# 웹 기반 시스템의 효율적인 유지보수를 위한 웹 개발 프레임워크의 설계

최재광*, 조은애**
*고려대학교 소프트웨어공학과
**고려대학교 컴퓨터학과
e-mail: {bondang*, eacho**}@korea.ac.kr

# A Design of Web Development Framework for Efficient Web based System Maintenance

Jae-Kwang Choi*, Eun-Ae Cho**
*Department of Software Engineering, Korea University
**Department of Computer Science & Engineering, Korea University

**요 약**

웹과 관련된 소프트웨어 산업이 대규모화, 다양화, 복잡화 되면서 소프트웨어 유지보수 비용이 증가하고 있다. 이로 인해 웹 기반 시스템의 효율적인 유지보수를 위한 다양한 방법들이 연구되고 있다. 최근에는 각 기업에서 웹과 Legacy 시스템과의 인터페이스를 위한 EAI(Enterprise Application Integration)방법들이 도입되고 있다. 그러나 이러한 각종 EAI 방법들의 도입에도 불구하고 웹 어플리케이션은 여전히 개발 및 테스트 시 Legacy 시스템의 서비스 개발 진척에 종속적이라는 한계가 있다. 본 논문에서는 이러한 웹 기반 시스템이 갖고 있는 유지보수의 문제점을 소프트웨어 산업현장에서 많이 활용되고 있는 6 시그마 기법을 통해 분석하였으며, 분석한 내용을 바탕으로 Legacy 시스템에 종속적이지 않는 효율적인 웹 개발 프레임워크(Efficient Web Development Framework)를 설계하여 개발 생산성을 향상시키도록 하였다.

## 1. 서론

최근 웹 기반 시스템 구축이 많아지고 있으며, 변경과 사용자 인터페이스(User Interface, UI) 부분의 리뉴얼에 대한 요청도 많아지고 있다. 이와 더불어 비용과 인력의 감축에 대한 압력 또한 적지 않다. 웹 기반 개발 시 대기시간 발생이 잦은 데, Legacy 시스템과 같은 타 시스템과의 인터페이스를 통한 개발 시, 인터페이스가 없는 내부 개발 보다 많은 개발시간이 소요되고 있으며, 고객이 요구한 납기를 지연시키는 원인이 되고 있다. 이와 같이 Legacy 시스템에 종속적이지 않는 웹 UI 자체 개발 및 테스트를 위한 효율적인 환경 구성은 웹 기반 시스템의 개발 생산성을 위한 필수적인 조건이라 할 수 있다[1][2].

웹 UI 부분 개발과 더불어 웹 기반 시스템에서 중요한 부분의 또 다른 하나는 웹 기반 시스템의 유지보수이다. 기업시스템에 대한 유지보수의 중요성은 오래 전부터 인식되어 왔다[3]. 1970년대 후반 UCLA대학의 Lientz와 Swanson 교수는 설문조사에 의해서 많은 기업이 개발보다 유지보수에 훨씬 더 많은 인력이 필요하다고 보고한 바 있다[4]. 이러한 현상은 그 후 30년 이상이 지난 현재 많은 신기술이 등장하였음에도 불구하고 개선되지 않고 있다[5]. 비용면에서 보면 유지보수는 개발과 비교해서 그 비용이 2/3에서 무려 4/5까지 달한다는 주장도 있다[6].

본 논문에서는 웹 기반 시스템의 유지보수를 수행하는 데 있어서 개발 생산성에 대한 효율화 분석을 위해 6 시그마의 DMAIC 기법 중 선행 2 단계를 활용하였으며, 웹 기반 시스템 유지보수 생산성 향상의 장기적 관점인 CTQ(Critical To Quality)를 선정하여 기업의 비전이나 목표를 달성하기 위하여 유지보수의 개선대상을 구체화하여 명확히 하였다. 또한 웹 어플리케이션 개발을 수행하는 데 있어서 향상된 방법으로 Legacy 시스템과의 병행 개발 시 서로간의 종속적 관계를 최소화 하기 위한 방법으로 웹 시스템 자체에 대해 개발 및 테스트 할 수 있는 방법을 제시하였다. 이렇게 함으로써 개발을 수행하는 데 웹 UI 부분과 Legacy 시스템과의 인터페이스에 대한 상호간의 개발 의존성을 최소화하였으며, 나아가 프로그램 개발시간을 단축할 수 있도록 하였다. 먼저, 효율성을 향상시키기 위해서 EWDF(Efficient Web Development Framework)를 제안한다.

뿐만 아니라 본 논문에서는 웹 기반 시스템에서 웹 UI 부분 개발 및 테스트에 초점을 두고 개발하여 I/O 를 수정하는 경우, 웹 UI 부분 자체가 변경되는 경우, 또한 Legacy 의 I/O 가 아직 설계되지 않았을 경우에도 효

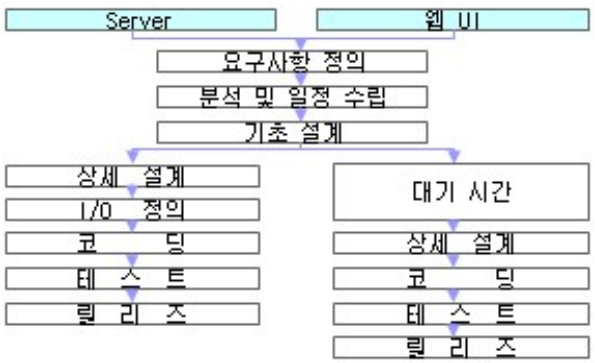
율적으로 개발할 수 있는 모델을 제안한다.

논문의 구성은 다음과 같다. 제 2 장에서는 기존 인터페이스 웹 기반 시스템과 6 시그마를 통한 웹 기반 시스템의 문제 정의에 대해서 기술한다. 제 3 장은 제안 모델의 설계를 위해 웹 유지보수 문제점을 통한 CTQ 도출과 EWDF 설계에 대해 기술한다. 제 4 장에서는 구현에 대해서 기술한다. 제 5 장에서는 비교 및 평가에 대해서 기술하였다. 제 6 장에서 결론과 향후 연구에 대해 제시하였다.

## 2. 관련 연구

### 2.1 기존 인터페이스 웹 기반 시스템

기존 Legacy 시스템과의 인터페이스 웹 어플리케이션 개발 및 테스트를 수행하기 위해 별도로 웹 UI 부분의 소스코드 수준에서 개발 및 테스트를 위한 추가적인 작업 절차를 거쳐야만 했다. 예를 들면, 웹 UI 에서 Legacy 시스템의 서비스와 연계되기 위해서는 공통 헤더, 입출력 데이터에 대한 인터페이스 레이아웃 전문을 각 프로그램 소스코드 수준에서 각각 정의하여 해당 서비스의 호출을 시도하며, 서비스 프로그램의 결과 값을 호출한 클라이언트 측에 되돌려 주는 방식이 보통이다. 즉, 클라이언트와 서비스가 동시에 개발이 시작되는 경우라 하더라도 해당 프로그램의 개발 및 테스트의 가능 시점이 서로 일치하지 않는다. 대량의 변경 개발이 수행되는 경우가 특히 그러한 데, 이는 규모가 더욱 커지고 복잡화 되어 가고 있는 유지보수 시스템들과 다양한 고객의 요구사항에 그 원인이 있다고 볼 수 있다. 더욱이 웹 UI 부분의 개발 및 테스트 착수시점의 차이는 Legacy 시스템의 I/O 설계 및 서비스 변경 완료 여부에 영향을 받는다. 따라서 서로간에 의존적이지 않은 개발 및 테스트를 수행할 수 있는 방법이 필요하다. Legacy 시스템과의 인터페이스 개발 은 서비스 부분과의 개발 및 테스트 시점의 불일치로 (그림 1)과 같이 웹 UI 부분에 크고 작은 대기시간이 발생하게 된다.




(그림 1) 기존 인터페이스 웹 개발 절차

### 2.2 6 시그마를 통한 웹 기반 시스템의 문제 정의

6시그마란 경영혁신과 품질 향상을 위한 프로그램중의 하나로서, 궁극적인 목적으로 객관적인 정보에 의거하여 프로세스를 개선하거나 합리적인 의사결정을 하도록 하는 일련의 활동이라 할 수 있다[7]. 이는 혁신적인 변화를 가져올 수 있는 명쾌한 해결책으로서 통계적 기법과 자료, 경영기법, 회의기법 등을 총체적으로 묶은 하나의 프로세스이다. 이와 같은 6시그마를 활용하여 웹 시스템의 효율적인 유지보수를 위한 모델을 설계하도록 확장할 수 있다. 6시그마의 DMAIC 기법의 5단계인 정의(Define), 측정(Measure), 분석(Analyze), 개선(Improve), 관리(Control) 중 선행 2단계는 다음과 같다.

- 1단계: 프로젝트를 선정하는 정의 단계로서 문제와 현상을 명확히 정의하고, 고객 및 VOC(Voice of Customer)를 구체적으로 파악하고 CTQ(Critical to Quality)로 구체화 시킴[8].
- 2단계: 1단계에서 도출된 CTQ를 가장 잘 대변할 수 있는 측정 가능한 지표 Y를 정의하여 현재 수준을 파악하고 성과지표 Y의 변동에 영향을 미치는 잠재 원인변수(X's)를 발굴하는 측정 단계이다[8].

## 3. 제안 모델 설계

### 3.1 웹 유지보수 문제점을 통한 CTQ 도출

먼저, 서론에 기술한 웹 유지보수 문제점들을 고객 관점의 요구사항과 비즈니스 관점의 요구사항을 구체적으로 파악하여 CTQ(프로그램 개발소요시간)를 도출하였다. 이는 프로그램 개발소요시간 단축을 통해 전체적으로 향상된 웹 개발 생산성을 기대할 수 있기 때문이다. 측정 단계에서의 특성요인도(Cause & Effect Diagram) 분석을 통해 프로그램 개발소요시간에 영향을 주는 잠재인자를 발굴하기 위해 A 사 웹 관련 시스템의 최근 1년 동안 200 여건의 SR(Service Request)을 기준으로 유형별 평균 개발소요시간을 측정하였다. 현수준(공정능력)을 확인한 결과 다음과 같이 웹 유형별 프로그램 본당 평균 개발소요시간의 차이에 대한 결과값은 <표 1>과 같다.

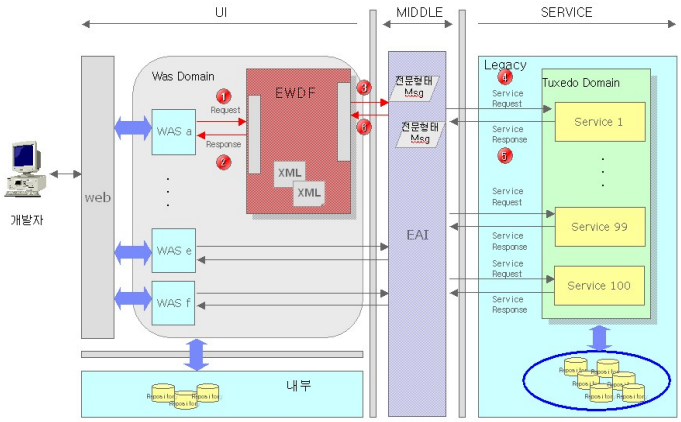
<표 1> A 사 웹 유형별 평균 개발소요시간

| 유형 | 평균 개발소요시간(h) |
|---|---|
| 내부 | 3.16 |
| Legacy 시스템 인터페이스 | 4.88 |
| 인증 | 3.18 |
| 상품정보 | 4.17 |
| 기타 | 3.07 |

이는 내부 개발 대비 Legacy 시스템 인터페이스 개발 시 웹 영역 개발과 비즈니스 영역에 해당하는 서비스 개발간의 개발 프로세스 차이와 테스트 방법 차이로 인해 프로그램 개발소요시간에 부정적 영향을 준다고 할 수 있다. 이와 같은 불필요한 소요시간에 대한 대부분의 대기 시간 발생에 대한 해결 방법으로 EWDF 를 설계하였다.

### 3.2 EWDF 설계

보통 웹 기반 시스템 프로세스는 Legacy 시스템과의 인터페이스가 이루어는 상황에서 (그림 2)와 같이 크게 UI 부분, MIDDLE 부분, SERVICE 부분으로 구성되며, 해당 기업의 아키텍쳐 설계에 따라서 MIDDLE 부분은 생략될 수 있다.




(그림 2) 웹 기반 시스템 프로세스

다음 (그림 3)은 EWDF와 연계된 부분을 UML의 유즈케이스 다이어그램(Usecase Diagram)으로 나타낸 것이다[9].


(그림 3) EWDF와 연계된 유즈케이스 다이어그램

(그림 3)의 유즈케이스 다이어그램에서 보면 관리자에 의해서 만들어진 테스트 케이스를 개발자가 우선 기능단위로 개발 및 테스트를 할 수 있도록 지원하며, 개발자가 최종적으로 EAI 원격 서비스와 인터페이스 테스트를 수행하도록 연계된다. EWDF(Efficient Web Development Framework) 설계에 대한 기본 가정은 UI 부분과 서비스 부분의 기본설계가 이루어져 상호간의 인터페이스 레이아웃 전문에 대한 확정이 이루어졌다고 정의한다. 제안 모델은 다음과 같이 구성되어 있다.

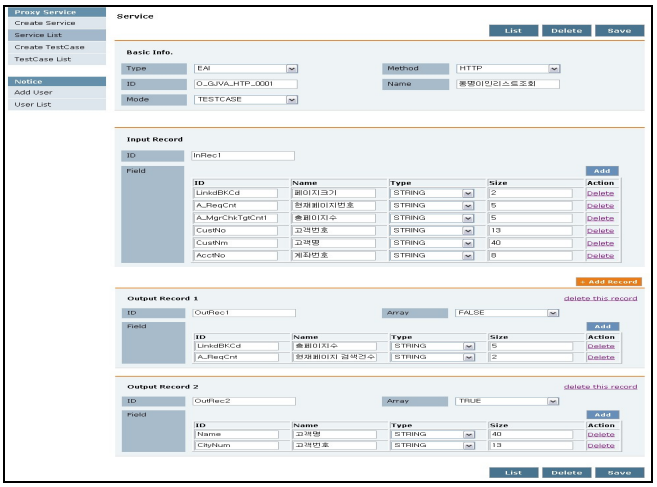
- UI 관리자가 EWDF에 Input 값을 입력한다.
- UI 관리자가 EWDF에 반복을 고려한 Output 값을 입력한다.
- 입력된 Input, Output 값을 한 쌍으로 EWDF에 가상의 인터페이스 XML 문서를 개별적으로 생성시킨다. 생성된 가상의 인터페이스 XML 문서는 UI 프로그램에서 호출 시 로드되며, Legacy 영역의 서비스 프로그램의 완료 여부와 상관 없이 개발 및 테스트를 수행할 수 있게 된다.
- 서비스 프로그램이 완료되면 EWDF와 인터페이스 테스트를 수행하여 UI 부분과의 테스트 이전에 서비스 프로그램의 인터페이스 Layout 전문 오류를 사전에 검출한다.
- 최종적으로 UI 프로그램과 가상의 서비스 역할을 담당하는 EWDF와 연계 테스트를 한다.
- EWDF와 서비스 프로그램의 테스트 완료 후 UI 프로그램과 서비스 프로그램과 연계되도록 한다.

이는 서비스 프로그램의 호출을 위해 개발 및 테스트를 위한 UI 부분 영역의 불필요한 개발 작업을 제거한다는 가정에서 설계되었으며[10], 서비스 프로그램의 완료 여부와 상관없이 용이하게 개발 및 테스트가 가능하도록 설계되었다.

## 4. 구현

설계된 EWDF의 구현된 환경은 다음과 같다. 웹 서버는 SUN사의 E4500로서 S/W는 SunOS 5.8, JDK 1.4.2, iPlanet 6.1을 사용하였으며, WAS(Web Application Server) 서버는 SUN사의 E4500로서 S/W는 SunOS 5.8, JDK 1.4.2, Jeus 4.2(Tmax사의 WAS)가 탑재되어 있는 것을 사용하였다.

제안된 EWDF는 Legacy 시스템과 인터페이스를 담당하게 되는 데, 중간에 최근 대부분의 웹 솔루션 개발들이 Middle Layer로 EAI를 두고 있으며[11], 이를 고려하여 EWDF가 설계 및 구현되었다. 우선 데이터 컬럼의 속성 및 사이즈 정의를 위해 메타 데이터를 (그림 4)와 같이 화면을 통해 입력한다.




(그림4) 메타 데이터 입력 화면

자료 입력 후 Save 버튼을 클릭하면 (그림 5)와 같이 자동으로 메타 데이터의 XML 문서를 생성한다.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SERVICE ID="O_GJVA_HTP_0001" NAME="동명이인리스트조회" TYPE="EAI" METHOD="HTTP">
  - <INPUT>
    - <RECORD ID="InRec1" ISARRAY="FALSE">
        <FIELD ID="LinkdBKCd" NAME="페이지크기" SIZE="2" TYPE="STRING" />
        <FIELD ID="A_ReqCnt" NAME="현재페이지번호" SIZE="5" TYPE="STRING" />
        <FIELD ID="A_MgrChkTgtCnt1" NAME="총페이지수" SIZE="5" TYPE="STRING" />
        <FIELD ID="CustNo" NAME="고객번호" SIZE="13" TYPE="STRING" />
        <FIELD ID="CustNm" NAME="고객명" SIZE="40" TYPE="STRING" />
        <FIELD ID="AcctNo" NAME="계좌번호" SIZE="8" TYPE="STRING" />
      </RECORD>
    </INPUT>
  - <OUTPUT>
    - <RECORD ID="OutRec1" ISARRAY="FALSE">
        <FIELD ID="LinkdBKCd" NAME="총페이지수" SIZE="5" TYPE="STRING" />
        <FIELD ID="A_ReqCnt" NAME="현재페이지 검색건수" SIZE="2" TYPE="STRING" />
      </RECORD>
    - <RECORD ID="OutRec2" ISARRAY="TRUE">
        <FIELD ID="Name" NAME="고객명" SIZE="40" TYPE="STRING" />
        <FIELD ID="CityNum" NAME="고객번호" SIZE="13" TYPE="STRING" />
      </RECORD>
    </OUTPUT>
  </SERVICE>
```

(그림 5) 자동 생성된 XML 문서

가상의 인터페이스 데이터 생성을 위해 Input Data, Output Data를 각기 다른 화면을 통해 입력한다. 이는 Legacy 시스템과의 실질적인 인터페이스 없이 UI 자체만을 위해 개발 및 테스트를 수행하기 위한 또 다른 XML 문서를 생성하게 된다. 자료 생성 시 메타 데이터의 속성값을 참조하여 Input Data, Output

Data를 생성한다.

## 5. 비교 및 평가

본 논문에서 제안 모델을 시험 적용, 결과를 확인하기 위해서 다음과 같이 3 가지 경우를 고려하여 기존시스템과 제안시스템을 비교, 평가하여 동일한 결과에 대한 변경 규모를 산출하였다. 또한 결과를 검증하고 개선효과를 확인하였다.

<표 2> 시험 결과(I/O 수정, UI 변경)

| 변경 구분 | 시스템 구분 | 변경 규모 |
|---|---|---|
| I/O 수정 | 기존시스템 | 프로그램 6본 |
| | 제안시스템 | 프로그램 3본, 데이터 3건 입력 |
| UI 변경 | 기존시스템 | 화면 변경 1개, Legacy 테스트 데이터 생성 |
| | 제안시스템 | 화면 변경 1개, 데이터 3건 입력 |

**(1) I/O 수정** : 프로그램 수정 사항이 I/O 에 제한되며 기존 I/O 변수의 길이 변경 또는 변수의 추가, 삭제인 경우

<표 2>와 같이 기존시스템에서는 ① 해당 화면(기능)에 대한 프로그램 소스 분석, 설계 ② 프로그램 6본 수정 ③ 컴파일 ④ 서비스 재기동의 절차를 거치게 된다. 그러나 제안시스템에서는 프로그램 3본 수정 및 EWDF의 화면을 통해 데이터 3건을 입력하는 것으로 절차 및 변경 규모가 감소 되었다.

**(2) UI 변경** : Legacy 와의 I/O 의 변경은 없으며 UI 변경에 따른 프로그램이 변경 되는 경우, 테스트 데이터의 존재 유무에 따른 테스트 소요시간

<표 2>와 같이 화면 1 개를 개발 및 테스트 하기 위해 Legacy 시스템에는 테스트 데이터를 생성하기 위한 소요시간이 제안시스템에서는 데이터 3 건의 입력으로 대체되었다. 이는 변경 규모가 커질수록 기존시스템 대비 제안시스템에서 개발 소요시간에 있어서 효율성이 높다.

**(3) Legacy I/O 미 설계** : UI 정의는 완료 되었으나 서비스 프로그램의 I/O가 미완료 되었을 경우

인터페이스 개발을 위한 UI 부분과 서비스 부분과의 I/O 가 설계되지 않은 상태이므로 변경규모와 소요시간의 시험 결과값을 얻을 수는 없지만 UI 정의가 완료된 상태이므로 UI 자체를 위한 가상의 Input Data, Output Data 를 미리 생성하여 UI 개발을 자체적으로 수행할 수 있다. 후속으로 I/O 정의가 완료되면 품질 향상 작업을 기대할 수 있다.

이상과 같이, 3 가지 경우의 시험을 통해 <표 3>은 기존시스템과 제안시스템 간 비교 평가를 보여준다.

<표 3> 비교 평가

| 비교항목 | 기존시스템 | 제안시스템 |
|---|---|---|
| 대기시간 | 많음 | 적음 |
| 편리성 | 낮음 | 높음 |
| 연속성 | 낮음 | 높음 |
| 결합도 | 높음 | 낮음 |
| 응집도 | 낮음 | 높음 |

<표 3>과 같이 기존시스템 대비 제안시스템은 Legacy 시스템의 서비스 개발 지연에 따른 대기시간이 적게 소요되며, 개발 및 테스트를 수행하는 데 개발자 입장에서 편리성이 더욱 높아지게 되었다. 또한 개발자가 중도에 다른 영역의 개발을 수행하더라도 EWDF 내에 있는 테스트 케이스를 활용해 현재의 개발자 및 타 개발자에게 개발의 연속성을 높일 수 있게 되었다. 이는 기존시스템에 비해 제안시스템이 웹 기반 개발을 수행하는 데 있어서 Legacy 시스템과의 상호 밀접한 개발 환경의 결합도가 낮은 반면, 웹 UI 개별 클래스 자체의 완전성에 대한 응집도가 높은 장점이 있다고 볼 수 있다.

## 6. 결론 및 향후 연구

본 논문에서 제안된 EWDF 를 통해 웹 기반 시스템의 UI 부분에서 I/O 수정, UI 변경, Legacy I/O가 미 설계된 경우에 기존시스템 대비 제안시스템에서 개선된 개발 및 테스트 기반을 제공하게 된다. 개인의 역량에 의존적인 기존의 웹 개발 방법에서 효율적인 개발과 테스트를 수행 할 수 있도록 보완된 인터페이스 모델을 추가함으로써 각 개발자마다 인터페이스 웹 개발 작업을 하는데 불필요하게 낭비되는 대기 시간이 감소되었으며, Legacy 시스템과의 인터페이스 개발에 소요되는 시간이 단축되었다. 향후 연구로는 제시된 모델을 확대 적용하여 적용한 시스템과 기존 시스템과의 성능 및 확장의 유연성, 유지 보수 시 개발 생산성을 증명하고 신뢰성을 확보하도록 하겠다. 또한 다양한 웹 인터페이스 기반을 고려하여 적용 가능 범위를 확대하도록 연구할 예정이다.

## 참고문헌

[1] 권용래, “소프트웨어 생산성의 명암”, 정보과학회지, Vol.17, No.12, 1999

[2] Hongjun Su, Stephen Jodis, Hong Zhang, “Providing an integrated software development environment for undergraduate software engineering courses”, Consortium for Computing Sciences in Colleges, Vol.23, No.2, 2007

[3] Lientz, B.P., and Swanson, E.B., "Problems in application software maintenance", *Commun. ACM*, Vol.24, No.11, pp. 763~769, 1981

[4] Lientz, B.P., and Swanson, E.B., "Discovering issues in software maintenance", *Data Manage*, Vol.16, No.10, pp. 15~18, 1978

[5] 송영재, 김귀정, 변정우, 서영준, 최한용, 한정수 객체지향모델링과 CBD 중심 소프트웨어공학, 이한출판사, 2003

[6] IEEE, *IEEE Software Engineering Standards COllection*, IEEE, 1993

[7] Michael L. George, “Lean Six Sigma, McGraw-Hill, 2007

[8] 주종문, 6 시그마프로젝트추진도구, 글로벌, 2007

[9] 한국소프트웨어기술진흥협회, UML 이론과 실제, 한국소프트웨어컴포넌트컨소시엄, 2006

[10] B W. Boehm, “Improving Software Productivity”, IEEE Computer, Vol.20, No.9, 1987

[11] 이장석, 홍정기, 지정권, “비즈니스 민첩성 향상을 위한 EAI 접근 방안 및 구현전략”, 정보과학회지, Vol.22, No.7, pp. 13~21, 2004