Ref: C0274

agriOpenLink: Semantic Services for Adaptive Processes in Livestock Farming

S. Dana K. Tomic, Domagoj Drenjanac, and Goran Lazendic, Forschungszentrum Telekommunikation Wien, Donau City Straße 1, 1220 Wien, Austria

Sandra Hörmann, Franz Handler, Josephinum Research, Rottenhauser Straße 1, 3250 Wieselburg, Austria

Wilfried Wöber, Klemens Schulmeister, University of Natural Resources and Life Sciences, Gregor-Mendel-Straße 33, 1180 Wien, Austria

Marcel Otte, Wolfgang Auer, MKW electronics GmbH, Jutogasse 3, 4675 Weibern, Austria

Abstract

Intelligent machinery and information and communication technologies (ICT) are crucial enablers for precision arable farming and automatized livestock husbandry. Technologies such as advanced robotics, sensor technology, and localization systems are being deployed to increase efficiency and reduce production costs based on efficient use of resources. Robots and sensors are vital sources of data that directly assist farmers in their work. For example, with the help of a localization application based on active ear tags a farmer can quickly identify any particular animal even within a large herd. On the other hand, farmers may have too many equipment to handle and operate, and they often need to manually input the very same data in different machines through different user interfaces (UI), and account for, and act upon data these machines offer them mostly for manual inspection. Accordingly, there is a growing need for a farm management platform that could provide a unified interface towards variety of different systems, as well as combine data from different sources and provide decision support to the farmer**.** This need is also motivated by the common awareness that the more **data sources get integrated** within the farm management decision support systems, the more optimal and sophisticated become the decisions or recommendations that such systems can offer. For example, in a dairy farming scenario an integrated system of a milking robot and an animal localization system with movement and rumination monitoring, can improve quality of collected milk, as well as the wellbeing and health of the animals. As the cost of designing decision support systems is generally high, it is essential that their architecture is open and future-proof to support integration of new innovative technologies as they emerge. Unfortunately, this vision of an open-platform is still far from reality. Approaches that are prominent today are based on single-vendor island solutions where new products can be integrated within the running system only if they are of the same vendor. On the other hand, to allow for a broader innovation adoption efficient plug-and-play of heterogeneous systems, processes and data is urgently needed. This paper presents a novel approach towards open interfaces and process models in agricultural production that is being developed within the project agriOpenLink. Our methodology is based on service and semantic technologies. To integrate heterogeneous devices we develop a common platform - a plugin server - and tools for plugin development. Each plugin publishes a number of semantic web services which offer data for integration. They run on the plugin server, and are formally described using the common ontology. The paper discusses our goals, methodology and results achieved so far, and illustrates the solution in the use case of dairy farming and livestock husbandry.

Keywords: process optimization, semantic services, ontology, dairy farming

1. Rational for Research and Goals

1.1 Challenges of Information Intensive Agriculture

Agricultural production is a field of constant innovations that are gradually transforming this work-intensive sector into a knowledge-intensive one. The drivers of this transformation come from two different intertwined areas – advanced machinery and data management.

Advanced agriculture machinery, such as automatically guided tractors or milking robots, combine robotic technology and a variety of smart sensors, and offer means to reduce the extent of costly manual activities. In addition, by automating the best practice workflows, these machines vastly improve reliability and quality of processes and products. Moreover, the novel equipment is a source of a high variety of data that can be captured, gathered, processed and presented to users. The availability of new data raises the need for data management technologies and methods that enable transformation and aggregation of data into information and into the knowledge.

The need for flexible, on-the-fly integration of data from different sources is becoming more and more prominent due to the fact that innovative solutions, such as e.g. new sensors, appear independently of each other, and a new value can only be created if they can seamlessly integrate together. The data created by different equipment essentially belong to a common production context, e.g., the milking, feeding and animal localization processes all belong to the dairy farming context. Accordingly, different equipment should be able to contribute with its own data to the common information context (e.g., the milking robot with the data about the milking efficiency, the feeding robot with the data about the feed mix and quality, and the localization system with the data about the mobility of animals) as well as to benefit from the common context data. In addition most of the equipment need to access some data from the common context, e.g., the identification data of the dairy cows, such as the unique number, name, RFID number, active ear tag number. Without the unifying platform for data integration, the user is expected to manually elicit / enter such data at all the related devices.

Currently, the challenge of integrating different heterogeneous equipment and different data is not solved. High level of integration is available only in single-vendor solutions. The pace of standardizing common data models is slow, and it focuses on syntax rather than semantics. Particularly in the dairy farming sector the standards are far from complete and are not being actively used (ISOagriNET). Moreover, the process of standardization, the way the standards are developed and maintained do not allow for fast changes.

On the other hand, system designers and integrators who are currently developing farm management systems for information intensive agriculture, can select from a broad variety of mature data management technologies. In doing so it is important to understand constraints of different existing approaches as related to data acquisition, processing and integration. The latest solutions that address challenges of system flexibility and heterogeneity similar to those of the agricultural sector, heavily build on the web service technology, and increasingly on semantic web service technology. Semantic technology offers distinctive benefits in systems that have to dynamically change and flexibly (re)configure. They offer high re-usability, and the cost of necessary extensions of system re-configuration is far lower than in the conventional systems.

1.2 The Goals of agriOpenLink

In addressing integration challenges in agricultural production, the agriOpenLink project heavily builds on the semantic and service technology, and pursues following goals:

(1) Design of a domain ontology to capture all important domain concepts both for the description and classification purposes. Such model is flexibly extensible with expert knowledge

codified in classes describing specific states/situations that are used for classification of instance in the process of the state detection (diagnoses).
(2) Implementation of tools for a collaborative maintenance of the ontology / knowledgebase.
(3) Design of a "plugin-based" infrastructure and necessary tools for flexible integration of any specific agricultural equipment within a platform for agricultural process optimization, as well as for publishing of equipment functionality by means of semantic services. In this context the open source approach has been taken: the process of plugin creation is supported by a plugin skeleton (extensible and reusable open source code) which implements common plugin functions.
(4) Design of a diagnosis infrastructure based on service discovery, service flexible chaining and reasoning on a knowledgebase. In this context a process is modelled with a chain of diagnosis steps (situation detection), and rule-triggered actions. This process model is a basis for flexible production process management and optimization.

While process management and optimization is the ultimate goal of agriOpenLink, in this paper we present results related to goals (1), (3), and (4), focusing on how semantic technology is used to create a common context formally specified with an ontology, how the semantic web service technology is used to create a platform that supports flexible integration of devices in the dairy farming production environment, as well as how diagnosis based on service orchestration and ontology and rule-based reasoning is realized.

1.3 Paper Focus and Organization

After a short introduction in Section 1, Section 2 reviews the ontology-based knowledge formalization (Section 2.1), and semantic web service based functional modelling (Section 2.2). Section 3 focuses on the results: a dairy farming ontology is briefly reviewed in Section 3.1, and the plugin concept and Section 3.2. A diagnosis workflow is illustrated in Section 3.4, as an orchestration of exemplary services. Section 4 concludes the paper.

2. Materials and Methods

2.1 Semantics in Knowledge Formalization

The first step towards flexible integration of data is a creation of a common context – a common knowledge model. Semantic Web approach for formalizing knowledge models and using it in a machine readable form includes knowledge description, rule standards, query standards, and variety of tools for storing semantic information (triplestore / knowledgebase) and reasoning on it (Fensel et al., 2011), (W3C, SW, 2014). Using web ontology language (OWL) (W3C OWL, 2013) for knowledge formalization is based on the description logic. An ontology specifies concepts in form of classes, data properties, and object properties. These are used to describe the entities in the system as instances of particular types (particular classes) with specific data properties, and being linked with other instances by means of object properties (logical relationships) of different types. The classes defined in the ontology can be of two types: the primitive and the defined type. The primitive type classes specify the basic hierarchy of concepts. The instances of these classes can have specific identifying properties, both data properties and object properties. These are sufficient properties, meaning that an instance of a specific type will have identifying properties. The defined classes, on the other hand, are bases for classification. These are defined with restrictions on specific properties, which specify necessary and sufficient conditions; instances that contain these properties and satisfy these conditions can be automatically classified as members of a specific class. These types are therefore not directly assigned to the instances; the membership to a defined class is inferred by reasoners, software tools that operate on the knowledgebase (Mishra & Kumar, 2011). In this way, the ontology and the knowledgebase are basis for inferring new knowledge through the process of reasoning. In addition to relationships specified within the ontology the knowledge can be further described in form of rules, by using the rule specification languages, such as Semantic Web Rule Language (SWRL, 2014). The rule languages introduce additional expressivity and are sometimes necessary. Extended expressivity of the knowledge description can be also achieved by using SPARQL (W3C SPARQL, 2013) specify constraints about the instances that satisfy the query. While semantic technology has already been used in agriculture (www.aims.fao.org), also including attempts to formalize dairy farming ontologies and models for process optimization in dairy farming (Athanasiadis, 2009), numerous challenges still exist. The innovation of the presented approach is focused on how ontology and semantic services are used to flexibly and dynamically create the common context on which on-the-fly diagnoses can be performed. The relevant methodology is reviewed in the next section.

2.2 Semantic Service Approach

Once the common knowledge model (ontology) is specified, the instances of primitive classes can be created and interlinked to create a knowledgebase. In the dairy farming scenario instances of the class Cow can be created based on the information inserted by a farmer in the process of animal registration. These instances can be associated with many valuable data that are currently contained within different systems. To extract these data from the devices and integrate them within a common context (as values of specific data properties), we pursue the Semantic Web service approach. With this approach, the data is accessed by means of a service interface, i.e., a SW service is a wrapper, and the data is accessed by invoking the service. Semantic Web Technology extends the Web Services technology, which are heavily used in different domains, in the Semantic Web design (Cardoso, 2007). The semantic extension extends the web service description with annotations – meta data based on an ontology ontology - so that services can be automatically found, chained together or matched against the semantically expressed triggers for service invocation. An efficient and simple semantic service model based on RESTfull Web service, the Semantic Automated Discovery and Integration (SADI) and its orchestration framework SHARE, are proposed and implemented in the area of the scientific workflows for bioengineering (Wilkinson, 2010). A SADI service consumes an instance or a set of instances of a specific Input Class and generates as a reply the instance(s) of the Output Class. A service can also use a Parameter Class for additional parameters. The business logic of the service translates Input to Output class in the stateless manner. For each SADI service that can be invoked there is an accompanying description in the service registry. This description allows the service invocation logic to find a service with a matching Input or Output classes, in particular to find services that need to be chained together to transform the instance of a particular input class (the input class of the first service in the chain) into the instance of a particular output class (the output class of the last service in the chain). For this service the SADI framework also integrates the SHARE framework for flexibly chaining services based on the semantic query that searches for instances of the specific type. Taking into account what type of instances is initially available in the knowledgebase, what type of instances is searched for, and the description of all available services, the SHARE engine selects and invokes the transformation chain. The SADI and SHARE framework are enabling technology for the agriOpenLink diagnoses process over the dynamically created common context.

2.3 Architecture

The architecture of the agriOpenLink platform follows the model-centered and layered paradigm. The lowest layer is the **Plugin Layer** (Section 3.2). All devices integrate within the platform by means of plugins controlled by the plugin server running at the farm computer. The layer above the Plugin Layer is the **Semantic Service Layer** (Section 3.2). Each plugin publishes semantic services as wrappers for its data access or control functionalities. Semantic services use classes from the common ontology. The next higher layer is the **Ontology and Context Interpretation Layer** (Section 3.1). This layer specifies the model of the common context including the domain ontology, rules and possible queries. At the top of this layer is the **Diagnose Layer** (Section 3.3) with the functionality of the diagnose engine which answers the diagnose queries by creating and executing flexible workflows of service invocations combined with the ontology-based and rule-based reasoning. The diagnose query comes from the next upper layer, **the Process and Application Layer,** in which the functionality of the process modelling engine is situated that supports creation of flexible process workflows as combination of diagnose queries and actions based on the results of queries.

3. Results and Discussions

The major results described in this paper include the dairy farming ontology, the plugin concept, and the diagnose approach based on orchestration of service invocations and reasoning.

3.1 Dairy Farming Ontology

The goals of developing the dairy farming domain ontology in agriOpenLink are as follows: (1) to create a relatively stable hierarchy of primitive classes relevant for the dairy farming domain and to use them to instantiate the knowledgebase (2) to define a set of data and object properties that can model different data available in the dairy farming system, and to use them to decorate instances of primitive classes, as well as (3) to define an extensible set of defined classes to codify expert knowledge supporting the diagnose model to be realized by the system, by defining restrictions on different properties, and (4) to define a set of defined classes to be used as Input and Output classes of semantic SADI services.

A first version of ontology that captures all primitive concepts of the sector has been created. For the description of the domain knowledge we use the Web Ontology Language (OWL), with its expressive syntax, and the editing framework Protégé (protege.stanford.edu). The primitive type classes that specify the basic hierarchy of concepts or types include classes such as Animal, Organization, Location, etc. In defining the properties we are taking into account existing data models including ISOagriNet (ISO, 2012), and the specification of information that can be extracted from the milking robot and the localization system. To specify defined classes on the other hand we have to codify specific expert knowledge. These are used for classification. As the focus of the project in this stage is more on the enabling technology for easy creation of any defined class, we are currently only identifying representative diagnosis queries related illustrative defined classes.

3.2 The Plugin Concept

The plugin concept has as a goal flexible integration of any particular equipment into a common platform. By plugging-in into the platform, a device can offer access to its functionality – data assess or control - via appropriate SADI services. These services are specified with Input, Output and parameter classes, and are registered in the service registry.

The agriOpenLink plugin concept has several components. The central component is the plugin REST and Registration Management (RRM) server which runs at the farm computer. RRM is a software component implemented in C++ that unifies the functionality of the REST server with dynamic capability to load and configure individual plugins, as well as service registration capability. The farm server is an industrial computer with communication interfaces suitable to connect devices within the farming environment. When the RRM starts it loads all the available plugins, and register services within the registry. The business logic of each individual plugin implements the communication with the related device at some appropriate interface, and extraction and processing of the relevant data from the device. For this purpose each plugin can create an instance of a relational database running on the Farm Computer in which a plugin can organize its data according to any internal scheme. The business logic of a particular plugin service is responsible for translating the instance of the Input Class (Input.owl file received through the HTTP POST call) with its identifying properties, into the data that the plugin further uses to either directly generate the response or query the internal database. The results have then to be translated into the properties with which the service decorates the instance of the Output Class (output.owl file). The services can be so defined as to support advanced access control mechanisms. It is important to notice that the device can specify and offer many different services, with any granularity of data. The semantic services act as interfaces towards monitoring or control functionality of the equipment, they be automatically discovered and flexibly orchestrated creating complex processes. The plugin framework is illustrated in Figure 1.

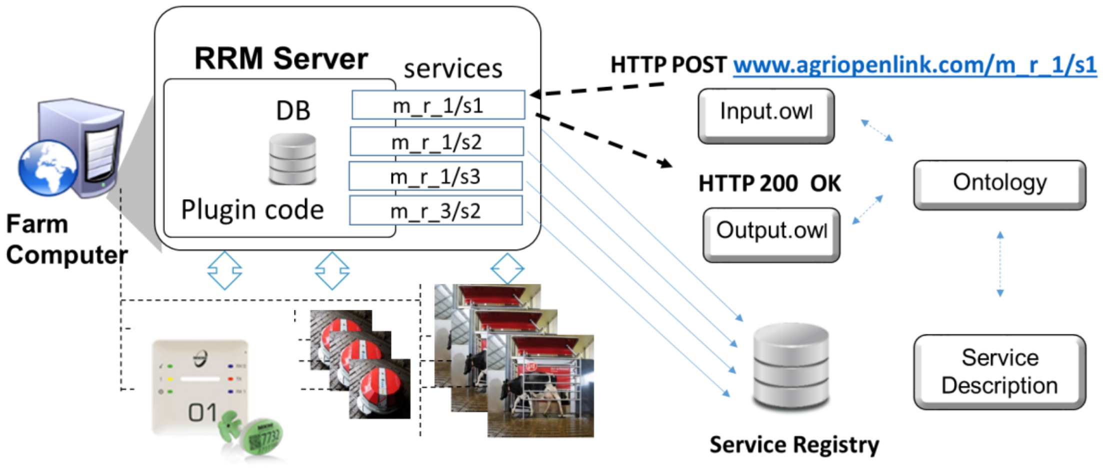

Figure 1: agriOpenLink Plugin Concept

3.3 The Diagnosis Workflow

Once that the RRM-based integration platform is in place, the diagnosis actions can be performed by the diagnosis engine. A common context is a dynamic combination of the data existing in the knowledgebase, the data available via the plugin services accessing the functionality of the farm robots, sensors or other integrated systems, and the data or functionality accessible via any accessible external SADI service. The data in the knowledgebase are for example the instances describing the animals and the farm; these are entered into the system by the farmer and maintained via services for animal registration or de-registration, etc. All other data may not exist in the knowledgebase nor be linked together at the time of the diagnosis query; they are linked together during the diagnosis step, which is triggered by the query. The query specifies a search for instances of special classes, in general satisfying specific constraints. It translates into a question whether a situation that is related to existence of such instances has occurred (instances are present) or not (none instances found). Any specific process can be further formally described as a workflow of diagnosis steps and action steps. In this paper we only focus on the realization of the diagnosis steps, and the methodology for describing the process is out of the scope. The query that triggers a diagnosis step can be a simple SPARQL search query or a more complex SPARQL query, specified according to the syntax of the SPARQL language. As already mentioned the expert knowledge about situations that should be queried and resolved can be codified in form of defined classes, queries and rules. To illustrate how a query regarding a specific situation results in a series of service matching (search), service invocations for data gathering and reasoning, an example is constructed and presented here.
The diagnostic query is expressed in the natural language in the following way: Find the instances of all cows that had above average milk production within the last month and being under special medical attention.

The relevant class descriptions in the system are:
- The class *Cow* comprises all instances of the cows in the system.
- The class *HighYieldCow* is a defined class specified as equivalent to the class Cow with the restriction on the property hasYield (value of type decimal).
- The class *LowMobilityCow* is defined as equivalent to Cow with the restriction on the property hasmPerDay (meter per day, value restriction e.g. < 45m).

- The class *CowMedicalAttn* is specified as equivalent to Cow with restriction on sentAlarm (Boolean value =1).

The relevant services in the system are:
- The service **S_HighYieldCowSentAlarm** takes as an input an instance of HighYieldCow with restrictions on properties hasID and hasAlarm and decorates this instance with the property sentAlarm (values 0 or 1) if the value of the property hasAlarm is higher than 10. In the same time this service sends the email to the veterinarian who is responsible for this particular farm.
- The service of the milking robot **S_HighYieldCow** is defined as taking the instance(s) of the class Cow restricted on the property hasID (value of type integer), and a parameter class specifying the observation period(start date, end date), and returning the instance(s) of the class HighYieldCow.
- The service of the localization system **S_MobilityCow** is defined as taking the instance(s) of the class Cow restricted on the property hasID, together with the parameter specifying the observation period and returning the input instance(s) decorated with the property hasmPerDay (value decimal) representing the length of the movements per day.
- The service **S_CowHasAlarmMobility** is specified as taking an instance of class Cow restricted on hasID and hasmPerDay and decorating this instance with property hasAlarm (value integer) the identification of alarm.

Based on the definition of classes we see that the query can be translated into a query for all instances of the HighYieldCowMedicalAttn.

After the query is sent to the Diagnosis Engine following inferences are made:
1. No instances of HighYieldCowMedicalAttn exist in the knowledgebase.
2. In order to generate an instance of HighYieldCowMedicalAttn the service S_CowSentAlarm shall be invoked. This class requires instances of the class HighYieldCow restricted on hasAlarm. Here the diagnosis engine has to find which service can generate a class with a restriction on hasAlarm property, and also how to generate HighYieldCow instances.
3. The matching service with hasAlarm as an output-defining property S_CowHasAlarmMobility is found by checking service descriptions. The input class is Cow restricted on hasID and hasmPerDay. Again it should be found what service specifies these properties in the definition of the output class restrictions.
4. The matching class S_MobilityCow is found as decorating the input instance with hasmPerDay. This class has an input class equivalent to Cow restricted on hasID.
5. The service to generate HighYieldCow from Cow is S_HighYieldCow.
6. Instances of class Cow exist in the knowledgebase.

In summary, the diagnosis engine identified that to resolve this query the instances of Cow class will be sent to S_HighYieldCow, then the output of it to S_MobilityCow, then the output of it to the reasoner to find out LowMobilityCow, then this output will be sent to S_CowHasAlarmMobility, then the output of it to S_CowSentAlarm, which output is the output to the query. During this query nothing changed within the knowledgebase. However, we can also specify as a part of the business logic that the service S_CowSentAlarm not only sends alarms but also decorates the corresponding instances in the knowledgebase with the property alarmOpen (value 1). In this case, this property value will stay valid until the visit of the veterinarian. During the visit, for all attended cows this property could be reset to zero.

4. Conclusions

Today, to remain competitive, farmers need to continuously improve efficiency and productivity of their production processes. This requires ability to embrace innovation as it emerges, that is, to flexibly integrate innovative efficiency-enhancing devices within their existing systems, and to use their data in an integrated form. From a technological perspective semantic, service-oriented, and Web technologies offer new possibilities to support the integration need, enabling the design of extensible systems which can reap the benefits of steady advancements in the fields of sensors, robotics, embedded processing, and information management technology. In this paper we presented the concepts and the realization of a data integration platform that is being developed within the project agriOpenLink. The solution currently under development integrates a number of different tools: (1) a tool for writing plugins for a new equipment that shall be integrated in the system, (2) for defining and maintaining the common domain model, and codifying the expert diagnosis knowledge by means of defined classes in the ontology, SPARQL queries, and rules, and (3) for designing semantic services of the plugins. As the first versions of the RRM server and the plugin design support tools are already developed we are starting to acquire experience in designing plugins, as well as to experiment with service orchestrations. For this purpose we are designing the examples of defined classes, rules and queries which will be relevant in our demonstration environment. The next steps include the design of the diagnosis engine and the process modelling tools.

Acknowledgements

The work presented here is partially funded by the Austrian Research Agency (FFG) within the project agriOpenLink (research fund 898398).

5. References

Athanasiadis IN, Rizzoli AE, Janssen S, Andersen E, Villa F. Ontology for Seamless Integration of Agricultural Data and Models. In proceeding of: Metadata and Semantic Research - Third International Conference MTSR 2009; Milan, Italy, October 1-2, 2009, 282-293.

Cardoso, J. Semantic Web Services: Theory, Tools and Applications. Information Science Reference, New York, NY, USA, ISBN: 978-1-59904-045-5, 2007.

Fensel, D, Foundations for the Web of Information and Services: A Review of 20 Years of Semantic Web Research, Springer, July 2011.

ISO17532: Stationary equipment for agriculture - Data communications network for livestock farming. Genf: Beuth Verlag,

Mishra, R. B. , Kumar, S., Semantic web reasoners and languages, Artificial Intelligence Review, April 2011, Volume 35, Issue 4, pp 339-368

Wilkinson, M.D., McCarthy, E.L., Vandervalk, B.P., Withers, D., Kawas, E.A., Samadian, S.: SADI, SHARE, and the in silico scientific method, BMC Bioinformatics(2010)7-7

World Wide Web Consortium, Semantic Web, http://www.w3.org/standards/semanticweb/, last accessed: December 24, 2013.

World Wide Web Consortium, SPARQL 1.1, http://www.w3.org/TR/sparql11-overview/ last accessed May 15, 2014.

World Wide Web Consortium, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission 21 May 2004

World Wide Web Consortium, OWL 2 Web Ontology Language Primer (Second Edition) W3C Recommendation 11 December 2012, http://www.w3.org/TR/2012/REC-owl2-primer-20121211, last accessed May 15, 2014