특집

조선생산실행시스템의 웹서비스 서버 및 모니터링 구축

이필립, 박영미, 정종필, 권기욱 (지노스)

1. 서 론

1.1 조선생산실행시스템 구축

자동화율이 높은 공장에서 구성하는 생산실행시스템(MES, Manufacturing Execution System)은 컨베이어 벨트나 분배기, 로봇 등의 자동화 기기에서 생산 정보를 수집하고 이를 분석한 후 적절한 생산 지시를 내리는 구조로 되어있다. 하지만, 조선소는 자동화된 생산 장비가 많지 않고 자동차나 핸드폰 제조업 같이 흐름 생산으로 이루어진 공정 비중이 높지 않다. 게다가 조선소 생산 정보의 핵심은 건조 과정에서 생산하는 블록과 부재가 쌓이는 공간으로, 해당 정보는 상황에 따라 유동적으로 변하게 되고 대상이 센서 등으로 자동 정보 수집이 어렵다. 생산성 변동의 비중 또한 가장 많은 영향을 주는 요소가 단위 시간에 투입한 노동량이기 때문에 정량화가 매우 힘들다. 따라서 조선생산실행시스템은 기존 생산실행시스템과는 다르게 사람에 의한 정보 입력의 적시성, 신뢰도와 정확성을 높이는 것을 목표로 하였으며, 이를 위해 입력 디바이스를 다양화 하여 시간과 장소를 구애 받지 않고 정보를 입력하며, 해당 정보의 정밀도를 기존 조선소 생산 시스템에 비해 정교하게 세분화 하는 것을 목표로 하였다.

1.2 조선생산실행시스템 서버

조선생산실행시스템 서버는 조선소 생산 현장에서 시시각각 변하는 정보를 관리하고 입력과 출력을 지원하는 역할을 수행한다. 더불어 조선생산실행시스템과 같은 기업정보시스템(EIS, Enterprise Information System)이라면 흔히 가지는 보안 관리, 계정 관리, 데이터베이스 통제와 같은 기능을 기반으로 구축하였으며, 관리 역할을 수행하기 위해서 각 기능의 사용 현황을 확인하고 분석할 수 있는 모니터링 기능을 포함하고 있다.

조선생산실행시스템 서버의 경우, 실제 조선소에서 이용하기 위해서는 조선소에서 기존에 이용하고 있는 APS(Advanced Planning System)이나 ERP(Enterprise Resource Planning)과 같은 기간 시스템(legacy system)과의 연동이 필수적이다. 하지만 이번 연구에서는 해당 시스템과의 인터페이스를 실제 구현하는 수준까지는 진행하지 않았고, 대신 현업 조선소 S사의 참여 아래 실무 데이터와 연동하고 효용성을 검증하는 단계로 진행하였다.

2. 본 론

2.1 조선생산실행시스템 서버 개요

조선생산실행시스템 서버는 조선소 생산 현장 정보를 다루고, 동시에 여러 사용자가 저장 중인 정보에 접근하여 입출력 및 수정이 가능해야 한다. 따라서 다른 EIS처럼 DBMS(Database Management System)을 기반으로 하는 것이 적당하다. 여기에 본 과제에서는 6가지 세부 분야에 대해서 조선생산실행시스템 기능을 제공하기 때문에 각 분야에 맞는 맞춤형 정보 제공과 관리 방법이 필요하다. 이런 기능을 제공하기 위해서는 조선생산실행시스템을 적용할 현장의 사용 시나리오를 면밀하게 검토하여 기능을 사용할 사용자에 맞추어 권한과 계정을 정교하게 설계해야 한다.

더불어 기존 조선소의 생산 현장 정보 관리 방법과 본 과제에서 개발한 조선생산실행시스템의 결정적 차이라고 할 수 있는 것이 입력 장비의 다양성이라는 것을 감안하면, 기존에 PC 기반 정보 시스템에 맞춰 개발한 서버와는 다른 방법의 개발 방법이 필요하다. 본 과제에서는 사용자가 기존처럼 PC 클라이언트를 이용해 서버에 정보를 입력하거나 조회하는 것도 가능하고, 스마트패드나 스마트폰을 이용한 정보 입력과 조회도 가능하도록 서버를 개발했다. 여기에는 플랫폼 만 놓고 보자면 Microsoft Windows 기반의 PC 환경과 스마트폰, 스마트패드를 고려하면 될 것 같지만 스마트폰이나 스마트패드는 크게 안드로이드 진영과 아이폰 진형으로 나뉘어 개발 및 구동 환경이 다르며, 같은 OS를 이용한다고 해도 스마트폰과 스마트패드는 장비 해상도가 다르기 때문에 한번에 소화할 수 있는 정보량이 달라지게 된다. 따라서 플랫폼 3가지와 해상도 2가지 경우를 조합한 경우에 대해 유연하게 적용 가능한 서버 개발 방법을 고려해야만 한다.

2.2 웹서비스 개념과 필요성

가장 널리 쓰이는 플랫폼에 대해서 서비스가 가능하도록

서버를 구축하기 위해서 3가지 플랫폼(MS-Windows, iOS, Android)에 대해 각각 서버를 구성하는 방법도 있으나 유지보수나 기능 개선을 고려하면 매우 비효율적일 수 밖에 없다. 그래서 조선생산실행시스템 서버의 경우에는 시스템과 클라이언트와의 인터페이스를 웹서비스(Web Services)로 구축하였다.

웹서비스는 소프트웨어 계층별 데이터 입출력을 위해 필요한 인터페이스를 특정 플랫폼에 종속되지 않고 표준 형식으로 구현하기 위한 표준이다. 이를테면, 시스템을 MS의 기술을 이용해서 구축하였다면 2000년대에는 DCOM을 이용해서 네트워크 분산 환경과 인터페이스를 구현하려고 했을 것이고, 닷넷 프레임워크를 이용해서 구축했다면 닷넷 리모트(.Net Remote) 등의 기술을 이용해서 분산 시스템을 구성해야 했을 것이다. 시스템을 Java 기반으로 구축하였다면 CORBA와 같은 분산 객체 표준을 이용해 인터페이스를 구현하는 것을 고려했을 것이다. 각각은 훌륭한 기술이지만 문제는 각 기술을 이용한 후에 해당 플랫폼에 종속이 되고 만다는 것이다. 이번 경우에도 서버의 애플리케이션 계층은 MS의 닷넷프레임워크로 구현하였지만, 스마트패드 클라이언트는 안드로이드 기반 장비를 고려하였기 때문에 안드로이드 개발 환경인 Java 계열과의 인터페이스가 필요하다.

하지만 웹서비스는 월드와이드웹(WWW) 표준 프로토콜인 HTTP(Hyper Text Transfer Protocol)를 이용하여 XML 기반으로 정보를 교환하므로 플랫폼 독립적인 인터페이스 구성이 가능하다.

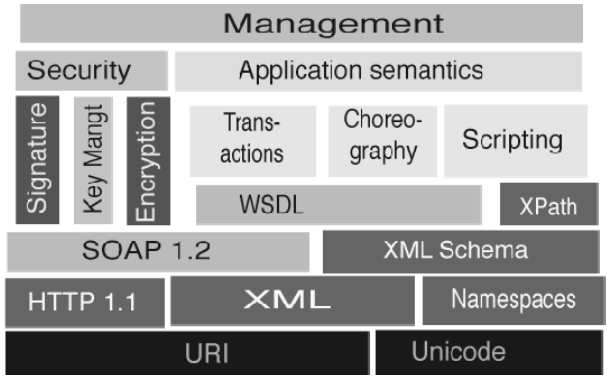

그림 1 웹서비스 구조 [www.w3.org]

월드와이드웹(WWW, World Wide Web) 표준을 관리하는 국제 단체인 W3에서 정의한 그림 1을 보면 표준 기술 기반으로 인터페이스에 필요한 모듈을 정의한 것을 알 수 있다.

그 중에서도 조선생산실행시스템 서버의 웹서비스는 기존 SOAP 표준 기반 웹서비스가 지나치게 무겁고 유연성이 부족하다는 문제점을 해결한 업계 표준(de facto) RESTful 웹서비스를 채용했다. RESTful 웹서비스는 객체 구조를 충실하게 정의한 SOAP 표준이 안정적이고 구조적인 장점이 있는 대신, 구조가 크고 복잡해 속도가 느리고, 유연한 기능 개선이 힘들다는 점에 착안하여 필수적인 정보만으로 신속한 정보 교환이 가능하도록 구성한 웹서비스 방식이다.

2.3 웹서비스 서버 구축 과정

웹서비스 서버 구축 과정에서 ㈜지노스는 전반적인 서버 환경 구축과 통합 개발 환경, 웹서비스를 구현하였으며, 각 기관들은 기관별 솔루션 개발과 개발에 필요한 기능을 설계하였다.

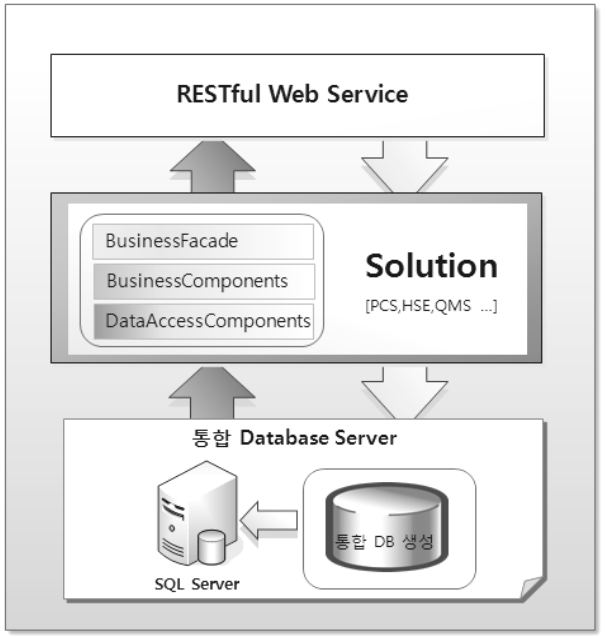

그림 2 웹서비스 서버 논리적 계층 구조도

서버는 Microsoft Windows Server 2003을 OS로 구성하고 통합 데이터베이스 구축을 위해 Microsoft SQL Server 2008 R2를 채용하였다. 각 기능별 개발을 맡은 기관의 데이터는 중복성을 최소화 하고 데이터를 효율적이고 일관성 있게 제공하기 위해 통합 관리할 수 있는 통합 데이터베이스를 구축하였다. 그리고 이 제공하는 서버에 플랫폼 독립적인 접근이 가능하도록 WCF 4.0 기반의 RESTful 웹서비스를 구현하였다.

기능별로 솔루션을 개발하였으며 체계적인 개발을 위해 CBD 방법론을 적용하였으며, 산출물로 사용자 인터페이스 설계서와 컴포넌트 설계서를 작성하며 시스템을 구성하였다. 각 솔루션은 크게 비즈니스 퍼사드, 비즈니스 컴포넌트, 데이터

액세스 컴포넌트로 구성하였으며, 계층은 기능별로 구분할 수 있다. 비즈니스 퍼사드는 웹서비스에서 접근할 수 있는 함수를 제공하며, 비즈니스 컴포넌트는 데이터를 목적에 맞게 가공하고 데이터 액세스 컴포넌트는 통합 데이터베이스에 접근하여 데이터를 관리하는 기능을 수행한다.

2.4 웹서비스 서버 구축 현황

웹서비스 서버는 RESTful 웹서비스를 웹서버를 통해 제공하게 된다. 웹서버는 Microsoft Windows Server 2003에서 제공하는 IIS(Internet Information Server) 6.0을 이용하였고, 웹서비스는 WCF(Windows Communication Foundation)를 이용해서 구현하였다. 다음 그림 3은 공용 기능에 대한 RESTful 웹서비스의 도움말 화면이다.

Uri	메서드	설명
Common_CheckUser={strEmployNumber}	GET	http://smartwork.xinnos.com/Restful/Common_CheckUser={STREMPLOYNUMBER}의 서비스
Common_CreateAuthority	POST	http://smartwork.xinnos.com/Restful/Common_CreateAuthority의 서비스
Common_CreateCommonEnum	POST	http://smartwork.xinnos.com/Restful/Common_CreateCommonEnum의 서비스
Common_CreateDepartment	POST	http://smartwork.xinnos.com/Restful/Common_CreateDepartment의 서비스
Common_CreateMobileDevice	POST	http://smartwork.xinnos.com/Restful/Common_CreateMobileDevice의 서비스
Common_CreatePicture	POST	http://smartwork.xinnos.com/Restful/Common_CreatePicture의 서비스
Common_CreateResource	POST	http://smartwork.xinnos.com/Restful/Common_CreateResource의 서비스
Common_CreateSystemRelatedDepartment	POST	http://smartwork.xinnos.com/Restful/Common_CreateSystemRelatedDepartment의 서비스
Common_CreateUser	POST	http://smartwork.xinnos.com/Restful/Common_CreateUser의 서비스
Common_DeleteAuthority={strSystemName}+{strSystemFunction}+{strDepartmentIDorUserID}	DELETE	http://smartwork.xinnos.com/Restful/Common_DeleteAuthority={STRSYSTEMNAME}+{STRSYSTEMFUNCTION}+{STRDEPARTMENTIDORUSERID}의 서비스
Common_DeleteCommonEnum={strSystemType}+{strEnumName}+{strEnumValueIndex}	DELETE	http://smartwork.xinnos.com/Restful/Common_DeleteCommonEnum={STRSYSTEMTYPE}+{STRENUMNAME}+{STRENUMVALUEINDEX}의 서비스
Common_DeleteDepartment={strDepartmentID}	DELETE	http://smartwork.xinnos.com/Restful/Common_DeleteDepartment={STRDEPARTMENTID}의 서비스
Common_DeleteMobileDevice={strCommonMobileDeviceID}+{strUserID}	DELETE	http://smartwork.xinnos.com/Restful/Common_DeleteMobileDevice={STRCOMMONMOBILEDEVICEID}+{STRUSERID}의 서비스
Common_DeletePicture={strPictureID}	DELETE	http://smartwork.xinnos.com/Restful/Common_DeletePicture={STRPICTUREID}의 서비스
Common_DeleteResource={strResourceID}+{strRevision}	DELETE	http://smartwork.xinnos.com/Restful/Common_DeleteResource={STRRESOURCEID}+{STRREVISION}의 서비스
Common_DeleteSystemRelatedDepartment={strSystemName}+{strDepartmentID}	DELETE	http://smartwork.xinnos.com/Restful/Common_DeleteSystemRelatedDepartment={STRSYSTEMNAME}+{STRDEPARTMENTID}의 서비스
Common_DeleteUser={strEmployNumber}	DELETE	http://smartwork.xinnos.com/Restful/Common_DeleteUser={STREMPLOYNUMBER}의 서비스
Common_LoginUser={strEmployNumber}+{strPassword}	GET	http://smartwork.xinnos.com/Restful/Common_LoginUser={STREMPLOYNUMBER}+{STRPASSWORD}의 서비스
Common_RequestActivity={strActivityID}	GET	http://smartwork.xinnos.com/Restful/Common_RequestActivity={STRACTIVITYID}의 서비스
Common_RequestActivityList	GET	http://smartwork.xinnos.com/Restful/Common_RequestActivityList의 서비스
Common_RequestAuthority={strSystemName}+{strSystemFunction}+{strDepartmentIDorUserID}	GET	http://smartwork.xinnos.com/Restful/Common_RequestAuthority={STRSYSTEMNAME}+{STRSYSTEMFUNCTION}+{STRDEPARTMENTIDORUSERID}의 서비스
Common_RequestBlockList={strStation}	GET	http://smartwork.xinnos.com/Restful/Common_RequestBlockList={STRSTATION}의 서비스
Common_RequestCommonEnum={strSystemType}+{strEnumName}	GET	http://smartwork.xinnos.com/Restful/Common_RequestCommonEnum={STRSYSTEMTYPE}+{STRENUMNAME}의 서비스
Common_RequestDepartment={strDepartmentID}	GET	http://smartwork.xinnos.com/Restful/Common_RequestDepartment={STRDEPARTMENTID}의 서비스
Common_RequestDepartmentList={strDepartmentType}	GET	http://smartwork.xinnos.com/Restful/Common_RequestDepartmentList={STRDEPARTMENTTYPE}의 서비스
Common_RequestLogout={strUserID}+{strPassword}	GET	http://smartwork.xinnos.com/Restful/Common_RequestLogout={STRUSERID}+{STRPASSWORD}의 서비스
Common_RequestMobileDevice={strUserID}	GET	http://smartwork.xinnos.com/Restful/Common_RequestMobileDevice={STRUSERID}의 서비스
Common_RequestPicture={strPictureID}	GET	http://smartwork.xinnos.com/Restful/Common_RequestPicture={STRPICTUREID}의 서비스
Common_RequestPictureList={strRelatedTableName}+{strRelatedID}+{strPurpose}	GET	http://smartwork.xinnos.com/Restful/Common_RequestPictureList={STRRELATEDTABLENAME}+{STRRELATEDID}+{STRPURPOSE}의 서비스

그림 3 조선생산실행시스템 RESTful 웹서비스 도움말

3. 서버 관리

3.1 서버 관리 기능

구축한 서버는 지속적인 관리가 가능해야 한다. 필수적이고 일반적인 관리 기능으로는 보안 관리, 계정 관리, 권한 관리 등이 포함될 것이다. 그런 관리 기능 중 중요한 것은 접속 정보를 기록하고 관리하는 것이다. 가깝게는 오류 발생시 상황을 확인하는 용도로 사용이 가능하고, 장기적으로는 빈번하게 사용하는 기능이나 서버의 부하를 확인하거나 통계 처리를 통해 더 많은 정보를 얻어낼 수도 있다.

본 과제에서 개발한 서버는 기본적으로 기능별로 제공하는 기능에 대해 로그 기록을 저장하고 있으며, 저장한 정보를 기반으로 기능별 사용 비중이나 시간 및 기간별 사용량 같은 간단한 정보를 확인할 수 있다.

관리 기능은 여기서 한 발 더 나아가 사용자가 호출하는 기능 사이의 연관성을 분석하여 단순한 빈도 처리만으로는 알 수 없는 정보를 분석하는 기능을 더하였다.

3.2 모니터링 기능 알고리즘

조선생산실행시스템 서버 모니터링 기능을 개발할 때 적용한 방법은 온라인 쇼핑몰에서 구매자의 구매 경향을 분석하는데 널리 이용하는 연관성 규칙(Association Rule)[Agrawal et al.]이라는 데이터마이닝 기법이다. 연관성 규칙은 구매자가 한 번에 구매한 두 개의 제품을 전체 제품과 비교하여 3가지 지표로 분석한다. 각각 지지도(support), 신뢰도(confidence), 향상도(lift)를 순서대로 구한 후 세 가지 지표의 크기에 따라 구매자가 두 제품을 함께 구매하는 것이 얼마나 연관이 있는지 파악할 수 있다. 이 방법을 통해서 온라인 쇼핑몰에서는 예상 못한 제품 구매 경향을 파악할 수 있고, 효과적인 판매 모델을 만들 수 있다.

조선생산실행시스템 서버에서는 연관성 규칙의 위 특성을 이용해서 서버에서 제공하는 모든 기능 중에서 한 클라이언트가 함께 사용하는 두 가지 기능의 연관성을 분석하는 기능을 구현하였다. 크게 6가지로 나눌 수 있는 기능 분류 중에서 수많은 세부 기능을 제공하는데, 같은 모듈에서 제공하는 기능이 연관성이 높으리라는 예상은 쉽게 할 수 있지만 다른 모듈의 기능 사이의 연관성은 예상하기가 힘들다. 조선생산실행시스템이 현장에서 구동되어 수많은 사용 로그가 쌓이면 설계 과정에서는 예상 못한 기능 사이의 연관성을 알 수 있게 된다.

3.3 모니터링 서버와 클라이언트 구조

모니터링 시스템은 로그 정보와 연관성 분석 결과를 제공하는 서버와 해당 정보를 그래프와 표로 가공하여 확인 가능한 클라이언트로 구성되어 있다. 여기서 모니터링 시스템 서버는 조선생산실행시스템 서버와 같이 RESTful 웹서비스로 구성되어 있으며, 클라이언트 시스템은 사용자들의 편의성과 접근성을 고려해 웹과 모바일에서 모두 접근이 가능하도록 jQuery와 jQuery Mobile을 이용해 구성하였다.

그림 4의 구조도에서 확인 가능한 것처럼 모니터링 시스템 클라이언트는 웹과 모바일에서 접근이 가능하며 접근하는 장비에 따라서 웹과 모바일을 구분하여 장비에 맞게 구성된 페이지를 보여준다.

그리고 모니터링 시스템 서버는 통합 데이터베이스와 모니터링 솔루션 그리고 웹서비스로 구성되었다.

웹서비스는 모니터링 솔루션에 접근하여 사용자가 원하는

정보를 XML, JSON 형식으로 값을 반환해 주며, 여기서 솔루션은 통합 데이터베이스에 접근해 데이터를 관리하는 Model과 데이터를 목적에 맞게 가공하는 Controller, 웹서비스에서 솔루션에 접근 할 수 있도록 함수를 제공하는 Facade로 구성하였다.

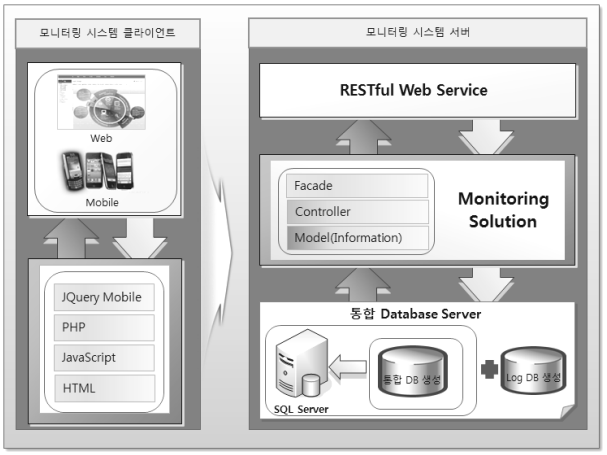

그림 4 모니터링 시스템 구조

3.4 모니터링 클라이언트 구현

모니터링 클라이언트 구현시 클라이언트 사이드 스크립트 언어로 jQuery와 jQuery Mobile을 사용함으로써 개발을 단순화할 수 있었다.

```
var ckb = document.getElementByName("ckb1");
var checkedValues = "";
for(var i = 0;i<ckb.length;i++){
    if(ckb[i].chekced)
        checkedValues += ckb[i].value + ",";
}
```

⇩

```
var checkedValues = "";
$("#ckb1:checked").each(
    checkedValues += $(this).val() + ",";
);
```

그림 5 JavaScript와 jQuery 코드 간결성 비교

jQuery는 JavaScript 라이브러리를 대표하는 언어 중 하나로 MIT 라이선스와 GNU 일반 공중 허가서 v2의 듀얼 라이선스를 가진 오픈 소프트웨어이다. jQuery는 크로스브라우징을 지원하며, Mootools와 YUI와 같은 다른 JavaScript 라이브러리에 비해 뛰어난 성능을 보여준다. 또한 그림5와 같이 jQuery의 대표적인 특징인 셀럭터를 통해 기존에 JavaScript에서 반복적으로 사용하던 'getElementByName'과 'for'문 등과 같은 함수들의 의존성을 줄이므로 코드 간결성을 확보할 수 있다.

모바일 기기를 위한 클라이언트 구현을 위해 jQuery Mobile를 추가로 사용했다. jQuery Mobile은 jQuery를 근간으로 개발돼 사용이 단순하며, HTML5를 지원하는 대부분의 모바일 기기를 지원한다. 또한 데스크탑 웹 디자인에 익숙한 개발자를 위해 ThemeRoller와 같은 UI 디자인 툴을 지원함으로써 모바일 UI를 효과적으로 개발할 수 있다.

3.5 모니터링 그래프 구성

모니터링 클라이언트은 그래프를 통해 크게 두 가지 분석 결과를 제공하도록 개발했다. 첫 번째는 기본적인 로그 분석으로 가능한 사용량을 제공하는 것이고, 두번째는 앞 장에서 다뤘던 연관성 분석 결과다.

사용량 분석 기능에는 사용량의 순위를 볼 수 있는 막대 그래프와 원형 그래프, 그리고 기간별 사용량의 변화를 볼 수 있는 선형 그래프, 3가지가 사용된다.

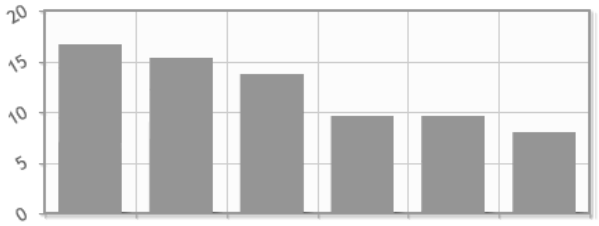

그림 6 기간별 사용량 분석 그래프

막대 그래프는 일정 기간 동안의 메서드와 동사에 사용량을 순서에 따라 정렬하여 보여준다. 원형 그래프는 앞에 막대 그래프에 표현된 사용량을 사용 비율로 보여준다.

선형 그래프는 특정 메서드와 동사의 사용량이 일정 기간 동안 어떻게 변화해 왔는가를 보여준다. 위의 3가지 그래프들은 모두 최근 한 달의 기간을 기본으로 하고 사용자 임의로 기간을 변경할 수 있다.

연관성 분석 기능은 전체 메서드와 동사에 대한 연관성의 정도를 볼 수 있는 버블 그래프, 특정 메서드와 동사에 대한 연관성 순위와 정도로 볼 수 있는 막대 그래프가 사용된다.

버블 그래프는 웹서비스 제공 메서드와 동사 사이의 연관성을 버블의 크기로 보여준다. 막대 그래프에서는 웹서비스 제공 메서드와 동사의 목록에서 선택한 하나의 메서드와 동사에 대해 연관성 3지표(지지도, 신뢰도, 향상도)를 높은 순서대

로 보여준다. 위의 막대 그래프 또한 최근 한 달의 기간을 기본으로 하고 사용자 임의로 기간을 변경할 수 있다.

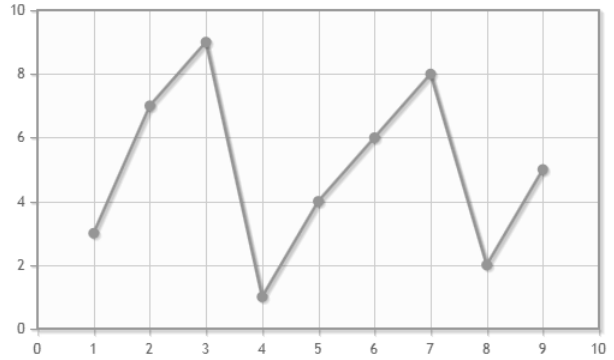

그림 7 기간별 변화량 분석

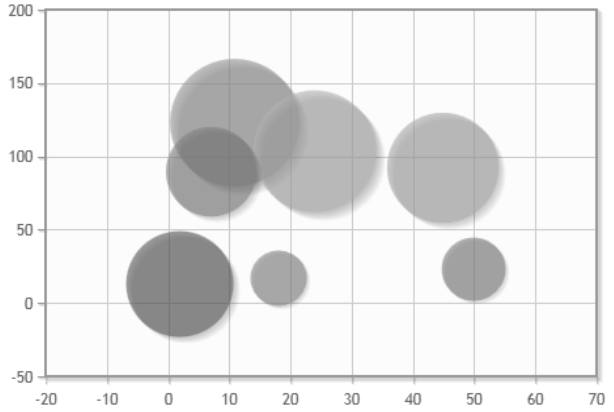

그림 8 연관성 분석 결과 그래프

4. 결 론

조선소 생산 현장 정보를 관리하기 위해 개발한 조선생산실행시스템의 서버 구축 과정을 정리하였다. 서버 구축 과정과 플랫폼 독립적인 인터페이스 구축을 위한 RESTful 웹서비스 기반 구현, 연관성 분석 기능을 포함한 모니터링 모듈 개발을 정리하였으며, 조선생산실행시스템에 적절한 기능을 제공함과 동시에 유연한 서버를 구축하기 위해 최신 웹 기술을 적용하였다.

구축한 서버는 실제로 조선소에 적용이 될 때 효용성도 증명할 수 있고 부족한 점을 찾아 기능을 개선할 수도 있다. 그동안 조선소에서 필요했지만 적용하지 쉽지 않았던 생산실행시스템에 대한 도전적인 연구 결과인 만큼, 실제 적용을 통해 시스템의 기능 향상을 포함하여 한국 조선의 고도화에도 조금이나마 이바지 할 수 있기를 바란다.

후 기

본 연구 내용은 지식경제부 글로벌전문기술개발사업 (과제번호 10039739, Smart Work기반 조선생산실행시스템 개발)으로 지원된 연구의 일부로 수행된 것을 정리한 것으로, 위 기관의 후원에 감사드립니다.

참 고 문 헌

W3 Consotium, http://www.w3.org/

Agrawal, R.; Imieliński, T.; Swami, A. (1993). "Mining association rules between sets of items in large databases". Proceedings of the 1993 ACM SIGMOD international conference on Management of data – SIGMOD '93.

이 필 립

■ 1979년생
■ 2008년 서울대학교 조선해양공학과 졸업
■ 현 재 : 지노스 PLM 연구소장
■ 관심분야 : 디지털 조선소, 가상 생산
■ 연 락 처 : 02-596-1488
■ E – mail : philippe_lee@xinnos.com

박 영 미

■ 1985년생
■ 2008년 한국성서대학교 소프트웨어학과 졸업
■ 현 재 : 지노스 박영미 사원
■ 관심분야 : 프로그램 개발
■ 연 락 처 : ***-****-****
■ E – mail : youngmi_park@xinnos.com

정 종 필

■ 1981년생
■ 2001년 한국산업기술대학교 컴퓨터공학과 졸업
■ 현 재 : 지노스 PLM 연구소 대리
■ 관심분야 : 소프트웨어 개발 방법론
■ 연 락 처 : ***-****-****
■ E – mail : jongpil_jung@xinnos.com

권 기 욱

■ 1988년생
■ 2013년 동양미래대학교 로봇시스템과 졸업
■ 현 재 : 지노스 PLM 연구소 연구원
■ 관심분야 : 미래 신기술
■ 연 락 처 : ***-****-****
■ E – mail : kiwook_kwon@xinnos.com