Open farm information system data-exchange platform for interaction with agricultural information systems

Joon Yong Kim[1], Chun Gu Lee[2], Seung Hwan Baek[2], Joong-Yong Rhee[1,2*]

(1. Research Institute for Agriculture and Life Sciences, Seoul National University, Seoul, Republic of Korea ;

2. Department of Biosystems Engineering, Seoul National University, Seoul, Republic of Korea.)

Abstract: Information systems for agriculture started from simple agricultural data storages. They have been improved to support farm managements or farm automations that employ optimal decision-making techniques by integrating various data. The interconnectivity of an information system is important to realizing additional values. The objectives of this study were to develop a data-exchange platform for Open Farm Information System (OFIS) and to suggest it to other farm management information systems (FMIS). OFIS is an integrated FMIS based on cloud computing technology and it uses the data-exchange platform to communicate with other information systems. OFIS data-exchange platform consists of two services: data-collection service and data-sharing service. For these services, data collection procedure for external data was analyzed and data sharing interface used to access internal data stored in OFIS was designed. In order to evaluate the data-exchange platform, two use cases were illustrated and a data consistency test was conducted. The use cases revealed that the platform was flexible and extensible in collecting from various data sources such as web services and serial communication devices. For the data consistency test, the data-collection service collected data from target data sources without any data loss, and the data-sharing service provided all expected data without error. As a result, this type of data-exchange platform would be useful for FMIS in terms of flexibility, simplicity, and extensibility.

Keywords: Farm management information system, data exchange, web service, system integration, open farm information system.

Citation: Kim, J. Y., C. G. Lee, S. H. Baek, and J. Rhee. 2015. Open farm information system data-exchange platform for interaction with agricultural information systems. Agric Eng Int: CIGR Journal, 17(2):296-309.

1 Introduction

Rapid development of information and communication technology (ICT) has influenced many areas of research including agriculture. In its early adoption in agriculture, the advantages of ICT were not clear. It was used to assist financial accounting or was believed to provide some potential gains (Batte, 2005; Kwon et al., 2014; Rolfe et al., 2003). Nowadays, computer systems have come to occupy a major position in agriculture and has been used from tiny sensor nodes to farm management information systems (FMIS) (Kaloxylos et al., 2012; Kim et al., 2013; Lee et al., 2010; Nash et al., 2009a; Schuster et al., 2011).

Agricultural data involve not only environment and crop information but also financial, biological, and technical information. Therefore, variety is one of the major characteristics of agricultural data. There are many types of agricultural data such as local weather, soil information, and so on. Steinberger et al. (2009) pointed out that the heterogeneous data structures of many agricultural systems used for data acquisition, GIS-based field indexing, different documentation tasks, and precision farming applications produce a variety of data formats and interfaces. Data volume is also an essential characteristic. Luck et al. (2013) stated the volume of agricultural machinery data would increase from 1.77 MB/ac to 14.4 MB/ac according to the resolution of precision agriculture would increase. These characteristics are reinforced as management processes in

Received date: 2014-12-02 **Accepted date:** 2015-04-07

***Corresponding author: Joong-Yong Rhee.** Research Institute for Agriculture and Life Sciences, Seoul National University, Seoul, Republic of Korea; Department of Biosystems Engineering, Seoul National University, Seoul, Republic of Korea. Email: jyr@snu.ac.kr

agriculture become more complex because of precision farming technologies, bigger farms, the sharing of machinery, and contracting work (Steinberger et al., 2009). Regarding agricultural data integration, Northern and Western European countries are considered data integrated countries. In most of these countries, databases for registering land parcels are shifting toward open systems, portals, or shared databases (Holster et al., 2011).

Schmitz et al. (2009a) stated that a standardized system for data exchange offers new possibilities for information-directed agricultural production to increase sustainability and minimize negative effects on the environment. It is essential to follow standards for integrating agricultural data. According to Nash et al. (2009b), five standards have been established for the transfer of agricultural data: ISO 11783 (ISOBUS), AgroXML, AgXML, eDAPLOS, and EDI-Teelt. The standards of the Open Geospatial Consortium (OGC), which were developed for sensor networks, can be used for agricultural sensor networks as well. Nash et al. (2009a) presented researches on applying standards from OGC and related initiatives to automate agricultural data processing. Most standards use extensible markup language (XML) as a data format and can be implemented as web services. OpenAPI, which refers to sets of technologies that enable websites to interact, is another way for agricultural data exchange. Shim et al. (2009) suggested an agricultural data exchange model that uses OpenAPI web service as the new paradigm of Web 2.0. A website called “Public Data Portal”, which provides information about OpenAPI services, is supported by the Korean government. The site contains information related to 7,776 data sets and 563 OpenAPI services provided by 786 organizations (MOSPA, 2014). In addition to Korea, many countries and companies support OpenAPI web services.

Some researches on data exchange between agricultural systems or mobile machineries have been conducted. Murakami et al. (2007) proposed an agricultural system infrastructure based on service-oriented architecture. The infrastructure includes a reference architecture, a standard language for data exchange between systems and services based on XML, and a service bus for use as a message-oriented middleware for connection with web services. Kaivosoja et al. (2014) developed a task controller prototype employing ISOBUS-compatible data messages. The controller utilizes external data from multiple sources during a spraying operation.

This study was initiated from a project to develop Open Farm Information System (OFIS), which is an integrated FMIS based on cloud-computing technology. Enabling OFIS to communicate with other information systems was essential. In order to collect data from various data sources including sensor networks and agricultural information systems (AIS), a general collection method was required. Moreover, it was necessary to provide internal OFIS data which could be used to acquire a farm-specific service from other information systems. The objectives of this study were to develop a data-exchange platform for OFIS and to suggest the data-exchange platform to other FMISs. The data-exchange platform consists of two services: data-collection service and data-sharing service. Most agricultural standards for data communication have not gained a large market share, and are largely national efforts (Nash et al., 2009b). In addition, no standard embraces all agricultural domains. Therefore, the data-sharing service could not follow a specific standard but the data-collection service should be sufficiently flexible to collect data using the standards.

2 Materials and methods

2.1 Data sources for collection

AIS is an information system that manages general agricultural information related to various crops and their varieties, agro-inputs, cultivation practices, and farming activities (Chaudhary et al., 2004). In addition, AIS represents a rich source of information maintained and published for the benefit of farmers and agro-professionals (Laliwala et al., 2006). Today, many AIS systems support OpenAPI for sharing their data. The OpenAPI was not designed to provide information to humans but rather to machines. Most OpenAPIs use hypertext transfer protocol (HTTP) as a data transfer protocol and XML or JavaScript Object Notation (JSON) for data format. Three popular Korean OpenAPI services, which are related to agriculture, were selected for use in this study: the National Crop Pest Management (NCPMS) OpenAPI service, a weather information service provided by the Korean Meteorological Administration (KMA), and the Okdab OpenAPI service provided by the Korean Agency of Education, Promotion and Information Service in Food, Agriculture, Forestry and Fisheries (EPIS). NCPMS is a specialized service related to pest and disease information and offers 15 functions for searching pest and disease information. It also includes four functions used to predict future occurrence of pests and diseases. The KMA weather service provides weather information about ambient temperatures, relative humidity, sky conditions, wind speed and direction, etc. The Okdab OpenAPI service primarily manages crop prices. Eleven of 19 services are related to crop price. Although all three services are known as OpenAPI services, they are different with respect to content, data formats and access techniques as summarized in Table 1.

Table 1 Data sources as OpenAPI services of AIS in Korea

Name	URL	Service item	Access method
KMA OpenAPI	http://www.kma.go.kr/wid/queryDFSRSS.jsp	Forecast weather by regional group	RSS (XML)
OKDAB OpenAPI	http://openapi.okdab.com/openapi_main.jsp	Agricultural dictionary service, Real-time agro-product price information, Agricultural news	Web service (XML)
NPCMS OpenAPI	http://npms.rda.go.kr/npms/OpenApiInfo.np	Pest/Disease information, Pest/Disease prediction	Web service (XML)

Target data sources can be not only web services, but also sensor systems. Since a weather station has several types of sensors to collects meteorological factors, it is a kind of sensor system having a sensor node. A weather station (WS-100, SNJ Corp, Korea.), which measures seven meteorological factors: ambient temperature, ambient relative humidity, dew point, solar radiation, wind speed, wind direction, and precipitation, was selected as a data source.

2.2 Data groups for sharing

FMIS is defined as a planned system for the collection, processing, storage, and dissemination of data necessary to accomplish the operational functions of a farm (Sørensen et al., 2010). The FMIS can exchange data with various sensors, mobile farm machineries, and other information services on the Internet. OFIS includes FMIS implemented as a web application. The interconnectivity with other AISs was one of the essential requirements of OFIS.

OFIS contains various data related to farm operations and internal services. To exchange data with other systems, the internal data were categorized into four groups. The first group contains basic farm information. It includes the location of the farm, contact information, information about crop fields and the main crop, and owner information. The second group of data contains local weather information. If the farm has its own weather station, it can be used to predict crop growth or make decisions for farming. The third group contains information about sensors that are installed on the farm. It includes both sensor system descriptions and sensor observations. The third group overlaps somewhat with the second group because a weather station has a set of sensors that measure meteorological factors. However, the two groups were separated because the weather information is crucial to crop farming as much as it takes a group. The last group contains working history. OFIS can store the daily working history of a farmer and manage his or her schedule. Table 2 shows the data groups and their items.

Table 2 Data groups and items of OFIS

Groups	Subgroup	Items
Farm information	Farm information	Name, address, main crop, contacts, and owner
	Fields information	Type, crop, shape, and description
Local weather	Weather station description	Location and status
	Weather	Temperature, humidity, dew point, rain, wind (speed/direction), and solar radiation
Sensor information	Sensor description	Location, type, and status
	Sensor observation	Observed time and value
Working history		Action, workers, equipment, resource, income, expense, date, attached file, and description

2.3 Data-exchange platform

A data-exchange platform consists of data-collection service and data-sharing service. The data-collection service is based on data collection procedure, which provides a general manner in which to collect data from many types of data sources. The data-sharing service operates based on data sharing interface.

2.3.1 Data collection procedure

In order to establish a data collection procedure, several cases of data collection were analyzed. Most data sources use standard data transfer protocols such as HTTP, RS232 and standard data formats such as XML, JSON. However, the content and its structure are defined by service providers. The methods to access data could be generalized based on the standards, but the methods to parse data could not be generalized. In some cases, the parsing method should be rewritten. After data parsing, some parts in the parsed data might be useless in OFIS or the unit of data is not match with OFIS. Various data processing methods are needed such as encoding conversion, XSLT translation, and arithmetic calculation. After data are processed, the data must obtain new metadata because it needs relationships with other saved data in OFIS. For instance, if the price of an apple at a certain joint market is collected, an identifier in OFIS should be assigned to the price information in order to be saved in OFIS database. After establishing a relationship between new data and the identifier, the new data can be saved.

Based on analysis results, four steps in the data collection procedure are identified: 1. Data are retrieved from a data source. 2. Data should be parsed and processed using different processing methods. 3. Data should be mapped with inside identifiers. 4. Data is then saved. Figure 1 shows the data collection procedure.

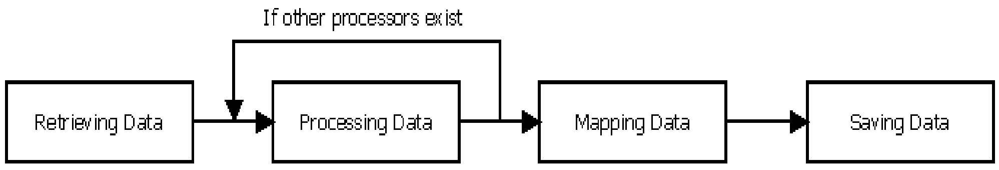


Figure 1 Data collection procedure in OFIS

2.3.2 Data sharing interface

In order to design data sharing interface, two standards for agricultural data representation were reviewed. AgroXML is a standardized language for data exchange in agriculture. It is based on XML and uses XML Schema as its definition language. AgroXML was developed by agricultural software makers, machinery companies, service providers, and the Association for Technology and Structures in Agriculture (KTBL) in Germany (Schmitz et al., 2009b). OGC is an international industry consortium of many companies, government agencies, and universities participating in a consensus project to develop publicly available interface standards for sensors and sensor networks (OGC, 2012). Representative standards of OGC for data exchange include sensor model language (SensorML) and Observations and Measurements (O&M). A sensor observation service provides a web-based interface for managing deployed sensors and retrieving sensor observation data (Na and Priest, 2007). AgroXML can be used for basic farm information and work-history data groups. In addition, OGC standards can be used for local weather and sensor data groups. However, no standard embraces all data groups in OFIS.

Data formats have evolved from being markup and display-oriented to supporting metadata describing the structural attributes of the information (Nurseitov et al., 2009). XML and JSON are representative data exchange formats. XML is a subset of the standard generalized markup language (SGML) and describes a class of data objects known as XML documents. Each XML document has both a logical and a physical structure (Bray et al., 1998). JSON is also designed as a data-exchange format. It is a lightweight text-based language-independent data-interchange format. It was derived from the ECMAScript programming language standard and defines a small set of formatting rules for the portable representation of structured data (Crockford, 2006). XML and JSON are readable by humans and easy for machines to parse and use. They are typically used to transmit data between a server and a web application. In the field of agriculture, XML and JSON are used in a various applications. Han et al. (2012) developed the CropScape web service, which offers functions to query, visualize, disseminate, and analyze crop and other specific land-cover classification data. CropScape uses XML-based standards of the OGC, and JSON is the data format. Arroqui et al. (2012) also developed a web-service-based whole-farm simulator accessible from an Android smartphone. It uses an XML-based simple object access protocol (SOAP), which is a protocol for exchanging information in a decentralized, distributed environment. White et al. (2013) provided an overview of the International Consortium for Agricultural Systems Applications (ICASA) Version 2.0 standard for describing experiments. JSON was used to store actual data describing an experiment. In addition, many other studies have used XML or JSON as a data format (Heindl et al., 2010; Kubicek et al., 2013; Montoya et al., 2013; Rafoss et al., 2010; Schuster et al., 2011).

Although JSON and XML possess unique strengths, the performance and resource utilization must be understood for choosing a data format (Nurseitov et al., 2009). For data sharing interface, JSON was chosen as a service-data format because it is faster, and uses fewer resources than its XML counterpart (Nurseitov et al., 2009). Instead of building applications, a web service is more dynamic and can find, retrieve, and invoke a distributed service dynamically (Murakami et al., 2007). A RESTful webservice is a service of allowing communication between a web-based client and server that employs representational state transfer (REST) architecture introduced by Fielding (2000). By REST convention, HTTP verbs are mapped to retrieve, create, update, and remove resources specified by the URL. It has gained in popularity because of its simplicity to use and implement contrast to protocols such as the SOAP (Cantelon et al., 2012). Based on these knowledge, the data sharing service was designed as a RESTful web service using JSON as a data format.

3 Results and discussion

3.1 Implementation

3.1.1 Data-collection service

The data collection procedure consists of four steps. A step would abstract as a process to get a data set, process the data, and transfer a new data set containing processed data to the next step. OFISBJunk is a class representing a data set. All steps receive an instance of OFISBJunk from the previous step and return a new instance of OFISBJunk storing processed data. Because of this abstraction, each step can be describes in same way and have the same parent class, called OFISBRoutine. OFISBRoutine has four abstract children classes: OFISBSource, OFISBProcessor, OFISBMapper, and OFISBSink. A child class of OFISBSource class retrieves data from various data sources. OFISBProcessor class has children classes that can parse data, change the encoding of data, transform an XML data format, and calculate data. OFISBMapper class has two children classes. One of these, called OFISBdataMapper, can map an identifier to one record of data. OFISSink class has children classes that can save data to two databases or a text file. Figure 2 shows a class design to implement a data collection procedure model. OFISBee is a main class that can read a configuration file and execute the configuration. OFISBStream represents a data collection procedure consisting of four or more routines which represent the steps.

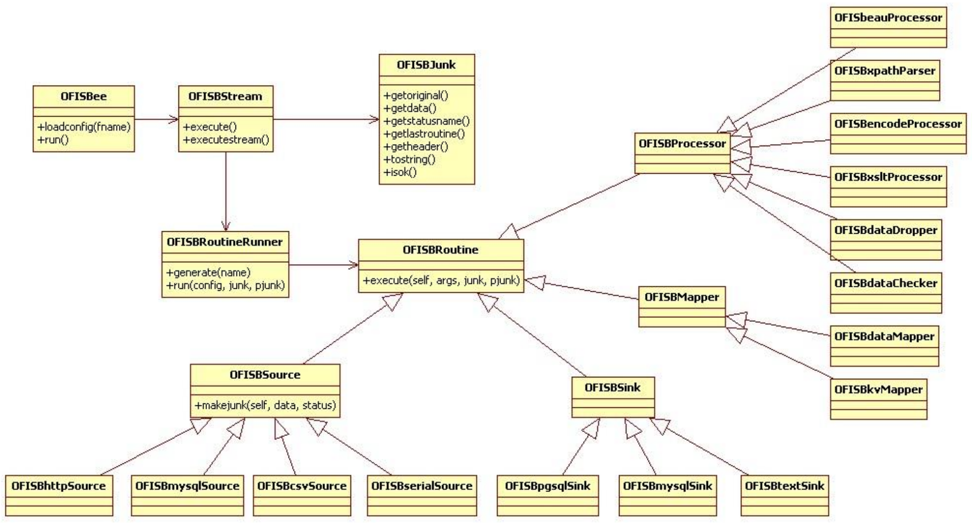


Figure 2 Class design to implement a data collection procedure model

The data-collection service was implemented using Python programming language (2.7.3, Python Software Foundation), which is a widely used general-purpose high-level language. The data-collection service was tested on two major operating systems: Ubuntu (12.04.4, Linux 3.2.0-57, Canonical Ltd.) and Windows (XP, Microsoft Corporation). All designed classes were implemented using standard libraries and the following seven external libraries: MySQL, psycopg2, ClientCookie, BeautifulSoup, libxml2, libxslt, and serial. It collects data from four types of data sources such as a web page or a web service, a database, a comma separated value format file, and a serial communication device. It has six types of data processing classes and two types of data mapping classes. Lastly, it can save data to three types of data storages. Figure 3 shows all implemented classes for each step of the data collection procedure.

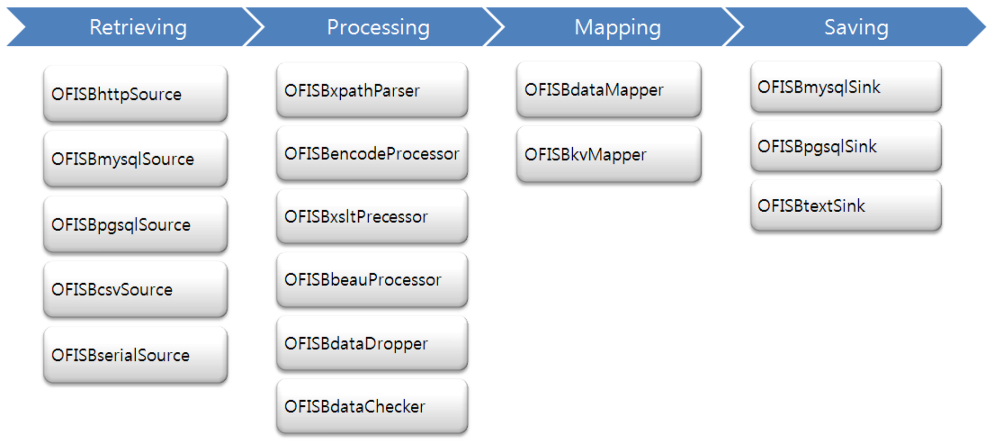


Figure 3 Implemented classes for each step of the data collection procedure

The data-collection service is flexible because it can support many possible combinations of classes. For example, a combination of OFISBhttpSource, OFISBxpathParser, OFISBdataMapper, and OFISBpgsqlSink enable the system to access an XML document from a web service, parse a document using XML path language (XPath), set proper metadata to the parsed data, and save data to a PostgreSQL database. At the combination, it is essential to know that the OFISBhttpSource could be used to retrieve data from any web pages or web services. Because all the classes are reusable, it can give flexibility to the data-collection service. Each combination is described by a configuration file, which is a text file containing sequences of classes and arguments for execution.

3.1.2 Data-sharing service

The data-sharing service was implemented as a RESTful web service unlike traditional software. To access four groups of data, 13 operations were designed. All operations employ a GET method, because the

Table 3 Designed operations for RESTful web service

Groups	HTTP Verb	Path	Action	Used for
Basic farm information	GET	/farm	Show	Display farm information
	GET	/farm/fields	Index	Display a list of fields
	GET	/farm/fields/:id	Show	Display information of a field
Local weather information	GET	/weatherstation	Show	Display description of the weather station
	GET	/weather	Show	Display current weather information
	GET	/weather/:datetime	Show	Display weather information at :datetime
	GET	/weather/:from,:to	Show	Display weather information between :from and :to
Sensor information	GET	/sensors	Index	Display a list of sensors
	GET	/sensornode/:id/observations/	Show	Display the latest observation of a sensor node
	GET	/sensornode/:id/observations/daily/:date	Show	Display daily statistics of observations of a sensor node
	GET	/sensornode/:id/observations/:from,:to	Show	Display observations of a sensor node between :from and :to
Working history	GET	/workhistory/:from,:to	Index	Display a list of working history
	GET	/workhistory/:id	Show	Display detail information of a specific work

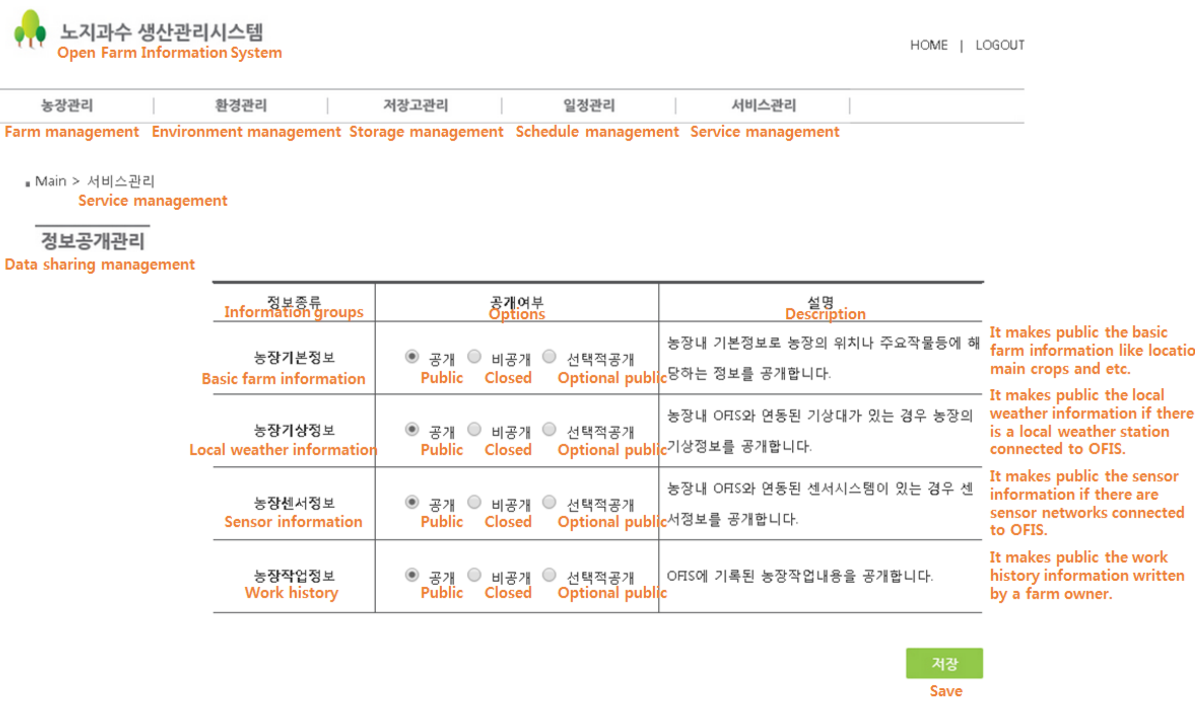
providing data are read-only. Table 3 shows all

A node.js framework, which is considered effective for developing a prototype server, was used to implement the data-sharing service. It is a high-performance concurrent program that does not rely on a mainstream multithreading approach, but rather uses asynchronous input-output with an event-driven programming model (Tilkov, 2010). At the core of the node.js framework is a streaming HTTP parser consisting of optimized C codes. This parser, in combination with a low-level TCP API that a the framework exposes to JavaScript, provides a low-level but flexible HTTP server (Cantelon et al., 2012).

In order to share internal data of OFIS, obtaining an

operations for the RESTful web service.

owner's permission is necessary. The user interface, which can be used to determine information to be shared, was implemented on OFIS. Each data group can have one of three options: public, optional public, and closed. The public option means that anybody can access the data group. The optional public option means that the data owner can give permission to a specific user when the user asks. Figure 4 shows a graphical user interface containing options the data owner can select. Because it was developed for Korean farmers, the base language is Korean. English translations (in orange) were added in Figure 4.



노지과수 생산관리시스템
Open Farm Information System
HOME | LOGOUT
농장관리 | 환경관리 | 저장고관리 | 일정관리 | 서비스관리
Farm management Environment management Storage management Schedule management Service management
Main > 서비스관리
Service management
정보공개관리
Data sharing management

정보종류 Information groups	공개여부 Options	설명 Description	
농장기본정보 Basic farm information	[✓ 공개 Public] 비공개 Closed 선택적공개 Optional public	농장내 기본정보로 농장의 위치나 주요작물등에 해당하는 정보를 공개합니다.	It makes public the basic farm information like locatio main crops and etc.
농장기상정보 Local weather information	[✓ 공개 Public] 비공개 Closed 선택적공개 Optional public	농장내 OFIS와 연동된 기상대가 있는 경우 농장의 기상정보를 공개합니다.	It makes public the local weather information if there is a local weather station connected to OFIS.
농장센서정보 Sensor information	[✓ 공개 Public] 비공개 Closed 선택적공개 Optional public	농장내 OFIS와 연동된 센서시스템이 있는 경우 센서정보를 공개합니다.	It makes public the sensor information if there are sensor networks connected to OFIS.
농장작업정보 Work history	[✓ 공개 Public] 비공개 Closed 선택적공개 Optional public	OFIS에 기록된 농장작업내용을 공개합니다.	It makes public the work history information written by a farm owner.

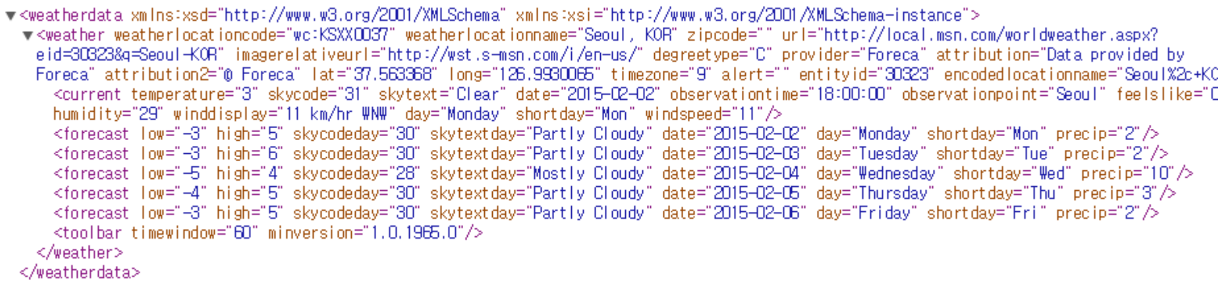
Figure 4 User interface for selecting data-sharing options

3.2 Use case

3.2.1 Data collection from MSN weather service

The data-collection service can be used to collect data from various data sources such as OpenAPI web services. In order to show the simplicity of the data-collection service, a use case to collect local weather from the MSN weather service was provided. Because weather is an important factor to directly affects growth characteristics and influences crop damage (Laurenson, 1990; Stewart et al., 1984), the use case would be reasonable.

In order to use the MSN weather service, a service URL and an address of a farm are required. A single instance of OFISBhttpSource can retrieve an XML document using the URL and the address.



```
▼<weatherdata xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 ▼<weather weatherlocationcode="wc:KSXX0037" weatherlocationname="Seoul, KOR" zipcode="" url="http://local.msn.com/worldweather.aspx?
  eid=30323&q=Seoul-KOR" imagerelativeurl="http://wst.s-msn.com/i/en-us/" degreetype="C" provider="Foreca" attribution="Data provided by
  Foreca" attribution2="© Foreca" lat="37.563368" long="126.9930065" timezone="9" alert="" entityid="30323" encodedlocationname="Seoul%2c+KC
    <current temperature="3" skycode="31" skytext="Clear" date="2015-02-02" observationtime="18:00:00" observationpoint="Seoul" feelslike="C
    humidity="29" winddisplay="11 km/hr WNW" day="Monday" shortday="Mon" windspeed="11"/>
    <forecast low="-3" high="5" skycodeday="30" skytextday="Partly Cloudy" date="2015-02-02" day="Monday" shortday="Mon" precip="2"/>
    <forecast low="-3" high="6" skycodeday="30" skytextday="Partly Cloudy" date="2015-02-03" day="Tuesday" shortday="Tue" precip="2"/>
    <forecast low="-5" high="4" skycodeday="28" skytextday="Mostly Cloudy" date="2015-02-04" day="Wednesday" shortday="Wed" precip="10"/>
    <forecast low="-4" high="5" skycodeday="30" skytextday="Partly Cloudy" date="2015-02-05" day="Thursday" shortday="Thu" precip="3"/>
    <forecast low="-3" high="5" skycodeday="30" skytextday="Partly Cloudy" date="2015-02-06" day="Friday" shortday="Fri" precip="2"/>
    <toolbar timewindow="60" minversion="1.0.1965.0"/>
  </weather>
</weatherdata>
```

Figure 5 Example of an XML document from the MSN weather service

Figure 5 shows an example of an XML document from the MSN weather service. A "current" tag contains current weather information, while "forecast" tags contain forecasted weather in five days. The document can be easily parsed by means of an instance of OFISBxpathParser. The parsed data requires metadata to save the data to a database in OFIS. PostgreSQL database (9.1.5, the PostgreSQL Global Development Group) with PostGIS (1.5.3-2) is used as a database in OFIS. This combination has been used in many agricultural studies (Blauth and Ducati, 2010; Nikkilä et al., 2010; Steinberger et al., 2009).

The data-collection service, which collects data from the MSN weather service, can be executed based on a configuration file. Table 4 shows that the configuration consists of four sections: source, processors, mapper, and sink. The sections represent the data collection procedure. The source section explains the method for obtaining the data from the MSN weather service. The processors section explains how to extract specific data items from the XML document. In this case, it uses XPath to parse the weather information. The mapper section explains how to provide metadata to extracted data. Lastly, the sink section explains how to save the mapped data. In this case, it saves the data in a "localweather" table using automatically generated SQL queries after the table is truncated. The database information in Table 4 is asterisked because it is considered private information.

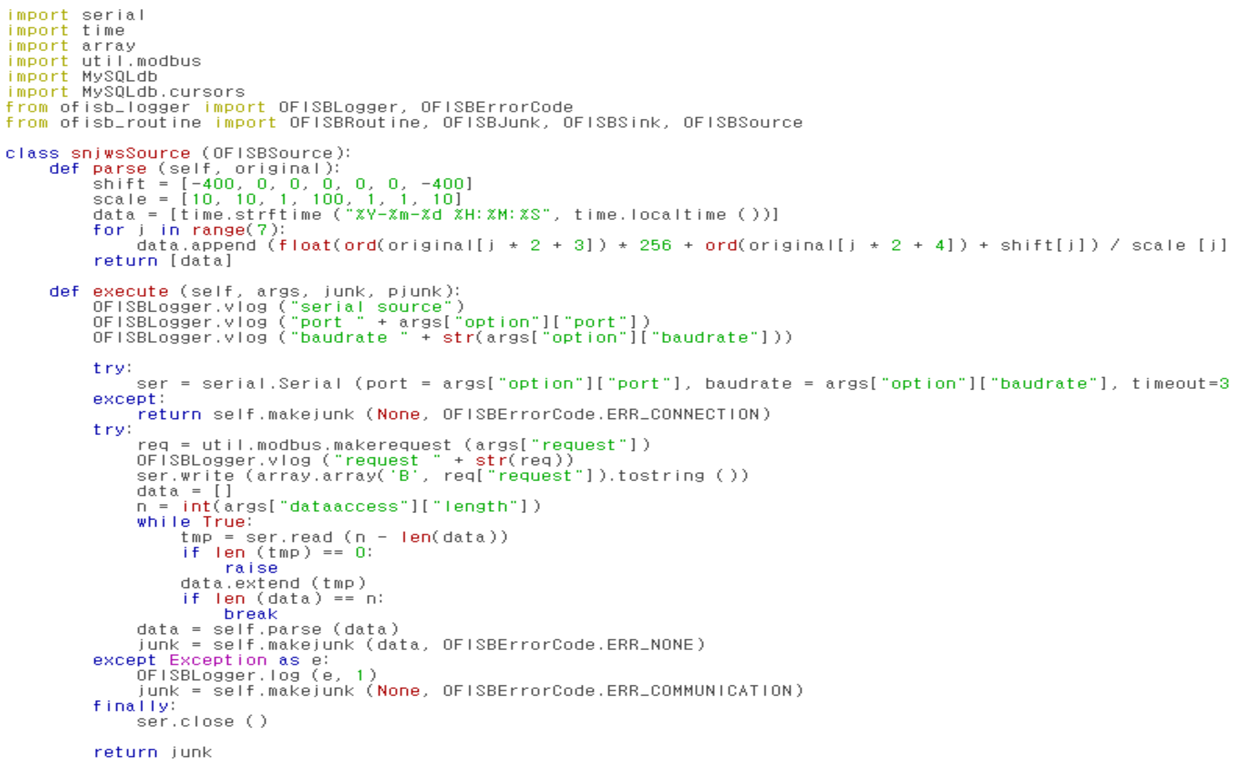
Table 4 Configuration to collect local weather information

```
{
    "source" : {
        "name" : "httpSource",
        "args" : {
            "dataaccess" : {
                "url" : "http://weather.service.msn.com/data.aspx",
                "method" : "GET",
                "args" : [
                    "weadegreetype=C&weasearchstr=Seoul"
                ]
            }
        }
    },
    "processors" : [{
        "name" : "xpathParser",
        "args" : {
            "format" : "xml",
            "start" : 1,
            "end" : 1,
            "xpaths" : [
                "/weatherdata/weather/current/@date",
                "/weatherdata/weather/current/@observationtime",
                "/weatherdata/weather/current/@temperature",
                "/weatherdata/weather/current/@humidity",
                "/weatherdata/weather/current/@windspeed"
      ]
        }
    }],
    "mapper":   {
        "name" : "dataMapper",
        "args" : {
              "observedtime" : {"type" : "datetime", "indexes" : [0, 1], "pattern" : "%Y-%m-%d %H:%M:%S"},
            "temperature" : {"type" : "float", "index" : 2},
            "humidity" : {"type" : "float", "index" : 3},
            "windspeed" : {"type" : "float", "index" : 4}
        }
    },
    "sink" : {
        "name" : "pgsqlSink",
        "args" : {
            "login" : {
                "username" : "********",
                "password" : "********",
                "dbname" : "********",
                "host" : "********"
            },
            "autoquery" : "true",
            "autoclear" : "true",
            "table" : "localweather"
        }
    }
}
```

This use case shows the simplicity of the data-collection service. The writing of programming code was not necessary but only writing a configuration file was required. It could simplify the data collection process and automate it.

3.2.2 Data collection from local weather station

The second use case shows collecting weather data from a weather station. The data packet of the target weather station follows the Modbus protocol standard. A Modbus message is a binary packet containing address, function, data, and 16-bit CRC checksum. Although the data-collection service supports a serial communication device as a data source using the OFISBserialSource class. However, the class only supports character stream with a delimiter not binary packet. It was necessary to develop a class that supports the Modbus protocol. Because the data-collection service was designed based on the four steps of the data collection procedure, a new source class inherited OFISBSource was developed. As shown in Figure 6, the source code of the new class called snjwsSource has only 40 lines and other classes can be reused. It means that most of the configuration shown in Table 4 could be used for this use case. It shows that the functions of the data-collection service can be extended easily. If it must collect data from a new data source, it may implement a new class to retrieve data from the new data source not an entire program because other classes were implemented independently and they can be used with the new class.



```
import serial
import time
import array
import util.modbus
import MySQLdb
import MySQLdb.cursors
from ofisb_logger import OFISBLogger, OFISBErrorCode
from ofisb_routine import OFISBRoutine, OFISBJunk, OFISBSink, OFISBSource

class snjwsSource (OFISBSource):
    def parse (self, original):
        shift = [-400, 0, 0, 0, 0, 0, -400]
        scale = [10, 10, 1, 100, 1, 1, 10]
        data = [time.strftime ("%Y-%m-%d %H:%M:%S", time.localtime ())]
        for j in range(7):
            data.append (float(ord(original[j * 2 + 3]) * 256 + ord(original[j * 2 + 4]) + shift[j]) / scale [j]
        return [data]

    def execute (self, args, junk, pjunk):
        OFISBLogger.vlog ("serial source")
        OFISBLogger.vlog ("port " + args["option"]["port"])
        OFISBLogger.vlog ("baudrate " + str(args["option"]["baudrate"]))

        try:
            ser = serial.Serial (port = args["option"]["port"], baudrate = args["option"]["baudrate"], timeout=3
        except:
            return self.makejunk (None, OFISBErrorCode.ERR_CONNECTION)
        try:
            req = util.modbus.makerequest (args["request"])
            OFISBLogger.vlog ("request " + str(req))
            ser.write (array.array('B', req["request"]).tostring ())
            data = []
            n = int(args["dataaccess"]["length"])
            while True:
                tmp = ser.read (n - len(data))
                if len (tmp) == 0:
                    raise
                data.extend (tmp)
                if len (data) == n:
                    break
            data = self.parse (data)
            junk = self.makejunk (data, OFISBErrorCode.ERR_NONE)
        except Exception as e:
            OFISBLogger.log (e, 1)
            junk = self.makejunk (None, OFISBErrorCode.ERR_COMMUNICATION)
        finally:
            ser.close ()

        return junk
```

Figure 6 Source code for data collection from a weather station

3.3 Data exchange evaluation

Data consistency refers to the maintenance of data uniformity as data moves between systems over a network. It is a major characteristic of the data-exchange platform, because it can collect data from data sources and provide internal data to other systems. In order to test the data consistency in a data-exchange platform, a test to compare the number of data records provided by a data source and the number of data records saved in OFIS was conducted. Five data sources of three interfaces were considered. Three data sources represent the OpenAPI data sources discussed in Section 2.1. The fourth data source was a weather station installed at a farm described in Section 3.2.2. The last data source was the data-sharing service of OFIS.

Table 5 Result of data consistency test

Data source	Data type	Data consistency	Interface
KMA (Forecast weather)	XML (RSS)	Exact matching	HTTP (OpenAPI)
OKDAB (Crop price)	XML	Exact matching	HTTP (OpenAPI)
NCPMS (Disease information)	XML	Exact matching	HTTP (OpenAPI)
Weather station (Sensor network)	Binary	Exact matching	Modbus
Data-sharing service	JSON	Exact matching	HTTP (OpenAPI)

Table 5 shows the results of the data consistency test. The data-collection service could obtain data without any loss, and the data-sharing service could provide all expected data without error.

4 Conclusion

OFIS as a FMIS requires a general method to collect sensor observations and data from AISs for agricultural decision making. In addition, it must possess the ability to share internal data to obtain farm-specific services from other systems. The data-exchange platform can help to satisfy these requirements to communicate with other information systems. In order to develop the data-exchange platform, several data collection procedures and standards for transmission and representation were analyzed. For the data-collection service, four steps in the data collection procedure were identified. Data can be: retrieved from a data source, processed, mapped with metadata, and saved. The data-collection service was designed using separated classes. The implemented classes could be assembled to several combinations depending on a configuration file describing a sequence of data collection steps. For the data-sharing service, OFIS data was categorized into four groups: farm information, local weather, working history, and sensor observations. This service was designed and implemented as a RESTful web service. In order to evaluate the OFIS data-exchange platform, two use cases were shown and a data consistency test was conducted. The use cases revealed that the platform is flexible and extensible in collecting from various data sources such as web services and serial communication devices. Five data sources were selected for the data consistency test: three OpenAPI services, a weather station as a sensor system, and a data-sharing service of OFIS. As a result, two services of OFIS data-exchange platform operates without any error and this kind of data-exchange platform would be useful for a FMIS in terms of flexibility, simplicity, and extensibility.

References

Arroqui, M., C. Mateos, C. Machado, and A. Zunino. 2012. RESTful Web Services improve the efficiency of data transfer of a whole-farm simulator accessed by Android smartphones. *Computers and Electronics in Agriculture*, 87: 14-18.

Batte, M.T., 2005. Changing computer use in agriculture: evidence from Ohio. *Computers and Electronics in Agriculture*, 47: 1-13.

Blauth, D.A., Ducati, J.R., 2010. A Web-based system for vineyards management, relating inventory data, vectors and images. *Computers and Electronics in Agriculture* 71: 182-188.

Bray, T., J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau. 1998. Extensible markup language (XML). World Wide Web Consortium Recommendation

REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210.

Cantelon, M., T. Holowaychuk, and N. Rajlich. 2012. Node. js in Action. Manning.

Chaudhary, S., V. Sorathia, and Z. Laliwala. 2004. Architecture of sensor based agricultural information system for effective planning of farm activities, Services Computing, 2004.(SCC 2004). In *Proc. 2004 IEEE International Conference on. IEEE*, 93-100. Shangai, China. Sept. 15-18 2004

Crockford, D. 2006. The application/json media type for javascript object notation (json). Internet Engineering Task Force (IETF).

Fielding, R.T. 2000. Architectural styles and the design of network-based software architectures. University of California.

Han, W., Z. Yang, L. Di, and R. Mueller. 2012. CropScape: A Web service based application for exploring and disseminating US conterminous geospatial cropland data products for decision support. *Computers and Electronics in Agriculture*, 84: 111-123.

Heindl, U., A. Jacob, M. Mueller, P. Racz, B. Stiller, E. Volk, and M. Waldburger. 2010. AgroGrid–Grid technologies in agro food business, grid and cloud computing. Springer, 173-190.

Holster, H., S. Horakova, B. Ipema, B. Fusai, G. Giannerini, F. Teye, D. Martini, L. Shalloo, O. Schmid. 2011. State of the art data exchange in agriculture in the EU27 & Switzerland agriXchange, 33.

Kaivosoja, J., M. Jackenkroll, R. Linkolehto, M. Weis, and R. Gerhards. 2014. Automatic control of farming operations based on spatial web services. *Computers and Electronics in Agriculture*, 100: 110-115.

Kaloxylos, A., R. Eigenmann, F. Teye, Z. Politopoulou, , S. Wolfert, C. Shrank, M. Dillinger, I. Lampropoulou, E. Antoniou, and L. Pesonen. 2012. Farm management systems and the Future Internet era. *Computers and Electronics in Agriculture*, 89: 130-144.

Kim, J., C. Lee, T.-H. Kwon, G. Park, and J.-Y. Rhee. 2013. Development of an Agricultural Data Middleware to Integrate Multiple Sensor Networks for a Farm Environment Monitoring System. *Journal of Biosystems Engineering*, 38(1): 17-24.

Kubicek, P., J. Kozel, R. Stampach, and V. Lukas. 2013. Prototyping the visualization of geographic and sensor data for agriculture. *Computers and Electronics in Agriculture*, 97: 83-91.

Kwon, T.-H., J. Kim, G. Park, C.G. Lee, A. Ashitiani-Araghi, S.H. Baek, and J.Y. Rhee. 2014. Survey on Informatization Status of Farmers for Introducing Ubiquitous Agriculture Information System. *Journal of Biosystems Engineering*, 39(1): 57-67.

Laliwala, Z., V. Sorathia, and S. Chaudhary. 2006. Semantic and rule based event-driven services-oriented agricultural recommendation system. In *Proc. Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on IEEE*, 24-24. Lisboa, Portual. 4 - 7 July 2006.

Laurenson, M. 1990. ODE-orchard decision environment. In *Proc. II International Symposium on Computer Modelling in Fruit Research and Orchard Management.* Acta Hort. (ISHS) 276, 301-304. Logan, Utah, USA. 1 July 1990.

Lee, W.S., V. Alchanatis, C. Yang, M. Hirafuji, D. Moshou, and C. Li. 2010. Sensing technologies for precision specialty crop production. *Computers and Electronics in Agriculture*, 74: 2-33.

Luck, J.D., S.K. Pitla, J.P. Fulton, and S. Shearer. 2013. Evaluation of agricultural field machinery data as a potential and beneficial source for big data to improve agricultural production practices. In *Proc. ASABE International Meeting*. ASABE, Kansas City, Missouri. 21 - 24 July 2013.

Montoya, F.G., J. Gómez, A. Cama, A. Zapata-Sierra, F. Martínez, J.L. De La Cruz, and F. Manzano-Agugliaro. 2013. A monitoring system for intensive agriculture based on mesh networks and the android system. *Computers and Electronics in Agriculture*, 99: 14-20.

MOSPA, 2014. Public Data Portal. Ministry of Security and Public Administration.

Murakami, E., A.M. Saraiva, L.C.M. Ribeiro Junior, C.E. Cugnasca, A.R. Hirakawa, and P.L.P. Correa. 2007. An infrastructure for the development of distributed service-oriented information systems for precision agriculture. *Computers and Electronics in Agriculture*, 58: 37-48.

Na, A., and M. Priest. 2007. Sensor observation service. Implementation Standard OGC.

Nash, E., P. Korduan, and R. Bill. 2009a. Applications of open geospatial web services in precision agriculture: a review. *Precision Agriculture*, 10(6): 546-560.

Nash, E., R. Nikkilä, L. Pesonen, K. Oetzel, W. Mayer, I. Seilonen, J. Kaivosoja, R. Bill, S. Fountas, and C. Sørensen. 2009b. Deliverable 4.1.1. Machine Readable Encoding for Definitions of Agricultural Crop Production and Farm Management Standards. EU, FutureFarm.

Nikkilä, R., Seilonen, I., Koskinen, K., 2010. Software architecture for farm management information systems in precision agriculture. *Computers and Electronics in Agriculture* 70: 328-336.

Nurseitov, N., M. Paulson, R. Reynolds, and C. Izurieta. 2009. Comparison of JSON and XML Data Interchange Formats: A Case Study. *Caine*, 2009: 157-162.

OGC, 2012. Open Geospatial Consortium.

Rafoss, T., K. Sælid, A. Sletten, L.F. Gyland, and L. Engravslia. 2010. Open geospatial technology standards and their potential in plant pest risk management—GPS-enabled mobile phones utilising open geospatial technology standards Web Feature Service Transactions support the fighting of fire blight in Norway. *Computers and Electronics in Agriculture*, 74: 336-340.

Rolfe, J., S. Gregor, and D. Menzies. 2003. Reasons why farmers in Australia adopt the Internet. *Electronic Commerce Research and Applications*, 2(1): 27-41.

Sørensen, C.G., S. Fountas, E. Nash, L. Pesonen, D. Bochtis, S.M. Pedersen, B. Basso, and S.B. Blackmore. 2010. Conceptual model of a future farm management information system. *Computers and Electronics in Agriculture*, 72: 37-47.

Schmitz, M., D. Martini, M. Kunisch, and H.-J. Mösinger. 2009a. agroXML Enabling Standardized, Platform-Independent Internet Data Exchange in Farm Management Information Systems, In: Sicilia, M.-A., Lytras, M. (Eds.), Metadata and Semantics. Springer US, 463-468.

Schmitz, M., D. Martini, M. Kunisch, and H.J. Mösinger. 2009b. agroXML Enabling Standardized, Platform-Independent Internet Data Exchange in Farm Management Information Systems. Metadata and Semantics, 463-468.

Schuster, E.W., H.-G. Lee, R. Ehsani, S.J. Allen, and J. Steven Rogers. 2011. Machine-to-machine communication for agricultural systems: An XML-based auxiliary language to enhance semantic interoperability. *Computers and Electronics in Agriculture*, 78: 150-161.

Shim, K.S., H.S. Ko, J.R. Kim, and J.I. Choi. 2009. The method of Agriculture Technology Information Service in Web2.0. *Food Business & IT services*, 1(1): 87-106.

Steinberger, G., M. Rothmund, and H. Auernhammer. 2009. Mobile farm equipment as a data source in an agricultural service architecture. *Computers and Electronics in Agriculture*, 65: 238-246.

Stewart, T.R., R.W. Katz, and A.H. Murphy. 1984. Value of weather information: A descriptive study of the fruit-frost problem. *Bulletin of the American Meteorological Society*, 65(2): 126-137.

Tilkov, S.S. 2010. Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, 14(6): 80-83.

White, J.W., L.A. Hunt, K.J. Boote, J.W. Jones, J. Koo, S. Kim, C.H. Porter, P.W. Wilkens, and G. Hoogenboom. 2013. Integrated description of agricultural field experiments and production: The ICASA Version 2.0 data standards. *Computers and Electronics in Agriculture*, 96: 1-12.