



Project no. **212117** Project acronym: **FUTUREFARM**
 Project title: **Integration of Farm Management Information Systems to support real-time management decisions and compliance of management standards**
 Instrument: **Collaborative project**
 Start date of project: **1st January 2008** Duration: **36 months**
 Thematic Priority: **THEME 2 FOOD, AGRICULTURE AND FISHERIES, AND BIOTECHNOLOGY**

Deliverable 4.3 Revision: Final
Specification of a Rules App to handle compliance assessment based on knowledge from repositories

Due date of deliverable: **31/12/2009** Actual submission date: **15/1/2010**

Work package 4: **Knowledge Management in the FMIS of Tomorrow**

Organisation name of lead beneficiary for this deliverable: **Rostock University**

Authors: **Edward Nash (UR), Raimo Nikkilä (TKK), Sascha Kluger (Agrocom), Kai Oetzel (Agrocom), Liisa Pesonen (MTT), Ilkka Seilonen (TKK), Jens Wiebensohn (UR)**

Accepted by Claus Sorensen, 7/2/2010

Accepted by Simon Blackmore, 23/2/2010

Project co-funded by the European Commission within the Seven Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Executive Summary

Previous deliverables from FutureFarm WP4 have defined an XML representation for agricultural management and production standards (D4.1.1) and two web-services, a catalogue and a rules server, allowing the definitions of the standards to be automatically retrieved via the Internet (D4.2). This deliverable provides the definition of a further component of the system which is being developed in order to manage knowledge of agricultural rules within Farm Management Information Systems. This component is responsible for the evaluation of adherence to rules and is given the name “Rules App”. The formal and detailed technical specification is given in an appendix to this document, which describes the component in an abstract form and how it interacts with other components in the system.

The specifications in D4.1.1 and D4.2 allow for the basic transfer of agricultural rules using a client-server architecture, without providing any mechanism for automated evaluation. The FMIS may therefore access knowledge and display it to the farmer. The purpose of the Rules App specification is to provide a uniform interface to enable the FMIS to firstly identify which concrete data items are required in order to evaluate compliance and secondly to pass that data to the Rules App in order to perform the evaluation.

The Rules App may be implemented as a software component integrated into the FMIS, as a separate component in a COM architecture or provided as a remote service via a web-service interface. The specification of the component interface is therefore technology-neutral, identifying only the operations, parameters and result types. As an initial concrete implementation specification, a concrete RESTful web-service interface specification based on the abstract specification is provided. This RESTful web-service will later be implemented within the FutureFarm project.

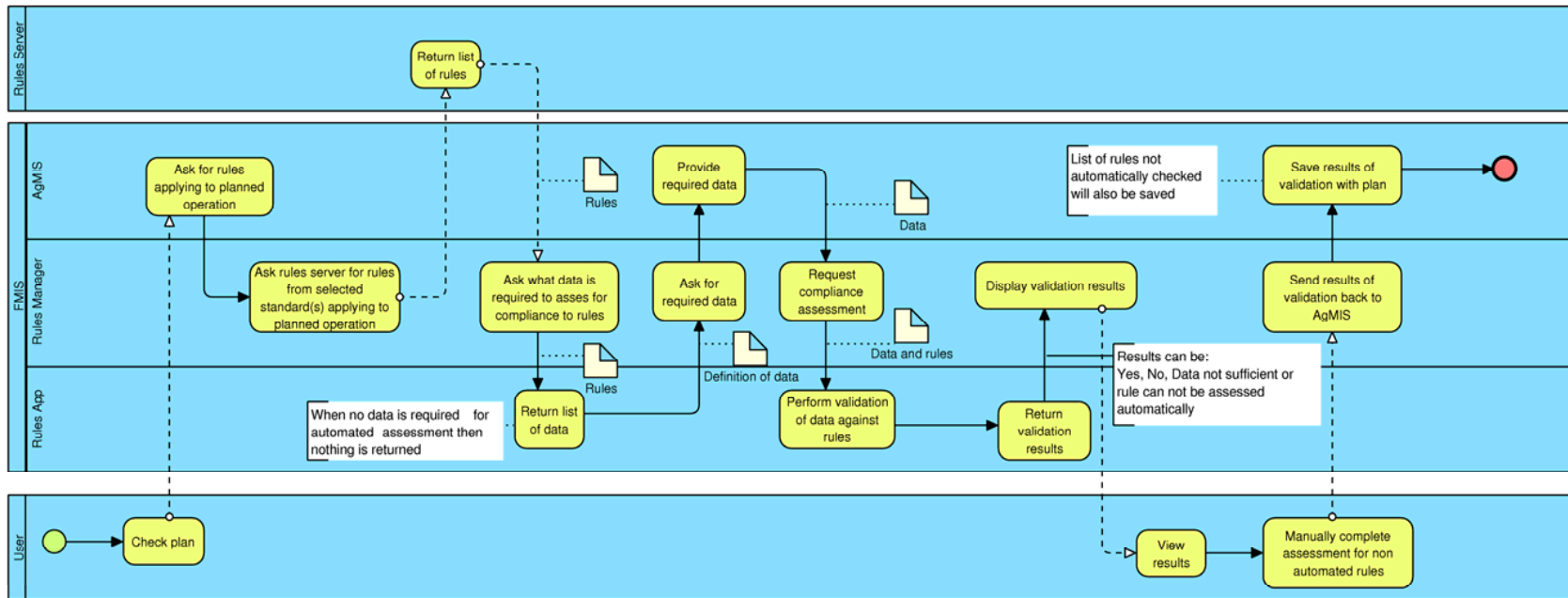


Figure 2. Expected interaction of other FMIS components with RulesApp

will now be considered in more detail. Note that further operations may also be specified for particular implementation forms, e.g. for web-service interfaces an operation to retrieve metadata is usually provided, that may not be necessary for a program library.

Identification of data required for evaluation of rules

Within the encoding of agricultural standards, each standard is considered as consisting of a number of discrete rules, each of which may be evaluated separately. The encoding of the rules identifies concepts used in defining the rules by means of an ontology, which may be specific to a particular standard, or a general ontology defining generic agricultural terms. In order to evaluate each rule, the concept from the ontology must be mapped to a specific data item available in the FMIS database, or available from some other source, and which is transferred in a format that the Rules App component can use. Depending on the implementation form of the Rules App, this may be an element from an XML-based data transfer format (e.g. agroXML), a particular programming class or data structure, or any other unambiguous definition of the data type required. Issues in implementing this operation are discussed in FutureFarm Deliverable 4.1.2.

Evaluation of compliance with rules

Since it is assumed that the Rules App is a stateless component, evaluation of compliance with rules requires fundamentally two parameter sets; the rules themselves and the data items identified as required for their evaluation. The actual evaluation of each rule is not straightforward, and is further complicated by the facts that the rules specified by agricultural standards are not all computable, or computable in a reasonable practical form, or the data supplied may not be sufficient. Four outcomes of a "successful" rule evaluation may therefore be identified (Table 1).

Outcome	Description
True	The rule evaluates to true with the given data (i.e. compliant to rule)
False	The rule evaluates to false with the given data (i.e. violation of rule)
Indeterminate due to data	The rule cannot be properly evaluated because the supplied data is insufficient to properly determine a result.
Indeterminate due to rule	The rule cannot be evaluated at all. This is because it is fundamentally flawed (incomputable) or this implementation of the Rules App cannot reliably evaluate it (e.g. due to missing models or other computational components).

Table 1. Possible outcomes of "successful" rule evaluation

For an indeterminate outcome, it is assumed that the Rules Manager component would provide a user interface to support the farmer in making a manual assessment of compliance.

Theoretically this operation could operate on an individual rule-by-rule basis, on whole standards, or on arbitrary sets of rules. It is suggested however, that the operation on arbitrary sets of rules is the best choice for two main reasons. Firstly, each individual data item may be needed for the evaluation of more than one rule, and allowing the rule-by-rule evaluation of arbitrary sets of rules therefore is likely to reduce the total amount of data retrieval and transfer required. Secondly, restricting the Rules App to evaluating

individual rules in isolation, rather than whole standards, means that the Rules App is a straightforward rules evaluator requiring no further logic for processing of interactions between rules (e.g. "requires some" as for GlobalGap "Minor Must" checkpoints). Since this latter evaluation is rather trivial, even a simple Rules Manager should be capable of achieving it, whilst the rules evaluator may in the best case be available as an existing software component from a domain other than agriculture, since this is effectively a generic function.

Implementation forms of the Rules App component

As can be seen in Figure 1, the Rules App component conceptually forms part of the "main" FMIS system – i.e. it is expected that it is part of the system that would actually be running on a PC on the farm, in contrast to the catalogues and rules servers which would be running on remote servers and accessed via web service interfaces. Nevertheless, the implementation of the Rules App must not be tightly coupled to the rest of the local FMIS. Although there are benefits to the Rules App being fully integrated with the rest of the FMIS, in particular in terms of access to data (i.e. if the Rules App has direct access to the FMIS database and live data sources, it may be able to automatically retrieve required or supplementary data), the actual implementation of a software engine for processing rules is not trivial, and it is considered that the majority of agricultural software companies would find this very difficult.

For this reason, the interface to the Rules App component is specified in an abstract form defining the operations, parameters and return values at a high level. From this abstract specification, concrete implementation specifications may be derived for individual program libraries or web service technologies. As the simplest form of web service interface, a RESTful (based on the REST principles proposed by Fielding, 2000) binding is proposed which uses the XML-based encoding of rules from D4.1.1 and any arbitrary data format (e.g. agroXML) for transfer of rules and data. Through implementing a proof-of-concept of this interface as a free software application, a ready-made component will be made available to agricultural software companies which they may in future incorporate into their products. It should be noted that a web service interface may also be accessed locally and not only over the internet, and so the implementation proposed in the FutureFarm project does not imply the deployment and ongoing maintenance of an internet-based rules evaluator.

Discussion

This document has presented the design criteria and rationale for a FMIS component for the evaluation of compliance to standards in terms of assessing data for its compliance with individual rules. The overall assessment of compliance to the entire standard must then be performed by the Rules Manager by aggregating the results from each individual rule.

The decision to define the Rules App as a separate component within the FMIS was made as it was realised that from a programming perspective this is probably the most complex part of the whole system, and having a loosely-coupled component which may then be integrated into any existing or future FMIS was therefore seen as the easiest way to encourage adoption of the proposed technologies. Since current agricultural software is

implemented in many differing programming languages, and future systems may move to an internet-based and service-oriented architecture (Murakami et al, 2007; Nash et al, 2009; Nikkilä et al, in press), the specification of the interface to the Rules App is initially defined at an abstract level. However, such an abstract specification is an insufficient basis for the actual implementation of a standardised software component and so concrete implementation specifications must be produced for each programming language or interface technology.

As the simplest form of platform-neutral component, a RESTful web-service interface specification has been defined for the initial proof-of-concept implementation. Although this may be bound in to any client system, it is expected that in future further implementation specifications should be produced for software components in specific programming languages (e.g. .net, C++, etc.) or for specific platforms. For these, the combination of the abstract specification and the REST implementation specification provide a solid foundation.

In practice, it is expected that the implementation of the Rules App, and even client applications to use this component, will prove to be somewhat problematic due both to the problems in rule formulation which are discussed in other publications from the FutureFarm project and to the problem of identifying and retrieving correct and sufficient data to allow evaluation where this is theoretically possible. On the one hand, this is a technical problem which may be overcome through advances in technology, whilst on the other hand this is a political problem in that rules are specified which are inexact or may not be easily evaluated. Solving this latter category of problem will require "buy-in" from decision makers to the concept of automated rules evaluation, with a clear demonstration of the advantages it would offer, potentially both to farmers for self-checks and to controlling bodies for official evaluations.

Appendix

- Rule evaluation interface specification

References:

Fielding, R., 2000. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, USA. Available at <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> (05/01/2010).

Murakami, E., Saraiva, A.M., Ribeiro Jr., L.C.M., Cugnasca, C.E., Hirakawa, A.R., Correa, P.L.P., 2007. An infrastructure for the development of distributed service-oriented information systems for precision agriculture. *Computers and Electronics in Agriculture* 58 (1) 37-48.

Nash, E., Nikkilä, R., Pesonen, L., Oetzel, K., Mayer, W., Seilonen, I., Kaivosoja, J., Bill, R., Fountas, S., Sørensen, C., 2009. Machine Readable Encoding for Definitions of Agricultural Crop Production and Farm Management Standards. FutureFarm Deliverable 4.1.1. Available at http://www.futurefarm.eu/system/files/FFD4.1.1_Encoding_of_standards.pdf (05/01/2010).

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

Nikkilä, R., Nash, E., 2009. Definition of an interface for a knowledge repository. FutureFarm Deliverable 4.2. Available at <http://www.futurefarm.eu/node/196> (05/01/2010).

Nikkilä, R., Seilonen, I., Koskinen, K., in press. Software architecture for farm management information systems in precision agriculture. *Computers and Electronics in Agriculture*.