韓國IT서비스學會誌
第5卷 第2號
2006年 8月



# ESB기반 SOA Application에 대한 S/W Architecture 관점의 평가와 개선 방안에 대한 연구

임철홍* · 홍도석* · 최정준*

# A Study of a Scheme to Assess and Improve ESB-based SOA Applications from the S/W Architecture Perspective

Chol Hong Im* · Do Seok Hong* · Jeong Joon Choi*

■ Abstract ■

In both inside and outside of our country, the interest of the new generation S/W architecture, the SOA(Service Oriented Architecture) is whooped up. We expect that the SOA will become general S/W architecure in several years, but the most compaines of S/W Industry still think that the SOA have much risk to adapt as their s/w architecture. In this paper, We performed the senario based verification of the SOA quantitatively and qualitatively, and presented the validity of the S/W architecture view and the method of adoption of the SOA.

Keyword : SOA, ESB, architecture, ATAM, CBAM

## 1. 서 론

S/W Architecture는 재사용성과 유연성을 증대시키고 발생 가능한 위험을 감소시키고, 품질 향상을 위해 계속적으로 발전하고 있다. Host, Client/Server, Web을 지나 이제는 SOA가 대세가 될 것이라 예상 하고 있다. IBM, BEA, MS와 같은 벤더를 중심으로 SOA를 지원 하는 솔루션이 활발 하게 출시 되고 있으나 실제 프로젝트 보다는 검증을 위한 POC(Proof Of Concept) 형태의 프로젝트가 많이 진행 되고 있다.

본 논문에서는 ESB기반 SOA Application을

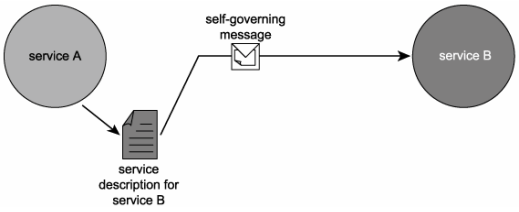
* SK C&C S/W공학 센터 기술전략팀

Architecture 관점에서 기존의 Application과 비교하기 위해서 B2B형태의 비즈니스 시나리오를 수행 하는 Architecture를 작성하고 정성적, 정량적 방법을 활용하여 평가를 수행하고 분석 하고자 한다. SOA적용 방안을 제시하여 Architecture 측면 장점 활용을 극대화 할 수 있도록 하여, POC 개념에서 벗어나 현실화된 Architecture로써 SOA가 도입 되고 활용 될 수 있도록 하고자 한다.

## 2. SOA 특성

### 2.1 SOA의 개념

SOA는 분산 객체의 형태로 존재하는 컴포넌트들간에 Message 통신을 통해 정보를 교환하는 느슨하게 연결된(Loosely Coupling) 형태의 S/W Architecture이다[11]. 개별 컴포넌트들은 미리 약속된 표준 Message인 SOAP(Simple Object Access Protocol)[10] 활용하여 송수신 하며 [그림 1]처럼 동작을 수행한다.

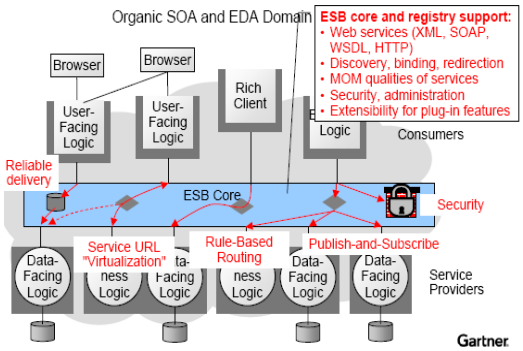



[그림 1] 컴포넌트간 표준 Message 교환

ESB(Enterprise Service Bus)는 SOA환경에서 여러 컴포넌트들 간의 연결을 지원하고, Message에 대한 Routing과 신뢰성 통신을 보장 한다[1]. ESB는 여러 서비스들이 연계 되는 Back Bone의 역할을 수행한다. ESB는 다음과 같은 특성을 가진다.

- XML 기반의 동기/비동기 SOAP messaging
- 안정적이고 신뢰성을 보장하는 messaging
- Intelligent routing and intermediaries
- 데이터 변환(XSLT, XQuery)
- QoS(Quality of Service) 및 보안

ESB의 구성과 동작은 [그림 2]와 같다[7].




[그림 2] ESB 구성도

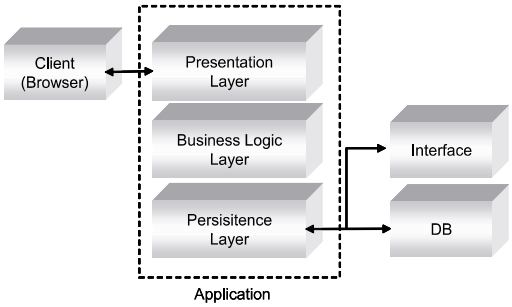
단위 서비스들은 ESB안에 설치되어 프로세스 기반 하에 Message Transformation, Routing, 외부 서비스 호출, Legacy 연동의 기능을 수행 하게 된다. ESB의 기능은 <표 1>과 같다[1].

〈표 1〉 ESB 기능

| ESB기능 | 설 명 |
|---|---|
| Transformation | XSLT에 따라서 Message 변환 |
| Content-Based Routing | 정해진 Rule에 따라 Process 상의 Path를 분기하여 처리 |
| 웹 서비스 | 외부 컴포넌트와 통신 |
| Legacy | 외부 Legacy와 통신 |
| B2B | B2B 프로토콜 기반으로 통합 |
| JMS | 비동기 통신 message 처리 |
| Work-list | Human Interaction(전자결제형태) |

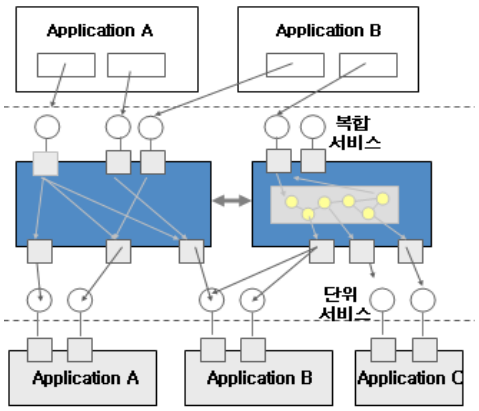
### 2.2 Web과 SOA Application의 특성 비교

SOA Application은 기존의 3-tier기반의 웹 Application과 다른 형태로 구성이 된다. 기존의 Application은 WAS(Web Application Server)를 Database와 연동하여 입력/수정/삭제/조회를 수행하는 방식이다. 3-tier기반 Layer Architecture형태의 Web Application에 대한 설명은 [그림 3]와 같다.




[그림 3] Web Application

ESB기반 SOA Application은 XML기반의 Message를 각 컴포넌트에 전송하여 입력/수정/삭제/조회의 처리가 가능하다. ESB기반 SOA Application에 대한 설명은 [그림 4]와 같다. ESB는 Function 중심의 단위 서비스를 연결 조합하게 되고, 이러한 ESB를 포함하는 Application A, B, C 등은 SOA Application의 형태(복합 서비스)로 동작하게 된다.




[그림 4] ESB기반 SOA Application

# 3. Architecture 평가 방법

## 3.1 ATAM

ATAM(Architecture Trade Off Analysis Method)은 품질 속성 요구사항과 비즈니스 목표달성을 위한 Architecture 결정 사항들을 평가한다. ATAM은 시나리오를 중심으로 품질 속성 요구사항을 찾아내고 Architecture가 특정 품질 속성들을 만족하는지를 분석하는데, 다양한 이해 관계자들과 함께 Risk, Sensitivity, Tradeoff 등을 분석한다[5].

ATAM은 <표 2>처럼 구성 되어 있으며, 평가팀과 이해관계자들, 프로젝트 의사결정권자들이 모여 Architecture 전반에 대한 평가를 하게 된다.

〈표 2〉 ATAM 평가 절차

| 그룹 | 스텝 |
|---|---|
| 소개 (Presentation) | 1. ATAM 소개 |
| | 2. 비즈니스 드라이버 소개 |
| | 3. Architecture 소개 |
| 조사와 분석 (Investigation and Analysis) | 4. Architecture 접근법 식별 |
| | 5. 품질 속성 유틸리티 트리 만들기 |
| | 6. Architecture 접근법 분석 |
| 시험 (Testing) | 7. Brainstorming과 시나리오 우선순위 매기기 |
| | 8. Architecture 접근법 분석 |
| 보고(Reporting) | 9. 결과 보고 |

## 3.2 CBAM

Architecture에는 달성해야 하는 품질 속성에 대한 기술적인 측면과 시스템을 구축할 때 드는 비용과 시스템이 제공하는 품질에서 얻을 수 있는 이득에 대한 경제적인 측면이 있다[4]. 비용과 이득을 고려한 Architecture 평가 방법인 CBAM(Cost Benefit Analysis Method)은 Architecture 접근법을 실현 할 수 있는 품질 속성이 가져다 주는 이익을 측정한다.

CBAM은 비용과 이익으로부터 ROI(Return Of Interest)를 계산하여 수익이 최대가 될 수 있도록 의사 결정을 지원 한다[2]. CBAM은 주로 ATAM 평가가 끝난 후 새롭게 제시된 대안 Architecture 접근법 들에 대해 ROI를 계산하여 대안 Architecture 접근법들 중 ROI가 높은 접근법 대안을 선택할 수 있도록 하고 있다[6]. CBAM의 실행 절차는 <표 3>과 같다[4].

〈표 3〉 CBAM의 실행 절차

| | |
|---|---|
| 시나리오 결정 | 1. 시나리오를 수집한다. |
| | 2. 시나리오를 정제한다. |
| | 3. 시나리오의 우선순위를 결정한다. |
| 효율-반응 값 곡선 작성 | 4. 선별한 시나리오의 효용-반응 값 곡선을 작성한다. |
| Architecture 접근법 전체 이익 계산 | 5. 시나리오를 담당하는 Architecture 접근법을 찾아서 예상 반응 값을 결정한다. |
| | 6. Architecture 접근법의 예상 효용을 계산한다. |
| | 7. Architecture 접근법의 전체 이익을 계산한다. |
| Architecture 접근법 선정과 검증 | 8. Architecture 접근법의 ROI를 계산하여 순위를 결정한다. |
| | 9. 비용과 일정을 고려해서 Architecture 접근법을 선정, 결과를 검증한다. |

# 4. 평가 대상 시스템 Architecture 정의

## 4.1 평가 대상 시스템 정의

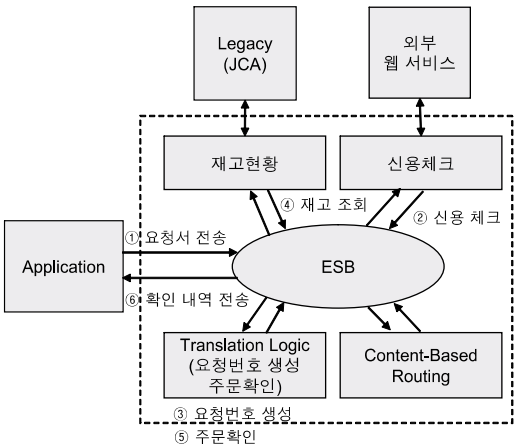
Architecture를 선정하고 평가하기 위하여 대상 시스템을 정의 하였다. SOA의 특성이 반영 될 수 있도록 정보 시스템의 내/외부의 통합 요구사항을 포함 하여 작성된 평가 대상 시스템의 기능은 <표 4>와 같다. 평가 대상 시스템은 보편적으로 활용되는 B2B 업무에서 주문 요청서를 접수하고, 확정하는 업무 단위로 선정 하였다.

## 4.2 대안 Architecture1 - SOA Application

ESB를 활용하여 SOA Application의 형태로 구축한 방안은 [그림 5]과 같다. 업무 또는 기능 단위의 컴포넌트를 구축하고 이를 ESB로 Binding하여 Message 형태로 연동이 된다. 신용 체크를 위해서는 외부 웹 서비스에 연동이 되도록 하였으며, 재고 파악을 위해서는 내부 Legacy 연동을 위하여 JCA (Java Connector Architecture)[3] 기반의 연동을 통해 수행 된다. ESB 및 기능 컴포넌트들 간의 모든 통신은 XML기반 Message의 형태로 수행이 된다.

〈표 4〉 평가 대상 시스템 정의

| 시스템 구성 | 설 명 |
|---|---|
| 관련 시스템 | 주문 요청 시스템, 접수 시스템, 재고 시스템, 신용 체크 시스템 |
| 프로세스 | 1. 주문 요청 시스템에서 주문 요청서 전송<br>2. 주문 요청 회사 신용 체크<br>3. 주문 요청 번호 생성<br>4. 재고 파악<br>5. 주문 확인 Message 전송 |
| 요건 정의 | 1. 주문 요청 시스템에서 주문 시스템으로 요청을 전송함(파일, 전자 문서)<br>2. 재고 시스템은 내부적인 Legacy를 활용함(실질적인 내부 Legacy형태에 상관없이 표준 인터페이스를 통해 통신 수행)<br>3. 신용 체크 시스템은 파일, 전자 문서 형태의 Request를 받고 응답<br>4. 주문 요청 번호는 날짜와 순번을 조합하여 정의함<br>5. 주문 수량 보다 재고가 적을 경우 안내문 출력 후 주문 거부함<br>6. 신용도가 “B”이상일 경우 주문 가능함. 미달일 경우 주문 거부함<br>7. 주문 확인 Message를 요청 시스템에 전송함 |

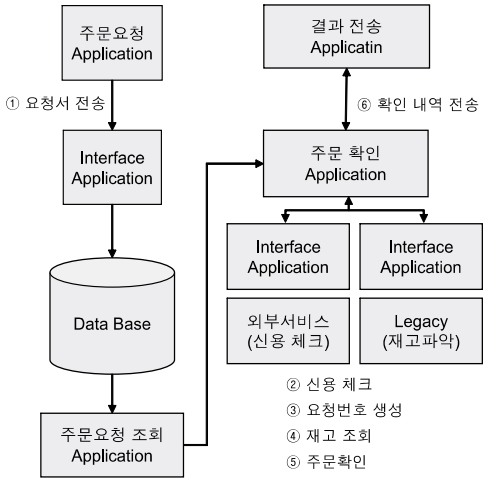



[그림 5] SOA Application Architecture

## 4.3 대안 Architecture2 - Web Application

일반적인 Web Application 형태로 구축된 방안

은 [그림 6]과 같다. SOA Application과 달리 기능 단위의 컴포넌트를 활용 하기 보다는 프로그램 안에서 Function 형태로 구축이 된다. 외부 연계는 웹 서비스를 활용 하는 방안도 가능 하지만, 파일 및 DB에 의한 연계 방안을 활용 하는 것으로 가정하였다. 웹 서비스를 포함하는 Web Application의 경우 초기 단계의 SOA Application이라고 볼 수 있기 때문에, 대안 Architecture1과 차별 점이 적어지게 된다.




[그림 6] Web Application Architecture

# 5. Architecture 평가 및 개선 방안

## 5.1 ATAM에 의한 정성적 평가

ATAM을 활용한 대안 Architecture 평가를 위해 Use Case 시나리오를 바탕으로 Quality Attributes Utility Tree를 작성한다. 작성된 Utility Tree를 기반으로 Architecture 접근법 분석을 통해서 Tradeoff, Sensitivity, Risk, Non-Risk를 식별 하게 된다. 그리고, Brainstorming에 의해서 Scenario의 순위를 주게 되고 Architecture 접근법의 재분석을 통해서 최종적인 Utility Tree가 작성되게 된다[8]. 최종적으로 작성된 Utility Tree는 <표 5>와 같다.

〈표 5〉 Quality Attributes Utility Tree

| Quality Attribute | Scenario | Rank (I,D) |
|---|---|---|
| Reliability | #1 데이터 전송에 대한 신뢰성 보장을 위한 에러 처리, 재전송, 전송 옵션, 동기/비동기 통신을 지원해야 한다. 구현 시간이 1D안에 가능해야 한다. | (H,M) |
| Availability | #2 시스템의 가용성을 96% 이상 보장 해야 한다. | (H,H) |
| Modifiability | #3 프로세스 및 Rule 변경에 따른 대응이 8H안에 가능해야 한다. | (H,H) |
| Performance (Real time) | #4 전체 프로세스 수행 시간이 1분 안에 처리 되어야 한다(자동화 처리 정도). | (H,H) |
| Inter operability | #5 다른 시스템과 연계 시에 인터페이스 호환을 보장하여 8H안에 연계가 가능해야 한다. | (H,M) |
| Performance | #6 동시 접속 100user 환경에서 100m/s 이내로 실행 Performance를 유지해야 한다. | (H,H) |
| Reusability | #7 유사한 프로그램 작성시에 재사용 가능한 컴포넌트 확보율이 30% 이상 되어야 한다. | (H,H) |

주) I : Importance, D : Difficulty

〈표 6〉 분석 결과

| Analysis Output | Architecture | 설 명 |
|---|---|---|
| Risk | SOA | #6 동시 사용자수의 증가에 따른 성능 저하와 장애 가능<br>#2 분산 객체 환경에서 시스템 가용성 문제 발생 가능 |
| | Web | #1 데이터 무결성 보장 한계 (Message 전송 개별 제어 필요) |
| Sensitivity Point | SOA | #6 개별적 트랜잭션에 대한 실행 성능 |
| | Web | #3 프로그램 수정을 최소화하는 유연성 부족 하고, 직접 Code를 수정 신규 배포 필요<br>#4 Real Time 처리 및 자동화 처리 어려움<br>#5 연계 인터페이스 호환성을 위한 추가 개발 필요<br>#7 재사용성 부족으로 생산성 및 유지 보수 문제 발생 가능 |
| Tradeoff Point | #1 전용 솔루션의 도입 시에 비용의 문제 발생<br>#3 프로세스나 룰의 변경이 많을 경우 코드 수정과 배포에 따른 장애 발생 가능성이 많아짐<br>#4 업무가 복잡하고 규모가 클수록 실시간 적인 처리가 중요해짐<br>#5 인터페이스 대상 시스템이 많을수록 호환성 문제 해결이 중요<br>#6 동시 접속 사용자 및 Traffic이 많을수록 Message 기반 처리에 한계<br>#7 기반 시스템의 경우 여러 시스템간 재사용 될 가능성이 많음 | |

ATAM의 분석에 의해서 시나리오 별로 Risk, Trade Off, Sensitivity에 대한 내용을 기술 하게 된다. 시나리오 별로 분석 결과를 종합하여 두 가지 대안 Architecture에 대하여 종합한 결과는 <표 6>과 같다.

## 5.2 CBAM에 의한 정량적 평가

ATAM에서의 시나리오를 정량적으로 검증하고 최종 ROI를 통한 Architecture 선정에 대한 근거 제시를 위하여 CBAM(CBAM2[9])을 사용한 평가가 유용하게 활용 될 수 있다[12]. CBAM을 이용한 평가 방법은 ATAM에서 결과를 기반의 시나리오를 정제하고 <표 7>에서처럼 반응 값(Response Goal)을 수립하고 <표 8>에서처럼 효용 값 (Utility Score)를 계산하게 된다. CBAM에서 Worst, Desired, Best에 대한 판단 범위를 제시하고, 이를 기반으로 효용 값(Utility Score)을 계산 하였다[12]. 본 연구에서는 Scenario Weight가 동일하다고 가정 하였으나, 여러 기관과 연계 되는 부분이 많은 경우 #5, 동시 사용자가 아주 많은 경우 #6, 자동화 및 실 시간 처리가 중요한 경우 #4의 비중이 더욱 높아지게 될 것이다.

〈표 7〉 반응 값(Response Goal) 수립 결과

| # | 수치 | Worst | Desired | Best |
|---|---|---|---|---|
| 1 | 구현 시간 | >1D | 1D | <1Min |
| | | 10 | 70 | 100 |
| 2 | 가용성 | <90% | 96% | >99% |
| | | 10 | 80 | 100 |
| 3 | 대응 시간 | >10D | 1D | <4H |
| | | 10 | 70 | 100 |
| 4 | 수행 시간 | >1D | 1Min | <1Min |
| | | 10 | 80 | 100 |
| 5 | 작업 시간 | >1D | 8H | <4H |
| | | 10 | 80 | 100 |
| 6 | 수행 시간 | >1Min | 100m/s | <10m/s |
| | | 10 | 70 | 100 |
| 7 | 재사용율 | 0% | 30% | 50%< |
| | | 0 | 70 | 100 |

〈표 8〉 효용 값(Utility Score) 계산 결과

| # | % | SOA | Web | 설명 |
|---|---|---|---|---|
| 1 | 15 | 100 | 70 | Web은 개별적 구현 필요 |
| 2 | 15 | 50 | 80 | 분산 객체 환경 가용성 |
| 3 | 15 | 90 | 70 | 설정 레벨에서 대응도 가능 |
| 4 | 15 | 100 | 80 | SOA의 경우 이벤트 기반 자동화 처리 가능 |
| 5 | 15 | 100 | 80 | 표준 기반 인터페이스 호환 |
| 6 | 15 | 50 | 90 | SOA 수행 성능의 한계 |
| 7 | 10 | 100 | 50 | SOA는 재사용 전제로 구축 |

두 가지 대안 Architecture에 대한 전체 이익을 계산 하기 위해서 개별적인 시나리오 별 효용 값(Utility Score)과 Weight를 곱해서 얻게 된 값의 총합으로 얻어지게 되며, 다음의 공식과 같이 표현된다. Bi = Sum(bi,j * Wj) (bi,j는 Architecture i에 의해서 j시나리오에 대한 효용 값이며, 주로 이전 Architecture에 의한 개선 효용을 말한다. Wj는 Weight를 말한다). SOA Application에 대한 전체 이익을 계산한 결과는 <표 9>와 같다.

〈표 9〉 전체 이익 계산 결과

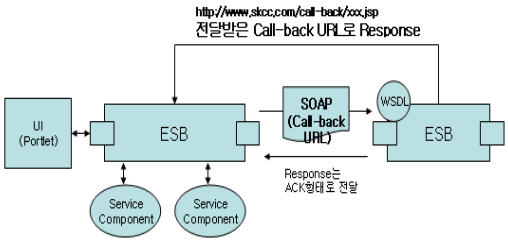
| 구분 | # | Weight | Architecture Strategy Benefit | | |
|---|---|---|---|---|---|
| | | | Raw | Normalized | Total |
| SOA | 1 | 15 | 100 | 1500 | 89.5 |
| | 2 | 15 | 50 | 750 | |
| | 3 | 15 | 90 | 1350 | |
| | 4 | 15 | 100 | 1500 | |
| | 5 | 15 | 100 | 1500 | |
| | 6 | 15 | 50 | 1350 | |
| | 7 | 10 | 100 | 1000 | |
| Web | 1 | 15 | 70 | 1050 | 75.5 |
| | 2 | 15 | 80 | 1200 | |
| | 3 | 15 | 70 | 1050 | |
| | 4 | 15 | 80 | 1200 | |
| | 5 | 15 | 80 | 1200 | |
| | 6 | 15 | 90 | 1350 | |
| | 7 | 10 | 50 | 500 | |

마지막 단계로 전체 이익이 계산 된 후에 Cost를 산정하여 ROI 계산하고 Architecture간 산정 결과를 비교 하게 된다. SOA Application 구축의 경우 Web Application 보다 많은 비용이 소요 되지만 업무 요구 사항과 정보 시스템 환경에 따라 프로세스 재사용, 통합 및 자동화를 통한 전체 이익을 증대시켜 충분한 ROI를 얻을 수 있다.

### 5.3 SOA 개선 방안

본 연구에서 ATMA, CBAM분석 결과 SOA의 Risk로 분석된 단위 트랜잭션 성능 문제에 대한 이슈를 해결 하기 위한 방안은 다음과 같다.

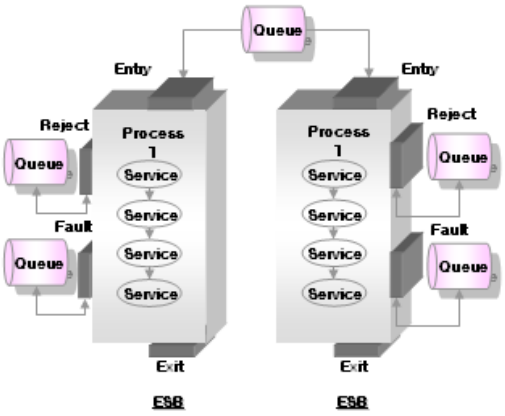
첫째, 평균적인 성능 측정을 통해 응답 시간이 긴 경우에는 [그림 7]과 같은 비동기 방식을 통한 통신을 수행 하도록 한다.




[그림 7] 비동기 방식 통신

둘째, 일반적 Application통합 환경을 구축하는 EAI (Enterprise Application Integration)처럼 해당 데이터를 다른 데이터베이스로 통째로 옮기는 형태가 아니라 서비스 컴포넌트를 호출 하고 결과를 받아오는 형태로 구축 하여 데이터 레벨의 통합이 아니라 어플리케이션 및 프로세스 레벨의 통합을 지향 하여, 컴포넌트간의 통신 및 Message 크기를 최소화 하도록 한다.

셋째, 다량의 부하가 예상 되는 경우에는 다수의 ESB를 배치하거나 업무 별로 프로세스를 나누어 ESB에 배치하여 [그림 8]처럼 Load Balancing이 되도록 한다. Load Balancing을 통하여 Message가 각 ESB당 적절한 입력/출력 Throughput을 유지하도록 해야 한다.

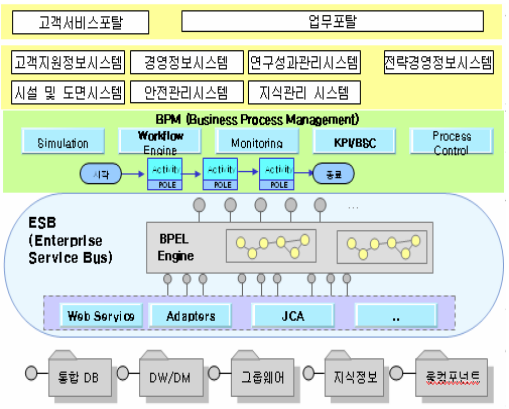



[그림 8] ESB Load Balancing 방안

## 6. SOA 적용 방안

### 6.1 SOA기반 차세대 시스템

현재 금융권을 중심으로 활발하게 도입이 추진되고 있는 차세대 시스템의 경우 주로 BPM 도입을 통해 업무 프로세스 최적화와 효율성 증대를 목적으로 하고 있다. 성공적인 BPM 구축을 위해서는 Workflow 통합과 함께 ESB를 도입 활용 하여 정보 시스템 내/외부적인 프로세스 기반의 시스템 레벨의 통합과 자동화가 필요하다.



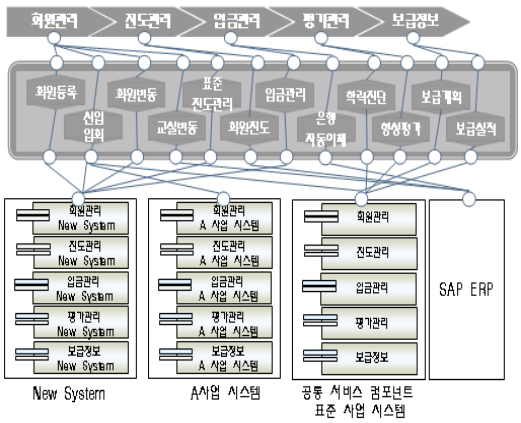

[그림 9] SOA기반 차세대 Architecture

본 연구에서 ATAM분석과 CBAM 분석을 통해 ESB기반 SOA 구축 시에 데이터 무결성 보장, 신뢰성 통신 보장, 유연성과 재사용성이 향상 될 수 있음을 알 수 있었다. [그림 9]은 연구소의 실시간 정보 시스템에서 활용된 SOA기반 차세대 Architecture 사례이다. ESB를 통해 내/외부 적인 정보 시스템을 통합하여, 컴포넌트 기반으로 구축하고 BPM(Business Process Management)과 연동 하여 프로세스 레벨의 통합을 구현한 사례(구축 중)이다.

### 6.2 SOA기반 공통 컴포넌트 시스템

기업의 신규 업무를 정보 시스템에 반영 시에 기존에 구축 되었던 것과 비슷한 형태의 중복 개발이 진행 되고, 업무 프로세스 변경 시에 중복 적용된 부분을 수정하는 작업이 진행 되어 비용 낭비와 효율성이 저하 되는 경우가 발생 했다.

이러한 문제를 해결하기 위해 비즈니스 프로세스를 분석하여 공통으로 활용 되는 Activity를 선별 하고 이를 공통 서비스 컴포넌트로 구축 활용하는 방안이 있다. [그림 10]는 ESB기반으로 공통 컴포넌트를 활용 하여 표준 적인 프로세스를 구축하고 사업 별로 다른 부분은 별도의 컴포넌트에서 호출 하여 사용 하는 사례(구축 예정)이다.




[그림 10] 공통 컴포넌트 활용 사례

## 7. 결 론

본 연구에서 SOA와 Web Architecture간의 정성적 정량적 분석을 통하여, 실시간 처리, 컴포넌트 기반 재사용, 업무 변경에 따른 유연성을 증대시킬 수 있다는 결론을 얻게 되었다. 다만 Web Architecture에서 충분히 활용 되고 있는 대용량 Traffic 처리가 어려운 단점이 있다.

SOA 사례의 부족과 구축 비용 및 시간에 대한 이슈로 관심에 비해 실제 도입은 아직은 활발하게 진행되고 있지 않지만, 차세대 Architecture로써 현재의 많은 S/W Architecture의 문제점을 해결할 수 있는 방안이며, 지속적인 연구와 노력을 통해 적용 방안을 발전시키고 실제 프로젝트에의 점진적인 적용과 검증이 필요 하다.

## 참 고 문 헌

[1] David A. Chappell, “Enterprise Service Bus”, O’Reilly, 2004.

[2] Jayatirtha Asundi, Rick Kazman, Mark Klein, “Using Economic Considerations to Choose Among Architecture Design Alternatives”, Technical Report CMU/SEI, 2001.

[3] J2EE Connector Architecture, http://java.sun.com /j2ee/connector.

[4] Len Bass, Paul Clements, Rick Kazman, *Software Architecture in Practice Second Edition*, Addison Wesley, 2003.

[5] Paul Clements, Rick Kazman, Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison Wesley, 2002.

[6] Robert L. Nord, Mario R. Barbacci, Paul clements, Rick Kazman, Mark Klein, Liam O’Brien, James E. Tomayko, “Interesting the Architecture Tradeoff Analysis Method (ATAM) with the Cost Benefit Analysis Method(CBAM)”, Technical Report

CMU/SEI, 2003.

[7] Roy Schulte, “The New Integration Scenario: Five Trends That Change How Application Software Works”, Gartner Application Integration and Web Service Summit, 2005.

[8] Rick Kazman, “ATAM: Method for Architecture Evaluation”, CMU/SEI-2000-TR-004.

[9] Rick Kazman, Jai Asundi, Mark Klein, “Making Architecture Design Decisions: An Economic Approach”, CMU/SEI-2002-TR-035.

[10] SOAP version1.2, http://www.w3.org/2000/TR/Group.

[11] Thomas Erl, *Service-Oriented Architecture : Concepts, Technology, and Design*, Prentice Hall PTR, 2005.

[12] Younbok Lee, Ho-Jin Choi, “Experience of Combining Qualitative and Quantitative Analysis Methods for Evaluation Software Architecture”, Proceedings of the ICIS ’05, 2005.

## ◈ 저 자 소 개 ◈

**임 철 홍 (imich@skcc.com)**

고려대학교 재료공학과를 졸업하고 공학대학원 전자컴퓨터 석사를 취득하였다. 정보처리 기술사 이며, 현재 SK C&C S/W 공학센터 기술전략팀에 재직 중이다. 주요 관심분야는 SOA, Semantic Web, S/W 아키텍처 등이다

**홍 도 석 (capcom@skcc.com)**

전북대학교에서 전산통계학 석사를 취득하였으며 박사과정을 수료 하였다. 현재 SK C&C S/W공학센터 기술전략팀에 재직 중이며 주요 관심분야는 SOA, Web Service, BRE, BPM 등이다

**최 정 준 (cjj@skcc.com)**

미국 뉴욕 주립대 스토니브룩에서 기술경영학 석사학위를 취득 하였다. 현재 SK C&C S/W공학센터 기술전략팀에 재직 중이며 주요 관심분야는 소프트웨어 모델링, 비즈니스 모델링 등이다.