온라인논문투고관리시스템(KISTI-ACOMS)의 통합 버전 관리

Unified Version Management of KISTI-ACOMS

정희석, 최명석, 박재원, 강무영
한국과학기술정보연구원

Jeong hee-seok, Choi myung-seok, Park jae-won, Kang moo-yeong
Korea Institute of Science and Technology Information

요약

현재 270개 이상의 학회에 설치된 온라인논문투고관리 시스템 KISTI-ACOMS는 개발 당시 예상에 비하여 훨씬 빠른 속도로 설치 학회 수가 늘어남으로 인하여 많은 양의 소스 코드를 관리하기가 점점 어려워지고 있다. 또한 이 프로그램의 본질적 특성상 빈번하게 학회에서 들어오는 수정 요구에 의한 소스 코드의 수정은 체계적인 프로그램 관리를 더욱 어렵게 한다. 이에 KISTI에서는 여러 학회의 서비스를 한 버전의 소스 코드로 통합 관리하는 체계를 고안하였으며 이 시스템의 개발로 유지/보수 측면에서 관리해야 할 소스 코드의 양이 대폭 줄어들 뿐 아니라 기존에 비하여 사용자들의 요구사항에 대하여 훨씬 능동적인 대처가 가능해질 것이다.

Abstract

KISTI-ACOMS(Online peer review system), which is installated for more than 270 learned societies, is getting harder to manage large amount of source codes, because the number of installation has been growing much faster than we expected at the development time. Moreover the fundamental problem of this system, that frequent requests of modification from societies comes with modification of source codes, makes it more difficult to manage this program systematically. Thereupon we formulated a system to provide services for many societies using one version of source code, and this system is expected to reduce the managed amount of source codes and to make us cope with requests of societies of more positive than until now.

I. 서 론

1. KISTI-ACOMS

KISTI 국내 과학기술 연구자들을 대상으로 학술 정보를 효율적으로 제공하기 위하여 1995년부터 학회 정보화 사업을 수행하며 2000년도부터는 학회 회원 관리, 학술 정보의 수집 및 다양한 서비스 등 학회에 필요한 여러 가지 업무와 학회의 학술지 발행에 이르는 모든 처리 과정을 온라인 방식으로 수행할 수 있는 온라인논문투고관리시스템을 개발하여 보급하여 왔다[1]. 2005년 모든 학회에 적용할 수 있는 시스템을 목표로 기존 KISTI-ACOMS 시스템을 확장하여 재개발하였으며[2], 그 이후 보급도 급속하게 늘어 2007년 10월 현재 273개의 학회에 대하여 설치되었고 101개의 학회에서 학술지 정규 논문 심사나 학술대회 개최에 활용하고 있다. 적극적인 학회들에서는 KISTI-ACOMS의 영문 서비스를 사용하여 영문 학술지 발행이나 국제 학술대회 개최를 하는 등 학회 위상을 제고하는데 본 시스템을 활용하기도 한다. 많은 학회들의 업무 프로세스 시스템화에 대한 열망에 의하여 KISTI-ACOMS의 보급 및 활용은 앞으로도 더욱 확대될 것이다.

2. 통합 버전 관리의 필요성

지금까지 KISTI-ACOMS를 하나의 학회에 설치를 한 다는 것은 사본이 하나씩 늘어나는 것을 의미했다. 이런 구조를 가지게 된 이유는 학회별 커스터마이징 가능성과 보안에 관한 과도한 우려가 얽혀 이루어진 것이다. KISTI-ACOMS에서 커스터마이징이란 실제로 사용하는 학회들 중에 표준 기능에 비해 일부 정보를 더 보여주거나 조작하기 위하여 로직을 고치거나 해당 학회 사람들이 가장 잘 알아들을 수 있는 안내문을 추가로 다는 등의 소스 코드를 수정하는 것을 의미한다. 비록 커스터마이징 내역이 장부로 관리되고는 있지만, 수시로 들어오는 모든 요구사항이 기록된다고 보기는 어렵다. 게다가 원래 KISTI-ACOMS 2.0 설계 당시 예상했던 보급 속도에 비해 훨씬 빠른 속도로 보급된 것 역시 요구사항 처리에 급급한 나머지 관리가 어렵도록 수정된 소스 코드를 양산하는 원인이 되었다. 이렇게 한 번 수정된 소스는 프로그램의 버그 수정 등 일괄적으로 이루어지는 작업에서 배제되므로 해당 학회에 대한 지속적인 지원이 어려워지게 한다. 하지만 기존에는 이런 측면에서 KISTI-ACOMS의 보급 뒷면에서는 이러한 소스

코드 버전 관리의 어려움이 논의되어 왔다.

본 연구에서는 이러한 어려움들을 효과적으로 해결하기 위하여 소스 코드의 통합 버전 관리가 가능해진 새 버전의 KISTI-ACOMS 개발 방법론을 제시하고 구현하였다. 2장에서는 통합 버전 관리 기술과 개발 관리 방법을 상세하게 설명하고 3장에서는 본 기술로 이루어질 효과를 예상하고 비교 평가하며 4장에서 향후 해야 할 일들에 대해 논의한다.

II. 구현 방법

1. 통합 버전 관리 기술

[표 1] ACOMS 개발을 위한 시스템 환경

프로그램명	사용 목적
Windows XP	OS (개발자 PC)
Linux	OS (서버)
JDK 1.5	프로그래밍 언어
Tomcat 6.0	서블릿 엔진
Eclipse 3.3	통합 개발 환경
서브버전	소스 코드 버전 관리 (저장소)

1.1 시스템 환경

KISTI-ACOMS를 개발하고 실행하기 위한 환경은 표 1과 같다. 이번 개발과 기존 개발의 특별한 차이라면 이클립스라는 통합 개발 환경을 사용하고 서브버전이라는 버전 관리 도구를 사용한다는 점이다.

1.2 ACOMS MV 모델과 컴퍼넌트 기반 프레임워크

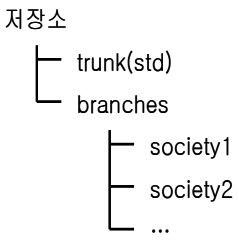
MVC(Model-View-Controller)는 실제로 애플리케이션의 작업을 하는 객체인 모델과 별개로 모델을 보여주는 뷰, 사용자와 뷰/모델 간의 인터페이스 역할을 하는 컨트롤러로 구성된다[3]. MVC는 모델에 독립적인 뷰의 선택을 더 유연하게 만들어 준다[4].

새 ACOMS 개발의 모델인 컴퍼넌트 기반 ACOMS MV 모델은 모델 2의 MVC와 기본 개념이 같다. ACOMS MV에서 서블릿에 대한 요청은 ACOMSServlet이라는 서블릿에서 처리하는데, 컨트롤러 역할을 하는 JSP에서 지정한 로직 모델에서 실행한 결과를 뷰 JSP에서 보여주는 것이다. 이 방식은 ACOMS MV 개발에 있어서 같은 로직 모델 클래스에 일부 로직 컴퍼넌트를 컨트롤에서 바꾸어 넘겨줌으로써 로직 모델의 중복 개발을 막고, 뷰나 로직 모델만 바뀌는 경우 나머지 소스 코드들은 표준 소스 코드를 그대로 이용할 수 있다는 장점이 있었다.

ACOMS MV 개발에서의 소스 코드 작성의 목표는 같은 일을 하는 기능의 코드를 중복해서 개발하는 일을 최소화하며 최대한 공통의 이해도를 높이기 위하여 구조화되고 통일된 형식의 코드를 작성하는 것이다. 이를 위하여 컴퍼넌트 기반 프레임워크를 만들었다. 프레임워크는 애플리케이션 개발자들이 정상적인 상황만 Exception 자동 처리 기능, 최적화된 자원 관리, 동적으로 SQL을 변형할 수 있어 재활용성을 극대화된 SQL 컴퍼넌트, 인터페이스를 통일시킨 메시징 컴퍼넌트, 권한 체크 기능, 개발자별로 로그메시지를 관리하기 위한 통합 로거, SQL과 파일 조작을 아우르는 트랜잭션 관리, 개발자 PC로 업로드된 파일을 일원화하여 서버에서 관리하기 위한 가상 파일 시스템 등을 포함한다. 이 프레임워크를 이용하여 웹 애플리케이션을 개발하는 개발자들이 해야 할 일은 로직의 정상적인 흐름에 따라 컴퍼넌트를 조합하고 배치하는 것이 되었다.

1.3 저장소 구성

서브버전을 이용한 저장소는 그림 1과 같이 구성된다. 표준 컨텍스트인 std가 trunk에 저장된다. 실제 학회 컨텍스트들은 branches에 저장된다. 각 컨텍스트들의 내용 중에서 저장소에 들어갈 것들은 Java나 JSP 등의 소스 코드, 설정 파일 등이다. 바이너리 파일인 Java 클래스 파일들은 저장소에 저장하지 않으며, 이클립스에서 자동으로 컴파일된 것들을 그대로 이용한다.



▶▶ 그림 1. ACOMS 저장소 구성

1.4 톰캣 기본 모듈 수정 및 환경 설정

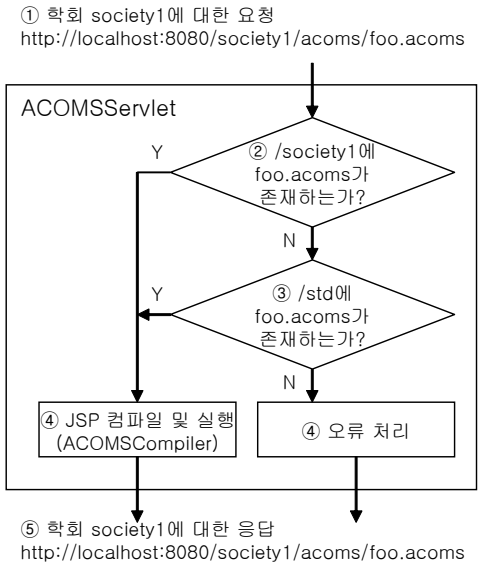
KISTI-ACOMS에서 각 학회는 톰캣 서버의 하나의 컨텍스트가 된다. 본래 톰캣은 컨텍스트별로 다른 컨텍스트의 JSP들을 참조하지 못하기 때문에, 기본적으로 제공하는 컴파일러를 사용하는 것만으로는 학회별로 컨텍스트를 사용해야 하는 ACOMS MV 모델의 통합 버전 관리가 불가능하다. 따라서 우리가 필요로 하는 기술을 구현하기 위하여 톰캣의 기본 모듈을 약간 수정하였다. 수정된 컴파일러는 톰캣의 공용 lib 디렉토리에 acoms_compiler.jar라는 라이브러리 파일 형태로 존재한다.

[표 2] Tomcat 6.0의 web.xml 추가 설정 내용

```
<servlet>
    <servlet-name>ACOMSMV-View</servlet-name>

<servlet-class>org.apache.jasper.compiler.ACOMSServlet</servlet-class>
    <init-param>
        <param-name>fork</param-name>
        <param-value>false</param-value>
    </init-param>
    <init-param>
        <param-name>compiler</param-name>

<param-value>org.apache.jasper.compiler.ACOMSCompiler</param-value>
    </init-param>
</servlet>
<servlet>
    <servlet-name>ACOMSMV-Control</servlet-name>

<servlet-class>org.apache.jasper.compiler.ACOMSServlet</servlet-class>
    <init-param>
        <param-name>fork</param-name>
        <param-value>false</param-value>
    </init-param>
    <init-param>
        <param-name>compiler</param-name>

<param-value>org.apache.jasper.compiler.ACOMSCompiler</param-value>
    </init-param>
</servlet>

<servlet-mapping>
    <servlet-name>ACOMSMV-Control</servlet-name>
    <url-pattern>*.acoms</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>ACOMSMV-View</servlet-name>
    <url-pattern>*.viewf</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>ACOMSMV-View</servlet-name>
    <url-pattern>*.view</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>ACOMSMV-View</servlet-name>
    <url-pattern>*.jspf</url-pattern>
</servlet-mapping>
```

톰캣의 설정 파일인 web.xml에 표 2와 같은 내용을 추가한다. 표 2의 소스 코드는 view, viewf, jspf를 확장자를 가진 ACOMSMV-View라는 JSP 파일들과 acoms라는 확장자를 가진 ACOMSMV-Control JSP 파일들을 ACOMSServlet를 통하여 ACOMSCompiler가 처리하게 한다.

ACOMSServlet을 통해 처리되는 과정은 그림 2와 같이 요청받은 위치(예: society1)에 JSP 파일이 있으면 바로 컴파일해서 실행하며, 없는 경우 std 컨텍스트에 같은 위치를 한 번 더 검사하여 있으면 실행하고 없으면 오류 처리하게 된다. ACOMSCompiler는 JSP를 .java 소스 코드로 바꾸는 과정에서 ACOMS MV 모델의 컴퍼넌트 관리와 개발자의 위한 소스 코드 등을 자동으로 .java 파일에 추가해 준다. 이렇게 자동으로 추가되는 부분들로는 화면에 출력될 용어 목록을 가져오는 소스 코드나 오류 처리 등이 있다.



▶▶ 그림 2. ACOMSServlet의 동작 과정

2. 개발 관리

2.1 개발

하나의 톰캣 컨텍스트는 이클립스에서 워크스페이스(디렉토리) 상의 한 프로젝트로 간주된다. 즉 이클립스의 워크스페이스인 ACOMS 홈 디렉토리에 표준 컨텍스트 std와 각 학회 컨텍스트들이 나란히 위치하게 된다. 한 학회의 ACOMS를 설치하는 작업은 이클립스에서 해당 학회의 프로젝트를 만들어 저장소에 올리고(서브버전의 커밋) 실제 서버에서 저장소에 들어있는 내용을 내려받는(서브버전의 업데이트) 프로세스가 된다. 자료의 동기화에는 이클립스의 서브버전 클라이언트 플러그인인 Subclipse가 사용된다.

1) 표준 std 컨텍스트

표준 컨텍스트에는 모든 학회에서 사용되는 공용 JSP와 Java 소스 코드가 존재한다. 표준 Java 소스는 'kisti.acoms.std'로 시작하는 패키지에 위치한다. 표준 컨트롤러 JSP 소스는 public_html/acoms 밑에, 표준 뷰 JSP 소스는 public_html/acoms/view/std 밑에 각각 위치한다.

웹 애플리케이션 개발자들은 자신의 PC에서 매일 컨텍스트의 소스 코드를 업데이트하고 담당 부분을 개발하고 저장소로 커밋한다.

2) 실제 학회별 컨텍스트

이 컨텍스트들에 존재하는 내용은 설정과 실제로 커스터마이징된 JSP와 Java 파일들로 한정된다. 커스터마이징되지 않은 컨트롤이나 뷰는 ACOMSServlet에 의해 std를 참조하게 되므로 각 학회 컨텍스트에는 존재하지 않는다. 학회별로 커스

터마이징된 Java 소스는 'kisti.acoms.컨텍스트명'으로 시작하는 패키지에 위치하며, 표준 로직에서 약간만 바뀐 것이라면 표준 로직의 해당 클래스를 오버라이드하되 필요한 부분만 재정의하게 되어 표준 클래스에 비하여 소스 코드의 양이 줄어든다. 커스터마이징된 컨트롤러 JSP 소스는 표준과 마찬가지로 public_html/acoms 밑에, 커스터마이징된 뷰는 public/acoms/view/컨텍스트명 밑에 각각 위치한다. 표준 컨텍스트에서 작성된 자바 클래스들은 jar로 묶이어 컨텍스트의 public_html/WEB-INF/lib 밑에 위치하게 된다.

학회 설치 담당자가 이클립스 프로젝트를 생성하고 서브버전에 올리면, 다른 개발자들은 이 이클립스 프로젝트를 내려받아 공동으로 작업할 수 있다. 개발자 PC에서 실제 학회별 컨텍스트 소스 코드의 업데이트는 필요시에만 이루어진다. 이후 개발 과정은 std 컨텍스트와 동일하다.

2.2 서버에서의 운영

서버에서는 ACOMS 홈 디렉토리 밑에서 컨텍스트 이름의 디렉토리를 만들고 저장소에서 해당 컨텍스트의 public_html 이하의 JSP와 Java 소스 코드, 설정 등을 업데이트한다. 소스 코드의 업데이트가 끝난 후 절차적으로 Java 클래스 파일들을 개발자가 이클립스에서 스크립트를 이용하여 컴파일된 것들을 묶어서 올리게 된다.

III. 예상 효과

KISTI-ACOMS는 500여개의 JSP 파일로 구성된다. KISTI-ACOMS는 상당히 많은 페이지의 소스 코드를 수정하는 경우도 있지만 학회당 보통 25개 내외의 JSP가 수정된다. 물론 JSP가 호출하는 로직에 해당하는 Java 소스 코드가 수정되는 경우도 있지만, 이러한 요구는 JSP에 비해 거의 적다.

여기서는 수치를 이용하여 관리하게 될 JSP 소스 코드의 분량을 예상하여 기존 방식과 ACOMS MV 방식의 개발의 효과를 비교 분석하였다. 비교 대상은 KISTI-ACOMS 내에서 다른 부분에 비해 많은 수정이 일어나는 학술대회의 JSP로 학회당 15개 가량의 JSP 소스 파일이 수정된다. 아래는 버전 통합 관리를 사용하였을 때 컨텍스트 수 x별로 f 개의 파일이 수정된다고 할 때 JSP 파일 수(J)와 라인(L) 수를 계산하는 식이다.

$$J_{acoms2} = x J_{acoms2}' \quad (1)$$

$$L_{acoms2} = x L_{acoms2}' \quad (2)$$

$$J_{acomsmv} = J_{acomsmv}' + 2f(x-1) \quad (3)$$

$$L_{acomsmv} = L_{acomsmv}' + 2f(x-1)\frac{L_{acomsmv}'}{J_{acomsmv}'} \quad (4)$$

식 (1)과 (2)는 기존 KISTI-ACOMS의 JSP 파일 수와 라인 수를 계산하기 위한 식인데, 산술적으로 컨텍스트 수 x를 곱하면 된다. (3)과 (4)는 통합 버전 관리 하에서의 JSP 파일 수와 라인 수를 산술평균에 근거하여 계산하기 위한 식이며, 파일 수가 $2f$인 이유는 기존 소스 코드에 비해 뷰와 컨트롤이 분리되어 있기 때문이다.

[표 3] KISTI-ACOMS 학술대회 기능의 JSP 코드의 양 비교

구분	기존 ACOMS		버전 통합 관리	
학회 수	1	300	1	300
JSP	301	90,300	429 (142.5%)	9,429 (10.4%)
총 라인	81,824	24,547,200	43,586 (53.3%)	911,344 (3.7%)
소스 라인	66,437	19,931,100	36,302 (54.6%)	759,042 (3.8%)
주석 라인	5,403	1,620,900	2,326 (43.1%)	48,635 (29.7%)
공백 라인	9,065	2,719,500	4,206 (46.4%)	87,944 (3.2%)
혼합 라인	919	275,700	752 (81.8%)	15,723 (81.8%)

표 3에서 x가 1일 때와 300일 때 각각에 대하여 f가 15일 경우를 비교하였다. 표에서 보듯 컨텍스트 수가 300개 상황에서 파일 수가 10분의 1로 줄어들었으며 총 소스코드의 라인 수는 3.7%로 줄어들었다.

이렇게 관리해야 할 소스 코드의 양이 줄어들면 웹 애플리케이션 개발자가 수정을 위하여 살펴봐야 하는 분량이 줄어들고 시스템 유지보수에 소요되는 시간이 절감되어 결과적으로 사용자의 요구사항을 보다 긍정적으로 검토할 수 있고, 신규 기능의 추가 개발에도 적극적으로 나설 수 있게 된다.

IV. 결 론

KISTI-ACOMS는 하나의 시스템으로 수많은 학회의 요구사항을 빠른 시간 안에 수용해주어야 한다는 점에서 시스템공학적으로 매력적인 연구대상이라 할 수 있다. 지금까지의 신청하는 학회가 있을 때마다 똑같은 분량의 독립된 버전이 생성되는 비효율적인 구조를 버리고, 앞으로는 모든 학회의 소스 코드를 하나로 관리하게 되므로 시스템의 유지/보수 측면에서 관리해야 할 소스 코드의 양이 대폭 줄어들 뿐 아니라 기존에 비하여 사용자들의 다양한 요구사항을 보다 수월하게 반영할 수 있게 되었다. 본 시스템을 서비스하게 되더라도 계속해서 더 합리적인 관리를 위하여 시스템 측면에서 불필요한 코드들을 정리하고 프로그램을 개선해 나가야 한다. 앞으로도 KISTI는 지속적으로 보다 합리적인 시스템 운영에 관한 기술과 정책 개발에 더욱 노력해야 할 것이다.

새로운 KISTI-ACOMS 서비스를 시작하게 되더라도 기존 버전의 표준 코드를 사용하는 학회들은 바로 적용이 가능하겠

지만, 자기의 특징에 맞게 시스템을 고쳤던 학회들은 시간을 가지고 이전 버전에서 작업했던 내용을 똑같이 작업해야 하므로 시간적인 측면에서 새 시스템의 사용이 지연이 불가피하다. 지금까지 KISTI-ACOMS를 운영하는 과정에서 무리한—하나의 요구를 들어줌으로써 시스템의 지속적인 유지/관리에 큰 어려움을 가져오는 요구를 하거나 업무가 집중되는 시기에 특히 많은 요구를 하여 다른 학회 업무에 차질이 생기도록 하는 경우들이 간혹 있었다. 실제로 학회가 필요한 것들을 최대한 구현해 주기 위한 KISTI의 노력도 절실하지만, 학회도 시스템 특성을 이해하고 실제로 일을 하는 KISTI가 시간적/비용적 측면에서 무리하다 싶을 정도의 요구는 자제할 필요가 있다. 앞으로도 학회와 KISTI 상호간에 서로의 이해를 넓히고 협조하기 위한 노력이 계속된다면 KISTI의 보다 나은 시스템의 개발을 통하여 학회에서는 보다 편리하게 업무를 처리할 수 있게 될 것이다.

▌참 고 문 헌▐

[1] 박재원, 최선희, 강무영, “KISTI-ACOMS를 기반으로 한 한국환경생태학회 온라인 논문투고관리시스템 개발 및 개선 방안”, 한국환경생태학회지, 제18권, 제4호, pp.456-464, 2004.

[2] “KISTI-ACOMS 2.0을 이용한 학술 정보 생산 및 유통의 전산화”, 한국멀티미디어학회논문지, 제8권, 제11호, pp. 1543-1555

[3] Krasner, G. and Pope, S., "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System," Journal of Object Oriented Programming, Vol. 1, No. 3, pp. 26-49, 1988

[4] Gamma, E., Helm, R., Johnson R. and Vlissides J., "Design Patterns: Abstraction and Reuse of Object-Oriented Design", ECOOP'93, Vol 707, pp.406-431, 1993

[5] “http://tomcat.apache.org/tomcat-6.0-doc/index.html”, The Apache Software Foundation