, Wrox Press, 2001.

[11] Robert Orfali, Dan Harkey, *Client/Server Programming with Java and CORBA*, WILEY Press, 1997.

[12] Ivar Jacobson, Grady Booch, James Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, 2001

[13] OMG Architecture Board MDA Drafting Team, "Model Driven Architecture: A Technical Perspective", http://cgi.omg.org/docs/ormsc/01-07-01.pdf

[14] Stephen J. Mellor, Marc J. Balcer, *Executable UML A foundation for Model Driven Architecture*, Addison-Wesley, 2002.