韓國IT서비스學會誌
第9卷 第2號
2010年 6月

JBI 기반 ESB 환경에서 효과적인 메시지 추적을 위한 메시지모니터링 프레임워크

최재현* · 박제원* · 이남용**

A Message Monitoring Framework for Tracing Messages on JBI-based Enterprise Service Bus

Jae Hyun Choi* · Jae Won Park* · Nam Yong Lee**

■ Abstract ■

In order to resolve the problems of traditional Enterprise Application Integration (EAI) for system integration and to establish flexible enterprise IT environments, Enterprise Service Bus(ESB) which have distributed architecture and support Service Oriented Architecture(SOA) has introduced. Particularly, JBI which developed by the Java Community Process is most widely used to implement ESB for advantages of Java technology. In ESB based on JBI, reliable message delivery is very important to ensure stability of services and systems because it is a message driven architecture. But, it is difficult to verify messages and trace messages when system fault or service error occurred because JBI specification is not enough to address them.

In this paper we has proposed the Message Monitoring Framework for JBI-based ESBs which for using in monitoring messages efficiently. It provides foundations for gathering and tracing message-related information about component installation, message exchange, service deploy by using proxy-based change tracking and delegation mechanism for data processing. The proxy which used in our solutions collects data about message automatically when it changed, and the delegation mechanism provides users flexibility for data processing. Also, we describe the performance evaluation results of our solution which is acceptable.

We expect to it enables users to ensure reliability and stability of the JBI-based ESB by systematic monitoring and managing messages being used to interact among components.

Keyword : Java Business Integration(JBI), Enterprise Service Bus(ESB), Message, Monitoring, Framework

논문투고일 : 2010년 04월 20일 논문수정완료일 : 2010년 05월 30일 논문게재확정일 : 2010년 06월 10일

* 숭실대학교 일반대학원 컴퓨터학과 박사과정
** 숭실대학교 컴퓨터학부 교수

1. 서 론

최근 기업들은 EAI를 기반으로 한 시스템통합의 문제를 극복하고, 보다 효과적으로 기업환경변화에 대응하기 위해 기업 내 표준 ESB(Enterprise Service Bus)[4] 구축에 많은 노력을 기울이고 있다. 이것은 ESB가 기존 EAI의 중앙집중형 연결방식에서 벗어난 버스구조의 시스템 연동 솔루션으로[2], 다양한 표준 프로토콜을 기반으로 재사용 가능한 컴포넌트들을 조립함으로써 유동적인 기업 IT 환경구축을 가능하게 하고[6-8]. 급격하게 변화하는 경영환경변화에 유기적으로 적응할 수 있도록 지원하기 때문이다.

따라서 기업들은 이러한 ESB 구축을 보다 신속하고 효과적으로 구축하기 위해 노력하고 있으며, 각 소프트웨어 진영들은 이러한 ESB 구축 과정에서 참조할 수 있는 표준화아키텍처를 개발하는데 노력을 기울이고 있다. 현재 이렇게 개발된 표준 아키텍처에는 JBI, SCA, SDO 등이 있는데, 이중에서 자바진영 제시한 JBI는 기존의 다양한 자바 기술을 손쉽게 통합할 수 있을 뿐 아니라 자바를 기반으로 한 신기술을 손쉽게 접목할 수 있어 서비스통합을 위한 차세대 오픈소스 ESB 아키텍처로 주목받고 있다[4].

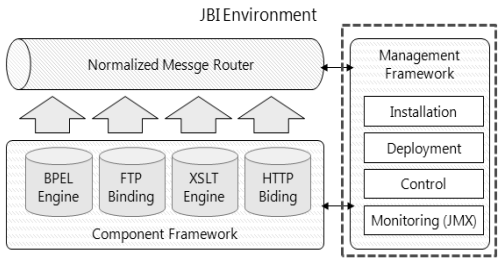
JBI를 기반으로 ESB를 구축할 경우 ESB상의 모든 IT기능들은 개별 컴포넌트로 정의되며, NMR (Normalized Message Router)이라고 정의되는 메시지교환미들웨어를 중심으로 서로 메시지를 교환함으로써 유기적으로 연동된다[9]. 교환되는 메시지에는 데이터뿐만 아니라 다양한 시스템제어와 관련된 내용들이 포함된다. 따라서 이러한 메시지의 안정적인 송수신은 곧 JBI 기반 ESB의 일관성 보장을 위한 핵심 요건이라 할 수 있다. 하지만, 현재까지의 JBI 명세에는 시스템 또는 서비스 장애 시 발생할 수 있는 메시지의 손실 및 오류를 추적하고 복구하기 위한 체계적인 모니터링서비스에 대한 고려가 다소 미흡하여, 장애 발생 시에 효과적으로 메시지 송수신을 추적하고 오류를 탐지하기에 다소 어려운 점이 있다.

이에 본 논문에서는 이러한 JBI를 기반으로 한 ESB에서 메시지 송수신을 효과적으로 추적하고 오류를 탐지하기 위한 메시지모니터링 프레임워크를 제안한다. 이러한 메시지모니터링 프레임워크는 JBI 메시지 프록시를 중심으로 메시지관련 정보를 수집하며, 메시지 상태 변화 및 송수신 과정에서 발생하는 다양한 이벤트를 처리하기 위한 별도의 이벤트 처리기들을 제공함으로서 사용자가 적절한 시점에 오류탐지 및 복구에 필요한 활동을 수행할 수 있도록 지원한다. 따라서 즉 본 논문에서 제안하는 메시지모니터링서 프레임워크를 활용할 경우, JBI 기반 ESB상에서 다양한 서비스들이 송수신하는 메시지를 효과적으로 모니터링 및 추적하고, 신속한 오류 탐지 및 복구가 가능할 것으로 판단된다.

2. 관련연구

2.1 JBI(Java Business Integration)

JBI는 기업 내의 SOA 환경 구축에 있어 가장 핵심기술인 ESB를 위한 자바기반의 아키텍처이다[9]. JBI의 기본이 되는 개념은 표준화된 플러그인 인터페이스를 통한 확장성으로, 기존 EAI에서 사용하고 있는 다양한 벤더들의 기술을 흡수할 수 있도록 다양한 인터페이스를 제공하고 있다. 하지만, 현재까지의 JBI 명세(JBI 1.0)에는 메시지 추적 및 모니터링에 대한 고려가 아직 미흡한 점이 있다.



[그림 1] JBI 환경에서의 관리프레임워크

이것은 JBI가 서비스관리를 위해 JMX(Java Management exTensions)기반 MBean (Management Bean)타입의 관리인터페이스만을 제공한다는 점이다. JMX는 자바컴포넌트를 외부에서 관리할 수 있게 해주는 기술로써, 컴포넌트의 상태를 파악하거나 메서드의 호출 등을 가능하게 하지만, 실제적으로 JBI 상에 존재하는 메시지들은 컴포넌트가 아닌 일종의 데이터이기 때문에 JMX로 이러한 메시지를 추적하거나 관리하는 것은 불가능하다. 즉, 메시지들은 서비스가 처리되는 매우 짧은 시간동안에만 존재하고 자동적으로 소멸되므로, 메시지 전달과정에서 일어나는 오류나 비정상적인 작동을 탐지하거나 모니터링 하는 것이 다소 어렵다. 또한 시스템이 중지되거나 오류가 발생하였을 경우 발생하는 메시지 손실은, 서비스의 비일관성 문제와 같은 또 다른 오류를 야기하게 된다. 따라서 JBI상에서 발생하는 메시지를 효과적으로 모니터링하고 오류를 탐지하는 것은 JBI 환경에서 서비스의 일관성 및 안정성을 결정하는 데 있어 매우 중요한 요소이며, 이러한 메시지의 추적을 보다 효과적이고 효율적인 방법으로 수행하는 것은 JBI 기반 ESB의 신뢰성 및 품질을 좌우하는 매우 중요한 요소라 할 수 있다.

2.2 모니터링에 관한 연구

특정 시스템을 구성함에 있어 시스템에 대한 가시성을 확보하고 오류와 문제를 탐색하여 신뢰성과 안정성, 효율성을 높이기 위한 일환으로써 모니터링에 관한 연구는 학계에서 끊임없이 진행되어 왔다. 특히 이러한 모니터링에 관한 연구들은 주로 시스템의 성능을 측정하고 이를 효과적으로 분배하여 시스템 효율성을 극대화하기 위한 연구들이 많이 이루어졌다. [10]과 같은 연구의 경우, 효율적인 모니터링을 하기 위한 아키텍처를 제시하고 있으며, [1-3, 11]의 경우 JBI가 아닌 타 프레임워크 또는 도메인에서 시스템의 성능모니터링을 위한 방법을 제시하고 있다. [12]와 같은 연구의 경우에는 모니터링 자체적인 성능 향상에 초점을 맞추고 있다. 그러나 이러한 연구들을 종합적으로 분석해 보면, 대부분의 모니터링은 성능측정 및 부하분배라는 시스템 효율성향상에 그 초점을 맞추고 있다. 즉, 시스템 성능에 대한 가시성확보와 성능향상을 위한 정보수집에 초점을 맞추고 있는 연구들로써, 본 논문에서 제안하고자 하는 실제적인 메시지(데이터)의 변화과정 추적하기 위한 모니터링 기법과는 다소 차이가 있다. 이에 본 논문에서는 효과적인 데이터 변화추적이라는 관점아래 JBI 프레임워크를 대상으로 효과적으로 모니터링 할 수 있는 방안에 대해 기술한다.

2.3 ESB의 메시지모니터링에 관한 연구

ESB는 일종의 분산 서비스 협업 모델인 SOA에서 다양한 서비스들의 협업을 위한 하부구조로 사용되고 있다. 따라서 대부분의 연구들은 ESB 자체에 대한 모니터링 연구보다는 이를 포괄적으로 수용하는 SOA 환경을 모니터링하고 관리하기 위한 기법들을 다루고 있다. 이러한 연구들은 모니터링 대상에 따라 그 형태를 다음의 2가지로 구분할 수 있다. 우선 첫 번째 형태는 SOA 환경에서 시스템인프라스트럭처와 애플리케이션 등의 시스템 구성요소의 성능정보를 모니터링하여 서비스의 수준을 확보하는 것이다[13, 14]. 이 같은 형태의 모니터링은 주로 SOA 환경에서 서비스의 상태를 모니터링하기 때문에 본 논문에서 다루고 있는 메시지모니터링과는 다소 거리가 있다고 할 수 있다. 두 번째 형태는 서비스 요청자와 서비스 제공자간의 요청 및 응답을 모니터링하고, 이를 서비스 수준에 정의된 메트릭과 비교 및 분석하여 서비스의 수준을 확보하는 것이다[15, 16]. 그러나 이러한 연구들은 서비스 요청자와 제공자 사이에서 일어나는 서비스협업적인 측면을 모니터링하고 관리하는 것으로, 실제적으로 모니터링을 효과적으로 수행하기 위한 세부적인 기술적 측면을 기술하거나 기법을 제시하고 있지는 않다. 이것은 기술적 측면의 경우 ESB

또는 SOA에서 다양한 기술들이 혼재할 수 있으며, 특정 환경에 대한 일반적인 모니터링 기법을 제안하는 것에 다소 어려움이 있기 때문이라 할 수 있다. 따라서 본 논문에서는 자바진영에서 ESB 또는 SOA를 구현하기 위한 표준으로서 제안한 JBI를 연구범위로 한정하여 JBI를 준수하는 다양한 ESB들에서 활용할 수 있는 체계적인 모니터링 프레임워크를 제안한다.

2.4 기존의 JBI 메시지모니터링기술

JBI에 기반한 오픈소스 ESB에서 모니터링에 활용되는 기술에는 크게 JMX, 로깅, 메시지큐의 세 가지로 구분할 수 있다. 첫째로, JMX를 이용한 모니터링은 JBI의 관리프레임워크에서 제공하는 MBean인터페이스를 이용하여 컴포넌트 및 서비스의 목록과 상태를 관리하는 것이다. 그러나 이것의 경우 JBI1.0 명세서에 정의된 내용에 따라 MBean인터페이스를 통해 관리 가능한 범위가 컴포넌트, 서비스로 제한된다. 즉, JMX를 이용한 모니터링은 컴포넌트 및 서비스 모니터링 서비스는 제공할 수 있으나 메시지를 모니터링 하고 추적하기 위한 부분에 대한 고려는 매우 취약하다. 둘째로, 로깅(Logging)을 이용한 모니터링으로 주로 로거(Logger)가 JBI 환경에서 메시지 교환 시에 발생되는 정보를 텍스트파일에 기록하는 방법이다. 하지만 이렇게 저장된 정보는 가공되지 않은 순수 텍스트로, 이를 체계적인 메시지관리나 추적위해 효과적으로 활용하기 위해서는 이를 처리하기 위한 별도의 로직을 마련해야한다. 또한 이 경우, 메시지 세부적인 변화보다는 일반적인 메시지의 송수신이나 JBI 환경에 대한 전체 상태정보를 기록하기 때문에 효과적인 메시지추적을 위한 정보로 활용되기에는 다소 부족한 점이 있으며, 단순히 정보기록만을 목적으로 하기 때문에 사용자가 필요에 의해서 특정시점에서 복구활동을 하기 위한 루틴을 적용하거나 상황에 따른 추가적인 로직수행을 전혀 지원하지 못한다. 셋째로, JMS를 이용한 메시지모니터링은 NMR에 송수신되는 메시지를 JMS큐에 저장하는 형태로, 이것은 JBI1.0 명세에서는 명시되지 않은 부분이다. 이것은 JBI 환경에서 메시지 송수신을 담당하는 NMR이 메시지 송수신 과정에서 JMS와 같은 메시지 큐를 사용하도록 함으로써, 메시지를 영속성 있게 보관하거나 추적할 수 있도록 한 기술이다. 그러나 이것은 단지 메시지가 서비스사이에서 이동하는 시점에서만 메시지를 저장할 수 있기 때문에는 메시지의 상태변화와정이나 오류를 체계적으로 추적하거나 탐지하는 데는 다소 부족한 점이 있다.

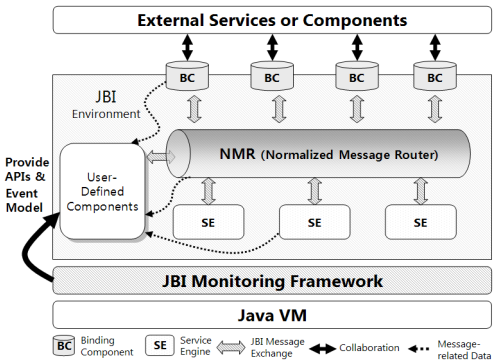
이처럼 JMX, 로깅, 메시지 큐를 이용한 기존의 모니터링 서비스 기술들은 메시지의 상태변화에서 송수신에 이르는 전 과정을 체계적으로 모니터링 및 관리하고, 이를 효과적으로 활용하는데 있어 다소 미흡한 점이 있다. 따라서 본 논문에서는 메시지관련 정보의 수집 및 모니터링에서부터 이를 체계적으로 저장 및 관리하고, 활용하기 위한 효과적인 메시지모니터링 프레임워크를 제시한다.

3. JBI 메시지모니터링 프레임워크

3.1 JBI 메시지모니터링 프레임워크 정의

JBI 메시지모니터링 프레임워크는 JBI에서 생성되고 사용되는 메시지의 모든 상태변화와 송수신 과정을 모니터링 및 추적하기 위한 프레임워크이다. 이러한 프레임워크에는 메시지를 효과적으로 추적하기 위해 필요한 객체 및 클래스들과 이를 활용하기 위한 다양한 모델들이 포함된다. 먼저, JBI 메시지모니터링 프레임워크는 사용자 또는 관리자에게 메시지의 상태 및 변화 값들을 추적할 수 있는 API들과 이벤트 모델을 제공한다. 이러한 API들은 메시지 추적을 위한 Proxy 객체의 생성에 관여하며, 이벤트 모델은 사용자 또는 관리자가 원하는 시점에서 특정 메시지 정보를 수집하여 원하는 형식으로 가공하여 원하는 곳에 저장할 수 있도록 지원한다. [그림 2]는 이러한 JBI 메시지모

니터링 프레임워크의 개요를 보여주고 있다. JBI 메시지모니터링 프레임워크는 자바를 기반으로 JBI 환경을 지원하는 하부 프레임워크로서 동작하게 되며, 메시지 교환을 담당하는 NMR과 이를 중심으로 실질적인 서비스를 제공하는 서비스엔진 컴포넌트 및 외부환경과의 연계를 담당하는 바인딩컴포넌트사이에서 발생하는 모든 메시지 교환에 대한 정보를 수집하여 사용자정의 컴포넌트에 전달하게 된다. 그리고 이렇게 전달된 정보는 JBI 모니터링 프레임워크 API 및 이벤트 모델을 바탕으로 효과적인 메시지 추적을 가능하게 한다.

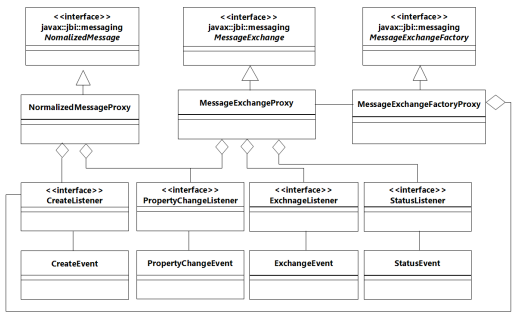


[그림 2] JBI 메시지모니터링 프레임워크 개요

3.2 JBI 메시지모니터링 프레임워크 설계

JBI 환경에서 메시지는 실제정보를 포함하고 있는 메시지객체(NormalizedMessage)와 메시지 교환객체(MessageExchange)로 구분되어 운영된다. 메시지객체는 JBI 환경에서 컴포넌트와 컴포넌트 사이에 이동하는 정보를 포함한 객체이며, 메시지교환은 메시지가 이동시에 송수신지의 정보 및 속성들을 함께 기록한 일종의 패킷객체이다. 따라서 본 논문에서는 JBI 환경에서 메시지를 효과적으로 추적하기 위해 이러한 두 가지 메시지관련 객체를 체계적으로 추적 및 모니터링 한다.

이를 위해 본 논문에서 제안하는 JBI 메시지모니터링 프레임워크는 소프트웨어 개발 패턴 중 Proxy 패턴[17]을 활용한 메시지 객체들을 정의한다. 이러한 Proxy 메시지 객체들은 JBI 메시지의 상태 및 정보의 변화과정을 세부적으로 추적하기 위한 것으로 일종의 스파이 에이전트이다. 즉, 이것들은 JBI 메시지 객체에 정보 수집을 위한 스파이코드를 삽입한 것으로 특정 시점 혹은 메시지 변화시점에서 원하는 정보를 얻기 위한 효율적인 방법이다. 그러나 이러한 스파이코드를 사용하는 경우, 모니터링을 필요로 하지 않는 메시지에 대해 오버헤드가 발생하거나 코드의 중복이 발생할 수 있기 때문에, 본 프레임워크에서는 Delegation 패턴[2]을 이용하여 Proxy 객체들에 필요한 추가코드를 최소화하였다. 즉, 정보수집을 위한 최소한의 코드만을 Proxy 객체에 삽입하고 나머지 로직은 외부로 위임하여, 효율적으로 Proxy 객체를 활용할 수 있도록 하였다.



[그림 3] JBI 메시지모니터링 프레임워크 메시지 추적관련 클래스다이어그램

[그림 3]은 JBI 메시지모니터링 프레임워크를 구성하는 클래스들 및 관계들을 클래스다이어그램으로 표현한 것이다.

MessageExchangeFactoryProxy, Message ExchangeProxy NomalizedMessage Proxy는 JBI에서 사용되는 메시지 관련 클래스들의 Proxy 클래스들로 JBI환경 내에서 메시지에 관한 모든 상태변화를 감지하고 정보를 수집하는 일을 담당한다. 이러한 Proxy 클래스들은 앞서 언급한 바와 같이 메시지 정보수집에 따른 오버헤드의 발생을 최소화하기 위하여, Delegation 패턴을 기반으로 정의

된 이벤트 리스너들을 포함하고 있다. 즉, 메시지에서 일어나는 변화를 이벤트로 간주하고, 해당 이벤트가 발생하였을 경우 필요한 로직을 외부에서 참조하도록 함으로써, Proxy 클래스에 삽입되는 추가로직을 최소화 한 것이다. 이벤트 리스너 클래스들은 <표 1>과 같이 JBI 메시지 정보를 수집하는데 효과적으로 활용된다.

〈표 1〉 메시지 정보수집을 위한 Proxy 클래스에 포함된 이벤트리스너들의 역할

리스너	역할 및 설명
Create Listener	◦ JBI 메시지 교환 및 메시지의 생성과 관련된 정보를 수집하기 위한 로직을 구현한다. ◦ onCreate() 인터페이스를 포함하고 있다. ◦ CreateEvent 객체를 사용하며, 이 객체에는 JBI 메시지가 생성되는 시점에서의 객체정보 및 관련정보가 포함되어 있다.
Status Listener	◦ JBI 메시지의 오류 및 상태변화와 관련된 정보를 수집하기 위한 로직을 구현한다. ◦ onMessageActive(), onMessage Done(), onMessageError() 3개의 인터페이스를 포함하고 있다. ◦ StatusEvent 객체를 사용하며, 이 객체에는 상태변화가 일어난 서비스 객체의 정보 및 메시지 정보가 포함되어 있다.
Exchange Listener	◦ JBI 메시지의 송수신 과정에서 정보를 수집하기 위한 로직을 구현한다. ◦ onMessageSent() 인터페이스를 포함하고 있다. ◦ ExchangeEvent 객체를 사용하며, 이 객체에는 송수신 객체의 정보 및 메시지 정보가 포함되어 있다.
Property Change Listener	◦ JBI 메시지의 속성 값의 변화 및 처리와 관련된 정보를 수집하기 위한 로직을 구현한다. ◦ onPropertyChange() 인터페이스를 포함하고 있다. ◦ PropertyChangeEvent 객체를 사용하며, 이 객체에는 변화된 속성에 대한 정보 및 변화가 일어난 서비스 객체의 정보 그리고 그 시점에서의 메시지 정보가 포함되어 있다.

각각의 이벤트 리스너들이 포함하고 있는 인터페이스들은 Proxy 클래스의 동작과정 중 어떠한 변화가 일어나는 시점에서 호출되며 이 때, 각 변화에 대한 정보가 담겨있는 이벤트객체가 매개변수로 전달된다. <표 2>는 이러한 이벤트객체의 정의를 보여주고 있다. 또한 JBI 메시지모니터링 프레임워크에는 이외에도 몇 가지 보조클래스들이 포함된다. 이러한 보조클래스들 및 역할은 <표 3>과 같다.

〈표 2〉 모니터링 프레임워크에 포함된 이벤트객체 및 설명

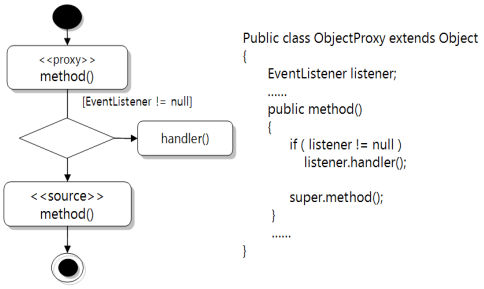
이벤트객체	역할 및 설명
Create Event	◦ 메시지 생성이 일어난 위치(객체정보)와 상태값들을 포함한다.
Status Event	◦ 메시지 상태 변화가 일어난 위치(객체정보)와 상태값들을 포함한다. ◦ 속성설정을 통해 해당 시점에서의 메시지 정보 또한 넘겨받을 수 있다.
Exchange Event	◦ JBI 메시지의 송수신에 관련된 정보를 포함한다. 여기에는 송수신이 일어난 시간, 송신객체 및 수신객체, 송수신 패턴, 메시지 내용 등이 포함된다.
Property Event	◦ 메시지 및 메시지 교환 객체에서 속성값의 변화에 관련된 정보들이 포함된다. 속성값의 이름 및 이전값과 이후값을 기록하게 되며, 메시지내용의 변화를 체계적으로 추적할 수 있는 기반을 제공한다.

〈표 3〉 모니터링 프레임워크에 포함된 이벤트객체 및 설명

클래스	역할 및 설명
Service Component Helper	◦ JBI 환경에 배치되는 컴포넌트들의 정보와 서비스 Endpoint들의 정보를 파악하고자 할 때 활용된다. ◦ JBI의 ComponenentLifeCycle 객체 및 ComponentContext 객체를 이용한다.
Deployment Helper	◦ JBI 환경에 배치된 서비스 어셈블리, 서비스 유닛, 상태들을 파악하고자 할 때 활용된다. ◦ JBI의 DeploymentServiceMBean 객체를 이용한다.

3.3 JBI 메시지모니터링 프레임워크의 동작

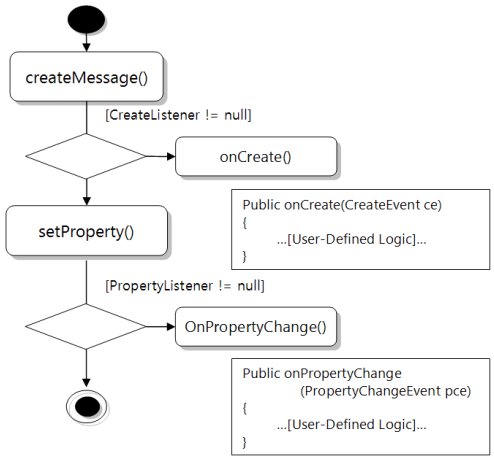
JBI 메시지모니터링 프레임워크는 JBI 관련 메시지의 모든 상태정보를 추적하기 위해 앞서 정의한 Proxy 클래스들을 이용한다. 이러한 Proxy 클래스들은 JBI 아키텍처내에서 서비스 일관성 확보를 위해 기존 클래스의 확장클래스로 정의되며, 이 확장클래스들 내에 메시지 관련 정보 추적을 위한 이벤트 리스너들이 설정된다. 리스너들은 아래와 같이 특정 메서드가 호출될 때 그 메서드를 호출하기 전에 리스너의 인터페이스를 호출함으로서 메시지에 관한 정보를 추적할 수 있게 한다. [그림 4]는 본 논문에서 사용하고 있는 Proxy를 이용한 객체추적의 기본 패턴을 보여주고 있다.



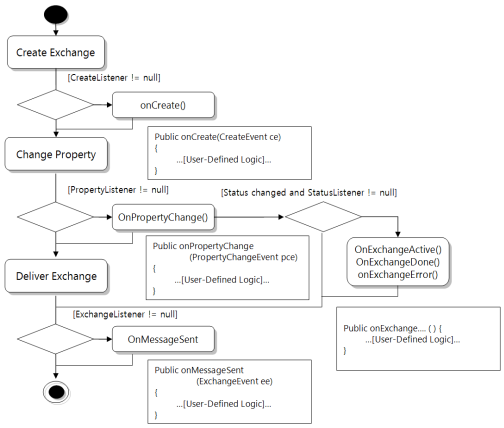
[그림 4] JBI 메시지모니터링 프레임워크의 Proxy의 동작 패턴

본 논문에서는 앞서 언급한 대로 JBI 환경에서의 메시지를 효과적으로 추적하기 위하여 메시지 객체와 메시지교환객체를 구분하여 Proxy 패턴을 적용하고 관련 정보를 수집한다.

[그림 5]는 메시지 객체에 적용된 Proxy 객체의 이벤트관련 동작과정을 보여주고 있다. 메시지 Proxy 객체는 메시지가 생성되는 시점에서 생성시에 CreateListener를 통해 해당 시점에서 파악된 정보를 CreateEvent 객체에 저장하여 onCreate()에 넘겨주게 되며, 또한 속성값이 변화하는 시점에서 PropertyListener를 통해 정보를 OnProperty Change()에 넘겨주게 된다. 관리자는 이렇게 넘어온 정보를 활용하여 필요한 로직을 구현함으로서 메시지를 추적하거나 관리할 수 있게 된다. 즉, 본 논문에서 제시한 모니터링 프레임워크는 이렇게 Delegation 패턴을 기본으로한 객체추적구조를 제공함으로써 단순히 Log를 사용하여 정보를 저장하거나 정해진 로직을 통해 관리를 수행하는 대신, 필요에 따라 적절한 관리 및 추적을 수행할 수 있도록 하여 보다 효과적이고 체계적인 객체추적 및 관리를 지원한다.



[그림 5] JBI 메시지 프록시 객체의 대한 Delegation 패턴기반 정보추적과정



[그림 6] JBI 메시지 교환 프록시 객체의 대한 Delegation 패턴기반 정보추적과정

[그림 6]는 메시지 교환객체에 적용된 Proxy 객

체의 이벤트관련 동작과정을 보여주고 있다. 메시지교환 Proxy 객체 역시 메시지 객체와 동일하게 생성되는 시점에서 생성 시에 CreateListener를 통해 해당 시점에서 파악된 정보를 CreateEvent 객체에 저장하여 onCreate()에 넘겨주게 되며, 속성값의 변화 그리고 메시지의 송수신이 일어나는 시점에서도 해당 이벤트 리스너를 통해 필요한 정보를 사용자 로직에 전달하게 된다.

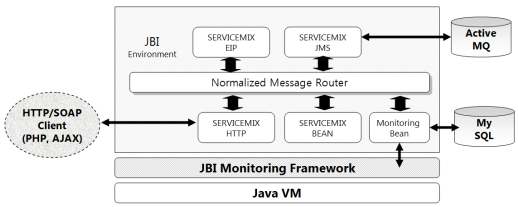
5. JBI 메시지모니터링 프레임워크 활용사례

본 논문에서는 JBI 메시지모니터링 프레임워크를 기반으로 한 모니터링 사례를 보여주기 위해 현재 JBI 구현체로 가장 많이 선호되고 있는 Apache ServiceMix를 활용하여 체계적인 모니터링서비스를 구축하였다. 본 논문에서 활용한 Apache ServiceMix는 JBI를 기반으로 서비스지향아키텍처와 이벤트기반아키텍처의 기능성을 결합한 오픈소스 ESB로, 다양한 컴포넌트들을 가볍고 손쉽게 임베디드 할 수 있으며, 또 다른 ESB내에 하나의 서비스로서 동작할 수 있다. 본 장에서 제시하는 JBI 메시지모니터링 프레임워크를 적용한 모니터링 서비스 구현사례의 환경은 <표 3>과 같다.

〈표 3〉 메시지모니터링 서비스의 구현환경

운영 환경	사양
서버	Apache ServiceMix 3.3
운용 언어	Java2, (JavaTM 2 Platform Standard Edition Development Kit 6.0)
클라이언트	php, Ajax
데이터베이스 및 메시지큐	MySQL 5.0, ActiveMQ 5.3.0

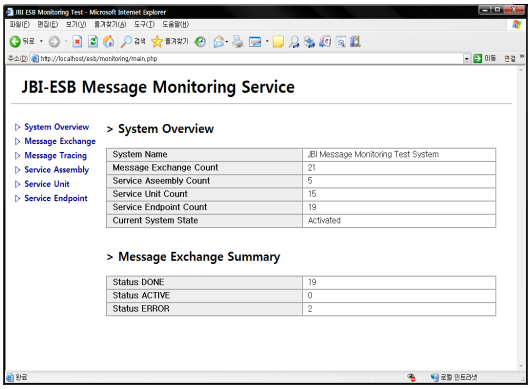
본 논문에서 제안한 JBI 메시지모니터링 프레임워크를 적용하기 위해, 가장 먼저 ServiceMix 구현체에서 내부적으로 사용하는 라이브러리에 모니터링 프레임워크를 지원하기 위한 라이브러리를 추가하였으며, 또한 ServiceMix내에서 활용되는 서비스엔진에서 이러한 추가된 라이브러리를 바탕으로 메시지 객체 및 교환객체를 활용할 수 있도록 코드를 수정하였다. 그리고 각 이벤트 핸들러에서는 이벤트객체로 전달된 정보들을 JDBC를 이용하여 데이터베이스에 저장하고, 이를 웹페이지에서 볼 수 있도록 하였다. 이렇게 본 사례에서 ESB 환경은 [그림 7]과 같이 구성되었다.



[그림 7] JBI 메시지모니터링 프레임워크 적용을 위한 구성

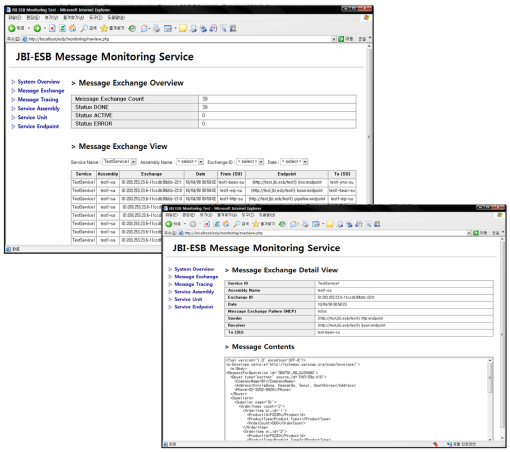
JBI 메시지모니터링 프레임워크에서 메시지추적을 위해 필요한 리스너들은 Monitoring Bean 객체에 구현되었다. 리스너들은 단순히 각각의 이벤트에 따라 넘겨진 정보를 MySQL에 저장하고, SOAP 메시지로 전달된 HTTP/SOAP 클래스의 요청을 받아 해당정보를 클라이언트로 전송하는 일을 수행하도록 하였다. HTTP/SOAP client는 PHP, AJAX를 기반으로 수집된 정보들을 효과적으로 표현할 수 있도록 하였다. 위의 환경에서 실제적으로 JBI 메시지를 모니터링하기 위해 구성된 서비스는 HTTP/SOAP 요청을 받아 SERVICEMIX-BEAN 컴포넌트에서 메시지의 속성값과 내용을 변화시키고 이를 SERVICEMIX-JMS를 통해 ActiveMQ로 전송하는 일을 수행하도록 하였다. 그리고 이러한 과정에서 메시지에 일어나는 변화와 메시지교환에 대한 정보를 MySQL에 저장하도록 하였으며, HTTP/SOAP 웹페이지를 통해 이러한 정보를 파악할 수 있도록 하였다. [그림 8]은 JBI 메시지모니터링 프레임워크를 기반으로 구성된 모니터링 시스템의 웹클라이언트 메인화면을 보여주고 있다. 여기서는 기본적으로 JBI 메시지모니터

링 프레임워크를 통해 수집된 JBI 정보들을 표현한다. 전체 메시지교환의 수, 서비스클래스들이 포함되어 있는 패키지인 서비스 어셈블리 및 서비스 제공점인 서비스 유닛과 앤드포인트들의 수를 보여주고 있다. 이러한 정보들은 JBI 메시지모니터링 프레임워크의 보조클래스를 이용하여 파악할 수 있다.



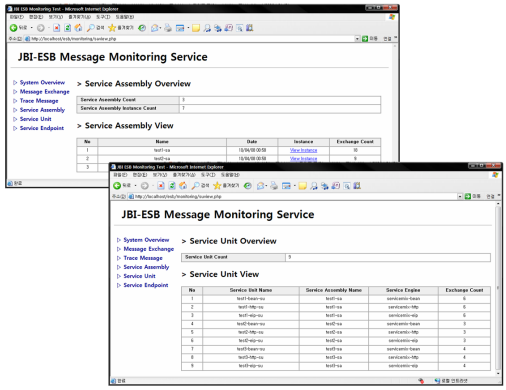
[그림 8] JBI 메시지모니터링 프레임워크를 기반으로 구성된 모니터링 시스템의 웹클라이언트 메인화면

[그림 9]는 메시지 교환(MessageExchange)객체에 대해 수집된 정보를 보여주고 있다



[그림 9] 메시지 교환(Message Exchange) 객체에 대한 수집된 정보를 보여주는 화면

이 정보들은 JBI 메시지모니터링 프레임워크의 메시지 교환관련 이벤트 객체들인 CreateEvent, PropertyChangeEvent, ExchangeEvent, Status Event 객체들에 의해 전달된 정보들 기반으로 구성한 화면으로 JBI 메시지모니터링 프레임워크를 통해 메시지 교환 객체들에 대해 어떠한 정보들을 파악할 수 있는지 알 수 있다. 메시지 교환일자 및 송수신지 메시지 내용 등 다양하고 체계적인 정보 파악이 가능하다.



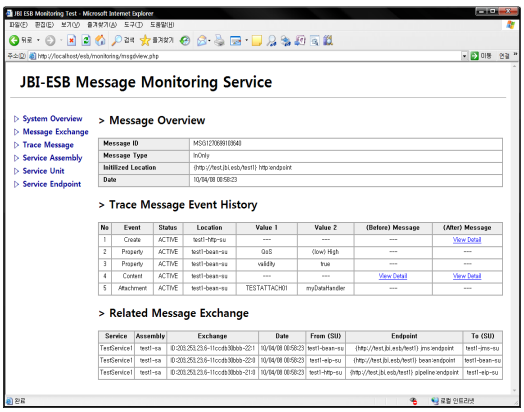
[그림 10] 서비스어셈블리, 서비스유닛에 대한 정보 및 관련 메시지 교환정보

[그림 10]은 JBI 환경에 배치된 서비스어셈블리 및 서비스유닛과 같은 컴포넌트서비스와 관련된 정보 및 이와 관련된 메시지 교환 정보를 보여주고 있다. 특히, 이것은 JBI 메시지모니터링 프레임워크의 보조클래스를 활용한 정보의 탐색을 보여주는 것으로, 보조클래스를 이용한 서비스컴포넌트 정보와 메시지 교환객체에서 수집된 정보를 조합하여, 각 서비스컴포넌트별 메시지 교환 수 및 정보를 파악할 수 있도록 하였다. 즉 이것은 JBI 메시지모니터링 프레임워크를 이용하여 각 서비스컴포넌트별 메시지 처리량 또한 손쉽게 파악할 수 있다는 것을 보여주며, 이를 통해 메시지 처리량을 기반으로한 서비스 성능 파악 및 관리에도 활용될 수 있다.

[그림 11]은 JBI 메시지모니터링 프레임워크를

이용한 메시지 세부추적결과를 보여준다.

이것은 JBI 메시지모니터링 프레임워크가 타 기술과 비교할 때 강점으로 볼 수 있는, 메시지 변화의 전 과정을 모니터링 할 수 있다는 것을 보여준다. 즉, 메시지가 생성된 후부터, 속성값의 변화나 메시지가 송수되는 과정을 체계적으로 파악하고 추적할 수 있도록 하여, JBI 환경에서 오류탐지 및 비정상정인 작동의 원인을 손쉽게 파악할 수 있도록 한다. 여기서 각 메시지들은 메시지가 생성되는 시점인 시간을 기점으로 메시지 아이디가 부여되며, 각 이벤트마다 발생한 정보들뿐만 아니라 해당 메시지와 연관이 있는 메시지교환객체들까지 체계적으로 파악할 수 있다. [그림 11]을 보면 2010/04/0 00 : 58 : 23에 생성된 메시지(ID : MSG127069910 3640)가 이후 QoS 속성 및 validity 속성값의 변화와 컨텐츠 및 첨부내용 변화가 차례대로 일어났음을 파악할 수 있으며, 이러한 변화과정에서 이전 값과 이후값을 파악할 수 있어 메시지 해석과 변경에 있어 오류가 있는지를 손쉽게 파악할 수 있다. 또한 이와 관련된 메시지교환이 총 3번에 걸쳐 일어났음을 알 수 있으며, 각 메시지 교환객체를 선택 시에는 메시지 교환객체에 대한 정보 또한 쉽게 파악할 수 있어 보다 종합적으로 편리한 메시지 추적이 용이하다.



[그림 11] JBI 메시지모니터링 프레임워크를 이용한 메시지 세부추적결과

6. JBI 메시지모니터링 프레임워크의 평가

앞서 언급한 사실과 같이 JBI 메시지모니터링 프레임워크는 실제적인 메시지모니터링을 제공하는 도구가 아닌 메시지모니터링을 가능하도록 지원하는 하부구조의 성격을 지닌다. 즉, 이것은 프레임워크 사용자가 원하는 방식대로 모니터링 기능을 구현할 수 있으며, 또한 필요한 시점에 원하는 로직을 수행할 수 있도록 지원함으로써 보다 유연하고 효과적인 모니터링을 가능하도록 할 수 있다. 따라서 JBI 메시지모니터링 프레임워크에 대한 평가는 기존의 기술들과의 확장성 및 활용성 측면과 기술적 측면을 종합적으로 고려할 필요가 있다. <표 5>는 JBI 모니터링 시스템을 구축하는 과정에서 본 논문에서 제시안 JBI 메시지모니터링 프레임워크를 이용하였을 경우와 기존의 JMX, Logger, 메시지큐를 이용하였을 경우를 비교한 결과이다. 이 표에 나타난 항목 중, 서비스정보 파악은 JBI 환경에 배치된 다양한 서비스어셈블리 및 유닛에 대한 정보를 모니터링 할 수 있는 것을 평

〈표 5〉 기존 JBI 모니터링 기술과 제안된 기술과의 비교

기술 / 항목	JMX	Logger	메시지큐	JBI 메시지 모니터링 프레임워크
서비스 정보파악	○	×	×	○
메시지 변화추적	×	○	×	○
메시지 교환추적	×	○	○	○
추적정보 영속성	일시적	영속적	영속적	영속적
추적정보 활용성	비효율적 (휘발성정보)	비효율적 (텍스트)	효율적 (객체화)	효율적 (객체화)
확장성	낮음	낮음	낮음	높음
적용 편의성	높음	낮음	중간	낮음

가하는 것으로, 이것을 지원하는 경우 서비스별, 어셈블리별과 같은 단위별 모니터링이 가능하여 보다 체계적으로 모니터링을 수행할 수 있는지를 비교한 것이다. 또한 추적정보의 활용성은 정보를 정해진 형태가 아닌 사용자가 원하는 형태로 가공하거나, 해당정보를 기반으로 다양한 사용자 뷰를 생성할 수 있는지에 대한 측면을 비교한 것이라 할 수 있다. 적용편의성의 경우에는 모니터링을 수행하기 위해 필요한 사전작업들의 양으로 결정되는 것으로, JMX의 경우에는 이미 JBI 명세에 그 표준안이 포함되어 있으므로, 높은 반면 로거나 본 논문에서 제시하는 프레임워크를 활용할 경우, 해당 라이브러리의 조정 작업이 필요하기 때문에 낮음으로 평가할 수 있다.

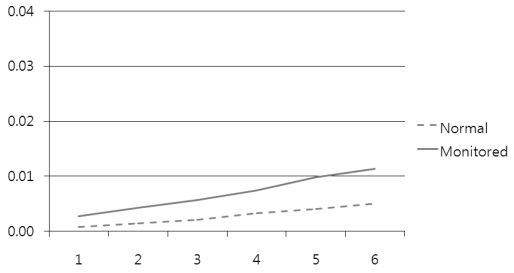
무엇보다 본 논문에서 제안하는 JBI 모니터링 프레임워크는 관리자가 JBI 환경내에서 생성되고 교환되는 메시지에 관한 정보 추적을 체계적이고 효과적으로 수행할 수 있도록 지원하는데 그 목적이 있다. 즉, 관리자가 메시지를 추적하는 과정에서 있어서 필요한 시점에 적절한 로직을 정의함으로써 메시지 변화를 추적하는 것을 용이하게 하고, 이를 통해 메시지 추적의 효율성과 편리성을 증대시키는 것이다. 기존의 JBI 메시지 모니터링 관련 기술들은 이러한 효율성 및 편리성 증대에 초점을 맞춘 하부기반구조를 제공하는 것이 아니라 직접적인 메시지모니터링에 관련된 기술을 제공하는 것이므로, 모니터링구조의 확장이나 정보의 활용에 다소 제약사항이 있을 수 있다. 예를 들어 로거의 경우에는 정해진 시점에 정보를 기록할 수는 있지만, 단순히 정해진 로직에 따라 텍스트 기반으로 한 정보의 저장만 수행하기 때문에 상황에 따른 사용자로직을 수행할 수 없고 단순한 모니터링에서 그 역할이 끝나지만, 모니터링 프레임워크의 경우에는 정보의 저장뿐 아니라 사용자가 속성값이 변화하는 시점에서 단순히 정해진 형태로 정보를 저장하는 것이 아니라 원하는 형태로 정보를 저장하고 필요에 따라 별도의 로직을 수행할 수 있는 진입점을 제공하기 때문에 보다 효과적으로 메시지를 추적하고 관리할 수 있다.

[그림 12]와 [그림 13]은 제안한 모니터링 프레임워크의 성능평가결과이다. 본 성능평가에서는 특정 서비스를 제공하기 위해 협력하는 하위 서비스컴포넌트의 수(N), 메시지의 길이(L), 그리고 메시지에 일어나는 속성변화의 수(C)를 달리하여 전체서비스의 응답속도(T)를 측정하였다. 또한 메시지교환정보와 메시지정보를 저장하기 위해 파일입출력을 사용하였으며, 성능평가를 수행한 시스템의 상은 <표 6>과 같다.

〈표 6〉 성능평가 시스템 사양

항 목	사 양
CPU	Intel(R) Core2Quad 2.33GHz
메모리	3.0GB
운영체제	MS Windows XP SP3

[그림 12]는 모니터링 프레임워크를 적용하였을 경우와 적용하지 않았을 경우의 서비스응답속도의 변화를 보여주고 있다. 이 성능평가결과는 N값이 1에서부터 6까지 변화시키면서 측정한 결과이며, 메시지교환 외에 메시지에 일어나는 속성변화수 C를 3으로 고정하여 측정한 결과이다.

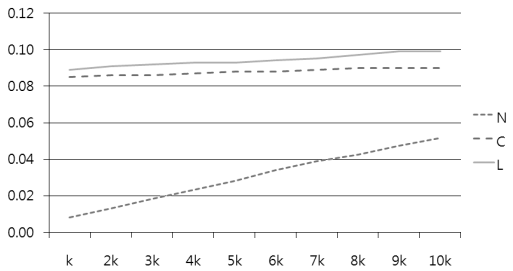


[그림 12] 모니터링 프레임워크 적용에 따른 성능평가결과

측정결과 서비스를 구성하는 컴포넌트의 수가 증가함에 따라 응답속도는 증가하는 경향을 보이지만, 하나의 서비스를 구성하는 일반적인 연계컴포넌트의 수가 10개 미만임을 감안할 때 성능차이

는 0.01초 미만이므로 모니터링 프레임워크를 사용한 메시지추적은 효율적이라 할 수 있다.

[그림 13]은 다양한 서비스구성에 따른 모니터링 프레임워크의 성능변화를 나타낸 평가결과이다. 여기서 k의 값은 N과 C의 경우 3이며, L의 경우 4096이다.



[그림 13] 서비스 구성에 따른 모니터링 프레임워크 성능변화

성능변화를 살펴보면, 모니터링 프레임워크는 연계컴포넌트의 수 N이 증가할 때마다 비교적 응답속도가 증가하고 있으며, 속성변화의 수 C와 메시지길이 L의 증가에 대해서는 응답속도의 변화가 메우 게 나타났다. 이것은 연계컴포넌트사이에서 메시지가 전송될 때, 모니터링 프레임워크가 전체 메시지교환을 저장하는 반면, 속성변화의 경우에는 속성변화에 대한 정보만을 저장하도록 설정하였기 때문이다. 그러나 이것은 어디까지나 모니터링 프레임워크를 사용하는 하나의 일반적인 경우로, 사용자가 모니터링 프레임워크를 기반으로 어떠한 모니터링 매커니즘을 정의하느냐에 따라 모니터링 프레임워크의 성능은 좌우된다.

7. 결 론

본 논문에서는 JBI 기반 ESB 환경에서 메시지 송수신을 효과적으로 모니터링하고 메시지 손실 및 오류를 추적하는데 효과적으로 활용될 수 있는 JBI 메시지모니터링 프레임워크를 제안하였다. 제안된 메시지모니터링 프레임워크는 JBI 상에서 교환되는 메시지의 모든 상태변화 및 송수신 정보를 체계적으로 탐지 및 모니터링하고 추적하기 위한 기반을 제공함으로써 사용자가 보다 편리하고 쉽게 메시지를 추적 및 감시할 수 있도록 한다.

본 논문에서는 이러한 모니터링 프레임워크의 구현을 위해 소프트웨어개발패턴 중 Proxy 패턴을 활용하여 모니터링의 하부기반 객체들을 정의하였으며, Delegation 패턴을 활용하여 보다 효과적으로 사용자가 원하는 모니터링시스템 구축을 지원할 수 있도록 하였다. 또한 이를 효과적으로 활용한 사례를 제시하였으며, 기존기술과의 비교평가 및 성능평가를 통해 타당성 및 효율성을 검증하였다.

따라서 본 논문에서 제안한 모니터링 프레임워크를 활용할 경우, JBI 환경에서 서비스 및 컴포넌트의 설치, 제거 등에 관련된 이벤트를 편리하게 추적하고, 정보를 수집할 수 있고, 단순히 JBI 내의 메시지 관련 정보의 수집에서 벗어나 사용자가 원하는 시점에 특정 활동을 수행할 수 있도록 하는 진입점을 찾는데 활용할 수도 있기 때문에, 신뢰성 있는 JBI 기반 ESB 구축에 많은 기여를 할 수 있을 것으로 판단된다.

향후 연구에서는, 단순한 정보수집차원에서 벗어나 보다 효과적인 오류핸들링을 지원하기 위한 확장된 프레임워크에 대한 연구를 진행할 것이며, 메시지 추적 외에도 많은 연구가에서 초점을 맞추고 있는 성능모니터링 및 실제적인 오류복구기능과 관련된 하부구조를 제시하는 방안도 연구될 것이다.

참 고 문 헌

[1] 권성현, 이병훈, 김재훈, 조위덕, “유비쿼터스 시스템을 위한 실시간 모니터링 에이전트”, 「한국정보과학회논문지C」, Vol.14, No.8(2008), pp.803-807.

[2] 이종대, 구용완, “분산된 서버 관리를 위한 실시간 모니터링 설계 및 구현”, 「한국인터넷정보학회논문지」, Vol.9, No.1(2008), pp.69-78.

[3] 신원, 김태완, 장천현, "비트마스킹 기법을 이용한 임베디드 모니터링 시스템", 「한국정보처리학회논문지D」, Vol.13-D, No.4(2006), pp. 613-618.

[4] David A., Chappell, *Enterprise Service Bus*, O'Reilly Media, 2004.

[5] Lee, J., K. Siau, and S. Hong, "Enterprise Integration with ERP and EAI", *Communications of the ACM*, Vol.46, No.2(2003), pp. 54-60.

[6] Schmidt, M. T., B. Hutchison, P. Lambros, R. Phippen, "The Enterprise Service Bus : Making service-oriented architecture real", *IBM Systems Journal*, Vol.44, No.4(2005), pp.781-797.

[7] Papazoglou, M. P. and W-J. van den Heuvel, "Service-Oriented Architectures : Approaches, Technologies and Research Issues", *The VLDB Journal*, Vol.16, No.3 (2007), pp.389-415.

[8] Naveen Erasala, David C. Yen, and T. M. Rajkumar, "Enterprise Application Integration in the electronic commerce world", *Computer Standards and Interfaces,* Vol.25 (2003), pp.69-82.

[9] Ron Ten-Hove, Peter Walker, *Java Business Integration (JBI) 1.0 Final Release*, Sun Microsystems, 2005.

[10] Tierney, B., R. Aydt, and D. Gunter et al., *A Grid Monitoring Service Architecture*, Global Grid Forum White Paper, 2001.

[11] Hongyan Mao, Linpeng Huang, and Minglu Li, "Web Resource Monitoring Based on Common Information Model", *Services Computing, APSCC 2006. IEEE Asia-Pacific Conference on Digital Object Identifier*, (2006), pp.520-525.

[12] Ke Wang, Zhong Xin Wu, Zhongzhi Luan, and Depei Qian, "Reducing the Cluster Monitoring Workload by Identifying Application Characteristics", *Proceedings of 2008 Seventh International Conference on Grid and Cooperative Computing*, gcc, pp.525-531.

[13] Wang, G., C. Wang, A. Chen, H. Wang, C. Fung, S. Uczekaj, Y. L. Chen, W. Guthemiller, and J. Lee, "Service Level Management using QoS Monitoring, Diagnostic, and Adaptation for Networked Enterprise Systems", *EDOC 2005 Proceedings*, (2005) pp.239-248.

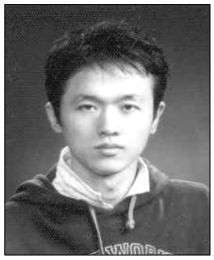
[14] Hershey, P. and D. Runyon, "SOA Monitoring for Enterprise Computing Systems", *Proceedings of EDOC 2007,* Annapolis, MD, 2007.

[15] Berbner, R., T. Grollius, N. Repp, O. Heckmann, E. Ortner, and R. Steinmetz, "An approach for the Management of Service-oriented Architecture (SoA) based Application Systems", *Proceedings of Enterprise Modeling and Information Systems Architectures (EMISA 2005)*, (Klagenfurt, Austria), (2005), pp.208-221.

[16] Cox, D. and H. Kreger, "Management of the Service-Oriented-Architecture Life Cycle", *IBM Systems Journal*, Vol.44, No.4 (2005), pp.709-726.

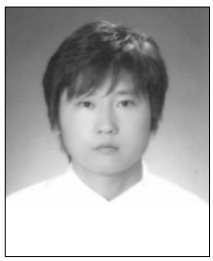
[17] Erich gamma, Richard Helm, Ralph Johnson, and John, Vlissides, *Design Pattern*, Addison-Wesley Pub.Co., 1994.

◈ 저 자 소 개 ◈



최 재 현 (uniker80@empas.com)

숭실대학교 컴퓨터학부를 졸업하고, 숭실대학교 일반대학원 컴퓨터학과에서 석사학위를 취득하였다. 현재 숭실대학교 박사과정에 있으며, 한국정보과학회, 한국정보처리학회 등 국내학술지와 외국학술지에 논문을 게재한 바 있다. 주요 관심분야는 소프트웨어 아키텍처 및 프로세스, 웹 서비스 및 서비스컴퓨팅, 유비쿼터스컴퓨팅 등이다.



박 제 원 (jwpark5656@hotmail.com)

숭실대학교 컴퓨터학과에서 석사학위를 취득하고, 현재 숭실대학교 박사과정에 있다. 한국정보과학회, 한국정보처리학회 등 국내학술지와 외국학술지에 논문을 게재한 바 있으며, 주요 관심분야는 소프트웨어테스팅, 소프트웨어프로세스, 웹 서비스 SOA/ESB 등이다.

이 남 용 (nylee@ssu.ac.kr)

현재 숭실대학교 컴퓨터학과 교수로 재직 중에 있으며, 고려대학교에서 경영정보학과 석사학위를 취득하고, 미국 미시시피 주립대학 경영정보학학과에서 경영학 박사학위를 취득하였다. 국군정보사령부 정보처 정보시스템분석 장교, 한국국방연구원 군수체계 및 정보체계연구부장을 역임하였으며, 한국정보통신기술사협회 회장직도 역임하였다. 주요 관심분야는 소프트웨어테스팅, 시스템엔지니어링, 경영정보시스템 등이다.