Computers and Electronics in Agriculture 100 (2014) 168–179

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

A cloud-based Farm Management System: Architecture and implementation

Alexandros Kaloxylos [a,*], Aggelos Groumas [b], Vassilis Sarris [b], Lampros Katsikas [b], Panagis Magdalinos [b], Eleni Antoniou [c], Zoi Politopoulou [c], Sjaak Wolfert [d], Christopher Brewster [e], Robert Eigenmann [f], Carlos Maestre Terol [g]

[a] Department of Informatics and Telecommunications, University of Peloponnese, Tripolis, Greece
[b] Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece
[c] OPEKEPE, Domokou 5, Athens, Greece
[d] LEI and Information Technology Group, Part of Wageningen UR, Wageningen, The Netherlands
[e] Aston Business School, Aston University, Birmingham, UK
[f] Huawei Technologies Duesseldorf GmbH, Germany
[g] Sector of Manufacturing and Retail, ATOS Research & Innovation, Spain

ARTICLE INFO

Article history:
Received 26 May 2013
Received in revised form 16 November 2013
Accepted 27 November 2013

Keywords:
Farm Management System
Future Internet
Generic enablers
Services' marketplace

ABSTRACT

Recent technological advances have paved the way for developing and offering advanced services for the stakeholders in the agricultural sector. A paradigm shift is underway from proprietary and monolithic tools to Internet-based, cloud hosted, open systems that will enable more effective collaboration between stakeholders. This new paradigm includes the technological support of application developers to create specialized services that will seamlessly interoperate, thus creating a sophisticated and customisable working environment for the end users. We present the implementation of an open architecture that instantiates such an approach, based on a set of domain independent software tools called "generic enablers" that have been developed in the context of the FI-WARE project. The implementation is used to validate a number of innovative concepts for the agricultural sector such as the notion of a services' market place and the system's adaptation to network failures. During the design and implementation phase, the system has been evaluated by end users, offering us valuable feedback. The results of the evaluation process validate the acceptance of such a system and the need of farmers to have access to sophisticated services at affordable prices. A summary of this evaluation process is also presented in this paper.

1. Introduction

It is not long ago that farmers started using information systems to organize their financial data and keep track of their transactions with third parties (Batte, 2005). In developed countries, it is now commonplace for farmers to use sophisticated systems to monitor their crops. Data is collected from locally installed hardware (which may provide temperature, humidity, soil moisture, luminosity measurements, etc.) or from third parties such as meteorological services (Wang et al., 2006). A number of proprietary systems are used to process these data and assist farmers to manage and even control their farms in an efficient manner (Allen and Wolfert, 2011; Nikkilä et al., 2010; Wolfert et al., 2010). These systems are called Farm Management Information Systems (FMISs) – (Robbemond and Kruize, 2012). A FMIS is a system used for collecting and processing data to execute the operations of a farm. These operations include strategic, tactical and operational planning; implementation and documentation; assessment and optimization of the work performed in the fields or on the farms. To improve the execution of these functions, various management systems, databases, software architectures and decision models have been proposed to serve these purposes (Beck, 2001; Nikkilä et al., 2010; Sørensen et al., 2011; Fountas et al., 2006).

Existing and future systems in general, operate under a specific business model (Teye, 2011; Sørensen et al., 2010). Their main goal is to provide or collect information to/from farmers, process it and provide a number of intelligent services. These services are usually tightly integrated with the system. Existing systems are proprietary solutions that use closed specifications. This imposes a constraint on farmers since they do not have the freedom to enhance or tailor their systems according to their needs. They are, consequently, forced to use only the functionalities from the commercial products they have purchased. Furthermore, external

* Corresponding author. Address: Department of Informatics and Telecommunications, University of Peloponnese, 22100 Tripolis, Greece. Tel.: +30 2710372205; fax: +30 2710372242.
E-mail address: kaloxyl@uop.gr (A. Kaloxylos).

0168-1699/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.compag.2013.11.014

service providers cannot develop new services and make them interoperate with existing systems since these do not provide the necessary means such as Application Programming Interfaces – APIs (Kruize et al., in press). The creation of a marketplace of applications developed by independent service providers and which end users can choose to use by giving them access, in a secure way, to their data could alleviate the aforementioned problems.

In Kaloxylos et al. (2012) we presented the notion of a Farm Management System (FMS) as a framework that can accommodate modular services and enable their interoperation. The services can be either simple ones (e.g., a meteorological service) or even complex ones like an existing FMIS. In this way a marketplace of services and applications offered by different providers can be created. While this idea is similar to Google Play and Apple's App Store, the main difference is that the services are not executed independently, but rather they can interact by having access to the same data sets in the cloud and even exchange information through the FMS.

In this paper, we present a proof of concept implementation that was developed in the context of the SmartAgrifood Project.[1] Our implementation assisted us to identify the technical issues of a cloud-based system that could serve as a marketplace for services for the farmers. The implementation of this architecture as well some of the supported services has been based on a set of domain independent software tools called Generic Enablers (GEs) that have been developed in the context of the FI-WARE Project.[2] The purpose of these tools is to provide to software developers the means to develop in a fast and reliable manner a variety of cloud-based services for the Future Internet.

The FMS has been designed from a usage-driven perspective. This means that end-users' needs were identified and user requirements were formulated as central design goals. Recurrent design workshops and repeated end-user evaluations during the entire development process have been undertaken. The process of a usage-driven design and evaluation process were based on a seven step design approach (Nurkka et al., 2007; Brewster et al., 2012) by which research and design efforts were combined to deliver a gradually maturing design output. During these steps, we organized national panels with farmers and ICT experts, performed interviews with end users, created mock-up GUIs and videos, and asked users to fill-in electronic questionnaires. For our use case, we focused on developing a number of modular services related to greenhouses. The final system has been installed and used by a greenhouse in Crete, Greece over a period of nine months where the system was evaluated.

The rest of this paper is organized as follows. First, we briefly describe the FMS architecture and provide an operational example. In Section 3, we provide the system specification and discuss implementation details. We also discuss how the GEs have been used and provide some insight into their usefulness during the development phase. Section 4 presents the evaluation results from the end users and finally, Section 5 concludes the paper and presents our future plans.

2. FMS architecture

As described in Kaloxylos et al. (2012), the overall FMS architecture consists of two main entities i.e., the Cloud FMS and the Local FMS (Fig. 1). The Cloud FMS is equipped with a number of GEs that are used to support operations related to the management of a greenhouse. It contains a services' repository so that developers of services can upload their services for users to discover and use them. The Cloud FMS also contains a module called "FMS Controller" that consists of a number of sub-modules concerned with data collection, statistical analysis of data, coordination of activities, and the creation of notifications and commands to be executed by farming equipment. The "Management Functions" module provides information about the underlying network infrastructure, so as to fine-tune the operation of the overall system according to the current networking conditions. This module also contains functionality for recording any activity between services and users so as to apply specific charging schemes. All services that are delivered and, cooperating through the Cloud FMS, are able to access the same data sets stored in the FMS controller. The services access these data in a secure way. Appropriate permissions are granted to a service when a user registers with it. This communication is realized through a Service Oriented Architecture – SOA layer that sets the communication links between the core modules of the FMS and any specific service.

The Local FMS is located with the end-users (for example inside a greenhouse) and is mainly used to aggregate sensors' and possibly machinery data and forward them to the Cloud FMS. Also it contains some of the FMS Controller functionality that is used when there is no Internet connectivity.

3. Operational example

Fig. 2 illustrates an operational example using a representation produced by the Archimate (2013) tool. On the business layer level, a farmer needs to monitor the crop and machinery, and in case there are some out of the ordinary events, there is a need to come up with a correction plan and implement appropriate actions (see Figs. 3 and 4).

On the service level, data are collected from a plethora of devices and machinery and are transferred through the Local FMS to the monitoring service of the Cloud FMS. These data are then analyzed and when any abnormal situation is detected, then appropriate alerts are produced. At the same time, the coordination module communicates with the appropriate services (e.g., an expert system such as an e-agriculturist, a task scheduling service, or a meteorological service) and communicates their recommendations and reports either to the farmer or directly to systems installed on the farm so as to be automatically executed if the farmer has configured the system to do so. All these steps and actions can be recorded by the FMS database. The type and the amount of information to be used by these external services is configurable by the farmer by giving appropriate permissions to the services. The access from external services to a farmer's information can be configured in way similar to the one used by Android and Apple applications (e.g., grant permission to collect automatically the location information, acceptance to access specific data like sensor values, etc.).

4. System specification: the case for a greenhouse

In this section we present in detail an instantiation of the FMS architecture that has been developed and used for the management of a greenhouse. The Greenhouse Management prototype is a Future Internet (FI) compliant framework that takes into account real data from sensors and provides them to a Farm Management System (FMS) in order to take smart decisions regarding actions that need to be taken. External services have access to the real data collected and produce recommendations and reports related to the smart management of the greenhouse. Notifications and alerts about the current situation and actions are forwarded to the farmer. A farmer is thus able to have total surveillance and management of his farm using services developed by different service providers. Our prototype has been implemented in order to integrate a number of innovative concepts. In particular:

[1] http://www.smartagrifood.eu/.
[2] http://catalogue.fi-ware.eu/enablers.

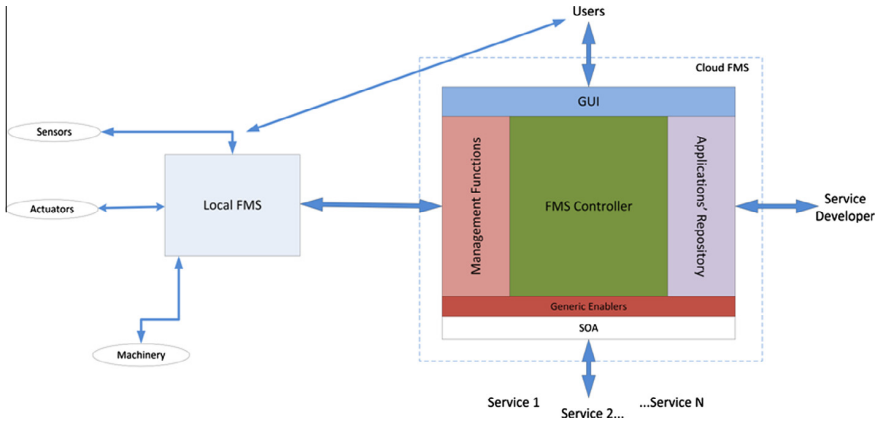

Fig. 1. Simplified FMS architecture.

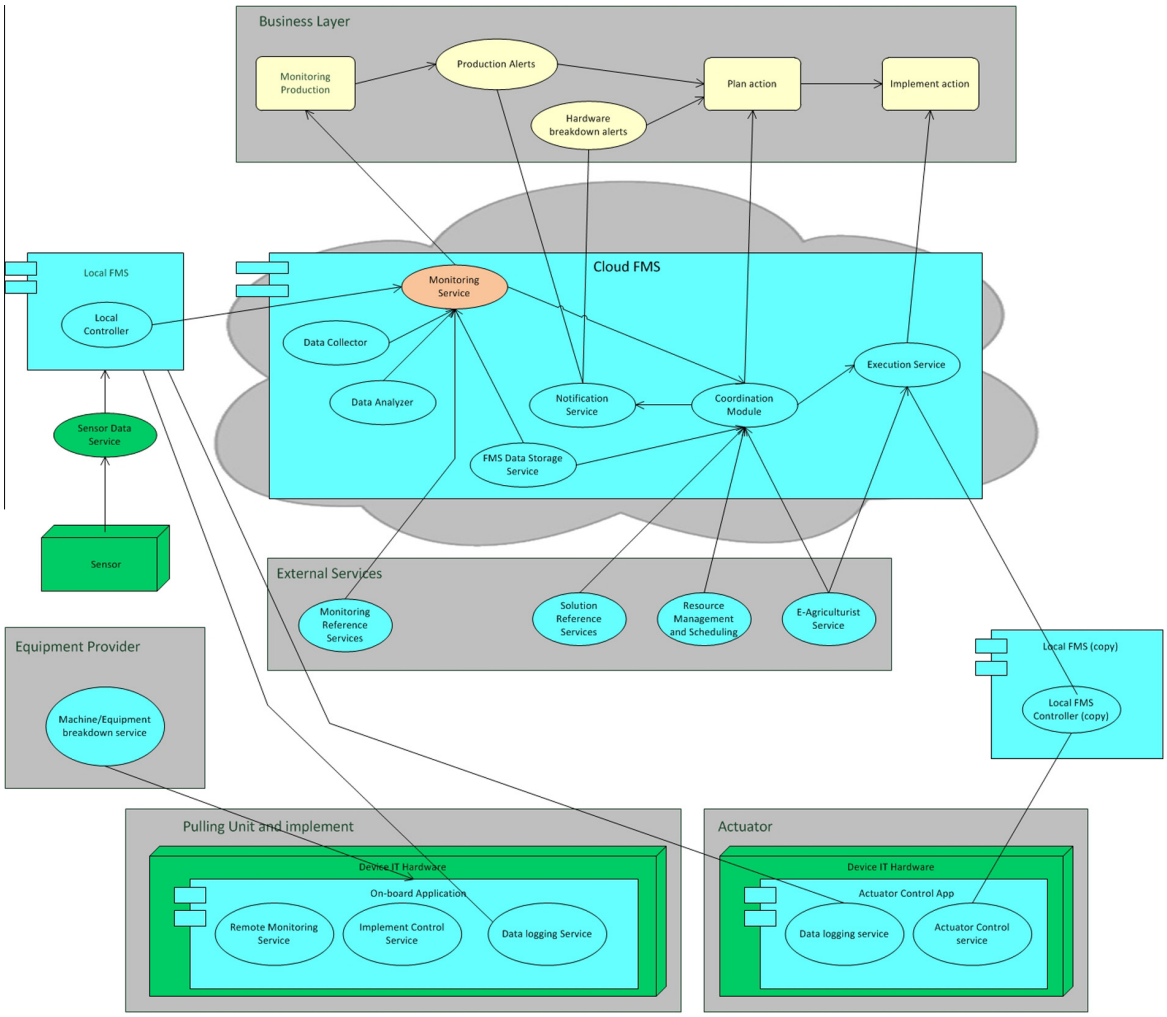

Fig. 2. An archimate model representation of an operation example.

- Lower investment cost since the intelligence of the system is located in the cloud and only sensors, actuators and low end computing components are needed in the greenhouse.
- Automatic communication of the system with any equipment or external service using a Service Oriented Architecture (SOA) approach.
- Storage of raw data and guaranteeing user-independence from any service provider.
- Service adaptation according to user preferences and end-device capabilities.
- One-stop market place facilitating the end-user in his everyday needs.

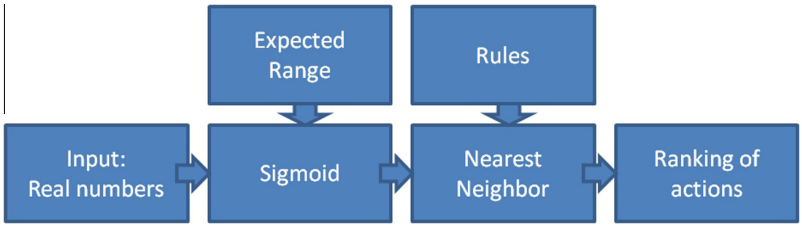

Fig. 3. Fuzzy logic approach.

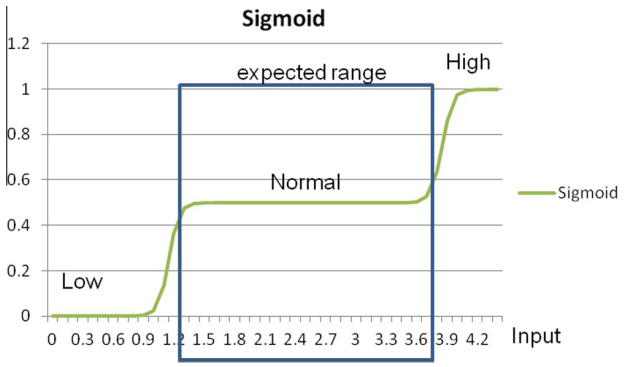

Fig. 4. Ranges.

- Integration of domain specific services (e.g., advisory services).
- Learning schemes focusing on improving operations through exploitation of accumulated data.
- Plug and Play with IoT solutions.
- Context aware networking that allows the system to adapt to different networking conditions (e.g., link failures).

The Greenhouse Pilot has been deployed in an actual greenhouse in Crete (Local FMS) and within the University of Athens premises (Cloud FMS). The greenhouse is approximately 10.000 m^2, having an almost rectangular shape. Inside the greenhouse we have deployed wireless sensor nodes. All data and information are presented through a simple web frontend, the Greenhouse Management web application. The user interacts with a web application without getting involved with the underlying complexity of the system. The deployed wireless nodes send their measurements periodically to the gateway, which is deployed on a Raspberry Pi[3] located at the farmer's office. The information is transmitted to the university premises. The processed information and the knowledge extracted are subsequently presented to the farmer via web-based portal.

Before describing the technical details of the implementation, we briefly present the GEs developed by FI-WARE and the capabilities that they offer. FI-WARE defines six large areas for which generic enablers are provided. Some of these are so fundamental that they will be applicable for a diverse set of tasks. Other enablers are adapted and fine-tuned for the agricultural case and thus, they provide advanced domain specific capabilities. These six areas are namely: Cloud hosting, Data and Context Management, Applications/Services Ecosystem and Delivery, Internet of Things, Interface to Network and Devices and Security. A detailed description for the GEs developed by FI-WARE can be found in FI-WARE MediaWiki (2013).

From these six categories we used a number GEs that were integrated in the pilot. A short description found in (FI-WARE GE Catalogue, 2013) about their functionality appears in the list below:

- **Cloud Edge**: Sort of a "Super Gateway", located at the edge between the WAN/Cloud and the LAN and able to locally execute applications.
- **Repository SAP-RI**: A service and application description repository.
- **Publish Subscribe context broker (SAMSON broker)**: Implementation of the Publish/Subscribe Context Broker GE, providing interfaces with which clients can do several operations like (a) register applications producing contextual data, such as a temperature sensor within a room, (b) update contextual information, for example send updates of temperature, (c) receive notifications when changes on contextual information take place or with a given frequency (e.g. get the temperature each minute).
- **Mediator TI**: A middleware application responsible for providing interoperability among different communication protocols and among different data models.
- **Service Composition and Application Mashup**: These two GEs offer a composition editor and execution engine that allows end users to create and run composite web applications.
- **IaaS Data Center Resource Management**: This GE offers the facilities to provide virtual machines, as well as associated computing, storage and network resources.

The abovementioned GEs have been used in three evaluation scenarios. In the first one, a service provider registers a service. All details required are provided in order to properly index and store his service (keywords, charging profile, etc.). The service description is formulated and transmitted in linked-USDL format. Minutes later, the farmer checks the marketplace in his end-user application, notices the new service and registers. The usage of the service is constantly monitored; the application provider can validate the logs and based on the actions performed charge the user (GEs validated: repository, mediator).

In the second scenario the user has deployed the Local FMS in his greenhouse. The devices constantly transmit information to the cloud. Information, before being stored, is assessed by the FMS Controller (statistical analysis sub-module). The latter, upon identification of a problematic situation, triggers a notification action that is forwarded to the farmer through any appropriate communication channel (GEs validated: Data Center, Publish Subscribe, Cloud Edge).

In the third scenario, the Service Composition and Application Mashing GE has been integrated and validated in the GUI frontend of the Greenhouse Pilot. It can be used to provide graphic mashups that exploit the capabilities of the widgets provided by the Mashup Factory (e.g. email). In the background, these widgets/services compose a new one in a service composition manner.

4.1. Implementation details

As mentioned in the previous sections, the deployed hardware is divided in two entities, the Cloud FMS (University premises) and the Local FMS part (greenhouse).

[3] http://www.raspberrypi.org/.

In the greenhouse, 5 WaspMote (Libelium, 2013) boards (ATmega1281 microcontroller, 128 KB Flash, 8 KB SRAM and 2 GB SD Card, Battery capacity 6600 mAh) each equipped with a 2 W Solar Panel[4] (80 × 100 mm) are deployed. Information is exchanged using a communication gateway, connected over a USB cable to the device implementing the concept of the local FMS and the Cloud Proxy GE. The gateway is an XBee-ZB (Xbee, 2013) module, implementing the Zigbee-Pro protocol. Four nodes are equipped with identical agricultural boards (Libelium sensors, 2013) and support the monitoring of temperature, relative humidity and soil moisture. The fifth one has a Libelium gases and monitors PH and CO^2 levels. Finally, in the farmer's office, a Raspberry Pi is deployed, for hosting the Cloud Proxy, with 256 MB RAM, an ARM1176JZF-S core CPU @ 700 MHz.

In the University premises, the server hosting the Liferay portal, has an Intel(R) Core(TM) i5–2320 CPU @ 3.00 GHz CPU, 4 GB RAM, and is running Ubuntu server 10.04, Linux kernel 2.6.32–38-generic with MySQL 5.1.41 and Tomcat 7.0.23. The server hosting the FMS controller, has an Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66 GHz CPU, with 4 GB RAM, running server 10.04 Linux kernel 2.6.32–38-generic with MySQL 5.1.41.

4.2. Software design

In order to meet the technological requirements based on the users' comments, we opted for several web-oriented technologies. The following programming/scripting languages are used in the current implementation: Java (J2SE, J2EE and JSP), HTML, CSS, JavaScript, and C/C++ for programming the sensors. Regarding the Graphical User Interface (GUI), the front-end is designed following the concept of a 'web-based portal'. The main reason for such an implementation option was the extensibility of the application to numerous types of devices. Additionally, it is also a more user-friendly way to support the marketplace GEs (e.g. each user could be registered with different services dynamically according to their needs). The portlets used in the context of the portal provide the user the required personalized experience. Each portlet is dynamic, supporting AJAX[5] and Javascript[6]/jQuery[7] combined with CSS3 and ALLOY UI (a UI meta-framework that incorporates all three aforementioned design languages[8]). As far as the localization of the system, Liferay provides a build-in way to localize the user interface. Liferay uses language keys for each language the administrator of the portal wishes to support. The user just selects his preferred language and the portal dynamically chooses the appropriate keys for each language. Those language keys are stored into property files.

Each portlet in the portal GUI is responsible for a sub-service of the whole system. The main point of this architecture is to have each service (portlet) totally independent from the other services. So, each portlet handles internally the information that should be represented to the user having access to the appropriate interfaces. Each portlet also uses a wide variety of design and scripting languages to represent it. HTML 5 coexist in the same JSP files for the representation of the web pages and jQuery/Javascript and ALLOY scripts are used to handle the events triggered from the user interaction with the screen. The modules of the Greenhouse Pilot at the backend are independently implemented as REST-ful web services (using Jersey[9]). External services are deployed in an OSGi[10] runtime environment (Apache Karaf[11]) and exchange data with the backend system with the assistance of the ESB architecture (Mule[12]). Authorization and authentication is accomplished with the use of Apache Shiro.[13]

As far as data exchanged between different sub-modules or subsystems is concerned, a thorough research regarding standardized XML schemata related to agricultural applications was made. The standardized XML schemata that we have used are the SensorML,[14] the agroXML[15] and the Observations and Measurements standard.[16] Data management in the Greenhouse Pilot is done using traditional Relation Database Management Systems (RDBMS). From a methodological point of view, we mapped the SensorML, agroXML and Observations and Measurements standards (all in XSD) to Entity-Relation (ER) schemata. Data management (access, updates and deletions) is therefore accomplished through SQL. Specifically, we have used MySQL ver. 14.14 distribution 5.1.61 for debian-linux-gnu for the Cloud FMS.

The reasons we opted for a traditional RDBMS were essentially (i) the ease of use, (ii) the maturity of the available solutions and (iii) re-usability (the Data Collector DB schema is essentially an extension of the Local Data Collector DB). Additionally, a lightweight RDBMS seems more suitable for a low-end computing component like the one envisaged for the Local FMS. This is why, for the Local FMS we have used SQLite (version 3.7.16) that is a serverless, self-contained database engine that is lightweight and widely used today by many software vendors.

4.3. Description of interfaces

In this subsection we describe at an abstract level, an indicative list of messages exchanged through the interfaces illustrated in Fig. 1. The messages that are exchanged in all interfaces are following the request–response pattern. For simplicity, we will indicate the request message and the corresponding response (see Table 1).

The interface between the Local FMS and the Cloud FMS mainly functions to receive sensor data created by the sensors at a farm. Problems regarding the operation of sensors can be detected and the Cloud FMS can be informed about them. This interface allows the monitoring of the Internet connection between Local FMS and Cloud FMS and identifying when communication failures occur. Furthermore, the Cloud FMS can request a remote update of the firmware of the sensor motes (see Table 2).

The User to Cloud FMS interface offers the user various capabilities. At a first level a farmer is able to monitor the conditions that appear on his/her farm in detail. In addition the farmer can be informed about results produced by the processing of these conditions, either by services or the FMS itself, in the form of notifications or alerts. Moreover, the user can have access to all kind of services, subscribe to them so that services can have access to user's data (see Table 3).

A service provider can register its service in the Cloud FMS by providing the details of the service and the relevant interfaces of the service so that its functionality can be integrated with the Cloud FMS functionalities. The service details can be altered and modified. A service provider can also see a list of all the services he/she uploaded and have an overview of them (see Table 4).

Services can communicate with the Cloud FMS and have access to the farm's data or the users' data. This can be done either by requesting the relevant information or by subscribing to specific channels where this information is published. In addition, services can upload data to the Cloud FMS such as notifications regarding

[4] www.seedstudio.com.
[5] Asynchronous Javascript and XML, http://www.w3schools.com/ajax/default.asp.
[6] Javascript standard: http://www.w3.org/standards/webdesign/script.html.
[7] jQuery: http://jquery.com/.
[8] ALLOU UI: http://www.liferay.com/community/liferay-projects/alloy-ui/overview.
[9] JSR 311 implementation/Jersey Framework: http://jersey.java.net/.
[10] OSGi Alliance: http://www.osgi.org/Main/HomePage.
[11] Apache Karaf: http://projects.apache.org/projects/karaf.html.
[12] Mule ESB: http://www.mulesoft.org/.
[13] Apache Shiro: http://projects.apache.org/projects/shiro.html.
[14] Sensor Model Language, http://www.opengeospatial.org/standards/sensorml.
[15] AgroXML, http://www.agroxml.de/.
[16] Observations and Measurements, http://www.opengeospatial.org/standards/om.

Table 1
Local FMS–Cloud FMS.

From	To	Message	Parameters	Response
Local FMS	Cloud FMS	Send_sensor_data	Sensorvalue list	Success or failure
Local FMS	Cloud FMS	Fault_of_sensor	Mote id, value type, value	Success or failure
Local FMS	Cloud FMS	Keep_connection_alive	Local FMS id, timestamp	Acknowledge
Local FMS	Cloud FMS	Internet_connection_back	Local FMS id, timestamp	Acknowledge
Cloud FMS	Local FMS	Firmware_update	Mote id	Firmware data, status

Table 2
User – Cloud FMS interface.

From	To	Message	Parameters	Response
User	Cloud FMS	Login	Username, password	Authentication token
User	Cloud FMS	Logout	Username, AuthToken	Success or failure
User	Cloud FMS	Show_farm_plan	User id	Farm sensors values
User	Cloud FMS	Show_notifications	User id	Notifications
User	Cloud FMS	Show_alerts	User id	Alerts
User	Cloud FMS	Show_my_services	User id	Services information
User	Cloud FMS	Update_my_location	User id, longitude, latitude	Success or failure
User	Cloud FMS	Subscribe_to_service	User id, service id.	Success or failure
User	Cloud FMS	Rate_service	Service id, user id, rating	Success or failure

the user so that the user can have access to it independently (see Table 5).

This interface allows the Local FMS to obtain the sensor values that are transmitted by the nodes to the gateway, by monitoring the serial port that the gateway is attached to (see Table 6).

Finally, a user can have direct access to the local FMS directly. This is rather important when the Internet link between Local and Cloud FMS is broken.

4.4. Services

In this subsection, we describe the implementation of the services we have developed as examples to be used in our pilot demonstration that are using the FMS core functionalities. Most services function in two layers; the front-end layer, and the back-end layer which incorporates all the business logic that produce the results. The front-end layer is a portlet that depicts the notifications produced by the backend. The backend layer uses the interfaces provided by the SOA layer which enable connectivity to the FMS controller. Specifically, in the backend layer, a service can fetch data from the FMS that could be, for example, sensor data or products data, and according to its functionality inform the farmer. Examples of such services, already deployed on the pilot, are the following:

4.4.1. E-agriculturist service

This service processes sensor data from the farm (in the back-end) and produces notifications (in the front-end) that inform the farmer of actions he should take. Using an expert system, and taking into account the current crops being cultivated, the service decides whether sensor data indicate appropriate conditions for the farm's crop and whether the farmer should implement any actions. This information is provided through the RESTful interfaces to the front end. The system provides the user sufficient functionality to delete, show and read the notification messages. Those notifications can also be sent with an E-mail. The implementation follows a

Table 3
Service provider – Cloud FMS.

From	To	Message	Parameters	Response
Service provider	Cloud FMS	Login	Username, password	Authentication token
Service provider	Cloud FMS	Logout	Username, AuthToken	Success or failure
Service provider	Cloud FMS	Register_service	Image, service name, service type, webpage, company name, city, address, telephone, email, country price explanation, price value, price currency, interaction interface, terms of use, user id	Success of failure
Service provider	Cloud FMS	Update_service_details	Image, service name, service type, webpage, company name, city, address, telephone, email, country price explanation, price value, price currency, interaction interface, terms of use, user id	Success or failure
Service provider	Cloud FMS	Delete_service	Service id, user id	Success or failure
Service provider	Cloud FMS	Show_my_services	User id	Services details list

Table 4
Services – Cloud FMS interface.

From	To	Message	Parameters	Response
Services	Cloud FMS	Request_farm_data	Service id, sensor id, mote id.	Sensor values list
Services	Cloud FMS	Subscribe_for_farm_data	Service id, sensor id, mote id, duration	Sensor values list
Services	Cloud FMS	Request_user_data	Service id, user id, type of data	Data, status
Services	Cloud FMS	Subscribe_for_user_data	Service id, user id, type of data	Data, status
Services	Cloud FMS	Upload_service_data	Service id, data, type of data	Success or failure

Table 5
Local FMS – sensor interface.

From	To	Message	Parameters	Response
Local FMS	Sensor gateway	Read_serial_data	Serial_Port	Sensor values

Table 6
Local FMS – user interface.

From	To	Message	Parameters	Response
User	Local FMS	Show_farm_plan	User id	Farm Sensors values
User	Local FMS	Show_notifications	User id	Notifications
User	Local FMS	Show_alerts	User id	Alerts

combination of the Fuzzy Rules methodology and Nearest-Neighbor approach (Hart et al., 2000; Ross, 2010) (Fig. 3).

For simplification it was decided to use a sigmoid mapping of the real-valued input data:

0 := low
0.5 := normal
1 := high
<empty> := do not care

The sigmoid mapping allows any value in-between the interval (0...1), thus a value of e.g. 0.6 indicates a state between "normal" and "high", with trend to be "normal". Thus, a ranking of best-matching rules can be achieved.

For each input value, an "expected range" is specified with a lower threshold and a higher threshold, e.g. for the temperature the normal range could be [22 °C...29 °C].

A set of rules has been created, mapping the sensor data to a set of recommended actions identified by a unique number. This service was a simple implementation that we developed to test in practice the concepts of the cloud and local FMS. The specific service though rather simple could be used for example, in the local FMS to execute actions whenever an Internet link failure occurs. In such a situation there is no connectivity to the Cloud FMS and thus no connectivity to more sophisticated expert systems.

4.4.2. Disease actions and alerts service

The current service processes the data from sensors on the farm in the backend layer. It takes into account various diseases and it applies a Naive Bayes classifier so as to produce the best possible result concerning a disease alert and an associated action for the current crop. This produces a notification for the result in the front-end. The system uses input that is similar to the input of the E-agriculturist. However, in contrast to the latter, this implementation is able to learn and update its knowledge base thus adapting to the micro-climate of the environment or to the requirements of a farmer. The design and implementation is based on a Naive Bayes classifier. In spite of their apparently over-simplified assumptions, Naive Bayes classifiers have worked quite well in many real-world situations, such as document classification and spam filtering (Metsi et al., 2006). They require a small amount of training data to estimate the necessary parameters. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

The implementation of the system comprises 3 phases, (i) knowledge base preparation, (ii) training of the classifier, (iii) classifier update. Throughout the phases we use the Weka[17] data mining and machine learning software. At first, we retrieve the categorization of the various parameters as well as the scenarios, associated actions and alerts like for the previous service.

Based on the scenarios, we have artificially generated a large number of tuples which follow the scenarios. Simply stated, each rule/scenario is replicated N times. Afterwards, we broke the input into two training sets, the Actions set and the Alerts set. Finally, using these sets, we train two classifiers which are subsequently used for classifying all incoming tuples. Applying this method the system is able to learn and evolve through time. The user can provide his input; the latter is accommodated in the knowledge base and all subsequent decisions are directly influenced by the new observation.

4.4.3. Weather service

The weather service takes into account the location of the farmer, and searches for relevant weather information at his area for the current and next days. A user has the ability to retrieve weather information also for any other place simply by typing the location name into the input box provided by the portlet. The weather service – by default – provides weather information for the current day and also a forecast for the next two days. Temperature, wind information, humidity all form a part of the information represented together with an appropriate image according to the weather. When the user uses the Weather Service the request is processed by the SOA interface. We have used the online free API[18] of the weather2 website with the user's query about the selected place and it returns the weather results of the selected place in XML format. These results are passed back through the GUI to the user.

4.4.4. E-prices service

A simple service enabling the farmer to consult the market in real time in order to decide the best price to sell his goods. The front end is a simple GUI that enables the user to enter the query. In the backend, the system queries stock-markets and web-hubs in order to retrieve the required information. In our implementation, as an example, the system sends the user's query to the Google Shopping web service[19] to retrieve the required information. This REST api allows querying for a product using free text search. The default response is returning the most relevant products descriptions

[17] http://www.cs.waikato.ac.nz/ml/weka/.
[18] www.myweather2.com.
[19] http://www.google.com/shopping.

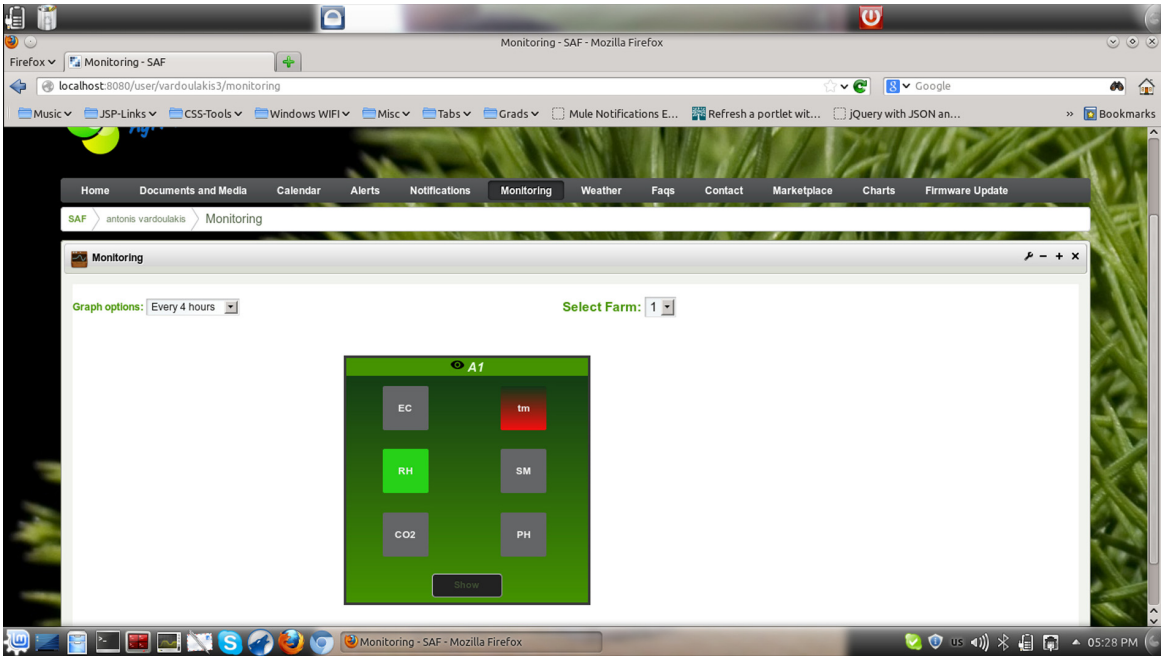

Fig. 5. Identification of a problem in one of the sensors – user interface.

according to the search query in JSON format and is shown to the user though the GUI.

4.5. GUI

The end user application is a simple web frontend, the Greenhouse Management web application. Thus, the user faces a user-friendly web application and can easily interact with the system without getting involved with the underlying complexity. A video presenting the GUI and some usage scenarios can be found at http://www.youtube.com/watch?v=dDq4RQYNiNs.

4.6. Usage scenarios

In this section, we provide a set of scenarios that exemplify the exploitation of the designed prototype. We identified eight distinct use cases which exploit the full set of classes, highlight the merits of the prototype and involve all GEs integrated in the pilot. It should be noted that the pilot itself supports a great variety of use cases, however due to space limitations we present only a small subset.

The system enables the user to visually assess the state of the farm's sensors by checking the user interface. In case an error has occurred, it is highlighted via a change in the color code (gre[20]en = everything is ok, gray = not used, red = problem). Fig. 5 depicts this situation.

The identification of such problems requires the exchange of a set of messages presented in Fig. 6. The local module, that is used for reading information from the serial port of a sensor forwards the extracted values to the Local Configuration and Communication component located in the Local FMS asking for validation. The latter is accomplished by the Error Detector module (part of the Local FMS) that signifies the existence of a faulty value in the data. The states together with the values are forwarded to the FMS Controller in the Cloud FMS via its Configuration and Communication component. The following procedure is similar to the a typical sensor values monitoring procedure with the distinct addition of the final alert message that is issued by the Notifier (part of the Cloud FMS) and sent to the user.

In case where a firmware update of the sensor can solve the problem a user can download and install new firmware in the faulty sensor node. For this to happen, a request is triggered by the user, using the web portal, and is captured by the FMS Controller (Cloud FMS). The latter provides the necessary file to the Local Configuration and Communication module (located in the Local FMS) that finally installs the firmware on the sensors.

In the following scenario we present how the system reacts to a network connection failure. Note that in the following figure, the Local Configuration and Communication module is part of the Local FMS while all other modules are part of the Cloud FMS. In this scenario, when a network connection failure occurs, the local FMS is able to identify the problem and trigger remedy actions. More specifically, a "local mode operation" is triggered, where all decision and actions are taken locally (i.e., by the Local FMS). Although, the Local FMS has less functionality than its cloud counterpart, it can still support simple operations (like collecting sensor values, use a simple advisory system). The network connection problem identification procedure is bi-directional in the sense that both the Local FMS as well as the FMS Controller of the Cloud FMS, both monitor the link quality. Therefore, when a network problem occurs, the Cloud FMS is aware and network status statistics are stored in the database. These statistics can be exploited by a third party, like a network provider in order to identify the root cause of the problem.

When the connection is up again, both edges are set to cloud mode and all values collected during failure time are transmitted to the FMS Controller. The entire process is depicted in Fig. 7.

When the user exploits the functionalities provided by an external service, their actions are logged with the use of the Mediator GE. The latter acts as a medium between the user and the service and logs all actions (e.g., http requests) which are in turn exploited by the FMS management functions. These logged actions are used

[20] For interpretation of color in Fig. 5, the reader is referred to the web version of this article.

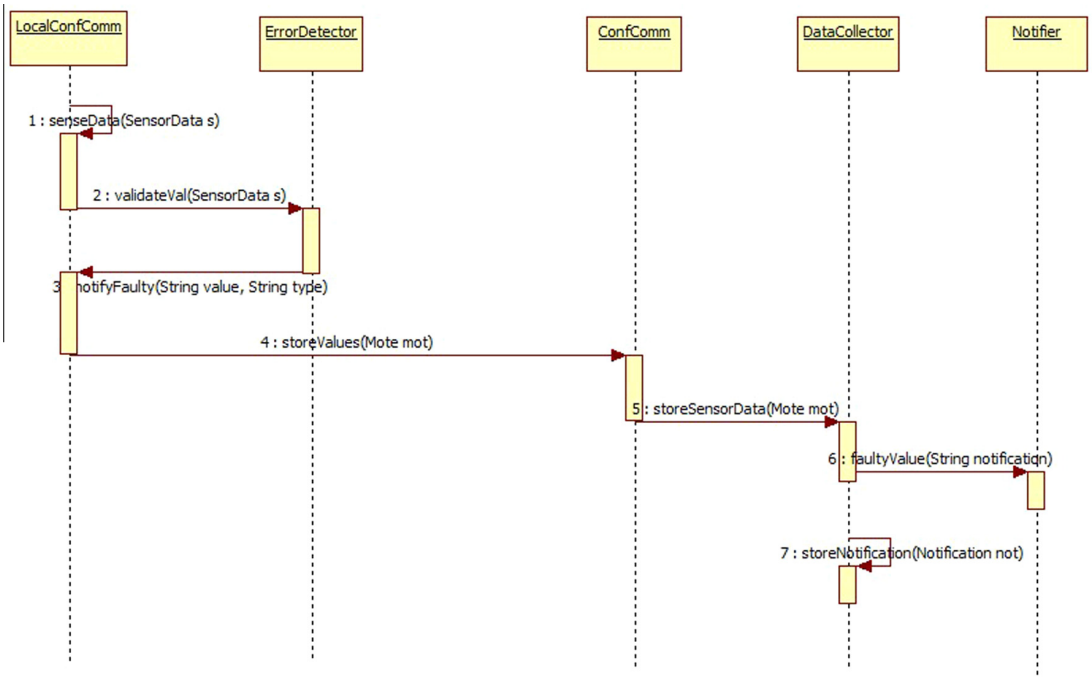

Fig. 6. Identification of a problem in one of the sensors.

for charging and billing the user according the tariff model provided by the service provider. This procedure is depicted in Fig. 8 for a scenario of a weather service usage.

5. Users evaluation of the FMS system

The FMS has been designed based on the user-centric design model referred to above so that end users were involved in all stages of the design and implementation process: concept specification, design of system functionality and software development. The evaluation process was continuous and adjustments were based on feedback collected from the following user groups: farmers, agriculturists, agronomists, ICT experts. Users involved in each group were selected based on several criteria such as the type of their activity (70% farmers, 10% agriculturists, 20% other specialists), the location, their age (13% less than 30, 57% is aged between 30 and 40, 20% between 40 and 50, 10% over 50), the infrastructure used on their farm, their familiarity with new technologies (40% have no or small experience, while 60% have a good knowledge of new technologies) or their intention to use them and their willingness to explore new business opportunities. In total 100 users were involved operating all over Greece (Attica, Crete, Peloponnese, North Greece, and Central Greece) with different levels of expertise in FMS systems and new technologies.

The method used for system evaluation was adapted to meet the following objectives: (i) introduce users to the use cases used to describe concepts identified in order to derive the main functional and non-functional requirements of the Smart Farming Ecosystem, (ii) identify real user needs and propose new concepts, (iii) obtain user feedback and reaction to the UI of the FMS and (iv) incorporate any changes that facilitate users' access and use of such a system in order to improve their daily work.

The user involvement employed a number of methods that included: interviews, electronic questionnaires and workshops. Semi-structured interviews (Wood, 1997) were used in the first phase of evaluation in order to assess the concepts of the FMS. They were used to gather the subjective perspectives from a different set of users. The second method was the online questionnaires (Nielsen and Molich, 1990) including both closed-ended and open-ended questions. The questionnaires were answered by individual users and focus groups (Mazza and Berre, 2007) set up in workshops. Four national discussion panels and a workshop were organized to evaluate the FMS functionalities and the graphical interface. The discussion panels where conducted during the evaluation process and included a team of 20 users involving representatives from ICT sector, farmers, logistics sector and service providers in the agricultural sector.

The evaluation took place in three phases. The objective of the first phase was to assess the use cases that described different usage scenarios. The users were introduced to the innovative ideas of Future Internet and Internet of Things and were provided with suggestions concerning how to use existing infrastructure more effectively. A group of 19 persons consisting of farmers, civil servants and representatives from farmer unions were involved. The participants' ideas and discussions led to alterations, modifications of these use cases and the suggestions of new ones such as the presentation of the products' prices and the traceability of the product across the market chain, a marketplace for purchasing the appropriate fertilizers after processing data from agro-soil environmental expert system.

Once this process was completed, a mock-up was implemented to present the functionalities of the FMS. At this stage 24 people (agriculturists, ICT experts, agronomists and farmers) were involved, evaluated the GUI and proposed new functionalities. Some of the questions asked referred to the technical solutions adopted, other applicable solutions that could be envisaged, and the applicability of the services developed. The observations of users' interactions resulted in data concerning errors, performance time and insights into the ease or difficulty of the tasks, the look and feel of the system and the user experience. Most of the proposed modifications were adopted during the implementation of the FMS.

The third phase concerned the evaluation of the functionalities developed after the refinements had been taken into account. The

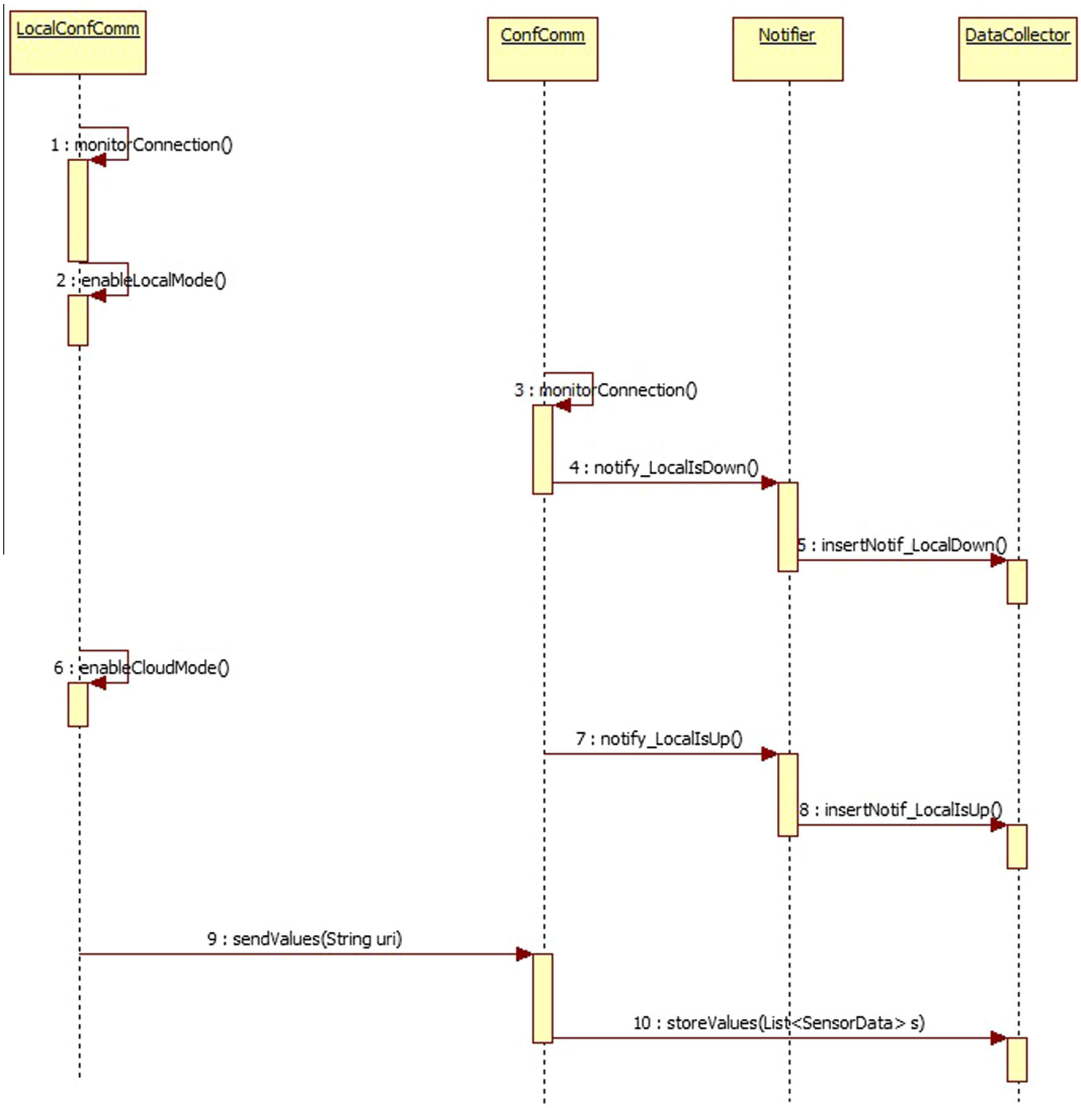

Fig. 7. Network connection failure and associated actions after restoring.

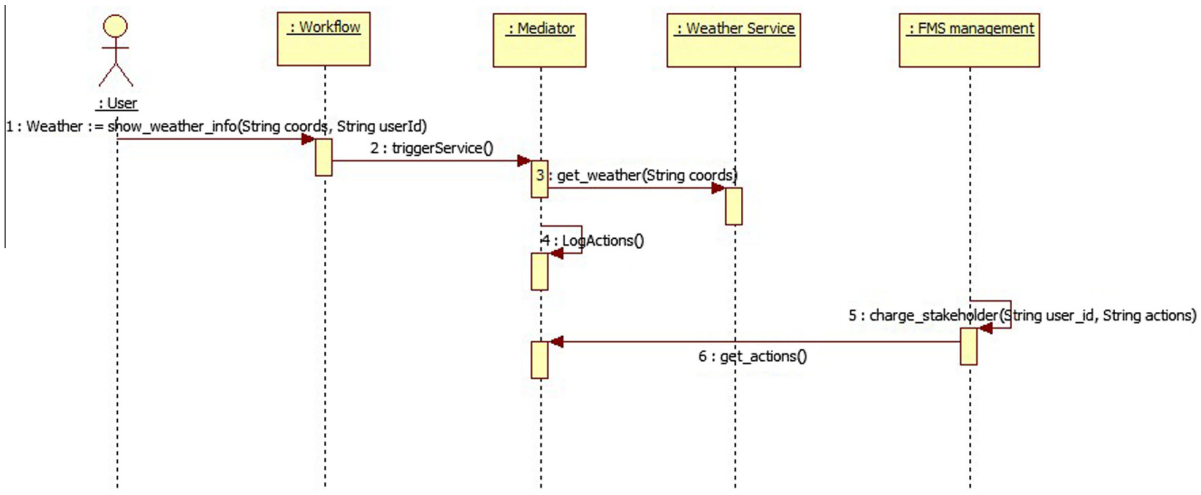

Fig. 8. Monitoring of service consumption via the Mediator GE.

majority of the participants (65%) stated that the use of the FMS will facilitate information sharing between the stakeholders and decrease the complexity in their daily processes. On the other hand, the low educational level of stakeholders is an impediment to the adoption of corresponding systems. Additional mechanisms, furthermore, need to be implemented to deal with confidentiality, integrity and availability of data.

The comments received from the evaluation process indicated a strong support for the concepts that were introduced and the offered services. 80% of the respondents believe that the system is useful and they can use it to complete some of their daily tasks. About 60% of the farmers are already using some kind of individual smart systems to support basic activities such as the temperature and humidity measurement or the irrigation of their fields. Most of the farmers lack access to an integrated expert system for decision support. This is due to (a) the lack of knowledge of this kind of systems (40% of the farmers have a low education level and about 60% have no or little experience with the Internet) and (b) the lack of trust in experts. The vast majority of farmers (88%) believe that such a system could reduce the cost of their work since it may contain adequate information for the farmer and is easy to use (90%).

Some of the functionalities that were found very interesting were the day-to-day calendar of the cultivation plan and the possibility to have an overall control of the farm through the Internet. In addition, farmers have the ability to monitor the sensed data and make decisions based on the solutions proposed by the system. Based on the users' comments new functionalities were added e.g. the presentation of the products' prices and the inventory of available products and supplies. A number of concerns were expressed for the following issues: cost of investment for sensors, cameras and other equipment and ownership of the data handled. 60% of farmers can invest about 1000–1500€ to install sensors or other expert systems while some of them are willing to invest even larger amounts of money if they are subsidized by the state.

The importance of data ownership and control was apparent. The FMS handles and processes various types of data. These data may originate from data collected by sensors installed in the farmer's farm, external services e.g. news/weather, data provided by experts that are stored in the FMS database or "new" data produced as a result of data processing. All these data have different ownership though many "actors" have access to other people's data that they can also process. Issues of copyright and ownership need to be taken into consideration especially for services generated from composition. A typical example of composition is the one that combines data collected from the sensors, an e-agricultural service and a meteorological service.

Regarding the ICT experts, all agreed that the most important innovations of the FMS are the creation of a market place for services developed from different providers and the incorporation of Future Internet technologies to easily deploy applications. Users pointed out that the FMS could extend its functionality in order to include more players along the food chain. Another issue that has been discussed was about the learning mechanisms of the system. It has been discussed that for the mechanism to be fairly evaluated more data from different crops is required.

6. Conclusions and future work

In this paper we have presented the implementation of a cloud-based FMS that allows the interconnection among services developed by different service providers. This creates a marketplace of services and applications that can be used by farmers. Thus, we have managed to create a more open system that is flexible enough so as to be tailor cut to the needs of every farmer. Also, by accommodating the main intelligence of the system in the cloud the requirements for the local installation are kept to a minimum resulting to cheaper solutions for the farmers. The FMS is validating a number of GEs developed by FI-WARE and adapted to support agricultural related tasks. During the design and implementation of the overall system we have also introduced a number of innovative concepts like the notion of a services' market place, network awareness in order for the system to adapt in malfunctioning Internet links, identification of malfunctioning sensor components, etc.

In order to turn into reality the concept of a market place of services developed by independent providers for the farmers, we believe that the most crucial issue is the one of open and standardized interfaces for the Cloud FMS. If the service developers have access to such interfaces, a farmer will be able to compose a working environment from interoperating services. Such concepts are currently under investigation in the context of the FI-PPP "FI-space" project that aims to create a platform for business to business collaboration among stakeholders in the food and product supply chain.

Acknowledgments

This work was performed in project SmartAgriFood that has received research funding from the Community's Seventh Framework program. The SmartAgriFood project is part of the Future Internet Public-Private Partnership (FI-PPP) program. This paper reflects only the authors' views and the Community is not liable for any use that may be made of the information contained therein. The contributions from colleagues of the SmartAgriFood consortium are hereby acknowledged.

References

Allen, J., Wolfert, J., 2011. Farming for the future: towards better information-based decision-making and communication – Phase I: Australasian stocktake of farm management tools used by farmers and rural professionals. New Zealand Centre of Excellence in Farm Business Management, Palmerston North <http://edepot.wur.nl/194811>.

Archimate <http://archi.cetis.ac.uk/> (7.04.13).

Batte, M., 2005. Changing computer use in agriculture: evidence from Ohio. Comput. Electron. Agric. 47 (1), 1–13.

Beck, H., 2001. Agricultural enterprise information management using object databases, Java, and COBRA. Comput. Electron. Agric. 32 (2), 119–147.

Brewster, C., Wolfert, S., Sundmaeker, H., 2012. Identifying the ICT challenges of the Agri-Food sector to define the Architectural Requirements for a Future Internet Core Platform. In: Proceeding of the Challenges Conference, Lisbon, Portugal.

FI-WARE MediaWiki, 2013: Materializing the FI-WARE Vision <http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Materializing_the_FI-WARE_Vision>.

Fountas, S., Wulfsohn, D., Blackmore, S., Jacobsen, H.L., Pedersen, S.M., 2006. A model of decision making and information flows for information-intensive agriculture. Agric. Syst. 87, 192–210.

Hart, P.E., Stork, D.G., Duda, R.O., 2000. Pattern Classification, second ed. Wiley, ISBN 978-0-471-05669-0.

Kaloxylos, A., Eigenmann, R., Teye, F., Politopoulou, Z., Wolfert, S., Shrank, C., Dillinger, M., Lampropoulou, I., Antoniou, E., Pesonen, L., Huether, N., Floerchinger, T., Alonistioti, N., Kormentzas, G., 2012. Farm management systems and the Future Internet era. Els. Comput. Electron. Agric. 89, 130–144.

Kruize, J.W., Robbemond, R.M., Scholten, H., Wolfert, J., Beulens, A.J.M., 2013. Improving arable farm enterprise integration – review of existing technologies and practices from a farmer's perspective. Comput. Electron. Agric. (in press).

Libelium motes. <www.libelium.com/waspmote/> (7.04.13).

Libelium Sensors, 2013. <http://www.libelium.com/products/waspmote/sensors>.

Mazza, R., Berre, A., 2007. Focus group methodology for evaluating information. In: 11th International Conference Visualization Techniques and Tools, Information Visualization, vol. IV, 2007, pp. 74–80.

Metsi, V., Androutsopoulos, I., Paliouras, G., 2006. Spam filtering with Naive Bayes – Which Naive Bayes?. In: Proceedings of Third Conference on Email and Anti-Spam, July 27–28, 2006, Mountain View, California USA.

Nielsen, J., Molich, R., 1990. Heuristic evaluation of user interfaces. In: Proceeding CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (1990), pp. 249–256.

Nikkilä, R., Seilonen, I., Koskinen, K., 2010. Software architecture for farm management information systems in precision agriculture. Comput. Electron. Agric. 70 (2), 328–336.

Nurkka, P., Norros, L., Pesonen, L., 2007. Improving usability of and user acceptance of ICT systems in farming. EFITA/WCCA Joint Congress in IT in Agriculture, Edinburgh.

Robbemond, R., Kruize, J.W. 2012. Data standards used for data-exchange of FMIS <https://sites.google.com/site/agrilabreferences/> (7.04.13).

Ross, T., 2010. Fuzzy Logic with Engineering Applications. John Wiley & Sons.

Sørensen, C.G., Fountas, S., Nash, E., Pesonen, L., Bochtis, D., Pedersen, S.M., Basso, B., Blackmore, S.B., 2010. Conceptual model of a future farm management information system. Comput. Electron. Agric. 72 (1), 37–47. http://dx.doi.org/10.1016/j.compag.2010.02.003.

Sørensen, C.G., Pesonen, L., Bochtisc, D.D., Vougioukas, S.G., Suomi, P., 2011. Functional requirements for a future farm management information system. Comput. Electron. Agric. 76 (2011), 266–276.

Teye, F., 2011. A conceptual model for collaboration – based management information systems, Master Thesis, Helsinki, Metropolia University of Applied Science.

Wang, N., Zhang, N., Wang, M., 2006. Wireless sensors and food industry – recent development and future perspective. Comput. Electron. Agric. 50, 1–16.

Wolfert, J., Verdouw, C.N., Verloop, C.M., Beulens, A.J.M., 2010. Organizing information integration in agri-food – a method based on a service-oriented architecture and living lab approach. Comput. Electron. Agric. 70, 389–405.

Wood, L., 1997. Semi-structured interviewing for user-centered design. Interactions 4 (2), 48–61.

xBee <http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/> (7.04.13).