상호운용성과 재사용성 증대를 위한 전자정부 표준 개발프레임워크[1] 구축 사례 연구

임철홍*, 전만성**, 이봉옥***, 김영우***, 이영곤****
*SK C&C, **LG CNS, ***삼성 SDS, ****한국산업기술대학교

A Study of Implementing e-Government Development Framework for Improving the Interoperability and Reusability

Im Chol Hong+, Jun Man Sung++, Lee Bong OK+++, Kim Young Woo+++, Lee Young Kon++++
+SK C&C, ++LG CNS, +++Samsung SDS, ++++KOREA Polytechnic University
E-mail : imich@skcc.com, msjun@lgcns.com, bongok.lee@samsung.com,
youngwoo@samsung.com, yklee777@kpu.ac.kr

요 약

전자정부 사업에서 개발프레임워크가 활발하게 도입되어 활용되어 왔으나 서로 다른 개발프레임워크 도입으로 인하여 전자정부 시스템간에 상호운용성이 보장 되지 못하고, 특정 개발프레임워크에 의한 사업자 종속성이 발생하게 되었다. 이러한 문제를 해결하기 위하여 전자정부 프로젝트에서는 표준화된 형태의 오픈 소스 기반 개발프레임워크를 구축하게 되었다. 본 논문에서는 표준 개발 프레임워크가 등장한 배경과 표준화 원칙들을 제시하고, 이러한 원칙들을 만족 시키기 위한 개발프레임워크의 실행환경과 개발환경에 대한 아키텍처 특성 및 기반기술을 제시하고자 한다.

1. 서론

어플리케이션 개발에서 상호운용성의 확보와 재사용성의 증대를 위한 노력이 계속해서 시도되어 왔으며, 이를 위해 소프트웨어 개발프레임워크의 적용이 보편화 되었다. 하지만, 서로 다른 규격을 지원하는 개발프레임워크가 많이 활용 되면서 하나의 조직에서도 서로 다른 개발프레임워크 활용으로 인한 상호운용성 문제가 다수 발생하고 있다. 이러한 문제는 여러 개발업체가 참여하는 국가 정보시스템에서 더욱 심각한 문제로 제기되어 왔으며, 이를 해결하기 위한 표준화된 개발프레임워크의 개발이 절실히 요구되었다. 이를 위해 개방형

[1] 개발프레임워크는 소프트웨어의 개발을 지원하고, 효율적 운영과 확장성을 보장하기 위한 기반 소프트웨어이다.

표준의 채택과 오픈 소스의 적극적인 활용, 국내 상용 솔루션 연계 등과 같은 표준 개발 원칙을 수립하였다. 표준 개발프레임워크는 내장 API를 활용하여 소스코드의 제어 및 실행을 담당하는 실행환경과 이클립스 기반의 GUI를 제공하고 소스코드의 구현, 테스트, 배포, 형상관리를 지원하는 개발환경으로 구성된다. 표준 개발프레임워크의 상호운용성, 재사용성, 유연성을 높이기 위해, 오픈소스 기반 기술을 채택하고 아키텍처를 설계 하였다.

2. 개발 프레임워크의 국가적 표준화

2.1. 개발프레임워크 현황

개발프레임워크는 응용SW 개발의 효율성, 유지보수의 용이성, 확장성 등을 위해 응용SW 개발 도구, 지침 및 SW 뼈대를 제공하는 공통 개발기반을 총칭한다[1]. 공통 개발 기반은 디자인 패턴과 아키텍처를 지원하는 기반 코드 형태로 구성되어 있다. 다음 그림1은 개발 프레임워크의 전체적 구성을 보여주고 있다.

그림1. 개발 프레임워크 개념

전자정부 사업(‘ 04~’ 07 수행 사업 중 정보시스템 구축 관련 사업) 개발프레임워크 적용 현황을 조사한 결과, 사업비 기준으로 71%에 해당하는 많은 전자정부 사업에 개발프레임워크가 적용되었음을 알 수 있다[2]. 그러나, 개발프레임워크의 보급에 따른 여러 문제점들도 발견 되었다. 먼저, 구축 업체마다 서로 다른 개발프레임워크의 사용으로 상호운용성이 보장되지 않았으며, 특정 SI 업체의 개발프레임워크 기술에 종속되어 확장 및 유지보수에 제약이 발생 되었다. 이러한 경우, 사업자를 교체하고자 하여도, 이미 구축된 개발프레임워크로 인해 사업자 교체가 원활하게 이루어지지 못하고 개발업체에 종속되는 문제가 발생되어, 후속사업자에게 진입장벽으로 작용하고, 최악의 경우에는 전면 재개발을 감수해야 하는 문제까지도 발생 되었다.

2.2. 표준화 원칙

기존의 개발프레임워크는 SI사업자에 의해 활용되거나, 솔루션 형태로 판매 되어 왔다. 국가 차원의 표준 개발프레임워크가 가지게 되는 특성상 많은 이해관계자들의 입장을 고려하여 솔루션을 포함하는 국내 SW시장에 대한 고려가 필요하였다. 또한, 중소개발업체에서도 기술적인 장벽이 없이 즉각 활용 될 수 있도록 개방형 표준 형태의 고려가 필요 하였다. 다음 표1은 전자정부 표준화를 위한 인식도 조사[3]의 결과로 도출된 표준화 요구 사항이다.

No	표준화 요구 사항
1	사용이 편리하고 기능이 풍부한 환경 제공
2	다양한 기술, 업무요건 수용 가능한 유연한 구조
3	신속한 기술지원 서비스 체계
4	지속적인 유지보수와 적시적인 버전 업 체계
5	개발표준(Open Standard)의 준수
6	관련기관 및 업계의 의견을 충분히 수렴
7	특정 사업자에 대한 종속성 배제
8	전자정부 개발프레임워크 인력양성
9	기술공개를 통한 중소기업 참여 활성화
10	오픈소스 프레임워크 기술 수용
11	업계의 최신 개발프레임워크 제품 기술 수용

표1. 표준화 요구 사항 조사 결과

위와 같은 표준화 요구 사항을 분석하여 국가적 차원의 표준화를 위한 원칙을 다음과 같이 수립

하였다.

- 공공 기관 정보화담당자, 대학교수, SW업계로 구성된 자문협의회 구성과 운영
- 상용 솔루션과 관련된 영역 배제 또는 필수적인 기능만 제공
- 상용 솔루션(UI Adaptor등)과 연동이 가능하도록 표준을 제공하고 연동을 보장
- 개방형 표준을 활용하고 오픈 소스를 적극 채용하여 기술 종속성 제거
- 전자정부 사업에서 잘 활용 되는 주요 WAS와 DB와의 호환성을 보장
- 분석 설계 산출물 및 소스코드의 공개를 통해 사용자의 활용성을 제고

3. 표준 개발 프레임워크 아키텍처

3.1 아키텍처 구성

표준 개발 프레임워크의 아키텍처는 기반 서비스가 직접 제공 되는 실행환경과 실행환경을 기반으로 업무 컴포넌트를 개발하는 개발환경, 프레임워크 기반 시스템의 운영을 모니터링 하는 운영환경, 개발 프레임워크 자체적인 개발과 운영을 위한 관리환경으로 구성 된다. 다음 그림2는 표준 개발 프레임워크의 전체 아키텍처 구성을 설명하고 있다.

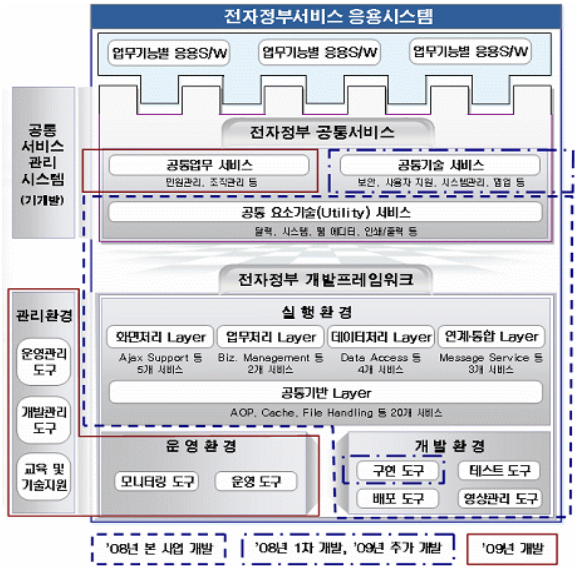

그림2. 전자정부 표준 개발프레임워크 아키텍처

다음 표2에서와 같이 실행환경은 5가지 Layer로 구성되어 있으며, Layer별로 대표적인 오픈소스를 선정, 채택하여 서비스를 제공 한다.

서비스그룹	설명
Presentation Layer	프로그램, 사용자간의 Interface를 담당. 사용자 화면 구성, 사용자 입력 정보 검증 수행 (Spring MVC, Ajax Tags 등)
Business Logic Layer	프로그램의 업무 로직을 담당. 업무흐름제어, 에러처리 수행 (Spring WebFlow 등)
Persistence Layer	프로그램에서 사용하는 데이터에 대한 제어를 제공 (iBatis, Hibernate 등)
Integration Layer	타시스템과의 연동 기능을 제공 (Apache CXF 등)
Foundation Layer	각 Layer에서 공통적으로 활용되는 공통 기능 제공 (Spring, Apache Commons 등)

표2. 실행환경 구성 Layer

개발 환경은 다음 표3에서와 같이 4가지 Tool로 구성되어 있으며, Tool별로 대표적인 오픈 소스를 선정, 채택하여 서비스를 제공 한다. 개발환경의 Tool들은 구현, 테스트, 배포, 형상관리의 형태로 정보 시스템 개발의 총 Life Cycle을 지원하며 실행환경을 편리하게 사용 할 수 있도록 기능을 제공 한다.

서비스그룹	설명
Implementation Tool	업무 프로그램 구현을 지원 하는 도구 (Eclipse, AmaterasUML 등)
Test Tool	구현된 업무 프로그램의 테스트를 지원하는 도구 (Junit, EasyMock 등)
Deployment Tool	구현 완료된 업무 프로그램을 실행 환경에 배포 가능한 형태로 패키징하고 패키징 파일을 실행환경에 배포하는 도구 (Maven, Hudson 등)
Configuration & Change Tool	형상 및 변경관리 지원 도구 (Subversion 등)

표3. 개발환경 구성 Tool

3.2 아키텍처 특성

표준 개발 프레임워크를 적용하여 시스템을 개발하게 될 경우, 시스템의 유연성, 재사용성, 상호운용성을 증대시킬 수 있다. 이러한 특성은, 개발 프레임워크의 특성상, 역할 별 코드의 엄격한 분할, 공동 코드의 동일한 인터페이스 호출, XML 방식에 의한 표준화된 프로파일 설정 등에 기인한다. 다음 표4는 아키텍처 특성을 지원하기 위해 활용 되는 기술을 설명 하고 있다. 이러한 기반 기술들은 전자정부 표준 개발프레임워크에서도 구현 되어 제공이 가능하다.

특성	향상 방법	기술
유연성	SQL문을 XML설정에 의해서 처리	SQL Map[4]
	DB와 객체 클래스를 매핑하여 객체지향 코드에 의해 DB접근 및 사용 지원	ORM[5]
재사용성 유연성	역할별 코드를 구분 비즈니스 로직과 화면 분리	MVC[6]
	컴포넌트 구성 지원, 인터페이스를 통한 기능 제공 창구를 일원화	DI[7]
	비즈니스 로직과 공통기술 로직을 분리 (공통기술 로직의 모듈화 및 재사용)	AOP[8]
상호 운용성	오픈소스 및 개방형 표준의 기술을 채택 활용	-

표4. 아키텍처 특성과 기반 기술

다음 그림3은 Presentation Layer의 C&C (Component-and-Connector)다이어그램이다. 이 그림에서는 MVC(Model-View-Controller) 패턴이 적용 되어 입력과 출력이 Controller에 의해 처리가 되며, Model 형태로 컴포넌트 Container와 정보를 교환하고, 결과물은 View에 의해서 사용자에게 전달 되는 형태의 처리과정을 보여주고 있다.

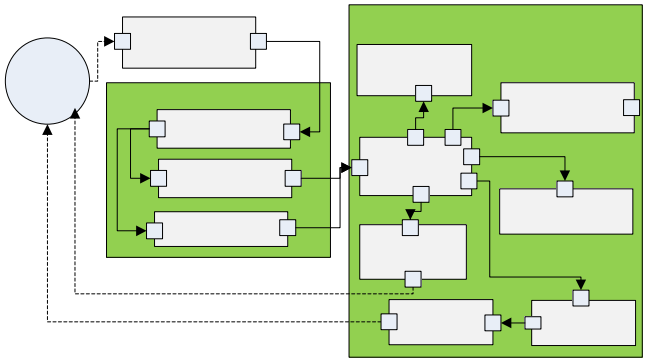

그림3. Presentation Layer C&C 다이어그램

다음 그림4는 Persistence Layer의 C&C 다이어그램이다. ORM과 SQL MAP중에 선택하여 활용 가능하도록 지원 한다. Query가 매우 복잡하고 데이터처리 중심적인 경우에 SQL Map의 활용이 효율적이다. ORM을 활용할 경우에는 Query에 의한 데이터 처리가 아니라 객체지향 프로그램 내에서 데이터 처리를 지원하는 형태가 된다. 두 가지 모두 DB나 SQL에 의해 코드가 종속적이 되는 것을 방지하고, 변경이 생겼을 경우 코드 자체를 수정하는 것이 아니라 SQL Map, Mapping 정보 수정을 통해 변경을 반영할 수 있는 유연성을 제공하게 된다.

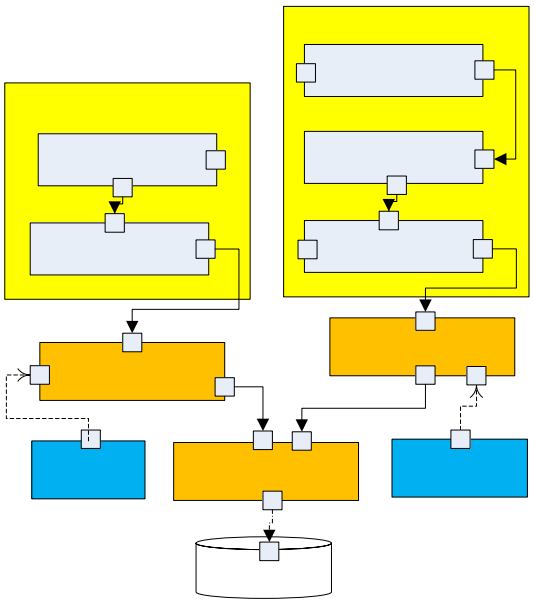

그림4. Persistence Layer C&C 다이어그램

4. 향후 계획 및 결론

현재 개발프레임워크는 실행환경 전체와 개발환경 80% 구축이 완료 되었다. 이후에는 관리환경, 운영환경 구현이 진행 될 예정이며, 전자정부 사업 적용을 통한 문제점의 해결, 안정화, 기능 개선 작업이 지속적으로 진행될 예정이다. 전자정부 사업 적용을 시작으로 전체 공공정보화 사업 적용 및 확산을 목표로 하고 있다. 표준 개발 프레임워크의 도입을 통하여 공공 정보시스템간의 상호운용성의 확보와 공통 컴포넌트의 공동 활용을 통하여 품질 및 재사용성 향상이 기대 된다. 성공적인 표준 개발 프레임워크의 적용을 위해서는 구현도 중요하지만 이를 지속적으로 유지보수하고 개선하여 표준 개발프레임워크의 적용을 확대하고, 적용 우수사례의 발굴을 포함하는 발주자 및 사업자의 지속적인 노력이 필요하다.

[참고문헌]

[1] Riehle, Dirk, " Framework Design: A Role Modeling Approach", Swiss Federal Institute of Technology, 2000

[2] " 전자정부 사업 개발프레임워크 적용 현황", 전자정부 공통서비스 관리체계 정립 ISP, 2008.

[3] " 전자정부 개발프레임워크 표준화를 위한 인식도 조사", 전자정부 공통서비스 관리체계 정립 ISP, 2008.

[4] Michael Klaene, " Using iBatis SQL Maps for Java Data Access", http://www.developer.com/lang/article .php/10924_3346301_1.

[5] Douglas Barry, Torsten Stanienda, "Solving the Java Object Storage Problem," Computer, vol. 31, no. 11, 1998.

[6] Steve Burbeck, " Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)", http://st-www.cs.uiuc.edu/users/ smarch/st-docs/mvc.html.

[7] Martin Fowler, " Inversion of Control Containers and the Dependency Injection pattern", http://martinfowler.com/articles/injection.html.

[8] Kiczales, Gregor; John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin , "Aspect-Oriented Programming", Proceedings of the European Conference on Object-Oriented Programming, vol.1241, 1997.