JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 19, No. 10 : 2350~2358 Oct. 2015

스프링 MVC에서 수평 개발 방법의 제안

양일등[1] · 김성열[2*]

A Proposal Of The Horizontality Development Method On The Spring MVC

Ill Deung Yang[1] · Seong Ryeol Kim[2*]

[1]Department of Computer & Information Engineering, Cheongju University, Chungbuk Cheongju-si Cheongwon-gu Daesung-ro 298, Korea
[2*]Department of Computer & Information Engineering, Cheongju University, Chungbuk Cheongju-si Cheongwon-gu Daesung-ro 298, Korea

요 약

MVC 디자인 개념이 포함된 스프링 프레임워크는 자바 기반 웹 개발의 표준이다. 엄격한 구조적 프로그래밍 형태를 유지해야하기 때문에 모든 웹 프로그램은 비슷한 구조를 가지게 된다. 그러나 모든 개발 인력들이 구조 유지를 위해 단지 소스 코드들을 단순 복사 하고 개발을 시작하기 때문에 업무배분에 용이하지만 중복코드 양산, 전문성 저하, 프로그램 성능 저하 등의 문제점을 가지고 있다. 이러한 문제점들을 개선하기 위해 우리는 스프링 MVC에 수평 개발 방법론(HDM)을 제안한다. 실험을 통해 HDM가 MVC 구조를 유지하면서 수평으로 업무를 분담하여 제기된 문제점을 해결할 수 있음을 증명하였다.

ABSTRACT

The Spring framework including the MVC design concept is the standard in web development in the JAVA environment. The programming style of the structure must be kept strictly, so every web program is able to employ the same structure. It is convenient to distribute jobs, but there are some side effects, like duplication of code, deterioration of professionalization, and deterioration of program performance. To ameliorate those problems, we propose the horizontality development method(HDM) on Spring MVC. We proved that the HDM can solve the problems through the experiment to keep the MVC structure.

키워드 : MVC, 스프링, 소프트웨어 개발

Key word : MVC, Spring, Software development

Received 02 July 2015, **Revised** 10 September 2015, **Accepted** 25 September 2015

*** Corresponding Author** Seong Ryeol Kim(E-mail:srkim@cju.ac.kr, Tel:+82-43-229-8490)
Department of Computer & Information Engineering, Cheongju University, Chungbuk Cheongju-si Cheongwon-gu Daesung-ro 298, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2015.19.10.2350 print ISSN: 2234-4772 online ISSN: 2288-4165

ⓒThis is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/li-censes/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright Ⓒ The Korea Institute of Information and Communication Engineering.

Ⅰ. 서 론

스몰토크 언어[1]에서 소개된 MVC(Model View Control)는 현대 프로그래밍 발전에 많은 영향을 주었다. MVC는 정보의 표현, 정보의 저장, 정보의 관리 주체를 분리 시켜 각각 모듈화를 추구한다. 그 중 웹 프로그래밍 분야에서 MVC를 적용한 스프링(Spring)이 많이 사용 되고 있다. 자바 언어 버전의 스프링 MVC 프레임워크는 JSP를 뷰로, 모델 및 컨트롤을 자바 클래스 파일로 구성하여 각 모듈별 개발을 분리시켜 개발 효율을 증가 시켰다. 특히 2009년 6월에 스프링을 기본으로 한 전자정부 프레임워크(EGOV)가 발표되었다. 전자정부 프레임워크는 자바 기반 웹 프로그램 개발 표준으로 지정되어 공공 기관을 중심으로 자바 기반 웹 프로그램 개발 표준 프레임워크로 사용 중이다. 하지만 전체 프로그램 업무를 배분할 때는 수직 개발 방법(VDM : Verticality Development Method)을 사용 하고 있다. 이러한 VDM는 업무를 기준으로 사용자 요청부터 내부처리 까지 하나의 프로세스로 구분하고 이를 각 개발 인력들에게 분배하는 방식이다. 이 구조를 유지하기 위해 개발 인력들은 사전 정의된 구조를 단순 복사(Copy-and-Paste) 하여 프로그램을 개발 한다. 이처럼 VDM는 업무 배분에 용이하지만 중복코드 양산, 전문성 저하, 프로그램 성능 저하 등의 문제가 발생 되었다. 따라서 본 논문은 기존의 MVC 개념을 유지하면서 이러한 문제점을 개선할 수 있는 수평 개발 방법(HDM : Horizontality Development Method)을 제안한다.

Ⅱ. 관련연구

2.1. MVC

MVC[1-3]는 Model, View, Controller의 약자로 제록스 팰러앨토 연구소에서 스몰토크 관련 연구를 하던 Trygve Reenskaug이 1979년 최초로 제안한 소프트웨어 공학 아키텍처 패턴이다. 이 패턴을 성공적으로 사용하면 사용자 인터페이스로부터 비즈니스 로직을 분리하여 애플리케이션의 시각적 요소나 그 이면에 실행되는 비즈니스 로직을 서로 영향 없이 쉽게 수정 할 수 있다. MVC에서 모델은 애플리케이션의 데이터를 나타내며, 뷰는 사용자 인터페이스 요소를 나타내고, 컨트롤러는 뷰와 모델의 실행 경로 및 상태를 관리한다. 예를 들어 컨트롤러는 모델에 명령을 보내어 모델의 상태를 변경할 수 있으며 관련된 뷰에 명령을 보내어 뷰의 표현 방법을 바꿀 수 있다. 모델의 변화가 있을 경우 적용 가능한 명령을 추가·제거·수정할 수 있다. 모델은 자신의 상태 변화가 있을 때 컨트롤러와 뷰에 이를 통보하며 뷰는 최신의 결과를 보여줄 수 있다. 뷰는 사용자에게 보여 지는 결과물을 생성하기 위해 모델로부터 정보를 참조 한다. MVC에 관해 2002년 11월 W3C는 미래의 웹 애플리케이션에 사용될 X폼즈(XForms) 아키텍처에 MVC 구조가 포함되도록 하였다. 이 규격은 XHTML 2.0 규격에 통합될 것이다. 현재 20개가 넘는 업체가 애플리케이션 스텍에 MVC가 통합된 X폼즈 프레임워크를 지원하고 있다.

2.2. 스프링 프레임워크

스프링 프레임워크(Spring Framework)[4-6]는 자바 플랫폼을 위한 오픈소스 애플리케이션 프레임워크로서 간단히 스프링(Spring)으로 불린다. 동적인 웹 사이트를 개발하기 위한 여러 가지 서비스를 제공하고 있으며 정부의 공공기관 웹 서비스 개발에 표준으로 사용되고 있는 전자정부 표준프레임워크의 기반 기술로 쓰이고 있다. 스프링은 MVC를 지원하며, MVC 형식에 맞게 프로그램을 작성하도록 하고 있어 개발 인력들은 컨트롤, 서비스, JSP, DAO(Data Access Object)등을 작성하게 된다.

2.3. 전자정부 표준 프레임워크

전자정부 표준 프레임워크는 2009년 대한민국 안전행정부에서 국내 공공부문 국가정보화사업(전자정부) JAVA[7] 플랫폼 기반 추진 시에 개발 프레임워크의 표준 정립되었다. 전자정부 표준 프레임워크는 응용소프트웨어의 표준화, 품질 및 재사용성을 향상을 목표로 하고 특정 업체의 종속성 심화와 사업별 공통 컴포넌트 중복 개발을 막기 위해 개발 했으며 아파치 라이선스 2.0[8]으로 배포했다. 최초 계획은 2007년 12월에서 2008년 6월까지 정보화 전략계획(ISP)을 수립하였고 2008년 11월부터 2009년 11월까지 1, 2단계 구축사업을 추진하여 대기업(삼성 SDS, LG CNS, SK C&C) 및 중소기업(티맥스 소프트 등 6개)이 공동으로 JAVA 플랫폼 기반의 표준프레임워크 실행/개발/관리

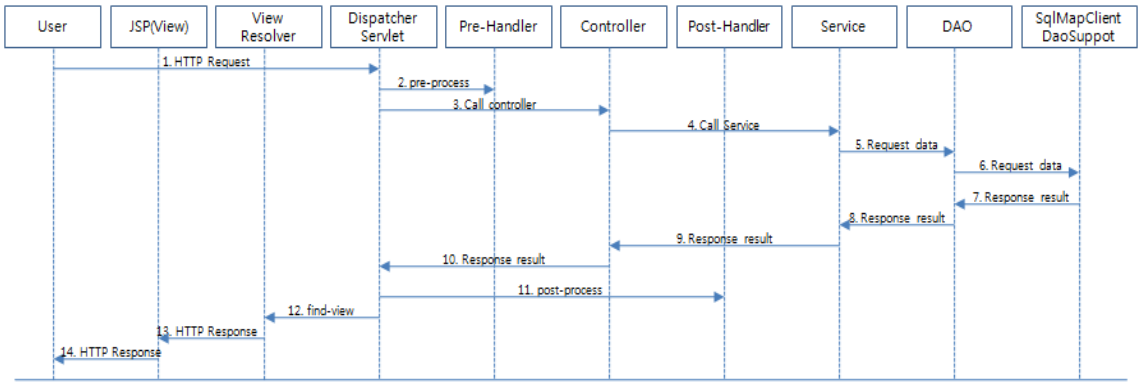

그림 1. 스프링 MVC의 처리 흐름도[9]
Fig. 1 The processing path of the spring MVC[9]

환경 및 공통 컴포넌트 172종을 개발하여 1.0 버전을 발표하였다. 그 후 2010년 4월부터 11월까지 3단계 구축사업을 통해 공통 컴포넌트 47종을 구축하고 CMMI 인증을 획득하였으며 2011년 4월부터 11월까지 4단계 구축사업을 통해 오픈소스의 버전 업그레이드, 경량화, 모바일 프레임워크 등을 구축하고 2.0 버전을 발표하였으며 배치 프레임워크를 포함한 2012년 5단계 구축사업이 완료되었다. 2014년 기준으로 표준프레임워크[9] 버전 3.1이 배포 중에 있다.

Ⅲ. HDM의 제안

3.1. VDM의 문제점

그림 1은 스프링 프레임워크의 웹 요청 처리 흐름도이다. 스프링 MVC 모델에서 데이터 액세스 오브젝트는 Model, JSP는 View, 컨트롤러와 서비스는 Control이다. 그림 1의 처리 흐름은 사용자(User)가 웹 처리 요청을 전송 하면 DispatcherServlet가 XML에 설정된 네이밍 패턴 정보에 따라 스프링에서 처리할 요청인지 검사하고 요청을 수락하며 Pre-Handler에 요청을 전달하고 전처리 작업을 수행 한다. Pre-Handler는 사용자가 전송한 URL정보를 기반으로 해당 컨트롤러에게 요청을 전송한다. 컨트롤러는 유효성 검사 및 웹 프로그램에 특화된 사전 작업을 처리하며, 비즈니스 로직 수행 필요시 해당 서비스를 호출한다. 호출된 서비스는 비즈니스 로직 수행 후 데이터의 조회 및 영구 저장 필요가 있을 경우 DAO를 호출한다. DAO는 파일, DB 등에 자료를 영구 저장하며 서비스에 데이터 요청 시 VO (Value Object)를 반환한다. 이처럼 개발 인력들은 컨트롤러, 서비스, DAO, VO, JSP 등을 각각 개발해야 한다. 또한 자바 언어의 패키지 개념에 맞게 네이밍 규칙을 결정하고 해당 파일들을 적절한 위치에 배치해야 한다. 가장 보편적인 네이밍 규칙은 도메인 이름에 대한 역순 배열이다. 예를 들어 도메인이 test.co.kr 이라면 이를 역순으로 배열하여 kr.co.test 패키지를 생성하게 되고, 이를 기준으로 각 폴더에 파일을 배치하게 된다. 각 컨트롤러와 서비스 및 기타 파일들은 각 업무에 맞게 배치된다. 예를 들어 A업무가 있다면 kr.co.test.a 폴더 하위에 control, service, dao 폴더를 각각 생성하고 하위 폴더에 컨트롤, 서비스, DAO 파일들을 배치한다. JSP 파일은 스프링 특성상 WEB-INF 하위에 별도 저장 하며 각 JSP 파일도 네이밍 규칙에 맞도록 배치한다.

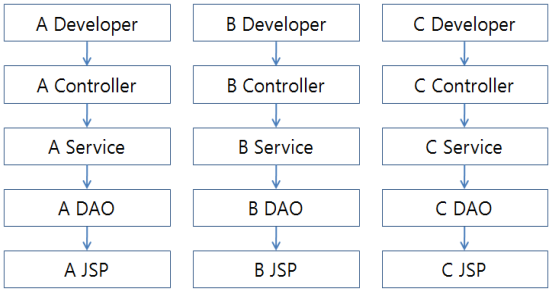

그림 2. 업무별 개발 업무 분담
Fig. 2 The division of development by the jobs

업무별 기준으로 MVC 모델에 맞게 작업을 분배하면 그림 2와 같다. 그림 2는 업무를 VDM로 분류한 것이며 각 개발 인력들은 각자의 업무에 맞게 MVC 구조로 개발한다. VDM는 업무 분배(Job Scheduling)가 매우 용이하다는 장점을 가지고 있다. 각 개발 업무 영역 할당은 화면 단위로 수행 되며, 단일 개발 인력이 컨트롤러부터 JSP까지 필요한 모든 구현을 수행해야 한다. 또한 VDM는 중복코드 양산, 개발숙련도 저하, 상호 업무이해 부족으로 인한 버그 양산 등의 문제점을 내포하고 있는데 구체적인 내용은 다음과 같다.

3.1.1. 중복 코드 양산

통상적으로 일정규모 이상의 접속자가 존재하는 사이트는 50명 이상의 개발 인력들이 웹 프로그램을 개발한다. 각 개발 인력들은 화면기준으로 독립적인 네이밍 영역에 필요 파일들을 생성하게 된다. 스프링 MVC 구조와 네이밍 규칙을 유지하기 위해 기존 프로그램 소스를 단순 복사 하여 자신이 필요한 내용을 수정 혹은 추가하게 된다. 예를 들어 A업무를 개발하는 AA개발 인력들이 회원 정보조회 기능이 필요해서 회원조회 DAO를 작성했다면 이후 다른 B 업무를 개발하는 BB 개발 인력들이 회원 정보조회 기능이 필요하다면 AA개발 인력들이 사전에 만들어 놓은 회원조회 DAO가 공유되지 않기 때문에 BB 개발 인력들도 유사한 회원조회 DAO를 새로이 개발해야 한다. 따라서 이렇게 작성된 중복 코드는 향후 유지보수 시에 A업무의 회원정보 DAO의 내용을 수정해야 한다면 반드시 B업무의 회원정보 DAO도 동일하게 수정해야만 한다. 만약 이 수정 작업이 누락되었다면 런타임에 오류가 발생하여 서비스에 장애를 발생시키는 요인이 된다.

3.1.2. 상호 업무 이해 부족으로 인한 버그 양산

공용으로 사용하는 DB 테이블이 존재할 경우 각 업무 영역에서 해당 테이블을 접근하는 DAO를 작성하게 된다. 모든 테이블은 주사용 업무가 존재하고 그것을 참조하는 보조 업무가 존재한다. 보조 업무를 수행하는 개발 인력들의 업무이해 부족으로 주 테이블에 부정확한 데이터를 입력하거나 혹은 조회된 결과를 부정확하게 표시할 수 있다. 이러한 문제는 사이트가 운영되고 일정기간이 지나야 파악되는 현상으로 민감한 정보 일 경우 그 문제가 매우 심각할 수 있다. 예를 들어 고객 과금 테이블에 입력된 정산 데이터가 천원 단위 절삭을 통해 저장되었는데 다른 개발 인력들이 이 부분에 대한 이해 없이 화면에 데이터를 그대로 표시하면 고객 요금의 편차가 발생하는 것으로 분석되어 운영상 문제점으로 보고될 수 있다.

3.1.3. 개발 숙련도 저하

웹 기술의 빠르게 변하고 다양한 기술들이 사용되고 있다. 특히 자바 기반 웹 프로그래밍 에서는 오픈소스와 실험적인 최신 기술들을 사용하여 개발 하는 것이 보편적이다. 예를 들어 DAO를 구성하는 기술중 myBatis[10]와 Hibennate[11] 등이 서로 다른 개념으로 VO를 구성한다. myBatis는 SQL 쿼리를 직접 사용하여 데이터를 모델링 하며, Hibernate는 오브젝트 기반으로 데이터를 모델링 한다. 또한 뷰에 사용되는 기술 중 ActiveX를 사용한 xinternet 계열 기술과 HTML5를 이용하여 순수 HTML과 CSS만을 사용하는 기술 등이 다양하게 존재한다. VDM에서는 이 모든 것을 단일 개발 인력들이 처리하기 때문에 빠르게 변화는 기술 트랜드를 모두 정확하게 파악하기는 매우 힘들다. 예를 들어 RDBMS의 발달로 현업의 대부분은 RDBMS를 사용하고 있다. 일정 규모 이상의 접속자가 존재하는 사이트의 경우 데이터가 1천만건 이상인 테이블이 다수 존재 하는데 다중 조인 연산을 통해 이 테이블들에서 복합 정보를 빠르게 산출하려면 DB에 대한 상당한 지식과 경험이 필요하다. 또한 VDM에서는 단일 개발 인력이 DB의 수정 및 접근과 관련된 DAO를 작성해야 한다. 따라서 개발 숙련도 저하에 따라 성능과 관계된 DB 조회 쿼리의 경우 데이터양이 소규모일 때는 문제가 발생되지 않지만 데이터양이 일정 수준 이상을 넘으면 성능 문제가 발생 될 수 있다. 이처럼 VDM는 단일 개발 인력들이 모든 영역에 대한 숙련도를 모두 갖출 수 없기 때문에 프로그램 품질 저하요인으로 작용될 수 있다.

3.2. HDM의 제안

VDM에서 제기된 문제점을 개선하기 위해 수평 개발 방법(HDM : Horizontality Development Method)을 제안 한다. 이러한 HDM는 각 개발 인력들의 전문영역을 고려하여 업무를 수평으로 분배한다. 전문영역이란 MVC에 의해 구분된 컨트롤, 서비스, DAO, JSP 등이다. 그림 3은 VDM를 도시한 그림 2를 HDM로 재구성

한 것이다. 이처럼 HDM는 VDM와 달리 업무를 모델 단위로 수평 분할한다. 따라서 제안한 HDM을 도입 적용하면 관리 주체가 일원화 되서 VDM에서 제기된 문제점들을 개선 할 수 있다. HDM 도입 시 장점은 다음과 같다.

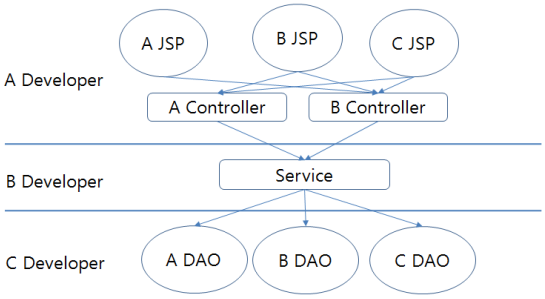

그림 3. 수평 개발 방법에 따른 업무 재 배치
Fig. 3 The relocation of the jobs by the horizontality development method

3.2.1. 중복 코드 제거

중복 코드가 많이 발생되는 곳은 서비스 부분과 DAO 부분으로 상단의 JSP와 하단의 영구 저장 영역 사이에 위치하여 수정이 가장 빈번한 영역이다. 이 영역에 대해 기능이 추가되거나 수정해야 될 경우, 해당 코드의 메인 로직에 옵션을 추가하고 옵션에 대한 세부 처리만을 추가할 수 있다. 예를 들어 DAO가 관리하는 쿼리에서 특정 조건을 추가할 경우 반환되는 컬럼명을 변경하려면 기존 방식은 새로운 DAO에 쿼리를 복사하고 쿼리 자체를 변경하여 컬럼명을 변경하는 방식이지만, 제안한 HDM를 적용하면 해당 모듈의 조건 절에 관련 인자값을 검사하는 코드를 추가 하고, 조건이 입력될 경우 컬럼명을 변경하여 변경된 컬럼명을 반환할 수 있다. 또한 서비스 부분에서도 옵션에 따라 처리 정보를 A 테이블과 B 테이블에 영구 저장하는 비즈니스 로직이 있을 경우, 기존 방식은 A 개발인력이 자신의 서비스 코드에 A 테이블에 저장하는 코드를 추가하고 B 개발 인력의 자신의 B 테이블에 저장 코드를 추가 했지만 HDM에서는 동일한 서비스 코드에서 옵션으로 A 혹은 B 테이블에 데이터를 저장하는 코드를 추가할 수 있다. 따라서 차후 유지보수에 해당 서비스를 수정하면 이를 참조하고 있는 다른 업무 영역에도 자동적으로 수정됨으로 유사한 코드를 여러 곳에서 수정할 필요가 없어진다.

```
A : public UserInfo getUserInfo( String userId );
B : public UserInfo getUserInfoByDeptCode
    ( String userId , String deptCode );
```

그림 4. VDM에서 사용자 정보 요청 함수 변경
Fig. 4 The change of the function of the requestion of user information on VDM

그림 4는 VDM에서 사용자 정보를 요청하는 함수의 변경 내용이다. VDM에서는 사용자 아이디에 추가로 부서 코드를 입력하여 사용자 정보를 요청할 경우, 기존 코드를 복사하여 신규 함수를 생성하고 신규 함수의 이름을 설정했다. 이것은 A함수가 관련된 코드에 영향을 주지 않기 위해 자신만의 사용자 정보 요청 함수인 B함수를 생성하기 때문이다. 이렇게 사용자 정보를 요청 하는 함수가 추가 생성되면 차후 관련 요구가 있을 경우 C혹은 D의 사용자 정보 요청 함수가 신규 추가 된다.

```
A : public UserInfo getUserInfo( String userId );
A : public UserInfo getUserInfo( String userId ,
    String deptCode);
```

그림 5. HDM에서 사용자 정보 요청 함수 변경
Fig. 5 The change of the function of the requestion of user information on HDM

반면 그림 5는 HDM에서 사용자 정보 함수를 변경하는 과정으로 기존 A함수를 관리하는 개발 인력에게 필요한 요청 사항을 전달하여 기존 A함수를 수정하게 된다. 또한 getUserInfo 함수명을 동일하게 사용하고 싶을 경우 객체지향 언어 기능인 오버로딩(OverLoading)을 사용하여 함수 이름 동일하게 생성하고 인자 형태가 상이한 A`함수를 생성할 수 있다.

```
A : SELECT * FOR USER WHERE ID = #ID#
B : SELECT * FOR USER WHERE ID = #ID#
    AND DEPTCODE = #CODE#
```

그림 6. VDM에서 사용자 정보 요청 쿼리 변경
Fig. 6 The change of the query of the requestion of user information on VDM

A`함수는 내부적으로 A함수를 호출하고 deptCode는 NULL로 설정하여 userId만 조건으로도 사용자 정보를 검색하는 코드로 수정할 수 있게 된다.

또한 그림 6은 iBatis에서 사용자 정보 조회 쿼리 예이다. VDM에서는 사용자 정보 조회 쿼리에 변경이 필요한 경우 기존 A쿼리를 복사하고 옵션을 수정하여 B쿼리를 생성하고 B쿼리를 호출하는 DAO와 VO를 새로 생성 하였다. 하지만 그림 7과 같이 HDM에서는 필요한 옵션값을 매개변수로 추가하여 동일한 쿼리를 수정 하기 때문에 CODE값이 존재 할 때만 조회 조건 변경이 적용되도록 수정 할 수 있다.

```
A : SELECT * FOR USER WHERE ID = #ID#
A : SELECT * FOR USER WHERE ID = #ID#
    <isNotEmpty property="CODE">
    AND DEPTCODE = #CODE#
    </isNotEmpty>
```

그림 7. HDM에서 사용자 정보 요청 쿼리 변경
Fig. 7 The change of the query of the requestion of user information on HDM

3.2.2. 업무 로직 분석 최소화

VDM는 특정 기능을 추가하거나 수정하기 위해서 컨트롤러, 서비스, DAO, JSP 등을 모두 분석하고 그 흐름을 정확히 이해하고 파악해야 한다. 이는 MVC 모델의 형태를 유지하지만 내부적으로는 모듈화가 적용되지 않아 코드 단위로 분석해야하기 때문이다. 그러나 제안한 HDM을 적용하면 자신에게 할당된 영역만을 분석하고 나머지는 API 명세서를 활용하여 분석할 수 있다. 이러한 API 명세서란 클래스, 메소드 등에 대한 상세 설명으로 개별 모듈을 참조하는 기본 문서이다. 자바의 경우 소스코드에 추가 주석을 작성하여 javadoc 컴파일러로 웹 문서 형태의 설명서를 산출할 수 있다. 이 API 명세서를 통해 각 모듈의 입력과 출력에 대한 설명을 참고하고 내부 코드에 대해서는 분석하지 않는다. 따라서 제안한 HDM에서 각 영역의 개발인력은 자신이 구현한 내용에 대한 입력과 출력을 명세하고 이를 전파하여 다른 영역을 개발하는 개발 인력들이 참조할 수 있도록 한다. 이처럼 입력과 출력이 명시된 API 명세서를 바탕으로 그것을 참조하는 개발 인력들은 업무 로직 분석을 최소화할 수 있어 이에 따른 각종 버그의 발생도 최소화할 수 있다. API 명세서 작성에 필요한 데이터를 입력하는 과정이 추가 작업으로 생각될 수 있지만, 모든 개발 작업은 마지막에 문서화 작업이 수행된다. 이런 문서화 작업에서 사전 생성한 API 명세서 자료를 그대로 사용할 수 있기 때문에 추가 작업이 아닌 필수 작업 성격으로 생각하는 것이 타당하다.

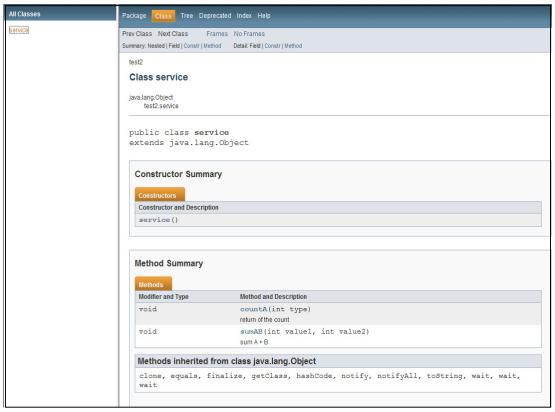

그림 8. javadoc으로 생성된 API 명세서
Fig. 8 API document that generated by javadoc

그림8은 service 클래스를 javadoc 컴파일 처리하여 생성한 API 명세서이다. 이처럼 자동 생성 기능을 통해 보다 상세하고 정교한 API 명세서를 작성하면 개발 인력들 사이에 보다 원할 한 의견교환의 자료로 사용할 수 있다.

3.2.3. 각 부분별 개발 숙련도 향상 및 전문성 확보

VDM에서는 모든 영역을 단일 개발 인력들이 다룬다. 따라서 빠르게 발전하고 변화하는 모든 신기술을 단일 개발인력이 모두 습득하기 어려우며 이로 인해 숙련도 및 전문성이 저하될 수 있었다. 제안한 HDM을 적용하면 MVC 모델 영역별로 업무를 분담하고 각 영역별 전문 인력을 배정하기 때문에 전체적인 개발인력의 숙련도 및 전문성을 확보할 수 있게 된다. 예를 들어 JSP 등은 모델러 혹은 코더의 영역으로 분리할 수 있는데, 이 영역에서는 사용자 친화적인 UI(User Interface)를 보다 심도 있게 다루게 되며 정보 표현을 사용자 친화적인 형태로 표현하는 것이 주요한 업무가 될 수 있다. 서비스영역에서는 트랜잭션의 확실한 처리가 매우 중요하다. 트랜잭션의 처리는 그 개념적 이해도 중요하지만

구성 환경에도 매우 민감하다. 스프링에는 트랜잭션을 다루기 위한 부가적인 개념들과 기술들이 포함되어 있고 지속적으로 발전하고 있으며, 이들 분야를 SA (System Architecturer)가 다루고 있다. 또한 DAO는 하부 영구 저장 영역과 매우 밀접한 관련이 있다. 하부 영구 저장은 높은 가용성과 실행 성능을 요구하며 동일한 결과를 도출하지만 그 과정에 문제가 있다면 바로 성능에 영향을 줄 수 있는 매우 민감한 사항이다. 이 부분은 DBA가 해당 영역을 관리할 수 있다. 따라서 HDM를 적용하면 각 영역에서 중요하게 다루어야 하는 전문적인 지식과 최신 개발기술들을 보다 심도 있게 적용할 수 있어 각 영역의 전문 인력의 개발 숙련도 및 전문성이 향상 될 수 있다.

```
$("example-basic").steps({
   headerTag: "h3",
   bodyTag: "section",
   transitionEffect: "slideLeft",
   autoFocus: true
});
```

그림 9. JQuery 화면 효과 예제
Fig. 9 A sample of the display effect by JQuery

그림 9는 JQuery[12]를 이용한 사용자 UI 코드를 보여 준다. 그림 9에 있는 코드를 실행하면 웹 화면에 동적인 느낌을 줄 수 있다. 이 같은 UI의 변화는 사용자에게 능동적인 느낌을 강조하여 보다 역동적인 화면을 제공할 수 있게 된다. 그러나 JQuery와 같은 기술은 화면 코딩 전문 개발자가 정교하게 코딩해야 하며, JQuery에 대한 전문 지식이 풍부해야 가능하다.

```
A : SELECT ID FROM USER_INFO  WHERE
    ID  LIKE '%#ID#%'
⇒
B : SELECT ID FROM USER_INFO  WHERE
    ID  = '%#ID#%'
```

그림 10. HDM에서 사용자 정보 요청 쿼리 변경
Fig. 10 The change of the query of the requestion of user information on HDM

그림 10은 HDM에서 사용자 정보 요청시 쿼리 변경 SQL 조건을 보여 준다. 대규모 데이터 조회 시에 LIKE 검색은 검색 속도 저하를 발생시킨다. 하지만 RDBMS에 전문 지식이 부족한 개발 인력이 SQL을 작성하고 다른 관련 부분을 고려하지 않을 경우, 차후 데이터가 증가될 때에 성능 문제가 발생할 수도 있다. 따라서 HDM로 업무를 적용하면 각 전문가들을 해당 영역에 배정하는 것으로 이러한 문제를 선결하는데 적합하다.

Ⅳ. 실험 및 평가

4.1. 실험 환경 설정

HDM의 우수성을 증명하기 위해 질병관리 본부 운영지원팀 6명의 협조를 얻어 실험에 참여시켰다. 각 개발 인력들은 10년 이상 개발 경험이 있지만 각 전문 분야는 모두 상이하였다. 다음 표1은 4가지 전문분야에 대해 각 개발 인력들의 능력치를 상중하(HML)로 표시하였다.

표 1. 개별 프로그래머의 능력 점검 표
Table. 1 A ability table of the each programmers

Type \ Programmers	A	B	C	D	E	F
UI	H	M	H	M	M	M
Business Logic	L	H	M	H	M	M
RDBMS	M	L	M	L	H	M
EGOV	M	L	L	L	H	H

항목의 구분은 MVC 모델의 각 영역을 기준으로 구분하였으며, 또한 전자정부 프레임워크의 운영 능력을 추가로 첨부하였다. 표1의 결과를 분석하면 A와 C개발 인력은 UI 적용 기술력이 뛰어났으며, B, D 개발인력은 비즈니스 로직 생성 능력이 뛰어났다. 마지막으로 E, F 개발 인력은 RDBMS 및 전자정부프레워크에 특히 경험이 많았다. 이를 근거로 개발인력을 A,B,E와 C,D,F를 두 그룹으로 분류 하였다.

실험을 위한 작업 대상은 질병관리 본부 통합 전산 시스템의 하위 시스템중 동물의약품 관리 시스템의 사용자 화면 3개를 선별하였다. 단 3번째 화면은 레포팅 기능이 있는 화면으로 레포팅 툴의 조작 경험이 요구되었다. 첫 번째 그룹은 VDM를 적용하여 3명의 개발인

력이 각각 화면 하나씩을 배정하고 화면에 필요한 모든 MVC 요구를 개발하도록 하였다. 두 번째 그룹은 HDM를 적용하여 화면, 비즈니스 로직, DAO로 구분하여 3명의 개발 인력들에게 작업을 배정하였다. 단, 기본 전자정부 설정은 각 그룹에서 E, F가 설정하고 이를 다른 개발인력에게 배포하였다. 이후 첫 번째 그룹은 각 개발인력이 필요한 내용을 수정보강 하였으며, 두 번째 그룹은 F 가 필요한 지원을 제공하였다. 각각의 내용이 변경될 때는 형상관리 시스템을 이용하였다.

4.2. 실험 결과 분석

그림 11와 그림 12는 각 그룹의 작업 진행 상황을 나타낸다. 각 그림의 Y축은 전체 공정률(TPR : Total Processing Rate)을 나타낸다.

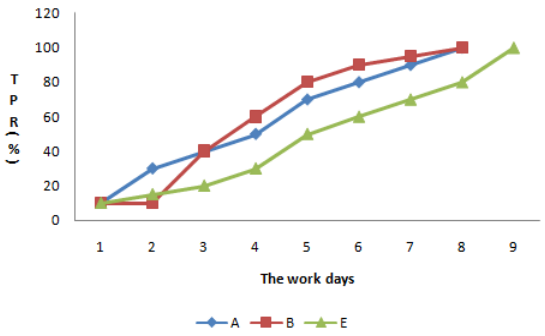

그림 11. 첫 번째 그룹의 작업 공정 그래프
Fig. 11 The work processing graph of the first group

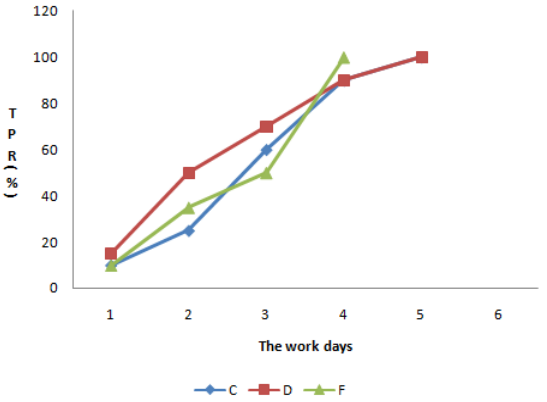

그림 12. 두 번째 그룹의 작업 공정 그래프
Fig. 12 The work processing graph of the second group

TPR이 100(%)에 도달하면 작업을 완료한 것이다. 또한 X축은 개발 일수를 나타낸다. 전체 공정률이 100(%)에 도달하기 까지 개발 일수를 표시 하였다. 매일 각 개발인력들에게 개별적으로 질의하여 공정률을 누적 수치화 하여 기록하고 이를 통해 각 개발인력의 공정률 진척에 따른 개발완료 시점을 수치로 표현하였다. 실험 종료 후 에는 개별 인터뷰를 진행하였다.

실험 결과를 분석하면 첫 번째 그룹에서 B개발 인력은 전자정부 프레임워크 설정에 시간이 많이 소요되었고, E개발인력은 화면 구성에 많은 시간을 소비하였다. 또한 A개발 인력은 당초 예상과 달리 모든 부분에서 평균적인 공정률을 나타내었다.

이는 A개발 인력이 UI구성에 전문성을 가지고 있지만 사교성이 매우 높아 필요한 정보를 적시에 얻는 것으로 파악되었다. 두 번째 그룹은 HDM의 특성상 상호 소통이 매우 필요하기 때문에 초반부터 의견교환 활동이 활발히 이루어졌다. 각 분야에서 평균적인 공정률을 나타냈으며, 서비스 로직의 수정으로 인해 D개발인력의 전체 공정률 완료가 다른 개발인력에 비해 하루 더 길었다. 첫 번째 그룹과 두 번째 그룹의 평균 개발기간은 8.3일과 4.3일 이었다.

또한 HDM를 적용하여 최초 개발을 수행한 두 번째 그룹에게 HDM에 대한 만족도를 설문으로 조사하였다. 표2는 설문조사 결과를 나타낸 것으로 3명 모두 매우 만족하는 것으로 조사되었다. 이처럼 HDM는 실용성 및 개발인력들의 만족도 측면에서 매우 좋은 성과를 나타내고 있다. 특히 첫 번째 그룹대비 두 번째 그룹에서는 모듈별 전문화가 이루어져 단위 테스트 시간이 월등히 단축되었다.

표 2. HDM 만족도 조사 결과
Table. 2 A research result of HDM satisfaction level

Type	Result(%)
Very satisfied	100%
Somewhat satisfied	0%
Neither satisfied nor dissatisfied	0%
Somewhat dissatisfied	0%
Very dissatisfied	0%

Ⅴ. 결 론

스프링 MVC은 웹 프로그램 개발에서 모델, 뷰, 컨트롤의 영역을 구분하여 각 영역별 개발이 가능하지만, 업무 분담 효율을 증대하기 위해 VDM을 적용하면 중복코드 양산, 전문성 저하, 프로그램 성능 저하 등의 여러 가지 문제점을 발생시켰다. 이에 본 논문에서는 VDM의 문제점을 분석하여 이를 개선할 수 있는 HDM을 제안하고, HDM 도입 시 가질 수 있는 장점으로 중복코드 제거, 업무로직 분석 최소화, 효율적인 프로그램 개발, 각 부분별 개발인력의 숙련도 향상 및 전문성 확보 등의 장점을 제안하고 실험을 통해 전체 개발효율이 향상됨을 보였다. 기존 VDM 대비 HDM는 구조적으로 문서화가 이루어지며, 개발 인력 상호간에 의사소통이 원활하게 이루어 졌다. 이를 통해 HDM 실험에 참여한 인력들의 만족도도 상당히 높음을 확인할 수 있었다. 따라서 제안한 HDM를 적용하여 프로젝트를 수행하면 MVC 모델의 장점을 살리면서 VDM 적용 시에 발생되는 여러 가지 문제점들이 대폭 개선될 수 있어 개발 시스템의 안정성 확보와 유비보수 비용을 획기적으로 절감할 수 있을 것으로 기대한다. 다만 HDM는 높은 수준의 전문성이 갖추어진 개발 인력의 적절한 배치 방법이 문제이긴 하나, 차후 연구에서는 개발인력의 업무 분야 타당성 검증 방법과 그에 따른 업무 영역 배치에 대한 연구를 수행할 계획이다.

REFERENCES

[1] wiki Document, http://ko.wikipedia.org/wiki/

[2] Ralph F. Grove, Eray Ozkan, “THE MVC-WEB DESIGN PATTERN,” *WEBIST 2011, Proceedings of the 7th International Conference on Web Information Systems and Technologies*, Noordwijkerhout, The Netherlands, 6-9 May, 2011.

[3] E. Althammer and W. Pree, “DESIGN AND IMPLEMENTATION OF A MVC-BASED ARCHITECTURE FOR ECOMMERCE APPLICATIONS,” *Int. Journal of Computers and Their Applications*, 2000.

[4] Terence Parr, “Enforcing Strict Model-View Separation in Template Engines,” WWW2004, May 17 – 20, 2004, New York, New York, USA.

[5] Spring Documents, Available: https://spring.io/docs

[6] Qbal H. Sarker and K. Apu, “MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application,” *International Journal of Hybrid Information Technology* Vol.7, No.5 (2014), pp.317-322.

[7] JavaAPI Document, http://docs.oracle.com/javase/7/docs/api/

[8] The Apache Software Foundation, http://apache.org

[9] eGovFrame Portal & eGoverment Standard Framework, http://www.egovframe.go.kr/.

[10] Mybatis Document, https://mybatis.github.io

[11] Hibernates Document, http://hibernate.org

[12] JQuery Document, http://jquery.org

양일등(Ill Deung, Yang)

2002년 청주대학교 컴퓨터정보공학과 공학사
2004년 청주대학교 컴퓨터정보공학과 공학석사
2010년 청주대학교 컴퓨터정보공학과 박사과정
2014년~현재 (주)인터페이스 정보기술 부설 연구소 책임 연구원
※관심분야 : 웹 시스템 설계, 소프트웨어 공학, 분산처리

김성열(Seong Ryeol, Kim)

1992년 숭실대학교 대학원 전자계산학과 공학박사
1984년~1990년 오산대학 전자계산과 교수
1997년~1998년 호주 QUT ISRC 객원 교수
1990년~현재 청주대학교 컴퓨터정보공학과 교수
※관심분야 : 웹 시스템 설계, 컴퓨터 보안, 소프트웨어공학