agriculture

Article

The O3-Farm Project: First Evaluation of a Business Process Management (BPM) Approach through the Development of an Experimental Farm Management System for Milk Traceability

Mauro Zaninelli [1,*] and Matías Reyes Pace [2]

1 Università Telematica San Raffaele Roma, Via di Val Cannuta 247, 00166 Rome, Italy
2 Centro de Tecnologías de la Información, Facultad de Ciencias Veterinarias y Pecuarias, Universidad de Chile, Santa Rosa 11735, La Pintana, 8820808 Santiago, Chile; matreyes@gmail.com
* Corresponding author: mauro.zaninelli@unisanraffaele.gov.it; Tel.: +39-06-5225-2552

Received: 30 June 2018; Accepted: 6 September 2018; Published: 8 September 2018

Abstract: The modeling of farm workflows, and the use of a business process management (BPM) paradigm, could enable improvement in the development of farm management information systems (FMIS). A rapid design of software applications could be possible and quick development, intrinsically service oriented, could be achieved through the use of a software suite for the implementation of BPM diagrams. As the first evaluation of this paradigm, an experimental FMIS was developed considering a "use-case" whose target was to develop a hardware and software solution for the traceability of milk. The outcomes of this activity have shown that the software application developed (O3-Farm) was able to provide all features of the database application previously used for the traceability of milk. At the same time, it was able to provide some new features such as increased usability, portability and efficiency. Also, the chance to integrate it with other possible software applications was increased as a result of a better sharing of agricultural data. This seems to suggest that a design, and a software suite, based on the BPM paradigm, could be a valid way for the development of FMIS also in line with the farm software environment models if its abilities to describe, use and deploy, workflows and software services are taken into consideration.

Keywords: farm management systems; business process management; workflows; data sharing; dairy farms

1. Introduction

In the last few years, many models of farm management information systems (FMIS) have been investigated [1–3]. These models generally design a hardware and software layout that should improve the sharing of data and the availability of software functionalities [4–6]. Often, they include a decision support system (DSS) [7,8] even though these support systems can be frequently used with independent FMIS [9]. In some cases, following a holistic approach, these models have become more complex and conceptual [10]. They are called farm software environments (FSE) and they have the scope to make farm informative systems more collaborative, linkable, interoperable and scalable [11], taking as an example what happened in the information and communication technology (ICT) sector with the development of heterogenic projects related to specific operating systems (OS), software applications, market places, mobile and smart devices, customers retention, etc. [12]. However, after their development, these models have been tested in specific contexts [13]. Therefore, it is difficult to know if one of these models will become a real tool able to reach a critical mass that could attract software houses and affect the development of new products [14].

A different approach to follow in order to reach a possible improvement of FMIS could be the study of farm workflows. In a workflow, it is possible to define: the sequence of activities (and transactions) that must be performed; the actors responsible for specific activities; data that have to be sent, received and shared (between actors and/or workflows); and the format to use for the messaging. Through the standardization of farm workflows, as done in other scientific sectors [15,16], interesting progress in the development of FMIS could be achieved because better data sharing and integration of software applications could be possible [17,18]. Furthermore, farm workflows could be investigated using a business process management (BPM) approach [19]. This paradigm is focused on the evaluation of different aspects, of a business organization, in order to promote process effectiveness and efficiency [20]. It can help to standardize and improve the business processes; follow the market and environmental changes; and carry out processes reengineering and/or cost reductions [21]. The application of a BPM approach is the basis of making a business organization computerized because it allows separating the logical layer from the technical implementation (i.e., the process modeling and the software application development [20,22]). Furthermore, it is intrinsically oriented to a design for services [19]. Therefore, it could be a useful method for the rapid development, and collaborative integration, of FMIS (in accordance with the scopes of the FSE models [11]) also considering the recent availability of software tools for the implementation of BPM schemas.

As a first evaluation of this paradigm from a practical point of view, an experimental FMIS was developed considering a "use-case" studied in a previous project [23]. The scope of the past project was to develop a hardware and software solution for the traceability of milk produced by farm members of the Fontina Cheese Producers' Consortium. They were, in general, farms of small sizes with milking systems of a low technological level. In the project, a portable milking machine was equipped with a radio frequency identification (RFID) antenna and an electronic milking unit able to evaluate, on line and for each teat, the electrical conductivity of milk [24]. The RFID antenna was connected to a Palm device (Palm Inc., Sunnyvale, CA, USA) where a customized plug-in was able to record: the cow ID, the milking can ID and the "teat-alarm" scores. A database application, developed in Access (Microsoft, Redmond, WA, USA), completed the technical solution in order to store milking and farming data. Through that software application, and the queries that were implemented inside it, a basic traceability between the farm supplies and the farm outputs (i.e., the milking cans), was reached. Nevertheless, some limits were observed. The technology used to implement the software application was imposed to develop a "stand-alone" solution. Data were accessible only by a unique PC and could not be shared outside the farm with a third party, like the cheese factories. No kind of role was defined and no farm workflow was implemented. As a result, real improvements in the management of the whole farm, and production cycle, were not reached. In this scenario, an organization of farm activities as workflows, and the use of a service oriented technology, could allow better results. For this reason, this use-case was considered a good candidate in order to evaluate, under a practical point of view, if a BPM approach could be a valid method for the development of a real FMIS. On the basis of our knowledge, no similar studies have ever been performed.

2. Background and Design Considerations

2.1. The O3-Farm Project

What is presented in this paper was a first step in the development of a more extensive project. Its general scope is to investigate new software technologies, and design approaches, in order to promote the development of software applications for farm management. The primary objectives of the project are: an improvement of data sharing, an improvement of data elaboration and an increased interoperability of software applications to be reached through a possible standardization of data structures, of communication and collaboration protocols, and of paradigms for the development of software applications, procedures and workflows.

2.2. The Business Process Management (BPM) Paradigm

Business process management allows users to define, implement, and control, business processes. Its main scope is to improve the efficiency of each process in order to reach a global increase in the effectiveness of the whole business organization.

The BPM adopts a standard notation for the description of a business process, called business process management notation (BPMN). It allows users to describe a process with a graphical schema, called a "diagram". Each diagram, generally includes:

1. A "Pool" where the workflow is performed; where all diagram elements must be positioned; and where the actors that are involved in the process are defined.
2. A "Swim lane": when two or more lanes are defined in a pool, each lane includes a subset of activities, of the workflow that has a specific role in the process.
3. "Events": an event is shown in a diagram like a circle. It is something that can happen when a process is in execution (i.e., a "case" is performed). It is usually activated by a trigger, such as a signal, a message, a timer, an error, etc.; and it has a result that generally affects the workflow. For example, an event can start a process, modify the sequence of activities in a process, conclude a process, etc. Furthermore, an event can be classified as "catching" (if it catches an incoming trigger), "throwing" (if it creates a result and/or a trigger) or "non-interrupting" (if it has a result that does not affect the workflow).
4. "Tasks": a task is shown in a diagram like a rounded-corner rectangle. It is an activity to perform in a workflow that cannot be spilt to a lower level. Different types of task can be used in a diagram, such as: a "human task" (when an action is expected by a human subject); a "service task" (when something, typically a data elaboration, is required to a system); and a "send or receive task" (when a message has to be sent or received from another pool). Furthermore, a task can be repeated many times as a standard loop or as a parallel/sequential multi-instantiation.
5. "Activities": an activity is also shown in a diagram like a rounded-corner rectangle. It is a part of a workflow that has to be carried out. It can be "atomic" (when it is a task) or "compound" (when it is a sub-process, a transaction, etc.).
6. "Gateways": a gateway is shown in a diagram like a rhombus. It is an element of a diagram that can control the path of the workflow. A gateway can be: "exclusive" (when it creates alternatives flows in a diagram); "inclusive" (when it allows alternative flows in a diagram that are all evaluated); and "parallel" (when it creates parallel flows without any evaluation of a specific condition).
7. "Flows": a flow is shown in a diagram like an arrow. It is used to connect two elements of a diagram. It can be a "sequence flow" (when it shows the execution order of the diagram activities); a "message flow" (when it highlights a messaging flow between two pools); or an "association" (when it connects a part of a diagram that is linked to an artifact, such as a text annotation, etc.).

Finally, other information can be considered as a part of a diagram or business process. For example: its name and its version can be useful to identify a change when a process that is already active in the business organization must be updated.

3. System Description

3.1. O3-Farm Main Features

The O3-Farm is a "web oriented" software application. It is accessible via the internet, using a common browser and the HyperText Transfer Protocol (HTTP). Its graphical theme is "responsive" to allow its use with any kind of internet device (PC, tablet, smart phone, etc.).

Data entry is shared between different groups of users and organized in business processes (i.e., farm workflows) in order to: assign each task to a specific group of users; perform in different times a process that involves more than one user; and allow the execution of more processes, or more instances of the same process, in a concurrent way.

The groups of users of the O3-Farm application describe the structure of a possible farm organization. For each of them, roles and responsibilities are well defined and are mapped in the farm workflows where the groups are involved.

3.2. O3-Farm Functional Description

After a web page for the "user-group" selection (Figure 1), each user is forced to follow a login procedure (Figure 2) that allows them to reach a different functional page on the basis of the group to which the user belongs. In the software application, three different groups of users are defined: "managers", "employees" and "milkers".

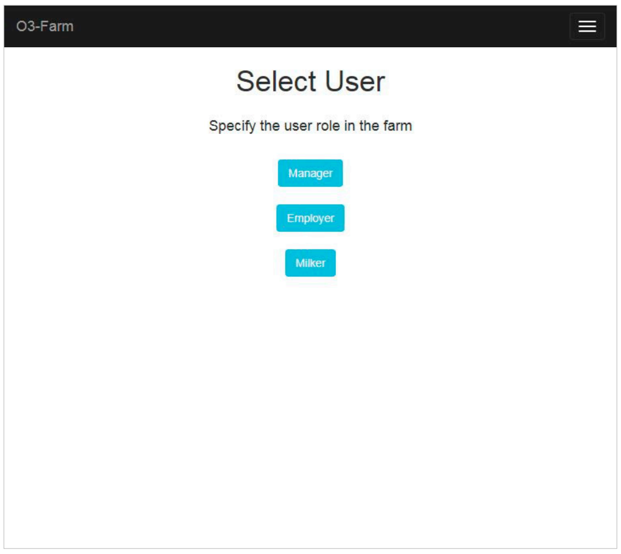


Figure 1. User-group-selection page.

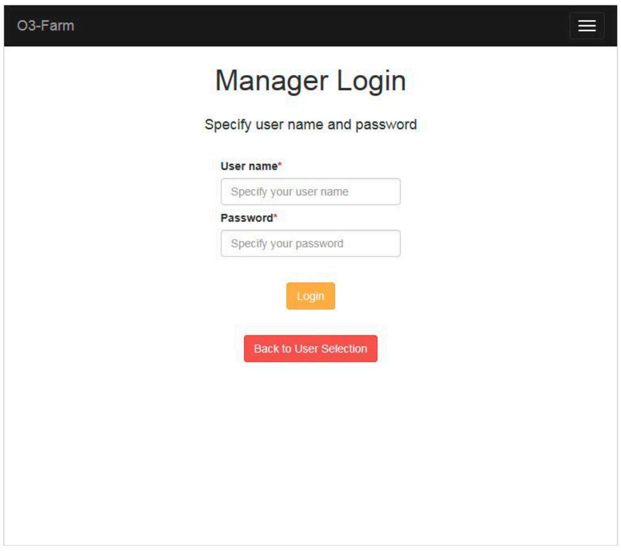


Figure 2. Example of a user login page.

Users of the group "managers" organize most of the farm data. After a login procedure, they reach a web page (Figure 3) that includes two functional areas. The first area, on the left side of the page, allows farm data to be stored. The second area, on the right side of the page, permits milking data to be managed and any information to be retrieved from the archive.

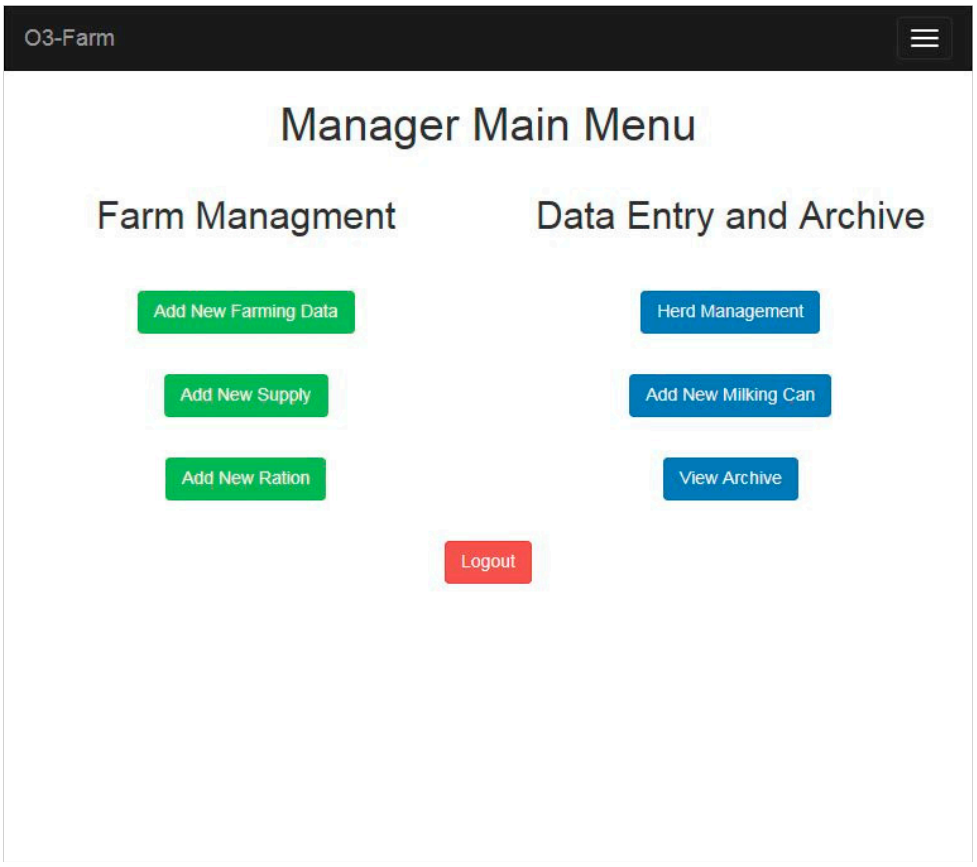
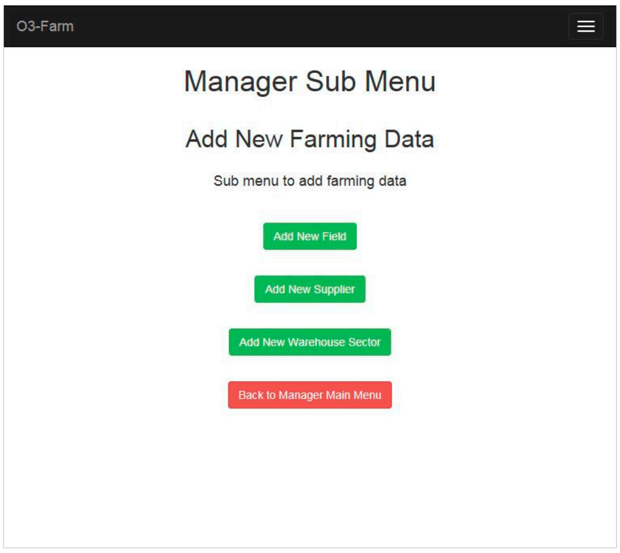


Figure 3. Manager main menu page.

Through the first functional area, called "Farm Management", a manager can:

1. "Add new farming data". Through a linked sub-menu (Figure 4), he/she can:
   a) "Add a new field" and define: a "Field code", its "Area", and its "Type" using a dedicated window selection box.
   b) "Add a new supplier" and store: its "Name", its "Address", and its "Value Added Tax (VAT) identification number".
   c) "Add a warehouse sector" and define: a "Progressive sector code" (useful to identify this sector of the farm warehouse), and a textual "Description".
2. "Add a new supply" and specify: the "Delivery date", the "Product type" and its "Description". When the supply is produced in the farm, the manager can select a "Field code". In the other cases, he/she can select a "Supplier name" (from a list of suppliers already stored in the O3-Farm application) and include: the "Lot", the "Seal", and the "Invoice" number.
3. "Add a new ration" for a group of cows. Each ration can be defined selecting up to six different ingredients. For each ingredient of the ration, the "Warehouse sector" where it is stored, and its "Quantity", are information requested. With this web form, the manager can also define: the "Number of daily distributions", the "Group ID" of the cows that will receive this ration, and the starting day using a "Ration date" box (Figure 5).


O3-Farm

Manager Sub Menu

Add New Farming Data

Sub menu to add farming data

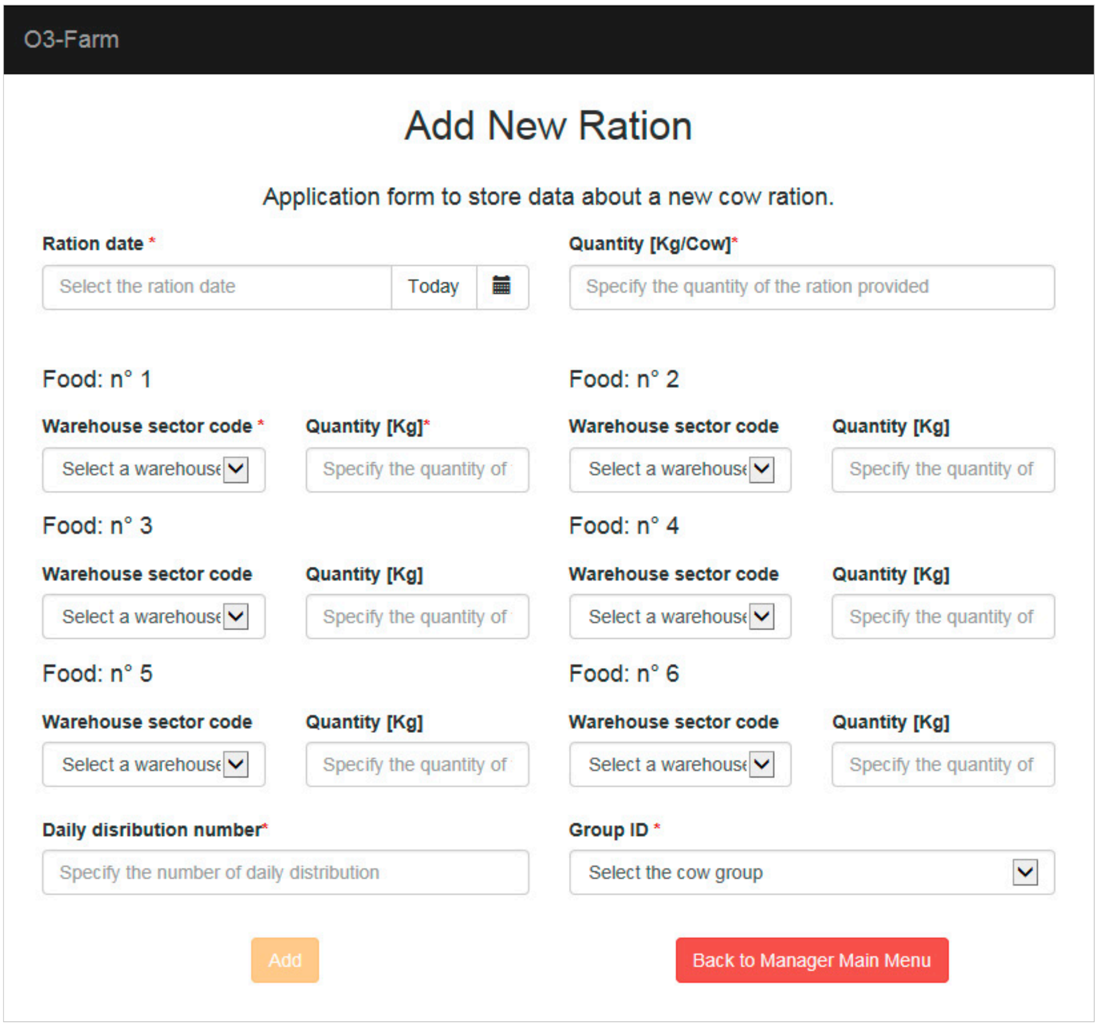
Add New Field

Add New Supplier

Add New Warehouse Sector

Back to Manager Main Menu

Figure 4. Manager sub-menu to add farming data.


O3-Farm

Add New Ration

Application form to store data about a new cow ration.

Ration date *: Select the ration date | Today

Quantity [Kg/Cow]*: Specify the quantity of the ration provided

Food: n° 1 — Warehouse sector code *: Select a warehouse — Quantity [Kg]*: Specify the quantity of

Food: n° 2 — Warehouse sector code: Select a warehouse — Quantity [Kg]: Specify the quantity of

Food: n° 3 — Warehouse sector code: Select a warehouse — Quantity [Kg]: Specify the quantity of

Food: n° 4 — Warehouse sector code: Select a warehouse — Quantity [Kg]: Specify the quantity of

Food: n° 5 — Warehouse sector code: Select a warehouse — Quantity [Kg]: Specify the quantity of

Food: n° 6 — Warehouse sector code: Select a warehouse — Quantity [Kg]: Specify the quantity of

Daily disribution number*: Specify the number of daily distribution

Group ID *: Select the cow group

Add

Back to Manager Main Menu

Figure 5. Application form that a manager can use to store a new ration for a specific group of cows.

Through the second functional area, called "Data Entry and Archive" (Figure 3), a manager can:

1. Make the "Herd management". Through a linked sub-menu (Figure 6), he/she can: "Add a new group" of cows (defining a "Group ID" and a textual "Description"), and "Add a new cow" in the herd of the farm. In this latter case, the manager must record: the "Name" of the cow, its "Race", its "Birth date" and its "Gestations number". Also the "Animal ID", and its "Group ID", are required.
2. "Add a new milking can" and define a "Milking Can ID". This ID is useful to identify where the milk yields of some specific cows have been stored during a milking procedure.
3. Retrieve data from the "Archive". Through a linked sub-menu (Figure 7), he/she can:

   a) "Retrieve milking data of the group of cows that have filled a specific can of milk" (Figure 8). Choosing the "Date of milking", the "Milking hour" (i.e., morning or evening) and the ID of the can, each manager can retrieve the milking data of cows whose milk yields have been used to fill a can of milk, in terms of: "Animal ID", "Yield" and recorded "teat alarm scores".
   b) "Retrieve supply data of a specific ingredient". Selecting the "Delivery date", and the "Warehouse sector", each user can display supply data in terms of "Product type" (i.e., produced in farm or bought from an external supplier) and, as a consequence, the "Field code" or the "Supply name", the "Lot", the "Seal" and the "Invoice" number.
   c) "Retrieve ration data of a specific group of cows". Choosing the "Milking date" and the "Group ID", a manager can retrieve data about the ration used, in terms of "Quantity" and "Warehouse sector" where each ingredient has been stored.
   d) "Retrieve milking data of a specific cow". Selecting the "Animal ID" of a cow that belongs to the herd, a user can display milking data like the: "Milking date", recorded "Yield" and all "teat alarm scores".

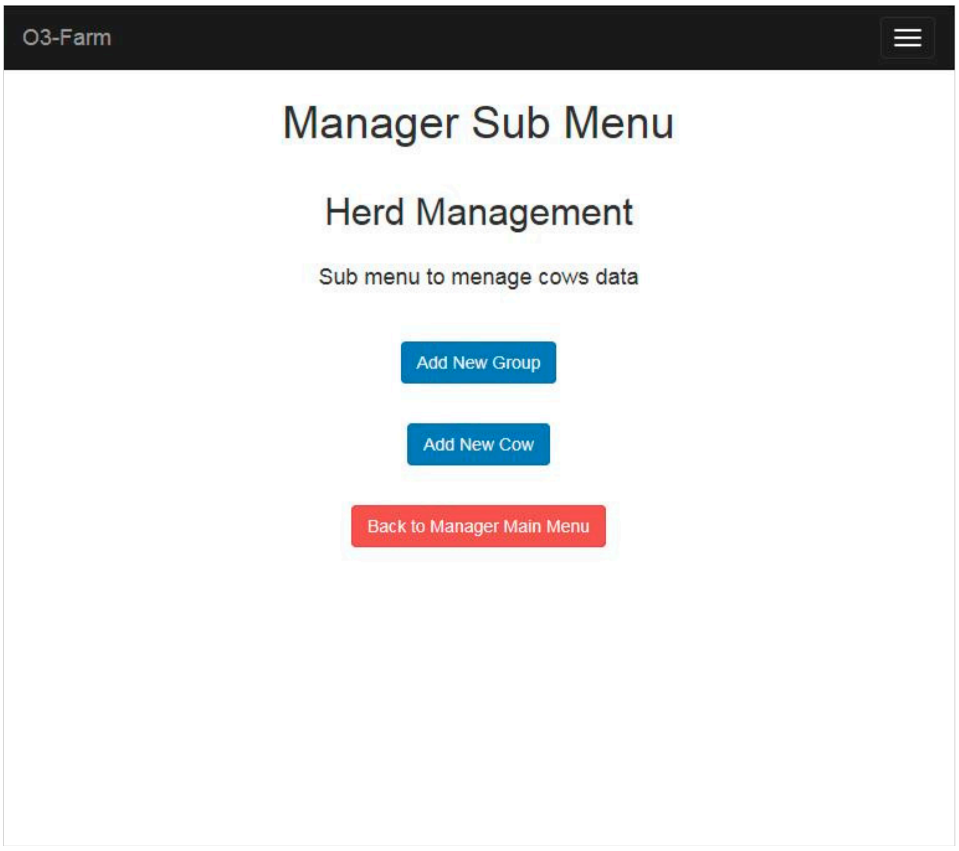


Figure 6. Herd management sub-menu available for managers to add a cow or a new group of animals.

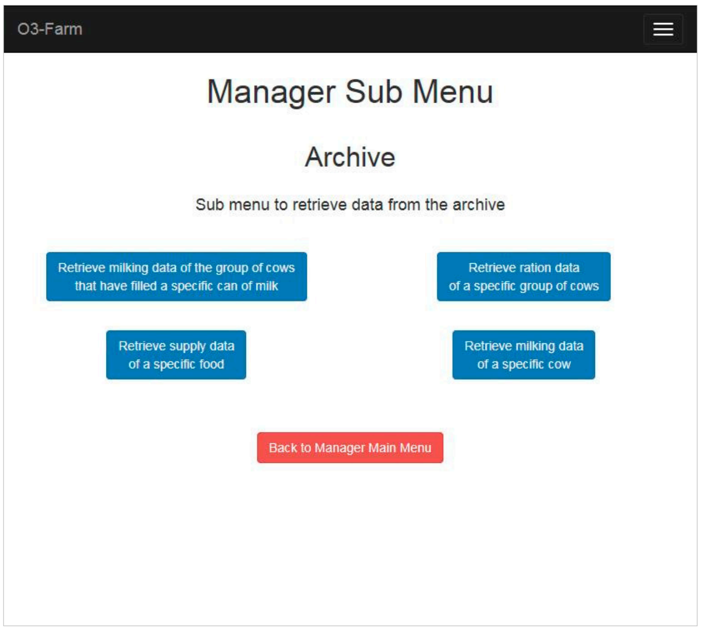
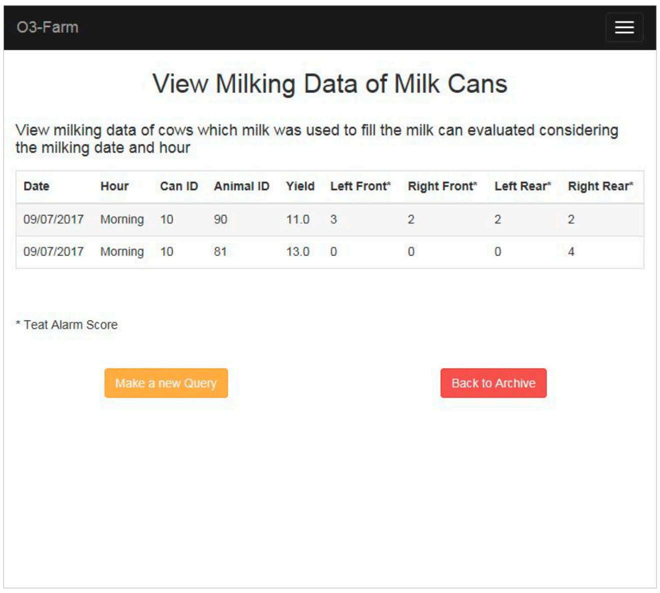

O3-Farm
Manager Sub Menu
Archive
Sub menu to retrieve data from the archive
Retrieve milking data of the group of cows that have filled a specific can of milk
Retrieve ration data of a specific group of cows
Retrieve supply data of a specific food
Retrieve milking data of a specific cow
Back to Manager Main Menu

Figure 7. Archive sub-menu available for managers to retrieve milking and agricultural data.



View Milking Data of Milk Cans

View milking data of cows which milk was used to fill the milk can evaluated considering the milking date and hour

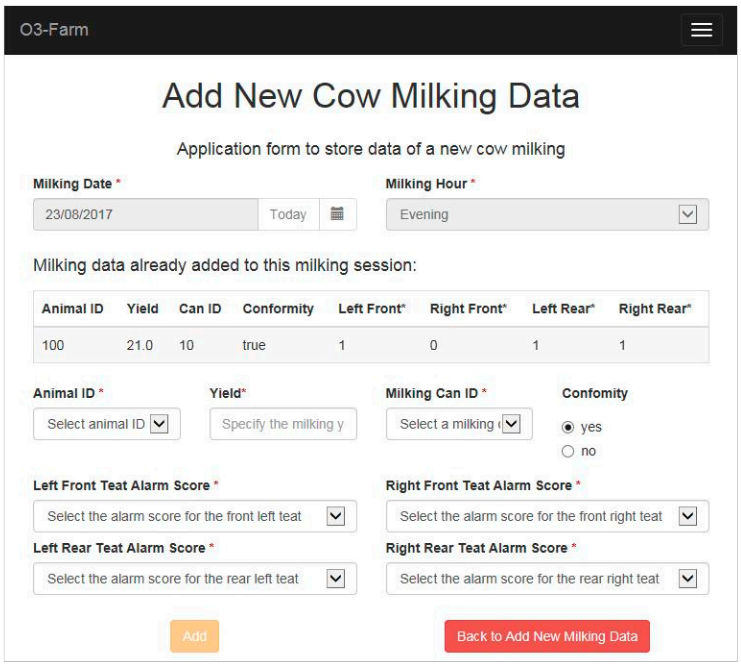
Date	Hour	Can ID	Animal ID	Yield	Left Front*	Right Front*	Left Rear*	Right Rear*
09/07/2017	Morning	10	90	11.0	3	2	2	2
09/07/2017	Morning	10	81	13.0	0	0	0	4

* Teat Alarm Score

Figure 8. Example of a set of milking data of the group of cows that have filled a specific can of milk.

The users of the group "employees" are responsible for the farm warehouse. After a login procedure, when a new supply is waiting to be stored in the farm warehouse, employees can reach a web page where a dedicated link is provided. Through this link, an employee can display supply data, such as: the "Delivery date", the "Product type" and its "Description". When the supply comes from the farm, the "Field code" where it has been produced is shown to the user. In the other cases, the "Supplier name", the "Lot", the "Seal" and the "Invoice" number are the data displayed on the screen. When the supply is stored in a specific sector of the farm warehouse, he/she can record this data using this page. Of course, when no other supply is waiting to be stored, the message "no further tasks are required" is displayed to users that reach the web page described above.

The users of the group "milkers" have the responsibility to add milking data, two times a day, after the end of milking procedures. When a new set of milking data are available, the milkers can reach a web page, after a login procedure, where a dedicated link is shown. Through this link, milkers can display a page (Figure 9) that reports: the "Milking date", the "Milking hour", and a set of milking data already stored for that milking. Using the input fields that are in the lower side of the page, milkers can add a new set of milking data that includes: the "Animal ID", the "Yield", the "Milking can ID" (where the milk of a specific cow has been stored), a "Conformity" evaluation of the milk produced, and all "teat alarm scores" recorded by the electronic milking unit. When this web page is reloaded, the new set of data is added to the list of milking data already recorded. The activity of data entry is suspended when data of all cows have been stored and no other milking session has to be recorded. In this latter case, if a user tries to reach the web page described above, he/she is informed that "no further tasks are required".



O3-Farm

Add New Cow Milking Data

Application form to store data of a new cow milking

Milking Date *	23/08/2017
Milking Hour *	Evening

Milking data already added to this milking session:

Animal ID	Yield	Can ID	Conformity	Left Front*	Right Front*	Left Rear*	Right Rear*
100	21.0	10	true	1	0	1	1

Animal ID *	Select animal ID
Yield*	Specify the milking y
Milking Can ID *	Select a milking
Confomity	[✓ yes] no
Left Front Teat Alarm Score *	Select the alarm score for the front left teat
Right Front Teat Alarm Score *	Select the alarm score for the front right teat
Left Rear Teat Alarm Score *	Select the alarm score for the rear left teat
Right Rear Teat Alarm Score *	Select the alarm score for the rear right teat

Figure 9. Form available for milkers to add milking data after a milking session.

3.3. Description of the Business Procedures

Data entry of the O3-Farm application is shared between different groups of users and designed in business processes. The most frequent process has a structure like the one reported in the diagram of Figure 10. This type of business process is used when a user must store a set of data in the database. When he/she submits a set of inputs, by a dedicated web form, the process starts and a "service task" allows data submitted to be collected and recorded. After this task, the process ends.

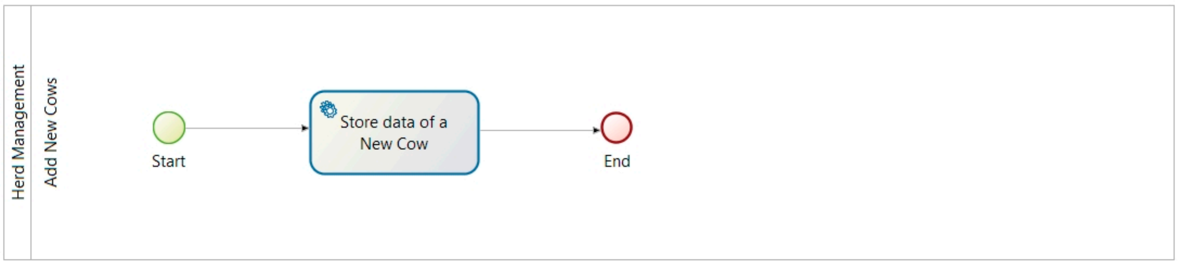


Figure 10. Example of a business process used to store a set of inputs in the software application database. In the diagram, using the business process management notation (BPMN), the business process "Add new cow" is reported.

Even though simple, this kind of business process involves different components of the software application. As shown in the activity diagram of Figure 11, the business engine instantiates a process when a user selects a specific function of the O3-Farm application. An instantiation form is displayed on the screen of the user in order to collect the data. When the user pushes the button "Add", the business processes engine starts the process. A new "business variable" is created, from a set of variables defined in the "business model", and is initialized with the inputs received by the web-form. A dedicated Groovy script, like the one reported in Figure 12, is used by the business processes engine to perform this activity. In a following step, the data are permanently stored in the database by an internal task performed by the business process engine. After this step, the business processes engine closes the process instance without the need for other user' activities.

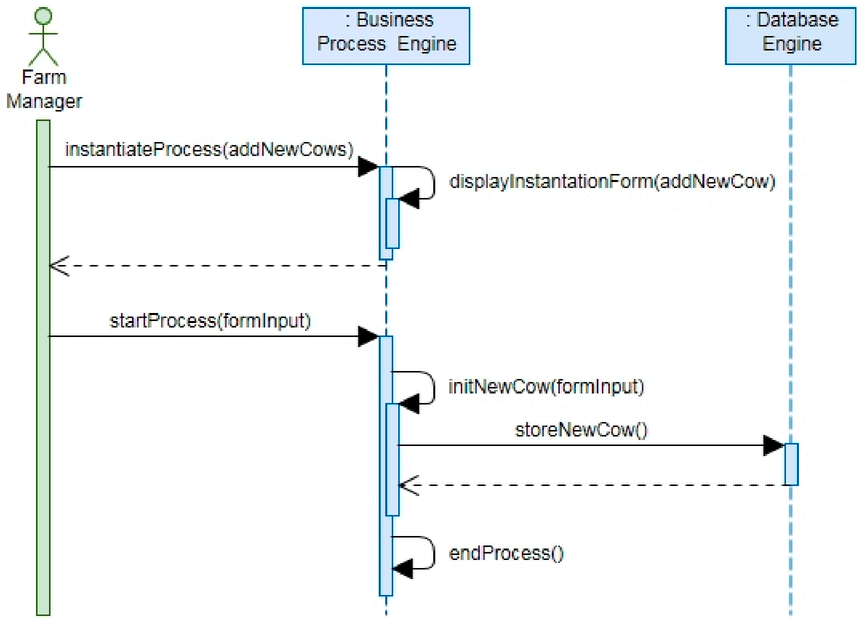


Figure 11. Activity diagram of a business process used to store a set of inputs in the software application database. In detail, the diagram shows the activities performed by different software components during the execution of the process "Add new cow".

```
def newCowVar = new com.company.model.Cows()
newCowVar.Name = newCowInput.Name
newCowVar.Race = newCowInput.Race
newCowVar.Birth_date = newCowInput.Birth_date
newCowVar.Gestations_number = newCowInput.Gestations_number
newCowVar.Animal_ID = newCowInput.Animal_ID
newCowVar.Group_ID = newCowInput.Group_ID
return newCowVar
```

Figure 12. Example of a Groovy script ("initNewCow") used to create and initialize a new business variable during the execution of the business process "Add new cow". This business variable is an instance of a business object (Cows) included in the business model of the software application.

A different business process, adopted in the O3-Farm application, is shown in Figure 13. This kind of process is used when a user of the group "managers" must retrieve a set of data from the database. The process starts through the submission of inputs collected by an instantiation form, filled by the user. These inputs are parameters used to build a query for the database. The "human task" is used to display the query result and catch the subsequent user actions (i.e., to make another query or to close a "case"). However, on the boundary of the human task, a "timer event" is provided. This timer is a "task timeout" that can close the human task if an action has not been selected by the user. In this way, any instance of the process can be closed after a reasonable interval of time.

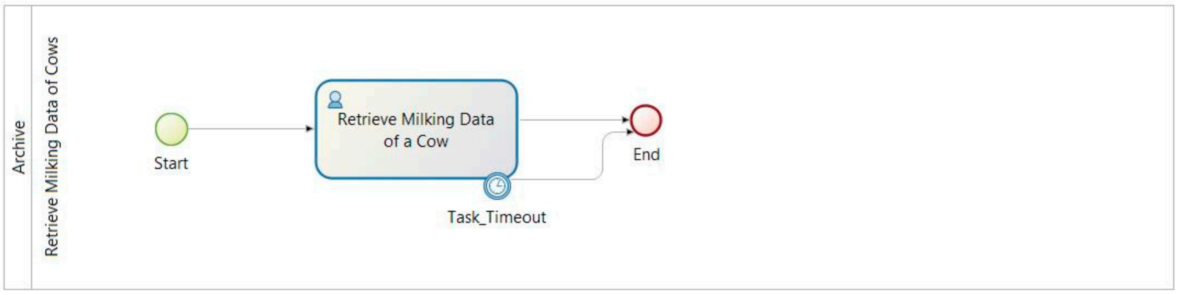


Figure 13. Example of a business process used by the software application to retrieve a set of data from the database.

The software activities required by the business process of Figure 13 are described also in the activity diagram of Figure 14. As shown in the diagram, when a farm manager selects this kind of function, from the web page of the software application, the corresponding process is instantiated by the business processes engine. An instantiation form is displayed on the screen of the user in order to collect data necessary to build the database query. When the user pushes the button "Submit", the business processes engine starts the process and retrieves the required data through a dedicated database connector. This connector is a software element that allows the process to read or write information from an external information system, like a database. Its configuration requires to define: the name of the Java Database Connectivity driver (JDBC) to use (org.h2.Driver—h2-1.3.170.jar), the JDBC Uniform Resource Locator (URL); and the name and password of the user that has the required privileges in the database to perform a query. The configuration of this specific connector adopts a dedicated query which Groovy scripts is reported in Figure 15. As it is shown in the figure, the result of the query is provided in a JavaScript object notation format (JSON) and it is returned as a java object "java.sql.ResultSet" to the business processes engine in order to be stored as a "process variable". When the result of the database query is available at process level, the business processes engine uses it to display the information requested on the screen of the user. All these software activities are repeated every time that a user makes a query, generating a new "case" of the same process for each query performed. In the same time, each instance of the process (i.e., each "case") is automatically closed by the business processes engine when a defined time is elapsed.

Another structure of the business process is shown in Figure 16. This process is used by the software application to store milking data after the end of a milking procedure. The start of this process is automatically provided by the system and scheduled two times a day, before each milking session. After the start of the process, the workflow is merged by an "exclusive" gateway and redirected to a gateway that makes a check on a condition imposed. This condition is set up at the beginning of the process considering the number of cows in the herd of the farm. When the number of iterations performed is lower than the herd size, the workflow reaches a human task: "Store data of a new cow milking". With this task, a milker can record a set of milking data that belongs to a cow. At the same time, a process variable for the counting of performed iterations is updated. After this step, the workflow comes back to the first "exclusive" gateway, to be merged, and it is redirected to another gateway to check the exit condition. When the number of performed cycles are equal to the herd size (i.e., number of milking data sets = number of cows of the herd) the workflow uses the default flow provided by the gateway "Check iteration". As a result, the process ends.

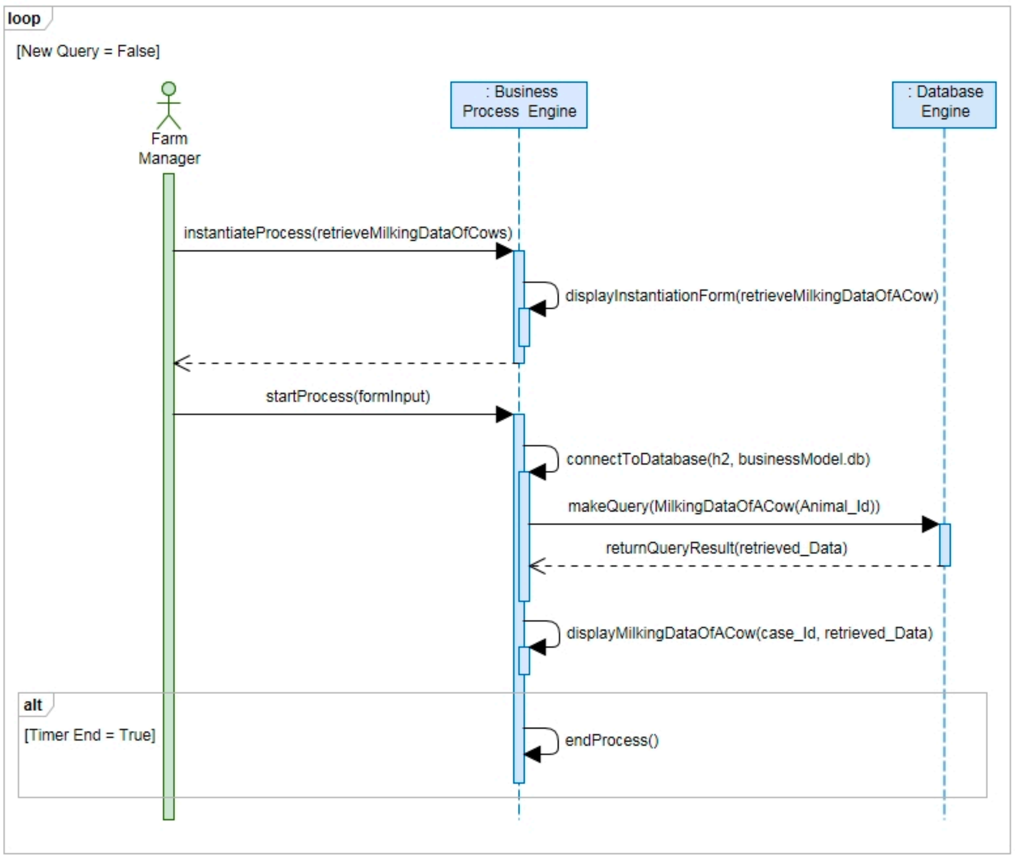
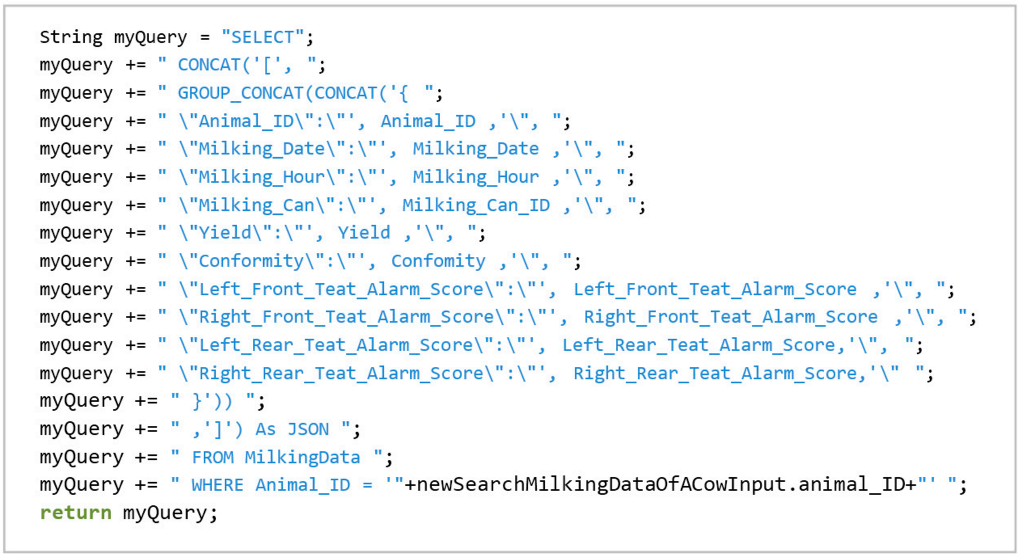

loop
[New Query = False]
Farm Manager
: Business Process Engine
: Database Engine
instantiateProcess(retrieveMilkingDataOfCows)
displayInstantiationForm(retrieveMilkingDataOfACow)
startProcess(formInput)
connectToDatabase(h2, businessModel.db)
makeQuery(MilkingDataOfACow(Animal_Id))
returnQueryResult(retrieved_Data)
displayMilkingDataOfACow(case_Id, retrieved_Data)
alt
[Timer End = True]
endProcess()

Figure 14. Activity diagram of a business process used to retrieve data from the database. In detail, the diagram shows the collaboration between different software components during the execution of the process "Retrieve Milking Data of a Cow".



```
String myQuery = "SELECT";
myQuery += " CONCAT('[', ";
myQuery += " GROUP_CONCAT(CONCAT('{  ";
myQuery += " \"Animal_ID\":\"', Animal_ID ,'\", ";
myQuery += " \"Milking_Date\":\"', Milking_Date ,'\", ";
myQuery += " \"Milking_Hour\":\"', Milking_Hour ,'\", ";
myQuery += " \"Milking_Can\":\"', Milking_Can_ID ,'\", ";
myQuery += " \"Yield\":\"', Yield ,'\", ";
myQuery += " \"Conformity\":\"', Confomity ,'\", ";
myQuery += " \"Left_Front_Teat_Alarm_Score\":\"', Left_Front_Teat_Alarm_Score  ,'\", ";
myQuery += " \"Right_Front_Teat_Alarm_Score\":\"', Right_Front_Teat_Alarm_Score  ,'\", ";
myQuery += " \"Left_Rear_Teat_Alarm_Score\":\"', Left_Rear_Teat_Alarm_Score,'\",  ";
myQuery += " \"Right_Rear_Teat_Alarm_Score\":\"', Right_Rear_Teat_Alarm_Score,'\"  ";
myQuery += " }')) ";
myQuery += " ,']') As JSON ";
myQuery += " FROM MilkingData ";
myQuery += " WHERE Animal_ID = '"+newSearchMilkingDataOfACowInput.animal_ID+"' ";
return myQuery;
```

Figure 15. Example of a Groovy script ("queryMilkingDataOfACow") used to retrieve milking data of a specific cow. The script builds a query using the "Animal_ID" provided by the user through the instantiation form "retrieveMilkingDataOfACow". This completes the configuration of the database connector used by the business processes engine to retrieve the required data.

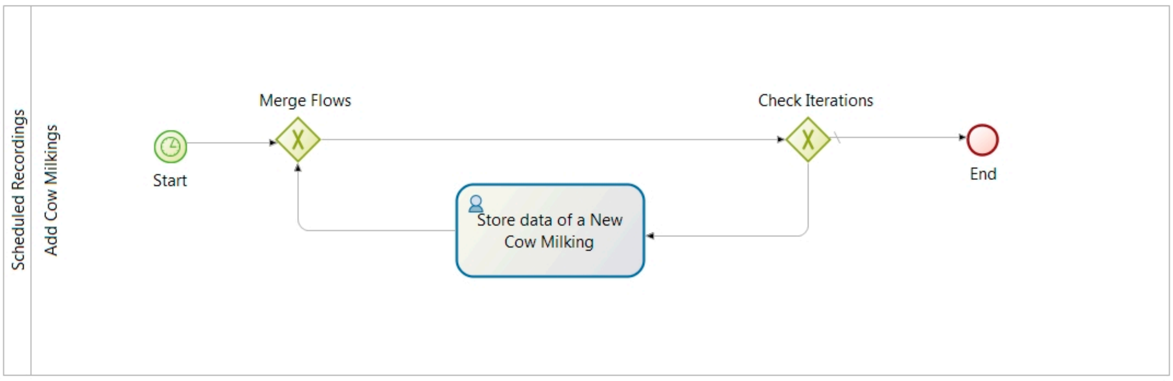


Figure 16. Diagram of the business process used to collect milking data of the farm herd.

Activities required by this business process are described also in the activity diagram of Figure 17. As shown in the diagram, the process ("Add milking data") is automatically instantiated (and started) by the business processes engine when a temporal condition is verified (i.e., two times a day before the start of milking procedures). To each case (i.e., an instance of a process), an incremental ID is assigned following the order of instantiation. When each process starts, the business processes engine retrieves the required data (i.e., the herd size) through a dedicated database connector. The same parameters, reported above, are used to configure this software connector. The query built, instead, is different because it has a different target: to return and count the number of records stored in a specific database table (called "Cows"). When the result of the database query is available at process level, two variables are created and initialized: a process variable called "remainingCows" and a business variable "cowsMilkingData" defined as a list of business objects "MilkingData". The Groovy script for the initialization of this business variable is reported in Figure 18. When a milker accesses the application to insert milking data, the web page displayed on the screen uses a sequence of calls to specific representational state transfer (REST) application programming interfaces (API). Through these calls, executed over the HTTP protocol, the software application can access to the process workflow and retrieve the process ID (knowing its name), the case ID (from a list of open cases that belong to a process characterized by a specific ID), and a task ID (from a list of pending tasks that belong to the same case, of a defined process, that have been assigned to a specific user, or group of users). When all these data are retrieved, the user can send to the business processes engine a set of new milking data. This sequence of activities are performed, by the software application, until some milking data are missing. When milking data of all cows of the herd have been stored in the database (i.e., the process variable "remainingCow" is equal to zero), the corresponding case is automatically closed.

Figure 19 shows a business process that requires the collaboration of users from different groups. The O3-Farm application uses this process to manage the warehousing of a new supply. The process starts when a manager submits a set of inputs by a dedicated instantiation form. The first task of the process is a "service task" that stores the submitted data. The workflow continues with a "human task" assigned to a user of the group "employees". With this task, an employee of the farm can record in the software application which is the sector warehouse where the supply has been stored. When this task is carried out, the process can be concluded.

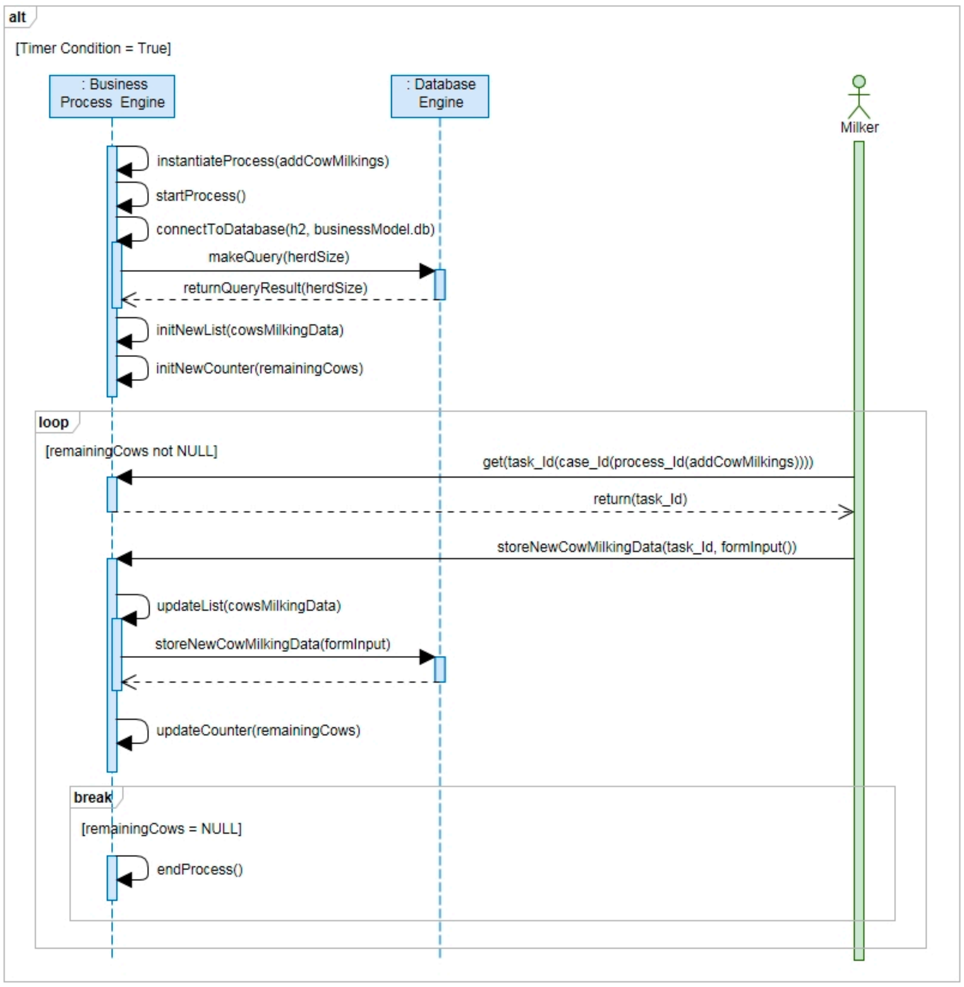
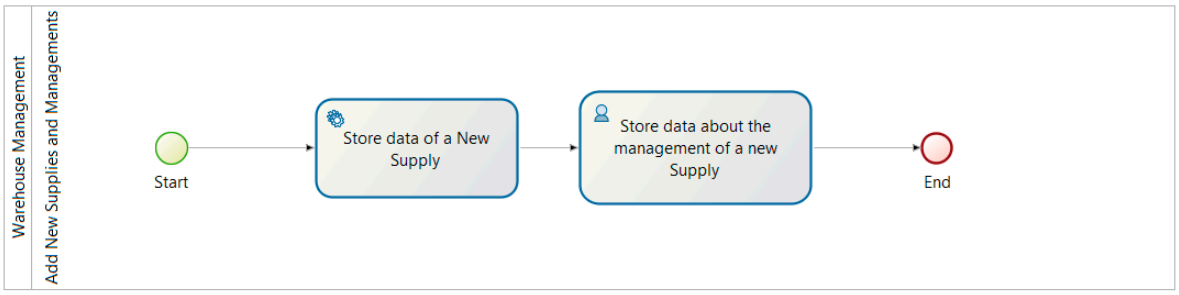

alt
[Timer Condition = True]
: Business Process Engine
: Database Engine
Milker
instantiateProcess(addCowMilkings)
startProcess()
connectToDatabase(h2, businessModel.db)
makeQuery(herdSize)
returnQueryResult(herdSize)
initNewList(cowsMilkingData)
initNewCounter(remainingCows)
loop
[remainingCows not NULL]
get(task_Id(case_Id(process_Id(addCowMilkings))))
return(task_Id)
storeNewCowMilkingData(task_Id, formInput())
updateList(cowsMilkingData)
storeNewCowMilkingData(formInput)
updateCounter(remainingCows)
break
[remainingCows = NULL]
endProcess()

Figure 17. Activity diagram of a business process used to record milking data after each milking session. In detail, the diagram shows the collaboration between different software components during the execution of the process "Scheduled Recordings".

```
def var = [] as List<com.company.model.MilkingData>;
for (int i = 0; i < herdSize; i++) {
   var[i] = new com.company.model.MilkingData([
        'Animal_ID' : null,
        'Milking_Date' : null,
        'Milking_Hour' : null,
        'Yield' : null,
        'Milking_Can_ID' : null,
        'Confomity' : null,
        'Left_Front_Teat_Alarm_Score' : null,
        'Right_Front_Teat_Alarm_Score' : null,
        'Left_Rear_Teat_Alarm_Score' : null,
        'Right_Rear_Teat_Alarm_Score' : null
        ]);
}
return var as List<com.company.model.MilkingData>;
```

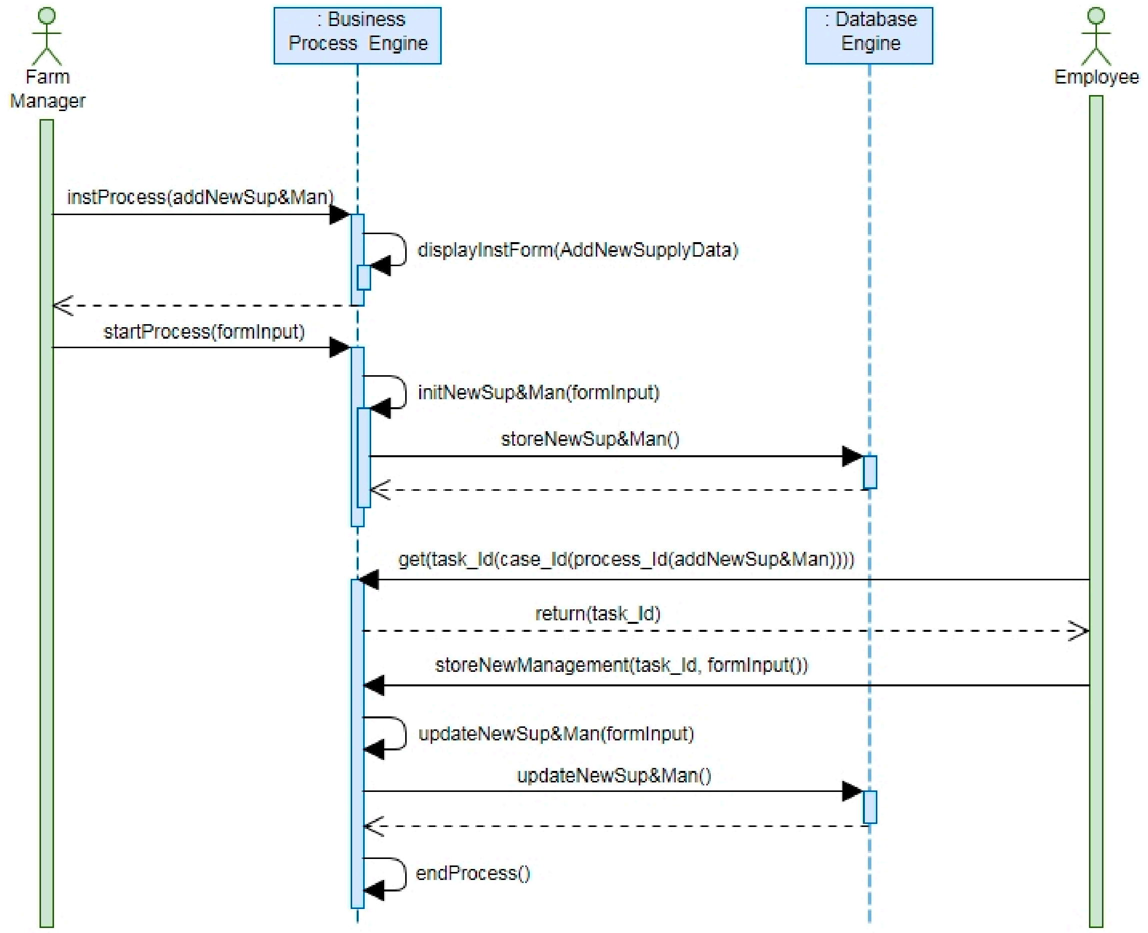
Figure 18. Example of a Groovy script ("initCowsMilkingData") used to create and initialize a business variable adopted to collect and store a set of cow milking data.



Warehouse Management
Add New Supplies and Managements
Start
Store data of a New Supply
Store data about the management of a new Supply
End

Figure 19. Diagram for the management of a new food supply in the farm warehouse.

Activities performed by this business process are reported also in the activity diagram of Figure 20. As shown by the diagram, the process ("Warehouse Management") requires an asynchronous collaboration of two users of different groups (managers and employees). A new case of the process is opened by a manager when he/she submits a set of inputs (related to a new supply of a food). When a case is opened, a new task is created and assigned to a user that has, in the organization, the role of "employee". To execute its task, the employee must retrieve the corresponding ID from the business processes engine. This activity is automatically performed by the web page displayed on its screen using a sequence of REST APIs provided by the business processes engine over the HTTP protocol. When these data are retrieved from the system, the employee can store a set of new data and, therefore, this allows the business processes engine to close the corresponding instance of the process.



Farm Manager
: Business Process Engine
: Database Engine
Employee
instProcess(addNewSup&Man)
displayInstForm(AddNewSupplyData)
startProcess(formInput)
initNewSup&Man(formInput)
storeNewSup&Man()
get(task_Id(case_Id(process_Id(addNewSup&Man))))
return(task_Id)
storeNewManagement(task_Id, formInput())
updateNewSup&Man(formInput)
updateNewSup&Man()
endProcess()

Figure 20. Activity diagram of a business process that involves an asynchronous collaboration of two users of different groups. In detail, the diagram shows the collaboration between different software components, and users during the execution of the process "Warehouse Management".

3.4. Database Layout

The database layout, used in the O3-Farm application, can be described by the entity-relationship diagram (ERD) reported in Figure 21. In the diagram: rectangles show the entities of the model. Rhombuses describe the relationships between entities and small circles display the attributes of entities or relationships. Numbers surrounded by round brackets describe the cardinality of relationships between entities, and black colored circles show the primary keys of the database.

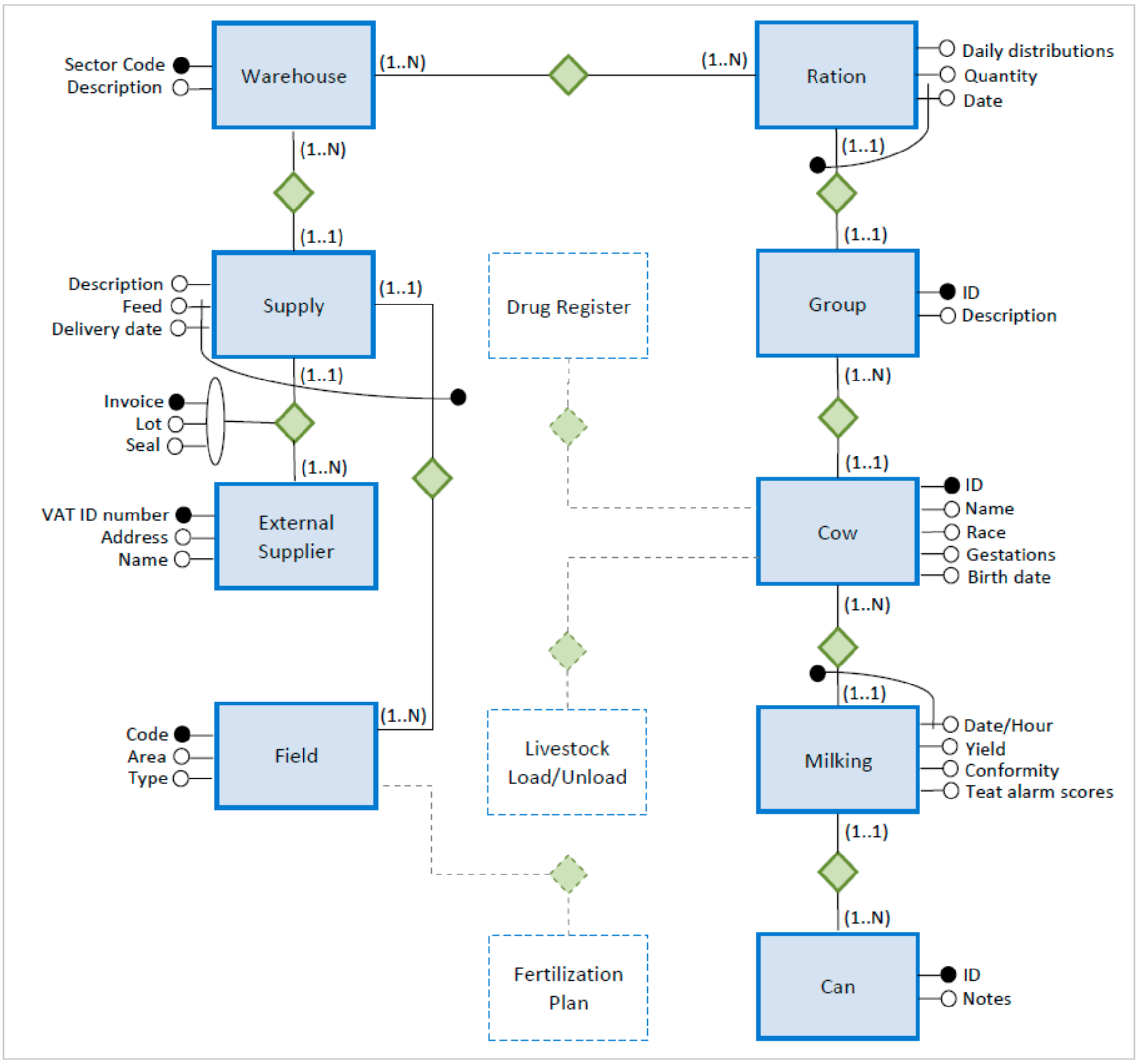


Figure 21. Entity relationship diagram (ERD) of the database layout.

The ERD explains how the database provides the traceability between the supplies that enter in the farm and what can be an output, namely, a can of milk. This is the smallest lot of milk that can be identified. It is modeled as an instance of the entity "Can" that has a one-to-many relationship with the entity "Milking" in order to collect data about the milk yields that were used to fill it. Each instance of the entity "Milking" has a one-to-one relationship with instances of the entities: "Cow", "Group" and "Ration". With this data structure, each milk yield can be linked to the supplies used to feed the cows that were milked. With the entities "External Supplier" and "Field", the database can store information about the suppliers and by their relationships with the entity "Supply", it is able to connect them to the farm outputs.

The ERD also shows, with dotted lines, possible relationships with external data sources that are generally available because they are requested by law or national regulations. Many times, they are paper registers that cannot be integrated with a software application. However, linking them to the data structure of the O3-Farm, full management of the farm could be achieved.

On the basis of the data structure shown in Figure 21, a business data model has been developed and used in the O3-Farm software application.

3.5. Software Tools Used

The O3-Farm application has been developed using the Bonita BPM software tool. The "Community" edition is open source and provides many tools for the development of BPM applications, such as: the "Studio", the "User Interface (UI) Designer" and the "Portal".

The Studio is the main tool for the development of business processes. It allows a complete business process to be designed by a graphical user interface. This tool permits the business data model of the whole project to be created and each business process to be set up defining: (1) its process variables, in terms of "Pool" or "Local" variables (i.e., with a task level scope); (2) inputs from users; (3) operations to perform on process variables during a case execution (basically through Java methods or Groovy scripts); (4) connectors to external resources (such as databases, ERP, SOAP web services, messaging systems, CMS, etc.) that can be used at the beginning or conclusion of a case; (5) who can initiate a process and/or perform specific tasks in terms of group of users or subset of filtered actors. Furthermore, the Studio allows the process to be validated and deploy to the "Processes Engine".

The UI Designer, instead, is a tool for the development of web content. It is basically used for the design of custom pages and forms. It has a graphical user interface that allows developers to build contents using common web elements, such as: input fields, selects, checkboxes, buttons, links, text areas, tables, containers, etc. The tool permits to define variables, with a scope in the custom page or form, that can be used to handle business data (of processes, cases or specific tasks) or to contain structural information that determines how the web elements have to behave. Available types, for these variables, are: strings and JSON for static variables; REST API, JavaScript expressions, and URL parameters, for dynamic variables that are initialized when the page is loaded and updated every time that the page is reloaded or when their dependencies changes.

Another tool provided by the software suite is the Portal. This tool have two different profile: "user" and "administrator". The administrator profile allows users to install and enable business processes; display data of opened and archived cases; show detailed information on pending, failed and done tasks; setup users, groups and roles of the business organization. Furthermore, this profile permits users to define web applications combining different software objects that can be uploaded as "Resources", such as: custom pages, forms, layouts, graphical themes and possible REST API extensions. The user profile, instead, has operative functions. With this profile a user can view: (1) all the business processes that he/she could manually start; (2) a list of open cases; (3) detailed information about each case, such as the case ID, the process name, the stating date, the name of the actor that has started the case and if there are tasks that are waiting to be carried out; (4) a list of tasks that can be classified as "To do", if not still assigned to a specific actor, or as "My tasks" or "Done Tasks".

Finally, the Bonita BPM platform is completed by a business processes engine and a set of additional software tools. The engine has the responsibility to execute the business processes installed by the portal. Apache Tomcat, and the database engine H2, are instead the tools that allow software applications developed in a "local host" setup to be tested.

3.6. Web Technology Adopted

The O3-Farm application has been developed and tested using a PC and the following versions of software tools: Bonita BPM Community Edition 7.5.2. (Bonitasoft SA, Grenoble, France); Apache Tomcat 7.0 (Apache Software Foundation, Wakefield, MA, USA); H2 1.3.170; Java 8 (Oracle Corporation, Redwood City, CA, USA); Internet Explorer 11 (Microsoft, Redmond, WA, USA); Google Chrome 6.0 (Google Inc., Mountain View, CA, USA).

4. Validation

In order to validate the O3-Farm application from a user perspective, five professional farmers were involved in the study. To each of them, the main functionalities of the old and new version of the software application for the milk traceability, were explained. After a training period, they were asked to carry out a list of operations, such as: to insert a defined set of data or to perform a particular search in the archive. These operations were performed by users in both systems, in order to allow them to make comparisons between the two software applications. However, each operation was performed in both systems, alternating the order of the system used (i.e., first operation performed: before, in the old version of the software application, and after, in the O3-Farm application; second operation performed: before, in the O3-Farm application, and after, in the old version of the software application; for the remaining operations, the same alternation, as for the operations one and two, was followed). The complete list of operations performed by each user is reported below:

1. Add a new cow to the farm herd.
2. Add a new group of cows in the farm herd.
3. Add a new ration for a specific group of cows.
4. Add a new field to the farm.
5. Add a new supply.
6. Add a new warehouse management related to a supply.
7. Add a set of milking data for ten cows of the farm herd.
8. Retrieve milking data of a specific cow.
9. Retrieve milking data of a group of cows that have filled a specific can of milk.
10. Retrieve ration data of a specific group of cows.

The above reported operations were performed separately by each user. At the end of the testing phase, his/her perspective was recorded through a user survey. The questionnaire consisted of three sections to evaluate contents, quality and efficiency. Each answer to the questionnaire was rated as "disagree", "neutral" or "agree". Answers to the questionnaire are reported in Table 1.

As reported in Table 1, results show that the majority of the users agreed that the O3-Farm application provides all necessary data to trace milk yields (60% of the testers referring to farming data and 80% referring to milking data). As a consequence, it could implement the traceability of milk in the production cycle evaluated (for the 80% of the testers). Also data entry and data retrieval improved permitting a better traceability of milk yields when compared with the software application previously developed (for the 80% of the testers). At the same time, the O3-Farm application showed some new features that could bring an increase of the efficiency of the traceability of milk yields. In detail, possible advantages identified by users were: (1) the organization of farm activities as workflows (80% of the testers); (2) farm workflows that could be performed by different users (100% of the testers), with a data entry that could be carried out in different times and from any part of the farm (80% of the testers); (3) the availability of data that could be potentially shared with other partners of the production chain (60% of the testers).

Table 1. Results of the survey performed at the end of a testing phase of the O3-Farm application carried out on a group of five professional farmers.

Questions [a]	Disagree	Neutral	Agree
	n (%)	*n* (%)	*n* (%)
1. Content			
How much do you agree with the following statements:			
• The O3-Farm application provides clear information.	0 (0%)	1 (20%)	4 (80%)
• The O3-Farm application provides sufficient information.	0 (0%)	0 (0%)	5 (100%)
• The O3-Farm application provides all necessary farming data in order to trace the milk yields in the production cycle evaluated.	0 (0%)	2 (40%)	3 (60%)
• The O3-Farm application provides all necessary milking data in order to trace the milk yields in the production cycle evaluated.	0 (0%)	1 (20%)	4 (80%)
• All considered, the O3-Farm application could implement the traceability of milk yields in the production cycle evaluated.	0 (0%)	1 (20%)	4 (80%)
2. Quality			
How much do you agree with the following statements:			
• The O3-Farm application improves the data entry because many fields, of the forms that must be filled, contain data that can be selected through a window selection box.	1 (20%)	1 (20%)	3 (60%)
• The O3-Farm application improves the data entry and retrieval, because each group, of the farm organization, has a different access to web pages for the data entry and retrieval.	0 (0%)	2 (40%)	3 (60%)
• The O3-Farm application improves the availability of farming data because data entry is facilitated and data could be accessible from any part of the farm, using also portable devices.	0 (0%)	1 (20%)	4 (80%)
• The O3-Farm application improves the availability of milking data because data entry is facilitated and data could be accessible from any part of the farm, using also portable devices.	0 (0%)	1 (20%)	4 (80%)
• All considered, the O3-Farm application could improve the data entry and retrieval, permitting a better traceability of milk yields in the production cycle evaluated.	0 (0%)	1 (20%)	4 (80%)
3. Efficiency			
How much do you agree with the following statements:			
• The O3-Farm application increases the efficiency of the traceability of milk because farm activities could be organized as workflows.	0 (0%)	1 (20%)	4 (80%)
• The O3-Farm application increases the efficiency of the traceability of milk because farm workflows could be performed by different users.	0 (0%)	0 (0%)	5 (100%)
• The O3-Farm application increases the efficiency of the traceability of milk because data entry could be performed by different users, in different times and from any part of the farm.	0 (0%)	1 (20%)	4 (80%)
• The O3-Farm application increases the efficiency of the traceability of milk because data could be potentially shared with other partners of the production chain.	0 (0%)	2 (40%)	3 (60%)
• All considered, the O3-Farm application could increase the efficiency of the traceability of milk yields for the production cycle evaluated.	0 (0%)	1 (20%)	4 (80%)

[a] Note for the testers: answer to the questions, related to the O3-Farm application, taking as a reference the old version of the same software application for the traceability of milk.

5. Discussion

The O3-Farm was able to provide all features of the database application previously developed. It permitted users to store all required data and to retrieve them, when necessary. As a result, it allowed them to reach a complete traceability of milk in a context as described by the use-case adopted for this test.

At the same time, the O3-Farm application provided some new features that increased its value if compared with the database application previously developed. It was a "web oriented" software application, accessible by the internet network—with a common browser—from any part of the farm. It has a "responsive" graphical theme that allowed the use of any kind of internet device, also mobile. Data entry was shared between different groups of users, whose roles and responsibilities were well defined and organized in farm workflows. In this way, each task that involved more than one user could be done in different times, and in parallel when more instances of the same process were activated. As a result, a better efficiency of farm management was reached. Furthermore, data sharing was increased. This could permit a better integration of software applications, and control of the whole production chain, if other actors (of the production chain) would add or use data collected [25].

In the development of the O3-Farm application, the BPM approach was followed. This paradigm allows users to split the functional layer from the implementation layer. This can improve the design of processes, and their organization as services, in order to increase the: integration, interoperability, and scalability of software applications. Nevertheless, as seen in this preliminary evaluation, the software development results in being more complex if compared to the development of a common database application. Every web page must be coded without supposing it to have an available server session. If a web page has the target to collect some data for a process, it must retrieve all necessary data in terms of: task ID, case ID, and/or process ID. This have to be done by a sequence of calls to the processes engine and to the existing instances (and variables) of a specific process. As a consequence, the coding of the web page results in being more complex and expensive in terms of programming time. Therefore, when this development approach is used, it is important to reach the right balance between the required development costs and the context of the software application that has to be developed. Nevertheless, it is clear that when this methodology is adopted, the final product is always more professional and technically robust.

However, the adoption of the BPM approach, in the development of FMIS, could facilitate the implementation of the FSE. This model has the target to make farm management systems more collaborative, scalable and also to identify a technical infrastructure where software applications, and/or services, could be used and shared. A way to reach this target is to improve the integration of software applications and make agricultural data more sharable. This result could be reached through a standardization of farm workflows, and farm data, by the collaboration of technical stakeholders, and international organizations, as done in similar contexts. For example, in order to reach a better integration of software applications for medical hospitals, different standards have been developed (Digital Imaging and Communications in Medicine, DICOM and Health Level Seven, HL7—[26,27]) and "technical frameworks", that describe hospital workflows, have been written (Integrating the Healthcare Enterprise, IHE [28]). These technical frameworks identify a subset of functional components of the healthcare enterprise, called "IHE Actors" [15]. The frameworks specifies their interactions, in terms of a set of coordinated messages that can be sent or received on the basis of the standards already in use (i.e., DICOM and HL7). When specific messages are sent or received by an actor, it performs a transaction. More transactions can be organized in "integration profiles". Each integration profile describes real-world scenarios or specific sets of capabilities of integrated systems [15]. When a software house has a new product that implements one or more actors, it can test and declare its compatibility with the corresponding Technical Framework in order to prove its possible integration with other software applications. In the same time, when a hospital want to buy a new product, it can be sure that the new product will collaborate with the others software applications, already in use in the hospital, only considering its declaration of compatibility. In this scenario, the BPM approach could be a valid tool to reach a similar result. It could allow the description of a farm workflow, under a functional and technical point of view, through the use of a BPMN. Furthermore, using a software suite for the implementation of BPM diagrams, it could be a useful way for the development of software applications for the farm management, as suggested by the outcomes obtained in this preliminary evaluation. Thus, it could be considered as a way to promote a FSE model, considering its ability to use new processes, and/or services, that could be developed, and sold, by different software producers.

6. Conclusions

The software application developed (O3-Farm) was able to perform all tasks of the database application previously developed for milk traceability. At the same time, it provided some new features that increased its usability, portability, efficiency; and its chance to be more integrated with other possible software applications through a better sharing of agricultural data. These positive outcomes suggest that the BPM approach, and the use of a software suite based on this paradigm, could be a valid way to develop FMIS. In addition, a good ability to deploy processes, and in the use of software services, was shown by this technology. This positive result suggests that its use, in the development of FMIS, could be also a way to reach the scopes of FSE models.

Author Contributions: Individual contributions of each author were as follows: Conceptualization, M.Z.; Software, M.Z. and M.R.P.; Validation, M.Z. and M.R.P.; Formal analysis, M.Z.; Writing the Original Draft, the Review and Editing version, M.Z.; Supervision, M.Z.

Funding: This research received no external funding and the APC was funded by authors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fountas, S.; Carli, G.; Sørensen, C.G.; Tsiropoulos, Z.; Cavalaris, C.; Vatsanidou, A.; Liakos, B.; Canavari, M.; Wiebensohn, J.; Tisserye, B. Farm management information systems: Current situation and future perspectives. *Comput. Electron. Agric.* **2015**, *115*, 40–50. [CrossRef]
2. Jones, J.W.; Antle, J.M.; Basso, B.; Boote, K.J.; Conant, R.T.; Foster, I.; Godfray, H.C.J.; Herrero, M.; Howitt, R.E.; Janssen, S.; et al. Toward a new generation of agricultural system data, models, and knowledge products: State of agricultural systems science. *Agric. Syst.* **2016**, *155*, 269–288. [CrossRef] [PubMed]
3. Lewis, T. Evolution of farm management information systems. *Comput. Electron. Agric.* **1998**, *19*, 233–248. [CrossRef]
4. Kaloxylos, A.; Groumas, A.; Sarris, V.; Katsikas, L.; Magdalinos, P.; Antoniou, E.; Politopoulou, Z.; Wolfert, S.; Brewster, C.; Eigenmann, R.; et al. A cloud-based farm management system: Architecture and implementation. *Comput. Electron. Agric.* **2014**, *100*, 168–179. [CrossRef]
5. Hori, M.; Kawashima, E.; Yamazaki, T. Application of Cloud computing to agriculture and prospects in other fields. *Fujitsu Sci. Tech. J.* **2010**, *46*, 446–454.
6. Paraforos, D.S.; Vassiliadis, V.; Kortenbruck, D.; Stamkopoulos, K.; Ziogas, V.; Sapounas, A.A.; Griepentrog, H.W. A farm management information system using future internet technologies. *IFAC-PapersOnLine* **2016**, *49*, 324–329. [CrossRef]
7. Fountas, S.; Wulfsohn, D.; Blackmore, B.S.S.; Jacobsen, H.L.L.; Pedersen, S.M.M. A model of decision-making and information flows for information-intensive agriculture. *Agric. Syst.* **2006**, *87*, 192–210. [CrossRef]
8. Lewis, K.A.; Bardon, K.S. A computer-based informal environmental management system for agriculture. *Environ. Model. Softw.* **1998**, *44*, 1–24. [CrossRef]
9. Nuthall, P.L. Expert systems for animal feeding management Part II: Falrmers' attitudes. *Comput. Electron. Agric.* **1996**, *14*, 23–41. [CrossRef]
10. Sørensen, C.G.; Bochtis, D.D. Conceptual model of fleet management in agriculture. *Biosyst. Eng.* **2010**, *105*, 41–50. [CrossRef]
11. Kruize, J.W.; Wolfert, J.; Scholten, H.; Verdouw, C.N.; Kassahun, A.; Beulens, A.J.M. A reference architecture for Farm Software Ecosystems. *Comput. Electron. Agric.* **2016**, *125*, 12–28. [CrossRef]

12. Kaloxylos, A.; Eigenmann, R.; Teye, F.; Politopoulou, Z.; Wolfert, S.; Shrank, C.; Dillinger, M.; Lampropoulou, I.; Antoniou, E.; Pesonen, L.; et al. Farm management systems and the Future Internet era. *Comput. Electron. Agric.* **2012**, *89*, 130–144. [CrossRef]
13. Nikkilä, R.; Seilonen, I.; Koskinen, K. Software architecture for farm management information systems in precision agriculture. *Comput. Electron. Agric.* **2010**, *70*, 328–336. [CrossRef]
14. Wolfert, S.; Ge, L.; Verdouw, C.; Bogaardt, M.J. Big data in smart farming? A review. *Agric. Syst.* **2017**, *153*, 69–80. [CrossRef]
15. Zaninelli, M.; Tangorra, F.M.M.; Castano, S.; Ferrara, A.; Ferro, E.; Brambilla, P.G.G.; Faverzani, S.; Chinosi, S.; Scarpa, P.; Di Giancamillo, M.; et al. The O3-Vet project: A veterinary electronic patient record based on the web technology and the ADT-IHE actor for veterinary hospitals. *Comput. Methods Programs Biomed.* **2007**, *87*, 68–77. [CrossRef] [PubMed]
16. Zaninelli, M.; Campagnoli, A.; Reyes, M.; Rojas, V. The O3-Vet project: Integration of a standard nomenclature of clinical terms in a veterinary electronic medical record for veterinary hospitals. *Comput. Methods Programs Biomed.* **2012**, *108*, 1–13. [CrossRef] [PubMed]
17. Murakami, E.; Saraiva, A.M.; Ribeiro, L.C.M.; Cugnasca, C.E.; Hirakawa, A.R.; Correa, P.L.P. An infrastructure for the development of distributed service-oriented information systems for precision agriculture. *Comput. Electron. Agric.* **2007**, *58*, 37–48. [CrossRef]
18. Sørensen, C.G.; Fountas, S.; Nash, E.; Pesonen, L.; Bochtis, D.; Pedersen, S.M.; Basso, B.; Blackmore, S.B. Conceptual model of a future farm management information system. *Comput. Electron. Agric.* **2010**, *72*, 37–47. [CrossRef]
19. Wolfert, J.; Verdouw, C.N.; Verloop, C.M.; Beulens, A.J.M. Organizing information integration in agri-food-A method based on a service-oriented architecture and living lab approach. *Comput. Electron. Agric.* **2010**, *70*, 389–405. [CrossRef]
20. Xu, L. Da Enterprise systems: State-of-the-art and future trends. *IEEE Trans. Ind. Inform.* **2011**, *7*, 630–640. [CrossRef]
21. La Rosa, M.; Hofstede, A.H.M.; Wohed, P. Managing process model complexity via concrete syntax. *IEEE Trans. Ind. Inform.* **2011**, *7*, 255–265. [CrossRef]
22. Bai, C.; Sarkis, J. A grey-based DEMATEL model for evaluating business process management critical success factors. *Int. J. Prod. Econ.* **2013**, *146*, 281–292. [CrossRef]
23. Tangorra, F.M.; Zaninelli, M.; De Santis, C. Development of HW and SW solutions for milk traceability. In *Computers in Agriculture and Natural Resources—Proceedings of the 4th World Congress, Orlando, FL, USA, 24–26 July 2006*; Zazueta, F., Ed.; American Society of Agricultural and Biological Engineers: St. Joseph, MI, USA, 2006; pp. 475–480.
24. Zaninelli, M.; Rossi, L.; Costa, A.; Tangorra, F.M.; Agazzi, A.; Savoini, G. Monitoring of goats' health status by on-line analysis of milk electrical conductivity. *Large Anim. Rev.* **2015**, *21*, 81–86.
25. Niederhauser, N.; Oberthür, T.; Kattnig, S.; Cock, J. Information and its management for differentiation of agricultural products: The example of specialty coffee. *Comput. Electron. Agric.* **2008**, *61*, 241–253. [CrossRef]
26. Digital Imaging and Communications in Medicine. *Ps 3.1—Introduction and Overview*; National Electrical Manufacturers Association: Arlington, VA, USA, 2017.
27. Health Level Seven. *Health Informatics—HL7 Version 3—Reference Information Model—Release 4*, 2nd ed.; Health Level Seven International: Ann Arbor, MI, USA, 2014.
28. Integrating the Healthcare Enterprise. *IHE Technical Frameworks—General Introduction*; IHE International: Oak Brook, IL, USA, 2014.