Development Experience of a Context-Aware System for Smart Irrigation Using CASO and IRRIG Ontologies

Quang-Duy Nguyen, Catherine Roussey, María Poveda-Villalon, Christophe de Vaulx, Jean-Pierre Chanet

applied sciences

MDPI

Article

Development Experience of a Context-Aware System for Smart Irrigation Using CASO and IRRIG Ontologies

Quang-Duy Nguyen [1,*,†], Catherine Roussey [1,†], María Poveda-Villalón [2,†], Christophe de Vaulx [3] and Jean-Pierre Chanet [1]

1 Université Clermont Auvergne, INRAE, UR TSCF, 63178 Aubière, France; catherine.roussey@inrae.fr (C.R.); jean-pierre.chanet@inrae.fr (J.-P.C.)
2 Ontology Engineering Group, Universidad Politécnica de Madrid, 28660 Boadilla del Monte, Madrid, Spain; mpoveda@fi.upm.es
3 Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS), UMR 6158 UCA-CNRS, 63170 Aubière, France; christophe.de_vaulx@uca.fr
* Correspondence: quang-duy.nguyen@inrae.fr
† These authors contributed equally to this work.

Received: 31 December 2019; Accepted: 21 February 2020; Published: 5 March 2020

Featured Application: This paper presents a smart irrigation context-aware system to be applied on an experimental farm in France. The experimental farm is an ecosystem for agricultural machines and digital information systems developed by an alliance of European organizations and institutes. Moreover, the two ontologies proposed by this paper, CASO and IRRIG, are open access and have the potential to be used by other applications such as smart buildings or smart cities.

Abstract: The rapid development of information and communication technologies and wireless sensor networks has transformed agriculture practices. New tools and methods are used to support farmers in their activities. This paper presents a context-aware system that automates irrigation decisions based on sensor measurements. Automatic irrigation overcomes the water shortage problem, and automatic sensor measurements reduce the observational work of farmers. This paper focuses on a method for developing context-aware systems using ontologies. Ontologies are used to solve heterogeneity issues in sensor measurements. Their main goal is to propose a shared data schema that precisely describes measurements to ease their interpretations. These descriptions are reusable by any machine and understandable by humans. The context-aware system also contains a decision support system based on a rules inference engine. We propose two new ontologies: The Context-Aware System Ontology addresses the development of the context-aware system in general. The Irrigation ontology automates a manual irrigation method named IRRINOV®. These ontologies reuse well-known ontologies such as the Semantic Sensor Network (SSN) and Smart Appliance REFerence (SAREF). The decision support system uses a set of rules with ontologies to infer daily irrigation decisions for farmers. This project uses real experimental data to evaluate the implementation of the decision support system.

Keywords: agriculture; smart irrigation; context-aware system; ontology; rules

1. Introduction

In the agricultural domain, farmers need to observe natural phenomena to engage in appropriate activities on their fields. For example, in traditional irrigation, farmers go to their fields to examine

the crop development stage and measure the soil moisture provided by probes in the soil. Then, they use practical experience or follow an irrigation method to estimate manually the water needs of their crops. Based on their estimations, the farmers decide whether to irrigate the fields. This conventional approach has two significant drawbacks. First, irrigation requires daily observations, often made by farmers. Manual observations are influenced by other factors such as the weather or the situation of farmers. For example, a daily observation could be skipped if the farmer is sick. Second, the resource shortage problem demands that farmers use water sparingly.

Context-aware systems (CASs) can overcome the above-mentioned situations. A CAS uses a wireless sensor network (WSN) to monitor environmental phenomena and uses those measurements for further processes. In a CAS, context refers to "any information that can characterize the situation of an entity. An entity could be a person, a place or an object that is considered relevant to the interaction between a user and an application, including the user and the application themselves" [1]. A CAS has two contexts: a low-level context and a high-level context [2]. The low-level context contains quantitative data. The high-level context contains qualitative data that synthesize a situation and ease the decision. For example, the statement "soil moisture is 160 centibar (cbar)" presents a low-level context; however, the statement "soil is dry" presents a high-level context.

The CAS has some peculiar characteristics. First, sensors in the system are heterogeneous. Each type of phenomenon demands a different type of sensor. For example, in agriculture, pluviometers measure rain quantity, and tensiometers measure soil moisture. Thus, the CAS should address some data interoperability issues. Second, the system should have the capability to process the raw measurement data and apply reason on them.

To address the above requirements, ontologies are adequate candidates. Some sensor ontologies are provided by several institutes of standardization. Their main goal is to propose a shared data schema that precisely describes measurements to ease their interpretations. These descriptions are reusable by any machine and understandable by humans. They propose a data modeling design pattern that is precise enough to answer any informational need. Thus, any sensor measurements described by these ontologies may be reusable by several decision support systems (DSSs). The sensor network becomes a data provider for several applications. For example, rain quantity measurements can be used by irrigation decision systems and crop disease management systems. Reusing existing ontologies enables the harmonization of sensor measurement descriptions. Moreover, those descriptions share the same data model. Thus, they are easy to integrate in a global data schema. Furthermore, some ontologies enable reasoning over data because they propose logical descriptions. In short, ontologies used in context modeling can solve the heterogeneity issues of various sensor measurements and infer the high-level context and decisions by applying rules. To the best of our knowledge, the proposed work is the first to reuse two well-known ontologies: the new version of the Semantic Sensor Network (SSN) [3] and Smart Appliances REFerence (SAREF) [4].

This paper presents the development method of a CAS that combines two engineering methodologies: ontology and information systems. This method was used to build a smart irrigation CAS that automates a manual irrigation method. This irrigation method, called IRRINOV®, was developed by Arvalis (Arvalis is an applied agricultural research organization dedicated to arable crops. It engages in many activities at 27 different local sites in France.) and its partners. The IRRINOV® method is widely used in many regions in France. The development of the system also includes the development of two new ontologies—the Context-Aware System Ontology (CASO) and Irrigation ontology (IRRIG)—and a set of rules for reasoning. The two new ontologies reuse well-known ontologies related to sensors and devices—SSN and SAREF. CASO specializes and extends SSN to describe the processing of context. CASO implements a generic context model that can be specialized to any observation of environmental phenomena. Using CASO simplifies the data processing and rule generating by dividing them in several steps. IRRIG specializes CASO by implementing the IRRINOV® method. The two ontologies model the sensor measurements and results of all measurement processing (cleaning, aggregation, reasoning, etc.). The system uses a set of rules for reasoning. Each rule

implements small inference steps. Thus, they are easy to understand, manage and correct. Finally, we propose a complete DSS that supports farmers in making daily irrigation decisions.

The rest of this paper is organized as follows. Section 2 presents the state of the art of several smart irrigation systems available in the research world. Section 3 introduces background information for the cycle of processes implemented in a CAS and the IRRINOV® method. The next section describes the development process of the complete system and the two ontologies. Section 5 provides an example of the development presented in Section 4. Section 6 presents and compares the results after implementing the system using real experimental data provided by Arvalis. Also, this section discusses the limitations of the IRRINOV® method, the limitations of the Arvalis dataset and the limitations of the DSS system. Finally, a brief conclusion sums up the system and presents the perspectives.

2. State of the Art

As mentioned in the introduction, an ontology is a shared data schema that provides a common understanding of data descriptions. Data descriptions based on ontologies are reusable by any machine and understandable by humans. Thus, we examined ontologies already used in irrigation decision systems to reuse them as data modeling patterns to improve the results of our data modeling activity.

Several expert systems already exist that can determine whether a crop needs water. For example, one expert system uses the common-KADS method to irrigate mango trees, as presented in [5]. Please note that this expert system is not connected to any sensors or actuators. Its goal is to help farmers decide when to irrigate. However, no information about the ontology publication is provided.

Semantic web technologies are already used in CASs dedicated to irrigation. For example, the FIWARE platform that links Internet of Things (IoT) devices to the cloud has been used in several different irrigation experiments [6]. The FIWARE cloud platform contains the SEPA SPARQL-based engine, which represents information in an RDF format and provides SPARQL queries. To the best of our knowledge, no information about the ontologies used in these experiments is available online.

The works of [7,8] present some ontologies dedicated to hilly citrus tree cultivation. One ontology addresses the irrigation decision. The ontology stores the computations of relative soil moisture based on the soil type and the crop growth stage. When the soil moisture reaches a given threshold, the farmers receive an alert from the system. In this case, the ontology is not published; therefore, it is impossible to reuse it.

To our point of view, the most successful CAS dedicated to irrigation is the one developed during the PLANTS project [9]. This system is installed in a greenhouse. It uses several types of sensors to observe raspberry plants. This system can control watering equipment to control irrigation in the four zones of the greenhouse. The PLANTS ontology aims to describe e-entities and their interactions [9,10]. An e-entity is a virtual representation of a physical object that can be either a raspberry plant or equipment. Every measurement is defined as the parameter of an e-entity. Sensor streams are not aggregated. The last measurements are used to update a parameter. The rules are used to derive the states of raspberry plants from various parameters. Watering equipment is controlled by the detection of "drought stress" in a specific zone. Thus, the only part of the ontology that we want to reuse is the plant state hierarchy. Unfortunately, this ontology is unavailable on the Web because it is the schema of the facts-base implemented in the Jess inference engine.

To conclude, while several works focus on processing sensor streams and computing aggregations, their results are not directly reusable, as our approach should handle the temporal and spatial aggregation of sensor streams for the irrigation use case. Additionally, the ontologies mentioned in previous works are either unavailable online or their domain is loosely related to our use case.

3. Background Information

This smart irrigation system is based on a CAS and the IRRINOV® method. First, the cycle of processes implemented in a CAS provides an overview of the global system: its devices, its phases of

processes and the transformation from data to context. Second, the IRRINOV® method specifies the requirements of the system and the basic algorithm for making decisions.

3.1. Cycle of Processing Dedicated to CAS

In irrigation, a CAS has three specific components. First, a WSN plays the role of sensing and monitoring the plot environment. Second, a DSS can (1) send notifications to farmers to support them in their decision-making process and (2) automatically make decisions and control the watering system. Third, watering devices are in charge of watering the soil in the field.

The CAS processes can be grouped and represented as a cycle of processes [11]. This cycle is called the CAS life cycle. It is divided into four phases: (1) acquisition, (2) modeling, (3) analysis, and (4) exploitation. Please note that this section improves our previous work about CAS life cycle [12] by decreasing the number of phases. The exchange between two phases can consist of data, the low-level context or the high-level context. The CAS life cycle for smart irrigation is depicted in Figure 1. The four phases of its life cycle are described as follows.

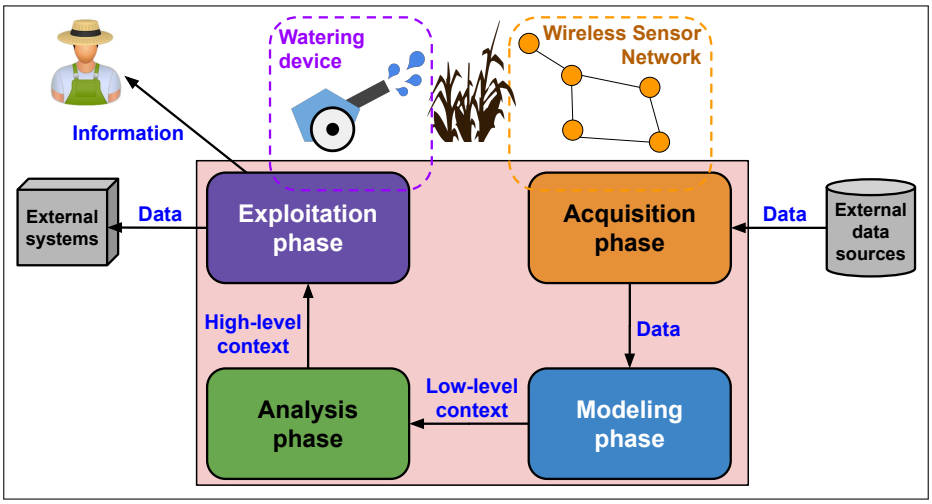

Figure 1. Life cycle of a smart irrigation CAS.

- **Acquisition phase** focuses on how the system retrieves and processes measurements from sensor devices and data from external data sources. The output of this phase are measurements and related data derived after raw data cleaning processes.
- **Modeling phase** focuses on the data model and the integration of input data into the system. The system is equipped with a storage service. The input data are organized with a common data model to become the low-level context. As shown in Figure 1, the input of this phase is data derived from the acquisition phase. The output of this phase is the low-level context. This paper considers ontologies as a candidate for the data modeling process.
- **Analysis phase** focuses on the transformation from the low-level context into the high-level context. The high-level context is an enrichment of the context with qualitative data. Such data summarize the situation of entities and support the decision process. The system uses a rules-based inference engine for the reasoning process. As shown in Figure 1, the input of this phase is the low-level context. The output of this phase is the high-level context. This paper considers Drools (https://www.drools.org/) as a candidate for the inference engine.
- **Exploitation phase** focuses on the use of the high-level context to distribute them to corresponding agents, which can be other devices or users. The input of this phase is the high-level context. The output of this phase can be human-readable content, a message to an external system or a reaction of a watering device to the environment.

3.2. IRRINOV® Method

This subsection presents the IRRINOV® (All versions of the IRRINOV® method are available at http://www.irrinov.arvalisinstitutduvegetal.fr/irrinov.asp) method. This method, developed by Arvalis and its partners, proposes a guide to make irrigation decisions based on measurements of soil moisture sensors and pluviometers oriented to farmers. The method aims at answering the following questions: (1) When should irrigation be started, or when should watering devices be installed on the plot? (2) When should we start each watering cycle ("Watering cycle" is the action of an irrigation system performing a watering activity on all the plots engaged in the system. "Watering cycle duration" is the number of days between two consecutive waterings of the same plot)? (3) When should irrigation be stopped, or when should farmers withdraw the watering devices?

A set of decision tables and recommendations are provided by IRRINOV® method to allow farmers to manage their irrigation system on a single plot. Numerous variants of the method are proposed depending on the soil, plot and crop types. This work uses the IRRINOV® method for the region Midi-Pyrénée, which is dedicated to maize crop plants on clay-limestone soil [13] and includes the following measuring equipment:

- One IRRINOV® measuring station composed of six Watermark (https://www.irrometer.com/sensors.html#wm) probes that measure the soil water tension (tensiometer). Three Watermark probes should be placed at 30 cm depth in the soil, and the other three should be placed at 60 cm depth in the soil. Figure 2 illustrates the prototype of the IRRINOV® station.
- One mobile pluviometer that measures the amount of water received by the crop during a watering.
- One weather station with a pluviometer to measure the quantity of water received by the crop during a rainfall.

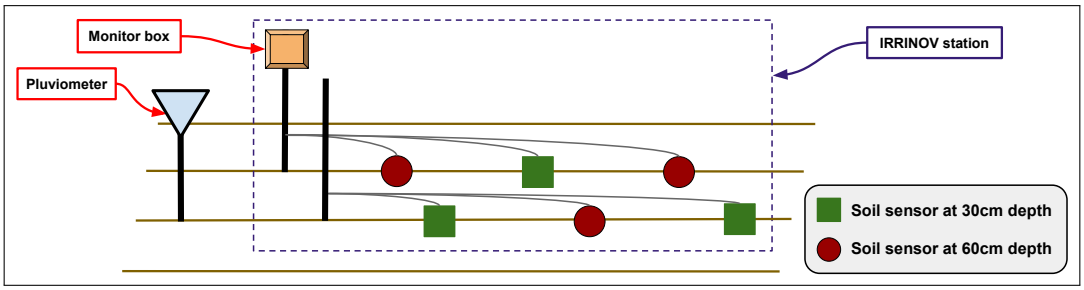

Figure 2. Prototype of the IRRINOV® station.

More information about IRRINOV® method is available at [12]. The goal of this section is to illustrate the processing defined un previous section. The IRRINOV® method specifies the time needed to install the devices on the plot. The IRRINOV® station and the mobile pluviometer should be placed in the plot when the crop reaches growth stage V2 (V2, V7, R1, and R5 are the code names of maize growth stages defined in [14]. They are respectively named "7 leaf", "10 leaf", "silking" and "dent stage with 50% moisture" in the Arvalis growth stage classification.). The measurements can start two or three days after installation. The Watermark probes are typically read once a week, but during dry weather, farmers can check the probes more frequently, for example, one observation per day. Irrigation should stop when the crop reaches the growth stage R5hg45. Please note that state R5 is defined in [14]; however, the state R5hg45 is defined in this project. This state means that the crops have moisture dents equivalent to 45%. The IRRINOV® method specifies the constraints to validate the measurements of Watermark probes. Those treatments are part of the acquisition phase. If the difference between the probe measurements is above 30 cbar, then one of the probes is malfunctioning, and the farmer should visit the field to correct the probe installation. The value read on a Watermark probe must be multiplied by the correction coefficient, which is specific for each batch of probes. For example,

probes from 2013 have a correction coefficient equal to 1.0. In this paper, we call the former value a raw data value. The latter value after this process is the measurement value. Please note that when the measurement value is 199 cbar, the IRRINOV® method considers that there is a contact problem between the Watermark probe and the soil. The measurement value represents the soil tension and has the cbar unit. It is part of the output of the acquisition phase.

The IRRINOV® method relies on decision tables to determine when to start a watering cycle depending on the soil moisture measurements and crop growth stage. An example of table that determines when to start a watering cycle for clay-limestone soil is provided in Table 1 which is, in fact, applied to maize crops when their growth stage is between V2 and V7. The decision table indicates that the farmer should start the first watering cycle when the median value of three probe measurements reach a given threshold.

Table 1. Decision table of the relation of the watering cycle duration and the soil moisture for maize at growth stage V2.

Watering cycle duration	9 to 10 days	6 to 8 days	below 5 days
Median of 30 cm depth probes	30 cbar	50 cbar	60 cbar
Median of 60 cm depth probes	10 cbar	20 cbar	20 cbar
Sum of the median of 30 cm depth probes and the median of 60 cm depth probes	40 cbar	70 cbar	80 cbar

The second column of the above decision table should be read as "If the plot has a watering cycle duration fixed between 9 and 10 days", and one of the two following conditions is satisfied:

(1) "When the median value of three probe measurements at 30 cm depth are above the 30 cbar value" and "when the median value of three probe measurements at 60 cm depth are above the 10 cbar value".

(2) "When the total (sum of the median of 30 cm depth measurements and the median of 60 cm depth measurements) is above 40 cbar".

Then, a watering cycle should start.

This project translates all the decision tables into rules to transform the manual IRRINOV® method into an automatic DSS. Those processing are part of the analysis phase. The exploitation phase will translate the irrigation decision into a command executable by the watering device.

4. CAS Development

The development methodology used to develop the smart irrigation CAS is the mini-waterfall methodology [15]. Additionally, CASO and IRRIG ontologies have been developed following the linked open terms (LOT) methodology [16].

A mini-waterfall is a methodology used for system engineering. It is inspired by the well-known waterfall methodology, which develops a system in five activities: (1) specification, (2) design, (3) implementation, (4) testing and (5) maintenance. The purpose of the mini-waterfall methodology is to repeat the sequence of the five activities multiple times to develop the system progressively.

LOT is a methodology used for ontology engineering. It was first proposed in [16] and further developed at [17]. LOT is inspired by agile techniques in which sprints aim to align ontology development with software development agile practices. This methodology focuses on (1) the reuse of terms (classes, properties and attributes) existing in already published vocabularies or ontologies and (2) the publication of the ontology following the linked data principles. LOT relies on the ontological engineering activities defined in the NeOn methodology [18] when available. LOT defines iterations over the following four activities: (1) ontological requirements specification, (2) ontology implementation, (3) ontology publication, and (4) ontology maintenance.

Figure 3 illustrates the smart irrigation CAS development methodology combined with the ontology development methodology. Ontology development has a life cycle called the LOT life cycle. The LOT life cycle is presented by the dashed yellow box. Each activity in this life cycle is represented by a solid yellow box. The solid black arrows between the boxes show the order of the activities; for example, the development team must perform the ontological requirements specification activity before the ontological implementation activity. The CAS development life cycle is called a mini-waterfall life cycle. It is presented by the dashed blue box. Each activity in this life cycle is represented by a solid blue box. The solid yellow arrow between one activity X in the LOT life cycle and one activity Y in a mini-waterfall life cycle means that the result of activity X is necessary to implement activity Y. For example, the design activity of the mini-waterfall cycle requires the result of the ontology conceptualization activity of the LOT life cycle. Otherwise, the dashed blue arrow from one activity Y of mini-waterfall to one activity X of LOT life cycle means that the result of the activity Y can contribute to activity X. It is necessary to note two items in this methodology. First, the LOT life cycle is triggered by the mini-waterfall life cycle, but it is independent of the mini-waterfall life cycle. Consequently, the LOT life cycle can be shorter than the mini-waterfall life cycle. Suppose that after each development life cycle, there is a new version. Therefore, there is nothing to prevent the ontology from having three versions at the same time, but the system can only have one version. Second, after the maintenance activities of both the LOT life cycle and CAS life cycle, the development team renews the sequence, i.e., starts with the first activity. However, they can skip the current activity if they find that the last time result of the same activity is sufficient.

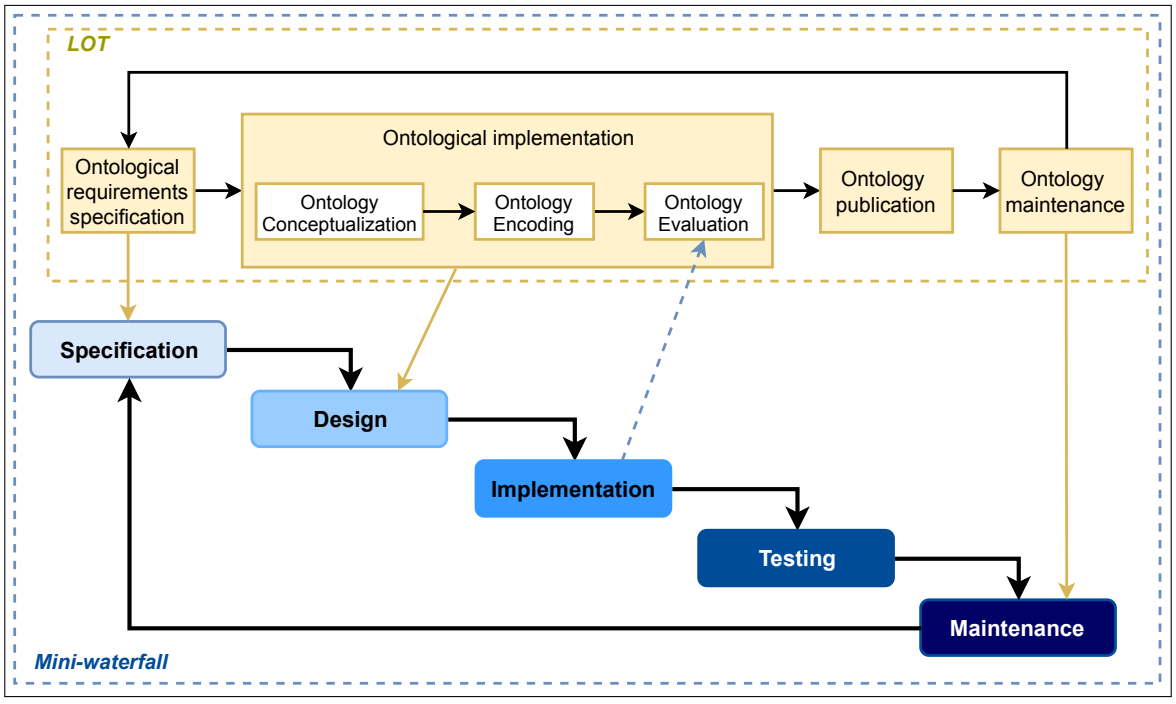

Figure 3. Mini-waterfall combined with LOT methodology to develop a system.

The specification activity analyzes and identifies all the requirements for the system and the associated ontology (global data schema). In the design activity, the system uses a network of ontologies to model the results of all data processing. Therefore, ontology implementation is part of the system design. In the CAS implementation activity, the system imports the ontologies. The result after a CAS implementation activity contributes to the ontology evaluation activity. If the CAS is implemented, then the ontologies cover all the requirements. The ontologies should be available in public URIs through ontology publication. Ontology evaluation and system testing are necessary to guarantee that the system works correctly. Finally, the system and the ontologies are maintained.

During the specification activity, we identify several sensor streams to be processed. The next activities of the mini-waterfall methodology (design, implementation and testing) were applied on one stream at a time to cover all the requirements. Then, a global algorithm was implemented that merges all the results of the stream processing.

4.1. CAS Specification

First, it is necessary to present the resources for system development. All the requirements of the system are extracted from the following resources:

- The IRRINOV® method version for the region Midi-Pyrénée, France.
- The Arvalis dataset: real experimental data from Arvalis and INRAE (INRAE (French National Research Institute for Agriculture, Food and Environment) was founded on 01/01/2020 from the fusion between INRA and IRSTEA. It supports the profound changes in agriculture, food and the environment in France, Europe and around the world through its research activities.). This is an excel file that contains recorded daily observations and activities of farmers in a maize plot T1 B1 - DKC4814 from 14/06/2013 to 06/10/2013 in Gaillac, France. Please note that the observations recorded in the Arvalis dataset are the daily aggregated values of the sensor measurements. In other words, the data in the dataset are already processed from the raw measurements.
- Farming journal ("Carnet de Terrain" in French): a document that contains several tables to record all the precise observations related to the plot. It is a kind of experimentation journal that was used frequently in research laboratories.
- Experiences and knowledge from specialists in the agriculture domain. The specialists are from Arvalis and INRAE.

This subsection focuses on two requirements: the data modeling requirement and the processing requirement. The data modeling requirement is needed for developing ontologies. The processing requirement is used to identify the type of processing applied on the sensor streams (cleaning, aggregation, comparison).

4.1.1. Ontology Requirement

First, the IRRINOV® documentation [13] was studied to identify the following:

- The type of sensors used and their measurements.
- The data needed to make the decision: type of soil, type of crop, sowing date, sensor localization and depth, quantity of rain per day, evolution of crop growth stage, and moisture measurements at different depths.
- The different processes applied to the sensor measurement streams, such as data cleaning, spatial and temporal aggregation, and threshold comparison.
- The different decisions produced by the method.

Second, it was necessary to communicate with specialists of Arvalis and INRAE to clarify the IRRINOV® method. Some of the questions formulated to the experts were as follows: What is the difference between an irrigation period and a watering cycle? What are the possible values of maize growth stages? How are we to observe a growth stage?

Third, to generalize the system, it was necessary to derive standard definitions and concepts in the agriculture domain. This work was carried out by querying a web retrieval system. The selected resources were articles referenced in publications and from well-known organizations. For example, the different maize growth stages are defined in a publication from the University of Iowa [14].

Fourth, the analysis of the Arvalis dataset provided a better understanding of the computing processes on sensor measurements, as well as the experimental details such as the type of soil and the code names of the sensors.

The first analysis enabled this system to define several process workflows corresponding to the type of sensor measurement stream. A workflow defines the aggregations (or any mathematical functions) and deductions applied on the streams. An example of temporal aggregation is the computation of the average on a 24-h window applied on a soil moisture stream produced by one probe. Then, the aggregation results are conceptualized into a property of an observed phenomenon. This system considers four main sensor streams described by observed phenomena and associated properties: (1) the moisture of the soil, (2) the quantity of rain, (3) the growth stage of the crop, and (4) the water need state of the crop.

Moreover, from the first analysis, an ontology requirement specification about the smart irrigation CAS was generated. These specific requirements were written as a competency questions document. An online spreadsheet (The spreadsheet is available on the GitLab repository of this project https://gitlab.irstea.fr/irrig/public/tree/master/CompetencyQuestions) was defined to share them. It contains the following fields:

- The identified code of requirements.
- The competency questions written in natural language. Various forms of the question were written to generalize them.
- Some examples of answers to identify the domain value.
- The explanation of the answer.
- The references of the vocabulary definitions.

Figure 4 presents an excerpt of the current competency questions. Please note that these competency questions are a large table with many columns and rows; we only show an excerpt as an illustration. For example, CQ1.1 expresses the need to know the type and brand of each sensor. CQ1.3 expresses the need to define the type of sensor streams used as inputs in the IRRINOV® method.

Specification Competency Questions document
Version 0.1

Code	Compentency Questions	Answer/Example	
CQ1	Description of measurement equiments		
CQ1.1	What is the identifier of the probe/sensor?	2013F13 03	
CQ1.2	Which type of measurement provides the probe/sensor? Which phenomenon and which associated property is measured by the probe/sensor?	Example: soil moisture at 30 cm-depth	
CQ1.3	What is the correction value of the probe/sensor? Does the raw measurement produced by a probe/sensor need a transformation before being considered as a valid measurement?	1	Tl va re
CQ1.4	What is the range of value provided by the probe/sensor? Is the measurement of the probe/sensor valid?	0 to 200	
CQ1.5	What is the unit of the probe/sensor?	char, mm, degree Celsius	

Figure 4. Screenshot of a part of the competency questions document containing the questions and answers from CQ1.1 to CQ1.4.

4.1.2. Processing Requirement

In studying the IRRINOV® method, we noticed that the irrigation decisions proposed by this method are based on several aggregation measurements: temporal aggregation and spatial aggregation. We notice that the decision workflows of processes follow the same pattern: provide a stream of sensor measurements, aggregate certain sensor measurements extracted from the stream, and then compare the aggregation results to a threshold. The thresholds enable us to define certain states. A state is qualitative data of a property that characterizes the observed phenomenon. The goal of the rules is to determine the state of each property. The design activity goal will be to translate each statement in decision tables and the associated threshold into one or several reasoning rules. The specification requirements activity identifies each process belonging to one workflow dedicated to one stream and

determines its input, its output and its parameters (thresholds). For example, the workflow dedicated to the soil moisture stream starts with a measurement process from a soil tensiometer probe. Then, those measurements are the input data of a sequence of aggregation processes. The last aggregation value is compared to a threshold to produce a state of a soil moisture entity. This example is precisely outlined in Section 5.

Second, the Arvalis dataset provided the parameters to precisely determine the aggregation processes used in the IRRINOV® method. Examples of such information are the watering cycle duration and the threshold for soil moisture.

4.2. CAS Design

This subsection focuses on explaining the conceptualization process of the ontologies. Later, the system design describing how to use the ontologies for data modeling is presented.

4.2.1. Ontology Conceptualization

First, the ontology requirements specification was oriented to the development of one ontology to support the smart irrigation CAS. However, it should be noted that a general part of the ontology could be reusable for other use cases; therefore, we decided to materialize the implementation in two ontologies, CASO and IRRIG. In the following, a typical approach for the conceptualization is detailed, and then the decisions about the ontology modularization used to create CASO and IRRIG are described. Please note that our ontologies have already studied well-known ontologies about sensor measurements [12]. They choose to reuse SSN since this ontology provides a precise description of measurements based on data modeling patterns. SSN defines several patterns that help to understand the data modeling issues.

The first step towards a conceptualization was to identify from the competency questions document those ontology elements that needed to be represented. More precisely, this step was carried out in the following order:

- Identification of concepts: We want to model all the processes applied to sensor streams. They are mainly aggregations and threshold comparisons. The goal of the comparison is to derive qualitative (symbolic) data. To differentiate quantitative from symbolic computation, we need to use deduction. A deduction provides qualitative data as a result. These results are the entity states that should be defined in some referential item. For example, maize growth stages are states defined in a certain thesaurus. In most cases, the state of an entity is derived because one value is above or below a threshold. We model thresholds as state boundaries.
- Identification of relations: To characterize a deduction, we need two new relations: (1) a relation that links a deduction to its resulting state and (2) a relation that links a deduction to the period when this deduction is valid, i.e., the period when the entity is associated with a state.
- Identification of cardinality restrictions applied for relations: This step also involves a decision on whether the relation at hand should be defined as functional. For example, the relation between a deduction and its result is functional. A deduction has only one result: the state of the entity derived by the deduction.
- Identification of individuals: Following the SSN pattern, these entities are characteristics of the feature of interest [3]. They are modeled as an instance of the class **ssn:Property**, which are linked to an instance of the class **sosa:FeatureOfInterest** by the object property `ssn:hasProperty`. To specialize an SSN to the irrigation use case, the involved properties in the use case, for example, air temperature or crop stage, have been defined as instances of properties.

Next, ontologies and particular ontology elements were identified to be reused. The following selected ontologies have been developed and maintained by standardization organizations.

- The W3C & OGC Semantic Sensor Network (SSN) ontology describes sensors, observations, samples and actuators [3].

- Smart Appliances REFerence (SAREF) (http://www.w3id.org/saref) is an ontology for smart appliances that contributes to semantic interoperability in the IoT domain [4]. This ETSI recommendation focuses on the representation of appliances and devices together with their functions, commands, services, states and profiles (It is necessary to mention that CASO and IRRIG reuse some concepts from SAREF and SAREF4AGRI, which are unavailable in the latest published versions of these ontologies, but it is known by the authors that they will be available in the next versions. SAREF4AGRI is an extension of SAREF for agriculture.).
- The W3C PROV ontology represents provenance information [19].
- The W3C SKOS vocabulary describes knowledge organization systems. It will be useful in defining properties and their states [20].
- The W3C OWL-Time (http://www.w3.org/2006/time), an ontology of temporal entities, describes the temporal properties of resources.

Another well-known ontology reused in this conceptualization is the Ontology of Units of Measure and Related Concepts (OM) [21].

In some cases, the reused ontology elements matched the intended use in the developed ontology; therefore, they were adopted without modification. Some examples of this case are the classes **sosa:Platform**, **sosa:FeatureOfInterest** or **om:Unit**.

However, in other cases, some additional properties or constraints had to be added to the reused elements. Then, a new subclass of reused concepts was created to attach the new semantics to the subclasses created. For example, the class **caso:Observation** is a specialization of the class **sosa:Observation** since **sosa:Observation** represents any computation activity in general. **caso:Observation** uses the object property `prov:wasInformedBy` to identify the inputs of the computation. The class **caso:Deduction** is a specialization of **caso:Observation**. A deduction is also a computation or a measurement that has a state for a result.

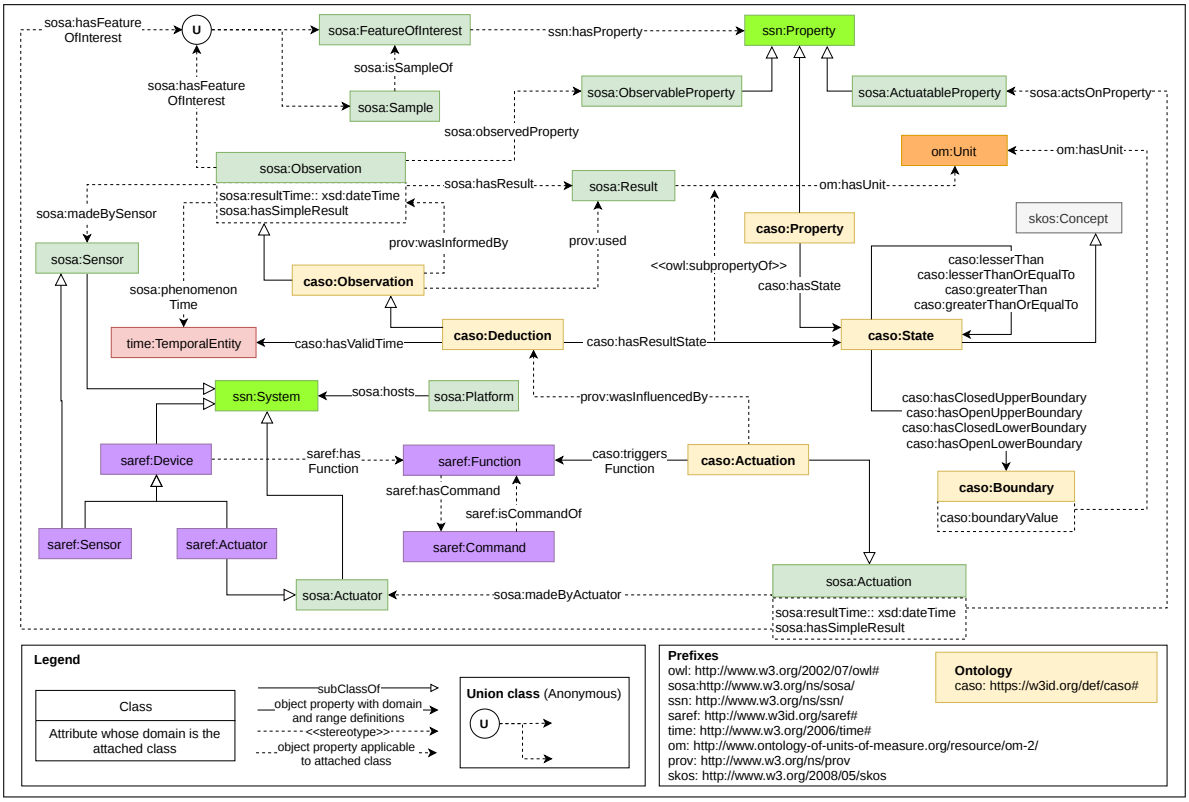

Figure 5. Overview of the CASO.

Finally, the conceptualization was modularized into two ontologies: CASO and IRRIG. The IRRIG ontology imports and extends CASO. The CASO ontology is illustrated in Figure 5. This IRRIG ontology is depicted in Figure 6. The entities defined in the CASO and IRRIG ontologies are preceded by the prefixes "caso" (yellow boxes) and "irrig" (blue boxes), respectively.

The goal of CASO is to model any context processing. IRRIG is a specialization of CASO used to describe all the context processing involved in the IRRINOV® method. For this reason, it specializes some elements of CASO, as shown in Figure 6. For example, IRRIG specializes the class **ssn:Property** in different subclasses related to moisture, stress, and growth. Several subclasses of **caso:Observation** specify different types of observations. Each subclass is dedicated to the observations of one specific instance of **ssn:Property**.

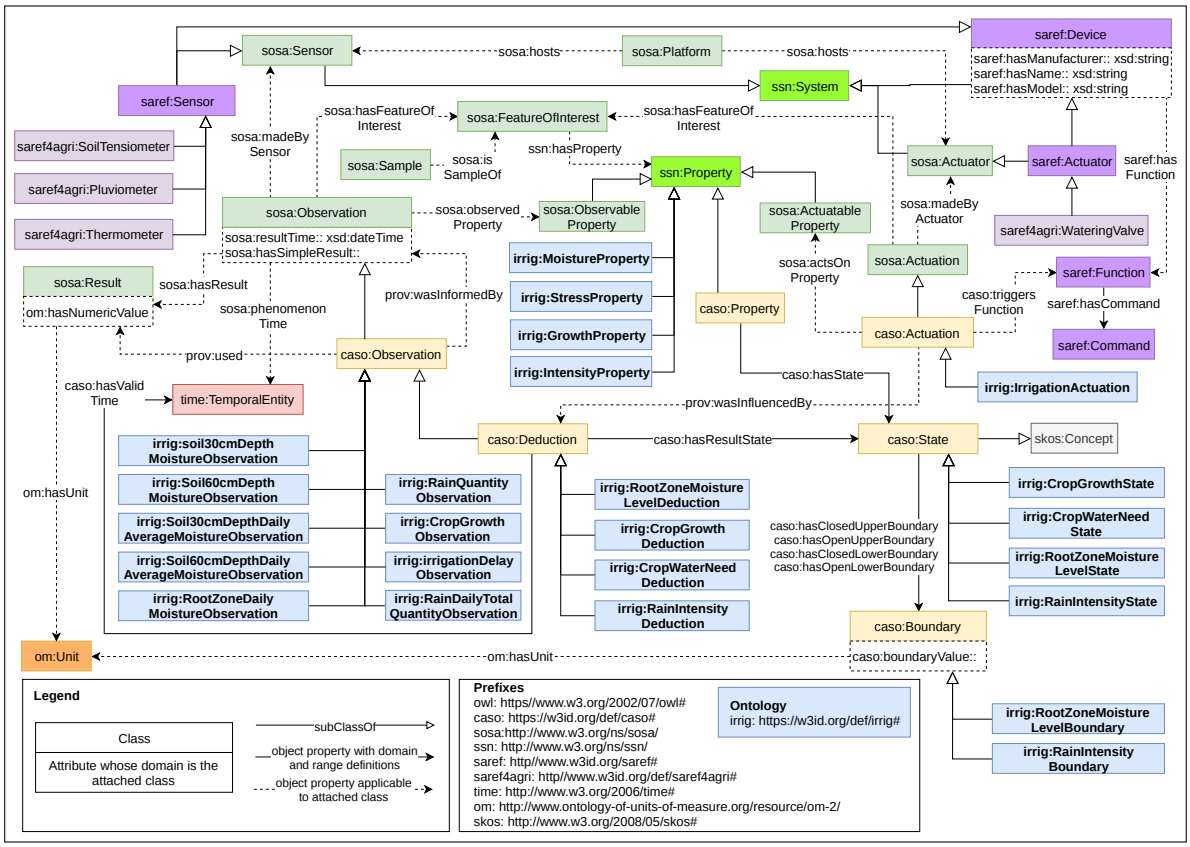

Figure 6. Overview of the IRRIG ontology.

During the conceptualization phase, several versions of the ontology model are produced and stored in a private repository of INRAE. Figures 5 and 6 are parts of the lastest version (v201912).

4.2.2. System Design

Two main steps used to model the smart irrigation CAS are (1) to identify the entities needed to model and (2) to choose the appropriate vocabulary and relation from CASO and IRRIG to model the determined entities.

First, we determined the entities needed to model the smart irrigation CAS. The following vocabulary should be considered entities:

- Feature of interest is the natural phenomenon observed by sensors.
- The property of the feature of interest can be a quantitative property (a sensor measurement or aggregation) or a qualitative property (a state that is a result of threshold comparison).
- Data processing actions and the actor that executes the action. The data processing may be a measurement, an aggregation or an inference.

- Result and the time related to the action.

The action could be a measurement, aggregation or deduction (threshold comparison or inference). On the one hand, the result of a deduction should be qualitative data that are a state of the entity. On the other hand, the result of the measurement and aggregation is numerical data.

Table 2 shows a list of entities needed to model. This table contains six columns:

- Workflow: there are four streams defined in this project: (1) crop growth, (2) crop water need, (3) soil moisture, and (4) rain quantity. Each stream specifies a dedicated workflow of processes.
- Feature of interest: the phenomenon that needs to be observed: crop, rain, soil, etc.
- Property: the aspect of the phenomenon that needs to be observed.
- Type of action: could be a measurement, an aggregation or a deduction.
- Actor: depending on the type of action, an actor could be a sensor, a software component that implements an aggregation, or an inference engine for the deduction.
- Type of result: could be a state or a numeric value.

Table 2. List of features of interest, properties and actors that require modeling.

Workflow	Feature of Interest	Property	Type of Action	Actor	Type of Result
Crop growth	Crop	Growth	Measurement	Human	State
		Growth	Deduction	Inference engine	State
Crop water need		Water Need	Deduction	Inference engine	State
	Sleeping	Sleeping duration	Aggregation	Component that gets the sleeping duration	Numeric
Soil moisture	Soil at 30 cm depth	Moisture at 30 cm depth	Measurement	Soil tensiometer	Numeric
		Daily average moisture at 30 cm depth	Aggregation	Component that gets daily aggregated moisture	Numeric
	Soil at 60 cm depth	Moisture at 60 cm depth	Measurement	Soil tensiometer	Numeric
		Daily average moisture at 30 cm depth	Aggregation	Component that gets daily aggregated moisture	Numeric
	Root zone	Daily average moisture at root zone	Aggregation	Component that gets daily aggregated moisture	Numeric
		Root zone moisture level	Deduction	Inference engine	State
Rain quantity	Rain	Rain quantity	Measurement	Pluviometer	Numeric
		Daily total rain quantity	Aggregation	Component that gets daily total rain quantity	Numeric
		Intensity	Deduction	Inference engine	State
	Delay	Delay duration	Aggregation	Component that gets the number of delay days	Numeric

In this table, in addition to the entities related to the agricultural phenomenon, it is essential to mention the two other entities: sleeping and delay. Sleeping is the period after a watering action of one plot. No more watering action is applied on this plot during the sleeping period. Delay is the period when a watering action is postponed due to a rain event.

The second step of the CAS modeling is to select the vocabulary in CASO and IRRIG to model the entities. In CASO and IRRIG, the features of interest, properties and state of properties were predefined. Thus, the remaining work creates the individuals for the appropriate actions, numeric results, actors and time that relate to the corresponding feature of interests and properties.

The list of modeled entities and their associated processes defined from the above analysis enabled us to design part of the CAS. To design inference processes (the rules), we build a state diagram for each workflow. This diagram presents the conditions for a property reaching a state. Moreover, this indicates the behavior of the system after the property reaches the state. The state diagram uses a set of information:

- List of the states for the concerned property.
- The conditions under which that property can reach a state.
- The actions of the system after the property reaches the state.

To design the state change of certain properties, we selected the UML state diagram. First, we drafted the state changes in chronological order. Based on this first draft, we designed the conditions of state changes based on the IRRINOV® decision table. Then, we checked each set of conditions to make them independent. The goal was to have the same set of conditions always reach the same state value. In the end, the state diagram took into account synchronization issues between the new sensor measurements, the aggregation processes and the state changes. For that reason, an Init state was created in each state diagram to represent a synchronization point for daily computation. Moreover, each state diagram has a clock to manage the synchronization between all the processes.

Section 5.1.4 presents the lastest version of the state diagram related to the RootZoneMoistureLevel property. The conditions of the state diagram will be translated into semantic web rule language (SWRL) rules during the implementation activity.

4.3. CAS Implementation

This subsection presents the process used to transform the model of the ontologies and the system into software products using encoding languages.

4.3.1. Ontology Encoding

Once the models were designed, the two ontologies were encoded in OWL [22] using Protégé [23]. The implementation of the ontologies included the declaration of metadata such as authors, dates, and licenses, according to [24]. The OWL code of the ontologies is available on two distinct GitHub repositories. The current implementation of CASO contains 27 classes, 40 object properties and 4 datatype properties reusing the SOSA, SSN, SKOS, SAREF, PROV, Time, and OM ontologies. The IRRIG ontology imports CASO and extends it with 31 classes, 4 properties and 71 individuals to model the irrigation use case states, properties and features of interest. Among them, four classes and three datatype properties are reused from SAREF4AGRI.

4.3.2. System implementation

This project developed a program called Ontogen. This program was written in Python and Java. In the Arvalis use case, Ontogen takes as input specific sensor measurements provided by Arvalis in an Excel file and produces the output as the state of CropWaterNeed property. The program itself can be divided into three modules:

- Virtual sensors: read data from Excel files.
- Aggregators: run functions to produce aggregated data.
- Inference engine: The core is OWLAPI (https://github.com/owlcs/owlapi) and SWRLAPI (https://github.com/protegeproject/swrlapi). This module uses a set of SWRL rules, and all the data provided from the previous module to infer the high-level context.

The aggregations are encoded in Python as a function. The deductions are executed by the SWRLAPI Drool inference engine integrated into Ontogen. In detail, the reasoning process of the inference engine in Ontogen can be explained as follows. First, Ontogen generates an OWL file that contains all the individuals and the IRRIG ontology necessary for the rule engine; it creates all the individuals that describe sensor measurements. It also creates the individuals, instances of **caso:Deduction**, necessary to define the daily deduction per instance of the class **caso:Property**. The file name is the concatenation of the feature of interest name and the beginning date. Second, it launches the Drools rule engine and applies it to the individuals previously created. The Drools engine updates the **caso:Deduction** instance. It links an instance of the class **caso:State** with an instance of **caso:Deduction** via the object property `caso:hasResultState`. Then, it creates a new OWL file that contains the results of the inference engine. The OWL file is updated from the previous file. The rule for Drools is coded in SWRL. There are a total of 25 rules in SWRL and two queries in semantic query-enhanced web rule language (SQWRL):

- 5 rules for crop growth.
- 1 rule for rain intensity.
- 1 rule for root zone soil moisture level.
- 16 rules for crop water need.
- 2 SQWRL queries for the results after the reasoning.

Each rule is put in a rule file with a ".rule" extension. The rule files are available on the GitLab repository of INRAE via the following link https://gitlab.irstea.fr/irrig/public/tree/master/Rule.

The DSS consists of (1) a knowledge base and an inference engine, mostly for high-level context inference, and (2) several functions for numeric data calculation. Except for the case of SleepingDuration, it is a numeric value, but the inference engine produces its value. The knowledge-base contains a rules-base and a facts-base. Technically speaking, the DSS is a part of the Ontogen program. Each day, the DSS takes phenomenon observations as input and produces an irrigation decision. Please note that the Arvalis dataset does not provide the sensor measurements but the daily aggregated values of those measurements.

Figure 7 shows the algorithm implemented by the DSS. At 06:00:00 on day d, the system triggers the DSS to start the computation. First, the DSS retrieves the observations relevant to days d-1 and d recorded in the Arvalis dataset. Please note that in the algorithm of Figure 7, the data retrieved at the instant `[d 06:00:00]` belong to the period `[d-1 00:00:00, d 06:00:00]`. The DSS checks the value of the evaluation of the CropGrowth property of day d-1. As the IRRINOV® method is valid only when the state of CropGrowth is from V7 to R5hg45, then the DSS has three possible cases: (1) if the state of CropGrowth is less than V7, then the DSS waits until the next day to reevaluate the CropGrowth, (2) if the state of CropGrowth equals R5hg45, then the DSS stops, and (3) if the state of CropGrowth is between V7 and R5hg45, then the DSS continues to retrieve rules to create a fact related to CropGrowth. The fact integrates rules as axioms. After determining the fact, the DSS can launch the Drools inference engine to infer the state of CropGrowth for day d-1. Next, the DSS runs the functions to obtain the value of the root zone average moisture of day d-1 and to obtain the number of delay days. The function to obtain the value of root zone average moisture takes the value of the average moisture of day d-1 corresponding to each soil tensiometer from the Arvalis dataset. Additionally, the function to obtain the number delay days takes the value of the total rain quantity of day d-1 from the Arvalis dataset. The DSS collects all calculated results and retrieves the other rules related to RainIntensity, RootZoneMoistureLevel and CropWaterNeed properties and updates them. Later, the Drools inference engine infers the state of the three mentioned properties and the value of SleepingDuration. Based on the state of CropWaterNeed, the DSS offers two solutions: (1) if the state of CropWaterNeed is Yes, the DSS suggests that users launch irrigation on day d; otherwise, (2) if the state of CropWaterNeed is No, NotApplicable or Unavailable, the DSS suggests that users do nothing. After producing the

suggestion, the system saves the results of all computations into the knowledge base. Finally, the DSS continues to wait until 06:00:00 of the next day (d+1).

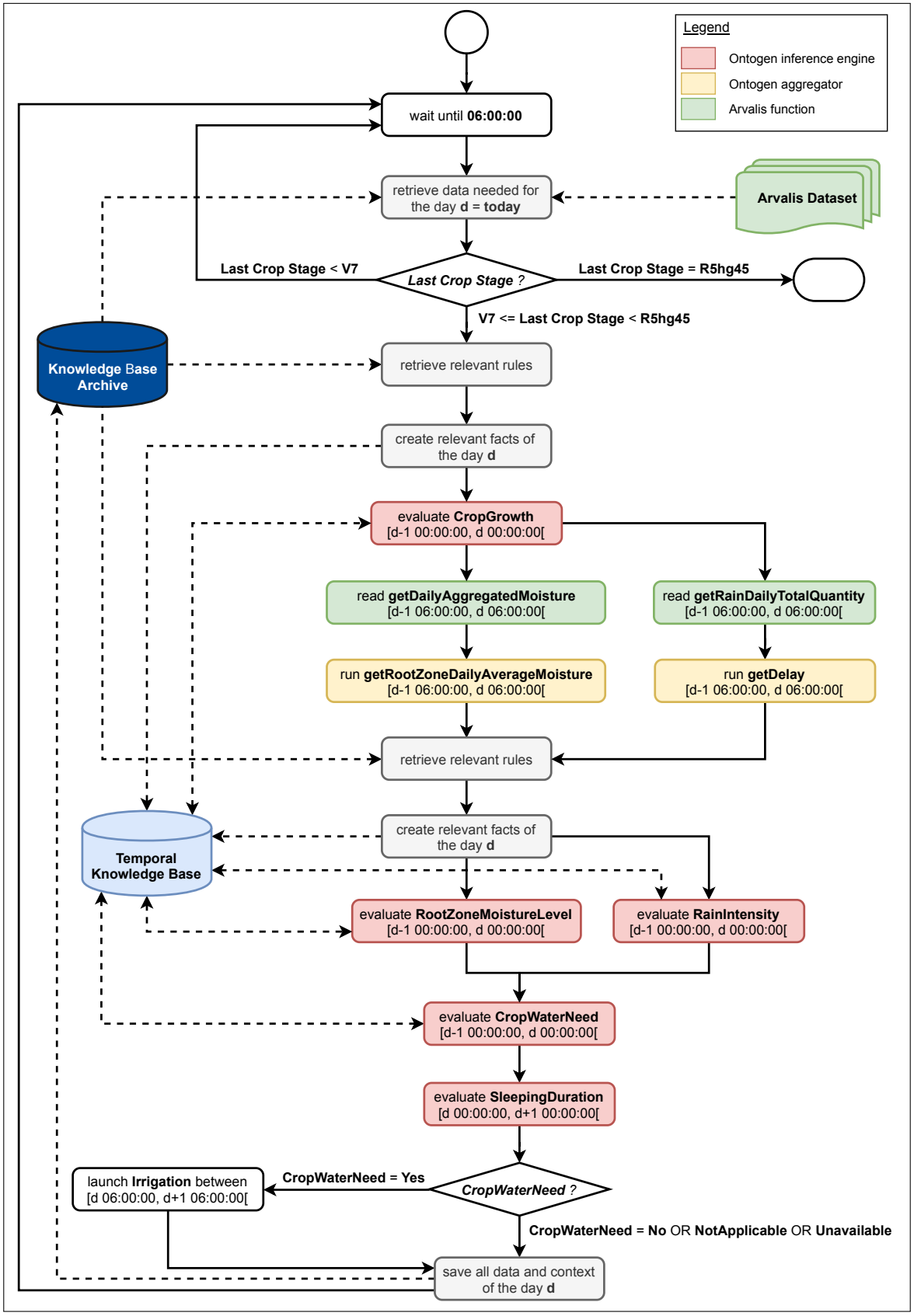

Figure 7. Algorithm of the DSS.

4.4. CAS Testing

CASO and IRRIG are examined using an ontological testing tool. The project also develops a group of unit tests and sample test scenarios to verify the working mechanism of the system.

4.4.1. Ontology Evaluation

The presented ontologies have been evaluated following three approaches: checking the absence of bad practices, coverage and data-oriented validation.

To check the absence of bad practices or common errors in the ontology, the online pitfall scanner OOPS! [25] has been applied to IRRIG and CASO. After executing the system, the following changes were made: (1) define some properties as the inverse of corresponding properties, (2) add metadata as `rdfs:label` and `rdfs:comment` for ontology elements, and (3) properly enter some ontology elements as OWL classes or properties.

Regarding the competency questions coverage, the IRRIG ontology, which as already mentioned contains CASO, has been compared to the CQ previously defined. For each CQ, the ontology elements need to answer the questions that have been identified.

Finally, we checked whether the data from our use case could be annotated with the ontologies. To do this, an RDF file representing one day of measurements from the Arvalis dataset in 2013 was generated. Some examples from this file are outlined in detail in Section 5.

4.4.2. System Testing

The project develops multiple unit tests and sample tests to evaluate the rules and aggregations. The sample test scenarios contain specific data from the Arvalis dataset.

- 20 test scenarios for crop growth workflow.
- 14 test scenarios for rain intensity workflow.
- 21 test scenarios for soil moisture workflow.
- 83 test scenarios for crop water need workflow.

4.5. CAS Maintenance

CASO and IRRIG are maintained based on a GitHub repository, and the smart irrigation project is maintained using a GitLab repository.

4.5.1. Ontology Publication and Maintenance

The key factor in maintaining the two ontologies is the interactions between our maintainer team and the community. Then, the CASO and IRRIG ontologies must be published online. The publication of the CASO and IRRIG ontologies relies on the https://w3id.org/ permanent identifiers initiative. The ontology publication follows the content negotiation mechanism. The URIs https://w3id.org/def/caso# and https://w3id.org/def/irrig# identify the two ontologies. The content negotiation mechanism deployed to publish the ontologies was transparent to ontology developers, as it was managed by OnToology [26]. OnToology is a web-based system that builds on top of Git-based environments and integrates a set of tools for documentation, evaluation and publication activities. The HTML published was generated by OnToology's integrated version of Widoco [27] and manually updated to include information about the ontology using a general description and diagrams.

The issue tracker provided by GitHub is the tool used to receive feedback and control issue lists. In addition to this portal, CASO and IRRIG developers also provide maintainer email addresses that enable another method for the community to contact.

In our road map, IRRIG is maintained for five years. CASO will be maintained even longer.

4.5.2. System Maintenance

A smart irrigation CAS is still in development by our team. We have implemented and tested all the processes located on the server. Now, we need to improve the development of the sensor and actuator network. To maintain the system and continue with development, we use a private GitLab repository. This repository is open to identified users. Thus, those users can indicate issues to the development team using the issue tracker.

5. Example of CAS Development

The development of CAS produces complex design works of many entities related to rain intensity, crop growth, root zone moisture level and crop water needs. Figure 8 shows the four workflows in this project. In a limited scope, this paper presents a part of the design activity that shows the transformation from soil moisture measurement to the state of the RootZoneMoistureLevel property. We select this transformation because it contains a temporal aggregation and a spatial aggregation. Moreover, the data used as the illustration in this part is the date 14/08/2013 in the Arvalis data.

A tensiometer measures the soil tension [28]. The soil tension affects the water extraction capability of crops:

- When the soil tension is low, the crops require less energy to extract soil water. In other words, when a tensiometer returns a high value, the soil moisture level is low.
- When the soil tension is high, the crops must use a large amount of energy to extract soil water.

The IRRINOV® method evaluates the water needs of crops by estimating the root zone moisture. Root zone moisture is the quantity of "water remaining in the depth of soil accessed by a plant" [29]. It depends on the soil type, root depth and irrigation method. Its goal is to evaluate the amount of irrigation water required by the plants [30]. The system uses six tensiometers to evaluate the root zone moisture.

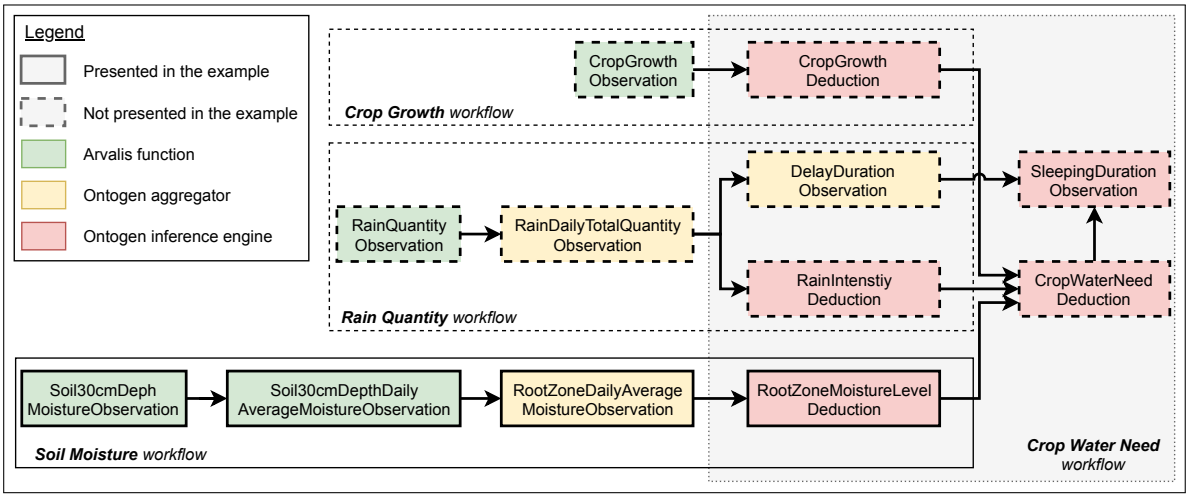

Figure 8. Four workflows of the CAS.

RootZoneMoistureLevel is an observational property of the root zone feature of interest. The RootZoneMoistureLevel property is qualitative data represented by a state. These states are determined based on tensiometer measurements. Each tensiometer is placed at a specific depth in the soil. Thus, the RootZoneMoistureLevel property depends on the following quantitative data properties:

- Soil30cmDepthMoisture: the moisture of a new feature of interest that represents the soil layer at a 30 cm depth (Soil30cmDepth).

- Soil60cmDepthMoisture: the moisture of a new feature of interest that represents the soil layer at a 60 cm depth (Soil60cmDepth).
- Soil30cmDepthDailyAverageMoisture: the daily average moisture of the feature of interest Soil30cmDepth. It equals the daily average of the property Soil30cmDepthMoisture.
- Soil60cmDepthDailyAverageMoisture: the daily average moisture of the feature of interest Soil60cmDepth. It equals the daily average of the property Soil60cmDepthMoisture.
- RootZoneDailyAverageMoisture: the daily average moisture of the root zone feature of interest. It represents a spatial aggregation of Soil30cmDepthDailyAverageMoisture and Soil60cmDepthDailyAverageMoisture properties.

The section aims to present some activities of the mini-waterfall methodology. Please note that the specification activity is already described in Section 4.1. This activity has produced a set of requirements. The design activity is divided into two parts: process conceptualization and ontology conceptualization. The ontology conceptualization activity uses ontologies to model the input and output of each process previously defined. The implementation activity presents the rule associated with the soil moisture stream. Finally, the testing activity shows some typical test cases.

5.1. Processes Conceptualization

This subsection presents in detail the processes related to the soil moisture stream. A sequence of processes builds a specific workflow, as shown in Figure 8.

5.1.1. Aggregation Processes of Soil Measurement

The first aggregation process is the computation of the daily aggregated soil moisture of the soil tensiometer. In the Arvalis dataset, each observation was stored as two datapoints: (1) the daily aggregated soil moisture in cbar and (2) the date of observation (year, month, day). Three soil tensiometers observe the moisture at a 30 cm depth, and three others observe the moisture at a 60 cm depth. The soil tensiometers used in the IRRINOV® method are Watermark probes. Watermark probe p measures the soil moisture at time t of day d. The unit of the Watermark probe is the cbar. Please note that a measurement value of 199 cbar indicates an error in the measurement process. Thus, the cleaning process is applied to remove all 199 cbar values. The function getValidMoistureProbe(p,t) returns only the valid moisture value measured by probe p at time t. If the moisture value is not valid, the function returns nothing. The function getDailyAggregatedMoisture(p,d) returns the average value of all the valid moisture values measured by probe p during the interval `[d 00:00:00, d+1 00:00:00[`. Table 3 presents the calculation of getDailyAggregatedMoisture(p,d) for one probe. Please note that the data recorded in the Arvalis data are, in fact, the value of the function getDailyAggregatedMoisture(p,d).

Table 3. The function getDailyAggregatedMoisture(p,d).

$$getDailyAggregatedMoisture(p,d) = \frac{\sum_{i=1}^{n} getValidMoistureProbe(p,t_i)}{n}$$

on the condition that $t_i \subset$ [d 00:00:00, d+1 00:00:00[

where

- d: a specific day, e.g., 14/08/2013
- t_i: a specific instant of day d, e.g., 08:00:00 on 14/08/2013
- getValidMoistureProbe(p, t_i): the valid moisture value measured by the probe p at the specific time t_i.
- n: the number of valid moisture values measured during day d.

Suppose that p1, p2, and p3 are the three probes at a 30 cm depth, and p4, p5, and p6 are the three probes at a 60 cm depth. The function getRootZoneDailyAverageMoisture(d) returns the sum of the

two median values: (1) the median of the getDailyAggregatedMoisture(p, d) values of three probes at a 30 cm depth, and (2) the median of the getDailyAggregatedMoisture(p, d) values of three probes at a 60 cm depth. Table 4 presents the calculation of getRootZoneDailyAverageMoisture(d).

Table 4. The function getRootZoneDailyAverageMoisture(d).

$$\begin{aligned} getRootZoneDailyAverageMoisture(d) = Median(&getDailyAggregatedMoisture(p1, d),\\ &getDailyAggregatedMoisture(p2, d),\\ &getDailyAggregatedMoisture(p3, d))\\ +Median(&getDailyAggregatedMoisture(p4, d),\\ &getDailyAggregatedMoisture(p5, d),\\ &getDailyAggregatedMoisture(p6, d)) \end{aligned}$$

where

- p1, p2, and p3: three watermark probes placed at a 30 cm depth
- p4, p5, and p6: three watermark probes placed at a 60 cm depth
- getDailyAggregatedMoisture(p_i,d): returns the average moisture value measured by the probe p_i during the interval [d 00:00:00, d+1 00:00:00[(see Table 3)
- Median(x, y, z): is the median value of x, y and z.

5.1.2. State of RootZoneMoistureLevel Property

The output of rules are the daily states of the RootZoneMoistureLevel property. For each day, a state should be determined. The list of possible states is as follows:

- RootZoneMoistureLevel.Init: This state represents the fact that the function getRootZoneDailyAverageMoisture(d) will be evaluated from new measurements of tensiometers. The tensiometers will perform several measurements per day. This state is not an output of the automata. It is a transitional state before determining the new state of the RootZoneMoistureLevel property
- RootZoneMoistureLevel.Saturated: The root zone is saturated by water. Crops do not need extra water. Moreover, any machine is allowed to access the plot.
- RootZoneMoistureLevel.VeryHigh: The root zone contains a significantly high amount of water. Crops do not need water regardless of its growth stage.
- RootZoneMoistureLevel.High: The root zone contains a high amount of water. Crops need water at specific growth stages.
- RootZoneMoistureLevel.Average: The root zone contains an average amount of water. Crops need water at specific growth stages.
- RootZoneMoistureLevel.Low: The root zone contains a low amount of water. Crops need water at specific growth stages.
- RootZoneMoistureLevel.VeryLow: The root zone contains an insignificant amount of water. Crops need water at specific growth stages.
- RootZoneMoistureLevel.Dry: The root zone is dry. Crops need water regardless of its growth stage.

The states of the RootZoneMoistureLevel property follow the order VeryLow < Low < Average < High < VeryHigh < Saturated. Those states follow the inverse order of the value of the function getRootZoneDailyAverageMoisture(d).

5.1.3. Threshold

Certain thresholds are used to determine the state of the RootZoneMoistureLevel property from the result of the function getRootZoneDailyAverageMoisture(d). Please note that these thresholds depend on the variety of maize crops and soil types.

- Th_ST_Minimum: The smallest value measured by a Watermark probe. In the Watermark's specification, it is fixed at 0 cbar.
- Th_ST_Saturation: This threshold value in the Arvalis experimentation is 10 cbar.
- Th_ST_VeryHigh: This threshold value in the Arvalis experimentation is 70 cbar.
- Th_ST_High: This threshold value in the Arvalis experimentation is 120 cbar.
- Th_ST_Average: This threshold value in the Arvalis experimentation is 140 cbar.
- Th_ST_Low: This threshold value in the Arvalis experimentation is 150 cbar.
- Th_ST_VeryLow: This threshold value in the Arvalis experimentation is 160 cbar.
- Th_ST_Maximum: This is set to 301 cbar, which is higher than any measurement value provided by any Watermark probe. The maximal measurement value that a Watermark probe can reach ranges from 0 to 200. However, to compute the soil tension, the manufacturer provides a coefficient. The measurement value is multiplied by the coefficient to obtain the soil tension value. The maximum coefficient provided by Watermark's manufacturer is 1.5 (from 2002 until now), and the maximal soil tension value is 300 cbar (200 cbar * 1.5).

The relations between thresholds and states of the RootZoneMoistureLevel property are presented in Figure 9. The blue squares represent the state of the RootZoneMoistureLevel property. The green squares represent the soil tension thresholds. The dashed line squares represent a RootZoneDailyAverageMoisture(d) value. Each of the squares' positions show the upper threshold and the lower threshold. For example, if the RootZoneDailyAverageMoisture(d) value is superior or equal to Th_ST_Low (150) and inferior to the Th_ST_VeryLow (160), then the RootZoneMoistureLevel state is VeryLow.

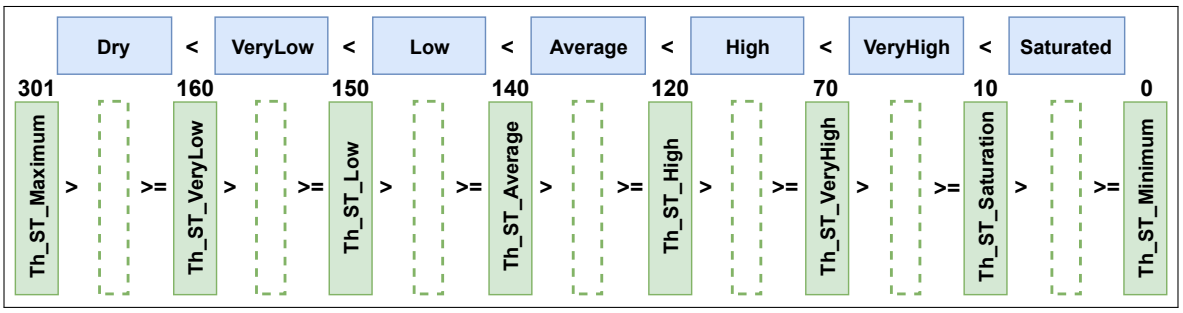

Figure 9. Relation between states and thresholds of root zone moisture level.

5.1.4. State Diagram

The state diagram presents the inference conception to deduce the state of the RootZoneMoistureLevel property based on the result of the function getRootZoneDailyAverageMoisture(d). In the diagram illustrated in Figure 10, there are eight states and ten transitions. The variable t_{soil} represents a clock dedicated to the RootZoneMoistureLevel property that has a time step of one day.

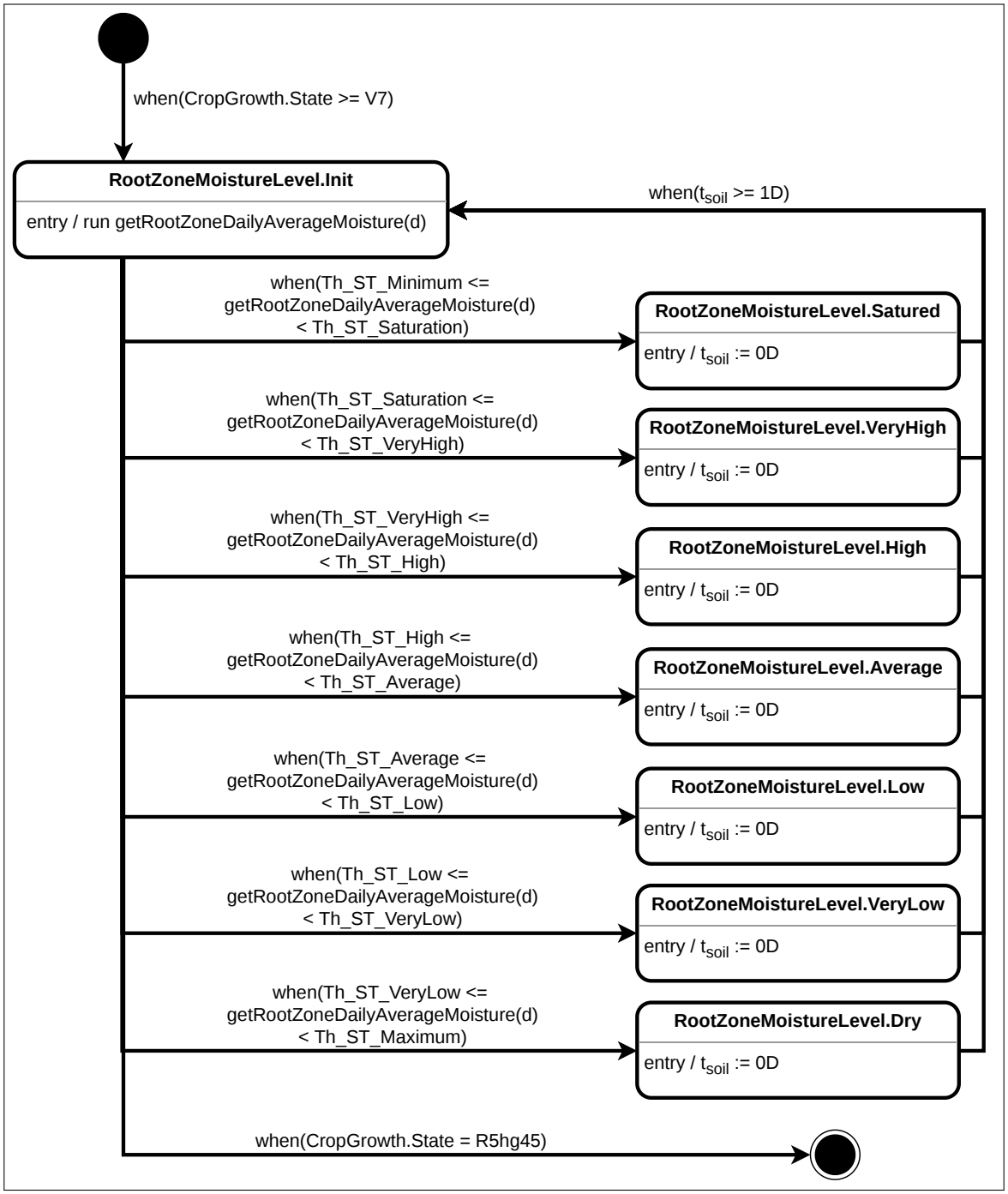

Figure 10. Automata of RootZoneMoistureLevel states and their transitions.

The transition from the initial state towards the state Init indicates when the state of the CropGrowth property is V7 or any upper state; then, the RootZoneMoistureLevel property reaches the state Init. After reaching this state, a new evaluation of the function getRootZoneDailyAverageMoisture(d) is performed based on daily tensiometer measurements.

The transition from the state Init towards the state Saturated indicates that when the value of the function RootZoneDailyAverageMoisture(d) is equal or superior to Th_ST_Minimum and inferior to Th_ST_Saturation, then the RootZoneMoistureLevel property reaches the Saturated state. After reaching this state, the clock t_{soil} is reset.

The transition from the state Init towards the state VeryHigh indicates that when the value of the function getRootZoneDailyAverageMoisture(d) is equal or superior to Th_ST_Saturation and inferior

to Th_ST_VeryHigh, then the RootZoneMoistureLevel property reaches the VeryHigh state. After reaching this state, the clock t_{soil} is reset.

The transition from the state Init towards the state High indicates that when the value of the function getRootZoneDailyAverageMoisture(d) is equal or superior to Th_ST_VeryHigh and inferior to Th_ST_High, then the RootZoneMoistureLevel property reaches the High state. After reaching this state, the clock t_{soil} is reset.

The transition from the state Init towards the Average state indicates that when the value of the function getRootZoneDailyAverageMoisture(d) is equal or superior to Th_ST_High and inferior to Th_ST_Average, then the RootZoneMoistureLevel property reaches the Average state. After reaching this state, the clock t_{soil} is reset.

The transition from the state Init towards the state Low indicates when the value of the function getRootZoneDailyAverageMoisture(d) is equal or superior to Th_ST_Average and inferior to Th_ST_Low, then the RootZoneMoistureLevel property reaches the Low state. After reaching this state, the clock t_{soil} is reset.

The transition from the state Init towards the state VeryLow indicates that when the value of the function getRootZoneDailyAverageMoisture(d) is equal or superior to Th_ST_Low and inferior to Th_ST_VeryLow, then the RootZoneMoistureLevel property reaches the VeryLow state. After reaching this state, the clock t_{soil} is reset.

The transition from the state Init towards the state Dry indicates that when the value of the function getRootZoneDailyAverageMoisture(d) is equal or superior to Th_ST_VeryLow and inferior to Th_ST_Maximum, then the RootZoneMoistureLevel property reaches the Dry state. After reaching this state, the clock t_{soil} is reset.

The transition from one state among Saturated, VeryHigh, High, Average, Low, VeryLow and Dry towards the state Init indicates that after one day ((t_{soil} >= 1)?), a state change is possible; in other words, a state is reached for a whole day.

The transition from the state Init towards the final state indicates that when the CropGrowth property is at the state R5hg45, then moisture measurements are terminated.

5.2. Ontology Conceptualization

The ontology IRRIG is used to model the input and output of our DSS. During the decision process, we need to model the following:

- The measurements of six tensiometers.
- The average of the measurements provided by a tensiometer for a whole day.
- The computation of the root zone moisture per day.
- The output of the deduction process, as the states of the RootZoneMoistureLevel property deduced by the rule engine.

This section uses a data sample on 14/08/2013 from the Arvalis dataset to illustrate the conceptualization. The sample (https://gitlab.irstea.fr/irrig/public/tree/master/Sample/) was modeled in Turtle format.

5.2.1. Tensiometer Measurement Model

The goal of this subsection is to model the tensiometer measurements or, more precisely, the result of the function getValidMoistureProbe(p,t). A tensiometer may be placed at a 30 cm or 60 cm depth. This section describes the valid measurement of a tensiometer placed at a 30 cm depth. This model reuses some elements of the IRRIG ontology. IRRIG defined several classes and instances used for modeling tensiometer measurements:

- irrig:featureOfInterest_Soil30cmDepth: this individual represents the soil layer at a 30 cm depth. This individual is an instance of the class **sosa:FeatureOfInterest**. The individual irrig:featureOfInterest_soil30cmDepth is linked to the individual

irrig:observableProperty_soil30cmDepth_moisture by the `sosa:hasProperty` object property (see Figure 11).

- **irrig:MoistureProperty**: this class is a subclass of **ssn:Property**. It represents the amount of water available inside the entity. The moisture aspect of the entity is intrinsic to and cannot exist without the entity. The moisture quality value should be expressed by a qualitative value such as a state or by a quantitative value (see Figure 11).
- irrig:observationProperty_soil30cmDepth_moisture: this individual represents the moisture property of the soil layer at a 30 cm depth. It is an instance of the class **irrig:MoistureProperty** (see Figure 11).
- **irrig:Soil30cmDepthMoistureObservation**: this class is a subclass of **caso:Observation**. It is a defined class that represents the act of carrying out a procedure (observation) to estimate or calculate a value of a moisture property related to the soil layer at a 30 cm depth. This moisture property is represented by the individual irrig:observableProperty_soil30cmDepth_moisture. Two necessary and sufficient conditions are defined: (1) All instances of the class **irrig:Soil30cmDepthMoistureObservation** are linked to the individual irrig:observableProperty_soil30cmDepth_moisture via the `sosa:observedProperty` object property; (2) All instances of **irrig:Soil30cmDepthMoistureObservation** are linked to the individual irrig:featureOfInterest_soil30cmDepth via the `sosa:hasFeatureOfInterest` object property. Each instance of **irrig:Soil30cmDepthMoistureObservation** is linked to a sensor to describe which tensiometer made the observation and how.

First, the measurement data are stored in the system under the IRRIG model. Figure 11 presents an example of one measurement from one soil tensiometer.

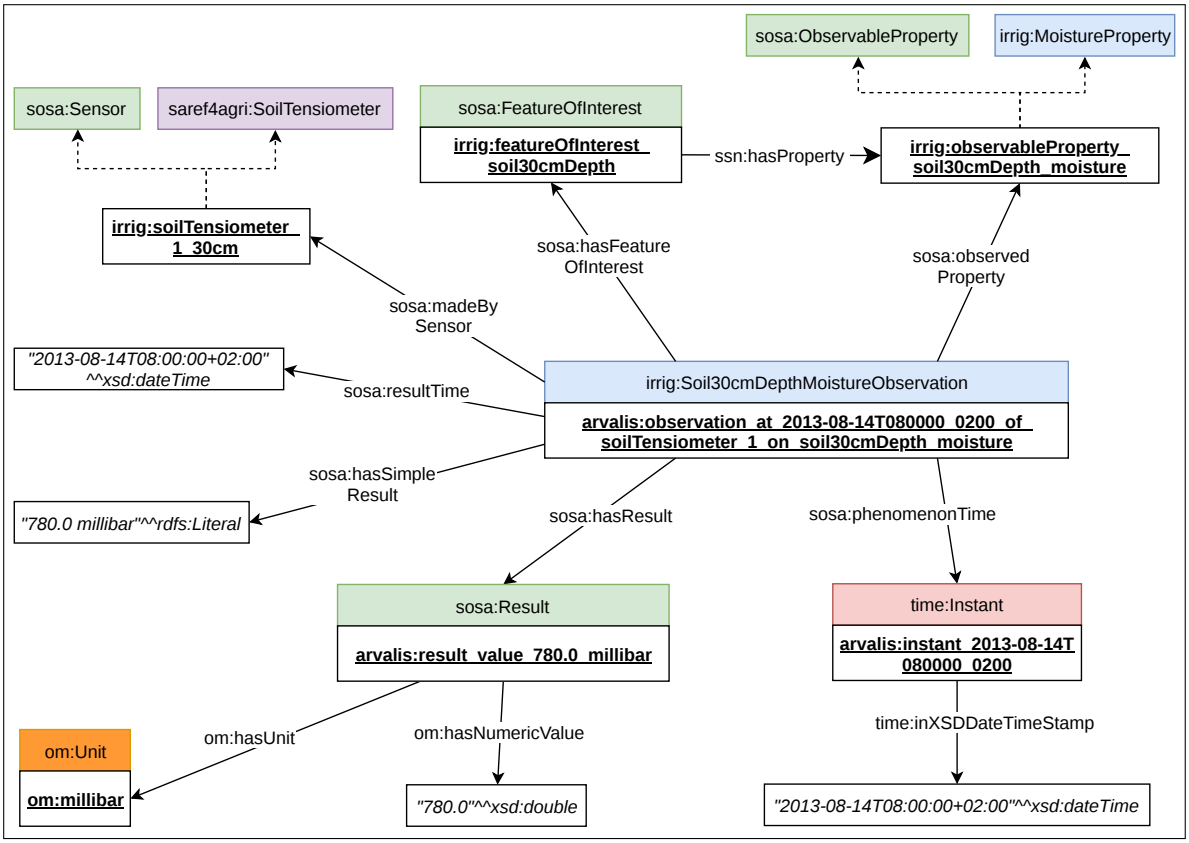

Figure 11. Example of an observation of a soil tensiometer.

An instance of the class **irrig:Soil30cmDepthMoistureObservation** represents the computation of the function getValidMoistureProbe(p,t). It is an observation of a moisture property related to the soil layer at a 30 cm depth, as shown in Figure 11. By definition, individuals of **irrig:Soil30cmDepthMoistureObservation** are linked to the individual irrig:observedProperty_soil30cmDepth_moisture, an instance of the class **irrig:MoistureProperty**, and to the individual irrig:FeatureOfInterest_Soil30cmDepth, an instance of the class **sosa:FeatureOfInterest**.

The observation individual is linked to the sensor that made the observation. We assumed that the node, which contains a tensiometer probe, also has the capacity to clean the probe measurements. A node tensiometer is represented by an instance of the class **sosa:Sensor**. It is also an instance of the class **saref4agri:SoilTensiometer**. Arvalis identifies their tensiometers by a number and depth, as presented in Section 4.1. We reuse this information to create a name for each instance of **saref4agri:SoilTensiometer**. For example, Figure 11 contains the individual arvalis:soilTensiometer_1_30cm.

A tensiometer measurement is an instantaneous observation. Please note that the measurements of tensiometers are unavailable in the Arvalis dataset. Thus, we can imagine that tensiometer measurements may happen at any time of the day. Figure 11 presents a measurement that happens at 08:00:00 on 14/08/2013. To store the time of the measurement, we create an instance of the class **time:Instant**. The *time:inXSDDateTimeStamp* data property stores the associated date time value. The `sosa:phenomenonTime` object property links the observation individual to the instant individual.

To present the time when the node tensiometer generates the measurement, we use the *sosa:resultTime* datatype property to links the observation individual to the `xsd:dateTime` value corresponding to a precise time. In Figure 11, this precise time is at 08:00:00 on 14/08/2013.

The result of tensiometer measurement is presented by an instance of the class **sosa:Result**. This result is described by a value and a unit. The instance of **sosa:Result** is linked to the data value via the *om:hasNumericValue* datatype property. The instance of **sosa:Result** is linked to the individual om:millibar, an instance of the class **om:Unit**, via the `om:hasUnit` object property. Please note that in the Arvalis dataset, the unit of tensiometer measurement is the cbar. However, to the best of our knowledge, this unit has not yet been defined in any ontology. Thus, we prefer to reuse the millibar (mbar), a unit that is already defined in well-known ontologies such as OM. For this reason, this model uses the millibar as the unit for moisture, as shown in Figure 11.

5.2.2. Function getDailyAggregatedMoisture Model

The goal of this subsection is to model the result of the function getDailyAggregatedMoisture(p,d). This function computes the daily average of all the measurements provided by one tensiometer. IRRIG defines several classes and instances used for modeling the daily average of tensiometer measurements:

- irrig:observationProperty_soil30cmDepth_dailyAverageMoisture: this individual represents the daily average of the moisture property related to the soil layer at a 30 cm depth. It is an instance of the class **irrig:MoistureProperty**. This individual is linked to the individual irrig:featureOfInterest_soil30cmDepth (see Figure 12).
- **irrig:Soil30cmDepthDailyAverageMoistureObservation**: this class is a subclass of **caso:Observation**. It is a defined class that represents the act of carrying out a procedure (observation) to calculate the average per day of the moisture property. This specific property is represented by the individual irrig:observableProperty_soil30cmDepth_dailyAverageMoisture. Two necessary and sufficient conditions are defined: (1) All instances of the class **irrig:Soil30cmDepthDailyAverageMoistureObservation** are linked to the individual irrig:observableProperty_soil30cmDepth_dailyAverageMoisture via the `sosa:observedProperty` object property; (2) All instances of **irrig:Soil30cmDepthDailyAverageMoistureObservation** are linked to the individual irrig:featureOfInterest_soil30cmDepth via the `sosa:hasFeatureOfInterest` object property.

Each instance of **irrig:Soil30cmDepthDailyAverageMoistureObservation** is linked to a sensor to describe which agent made the observation and how.

Figure 12 presents an example of the getDailyAggregatedMoisture(p,d) result.

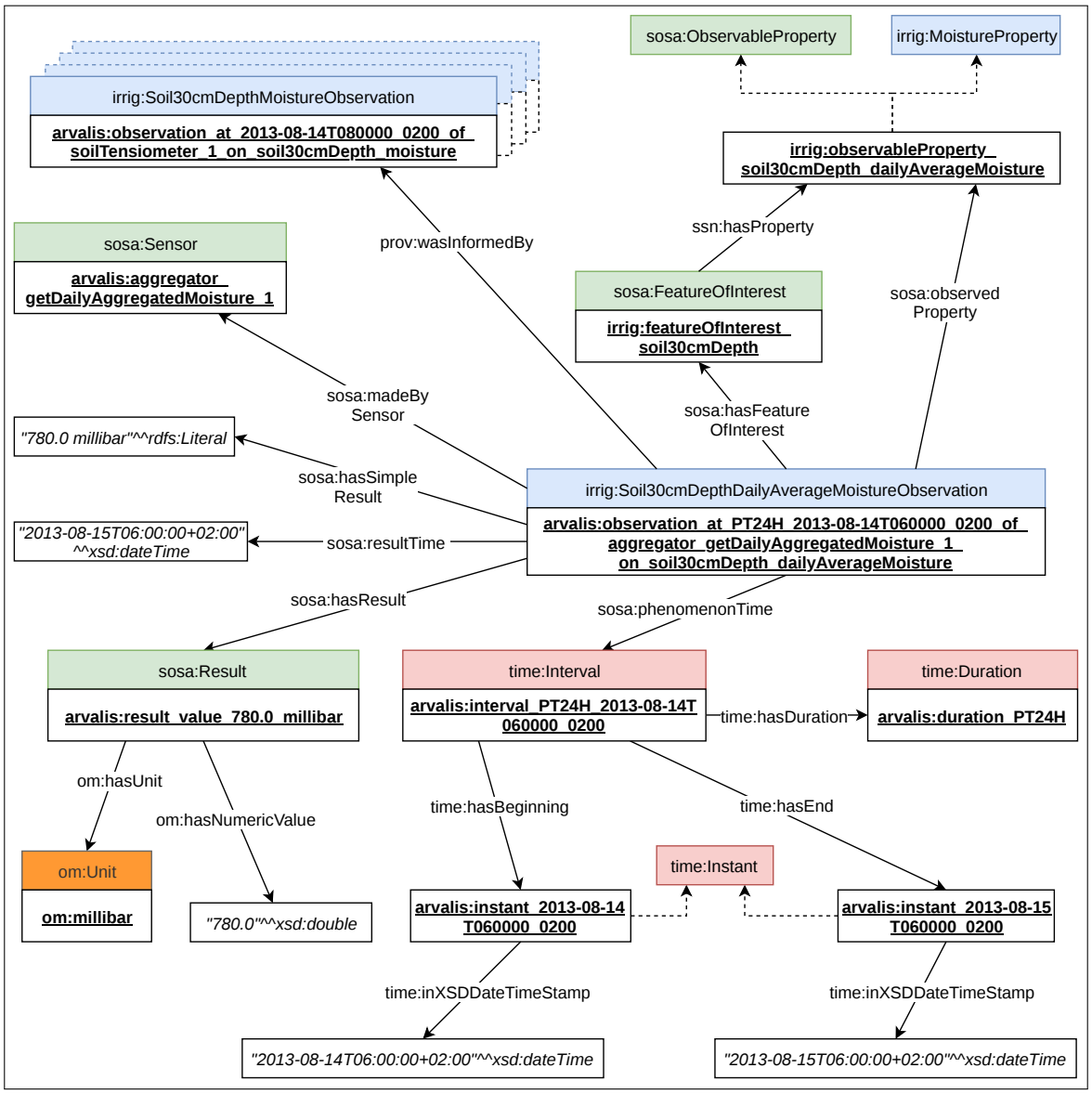

Figure 12. Example of the computation of getDailyAggregatedMoisture(p,d).

The instance of the class **irrig:Soil30cmDepthDailyAverageMoistureObservation** represents the computation of the daily average of all measurements provided by one tensiometer, as shown in Figure 12. By definition, individuals of **irrig:Soil30cmDepthMoistureObservation** are linked to the individual irrig:observedProperty_soil30cmDepth_dailyAverageMoisture, an instance of the class **irrig:MoistureProperty**, and to the individual irrig:featureOfInterest_soil30cmDepth.

The observation individual is linked to the sensor that produces the result, i.e., a software program that computes the daily average of several measurements. We do not have information on the agent that produces this computation in the Arvalis dataset. It is possible that the same software, located on a server, computes the average for all the measurements. Alternatively, the IRRINOV station may contain two different processing devices. Each device can compute the average for all the tensiometers located at the same depth. To avoid errors, we create a software agent for each tensiometer based on the tensiometer name: arvalis:aggregator_getDailyAggregatedMoisture_1. The individual that

represents the software is an instance of the class **sosa:Sensor**. The observation individual is linked to the software agent via the `sosa:madeBySensor` object property, as shown in Figure 12.

The software program produces the result once per day at a precise time. In the Arvalis dataset, there is no information about the hour, minute and second of the computation. For synchronization purposes, we fix it to the time of the deduction computation `[d+1 06:00:00]`. At this time, all the tensiometer measurements were available for the computation of the average. To store the time of the computation, we use the *sosa:resultTime* datatype property to links the observation individual to the `xsd:dateTime` corresponding to the precise time `[d+1 06:00:00]`. This is presented in Figure 12.

The computation calculates the daily average. Figure 12 presents the average for the 14/08/2013. We represent the day as a period of 24 h: `[d 00:00:00, d+1 00:00:00[`. Thus, we create an instance of the class **time:Interval**, which is described by two instances of **time:Instant**. The `sosa:phenomenonTime` object property links the observation individual to the **time:Interval** instance.

A computation of daily average is based on several tensiometer measurements. Thus, the instance of the class **irrig:Soil30cmDepthDailyAverageMoistureObservation** is linked to several instances of the class **irrig:Soil30cmDepthMoistureObservation** via the `prov:wasInformedBy` object property, as shown in Figure 12.

The result of the computation is represented by an instance of the class **sosa:Result**. This instance is described by a unit and a value, as shown in Figure 12.

5.2.3. Function getRootZoneDailyAverageMoisture Model

The goal of this subsection is to model the result of the function getRootZoneDailyAverageMoisture(d). The value represents the evaluation of average daily moisture related to the root zone. First, the computation is relative to the daily average moisture property, as presented in the previous section. Second, the feature of interest is no more a soil layer at a specific depth, but a volume of soil named root zone, as explained in the introduction paragraph. IRRIG defines several classes and instances used for modeling the measurements of the daily average moisture of root zone:

- irrig:featureOfInterest_rootZone: this individual represents the root zone. This individual is an instance of the class **sosa:FeatureOfInterest**. The individual irrig:featureOfInterest_rootZone is linked to the individual irrig:observableProperty_rootZone_dailyAverageMoisture by the `sosa:hasProperty` object property (see Figure 13).
- irrig:observationProperty_rootZone_dailyAverageMoisture: this individual represents the daily average of the moisture property related to the root zone. It is an instance of the class **irrig:MoistureProperty** (see Figure 13).
- **irrig:RootZoneDailyAverageMoistureObservation**: this class is a subclass of **caso:Observation**. It is a defined class that represents the act of carrying out a procedure (observation) to calculate the daily average of the moisture property related to the root zone. The specific moisture property is represented by the individual irrig:observableProperty_rootZone_dailyAverageMoisture. Two necessary and sufficient conditions are defined: (1) All instances of the class **irrig:RootZoneDailyAverageMoistureObservation** are linked to the individual irrig:observableProperty_rootZone_dailyAverageMoisture via the `sosa:observedProperty` object property; (2) All instances of **irrig:RootZoneDailyAverageMoistureObservation** are linked to the individual irrig:featureOfInterest_rootZone via the `sosa:hasFeatureOfInterest` object property. Each instance of **irrig:RootZoneDailyAverageMoistureObservation** is linked to a sensor to describe which agent made the observation and how.

Figure 13 presents an example of the getRootZoneDailyAggregatedMoisture(d) result.

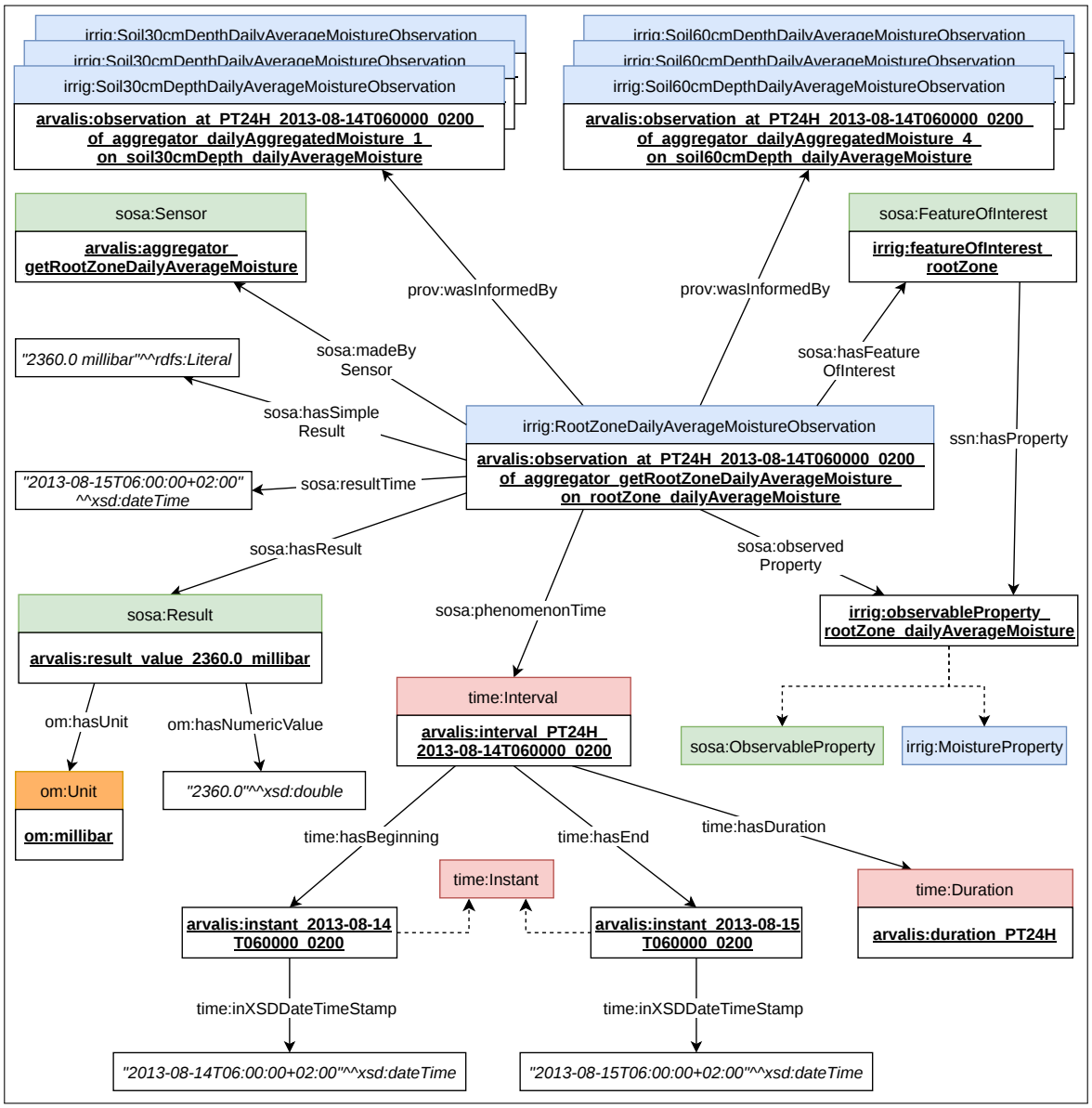

Figure 13. Example of the computation of getRootZoneDailyAverageMoisture(d).

An instance of the class **irrig:RootZoneDailyAverageMoistureObservation** represents the computation of the daily average moisture for the root zone. By definition, individuals of **irrig:RootZoneDailyAverageMoistureObservation** are linked to the individual irrig:observedProperty_rootZone_dailyAverageMoisture, an instance of the class **irrig:MoistureProperty**, and to the individual irrig:featureOfInterest_rootZone, an instance of the class **sosa:FeatureOfInterest**, as shown in Figure 13.

The observation individual is linked to the sensor that produces the result, i.e., a software that computes the equation presented in Table 4. The software program is named arvalis:aggregator_getRootZoneDailyAverageMoisture. This individual is an instance of the class **sosa:Sensor**. The observation individual is linked to the sensor via the `sosa:madeBySensor` object property, as shown in Figure 13.

The software computes at a precise time when all its inputs are available. There is no information about the hour, minute and second of the computation. For synchronization purposes, we fix it to the time at `[d+1 06:00:00]`. To store the time of the computation, we use the *sosa:resultTime* datatype

property to links the observation individual to the *xsd:dateTime* value corresponding to the precise time `[d+1 06:00:00]`. This is presented in Figure 13.

The computation calculates the daily average. Figure 13 presents the computation for the 14/08/2013. We represent the day as a period of 24 h: `[d 00:00:00, d+1 00:00:00[`. We reuse the instance of the class **time:Interval** defined for the description of the daily average of tensiometer measurements (see Section 5.2.2). The `sosa:phenomenonTime` object property links the observation individual to the **time:Interval** instance.

The result of the computation is presented by an instance of the class **sosa:Result**, as shown in Figure 13.

5.2.4. RootZoneMoistureLevel Deduction Model

The goal of the subsection is to model the output of the rule engine, which infers the state of the RootZoneMoistureLevel property based on the result of the function getRootZoneDailyAverageMoisture(d). IRRIG defines several classes and instances used for modeling RootZoneMoistureLevel deductions:

- irrig:ObservationProperty_RootZone_MoistureLevel: this individual represents the moisture property of the root zone. It is an instance of the class **irrig:MoistureProperty**. It is also an instance of the class **caso:Property** because its associated values are states, represented by instances of the class **irrig:RootZoneMoistureLevelState**. This individual is linked to the individual irrig:featureOfInterest_rootZone (see Figure 14). The individual irrig:ObservationProperty_RootZone_MoistureLevel is linked to an instance of the class **irrig:RootZoneMoistureLevelState** via the object property `caso:hasState`.
- **irrig:RootZoneMoistureLevelState**: this class is a subclass of **caso:State**. It represents the qualitative value of the RootZoneMoistureLevel property which changes over time, summarizing a set of information about that property. Several instances of this class are presented in Section 5.1.2.
- **irrig:RootZoneMoistureLevelDeduction**: this class is a subclass of **caso:Deduction**. It is a defined class that represents the act of carrying out a procedure (observation) to estimate the state of the moisture property for the root zone. The moisture property is represented by the individual irrig:observableProperty_rootZone_moistureLevel. Two necessary and sufficient conditions are defined: (1) All instances of **irrig:RootZoneMoistureLevelDeduction** are linked to the individual irrig:observableProperty_rootZone_moistureLevel via the `sosa:observedProperty` object property; (2) All instances of **irrig:RootZoneMoistureLevelDeduction** are linked to the individual irrig:featureOfInterest_rootZone via the `sosa:hasFeatureOfInterest` object property. Each instance of **irrig:RootZoneMoistureLevelDeduction** is linked to an instance of the class **irrig:RootZoneMoistureLevelState** by the `caso:hasResultState` object property, is linked to an instance of **irrig:RootZoneMoistureObservation** by the `prov:wasInformedBy` object property, and is linked to a sensor to describe the agent that made the deduction.

Figures 14 and 15 present an example of the RootZoneMoistureLevel deduction.

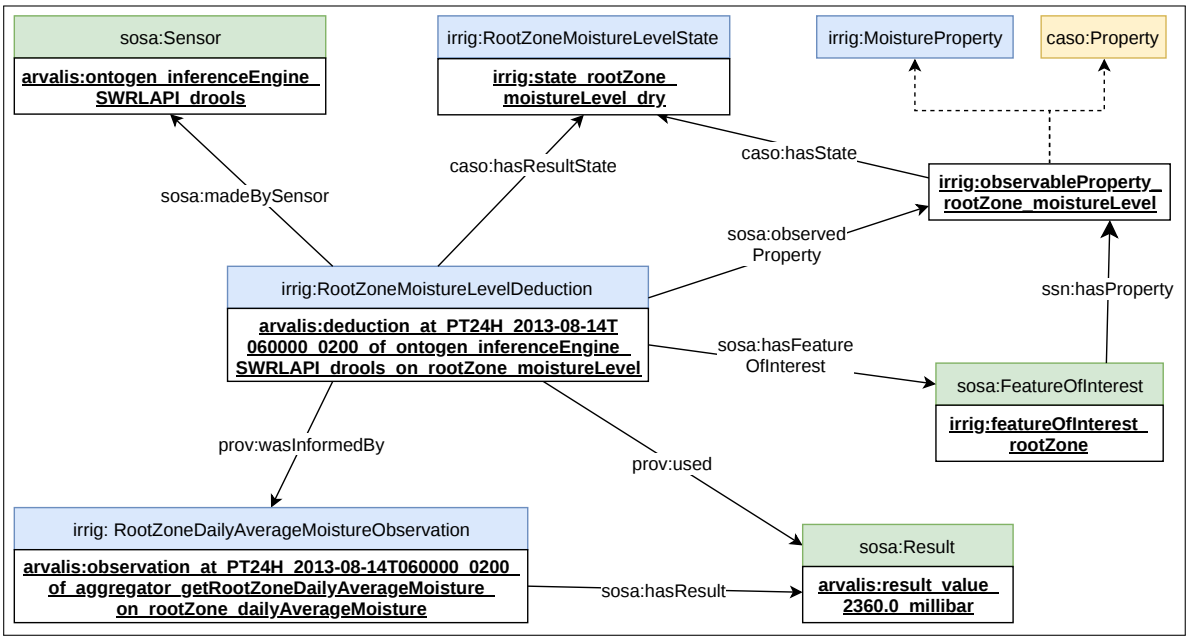

Figure 14. Example of the deduction of root zone moisture level (part 1).

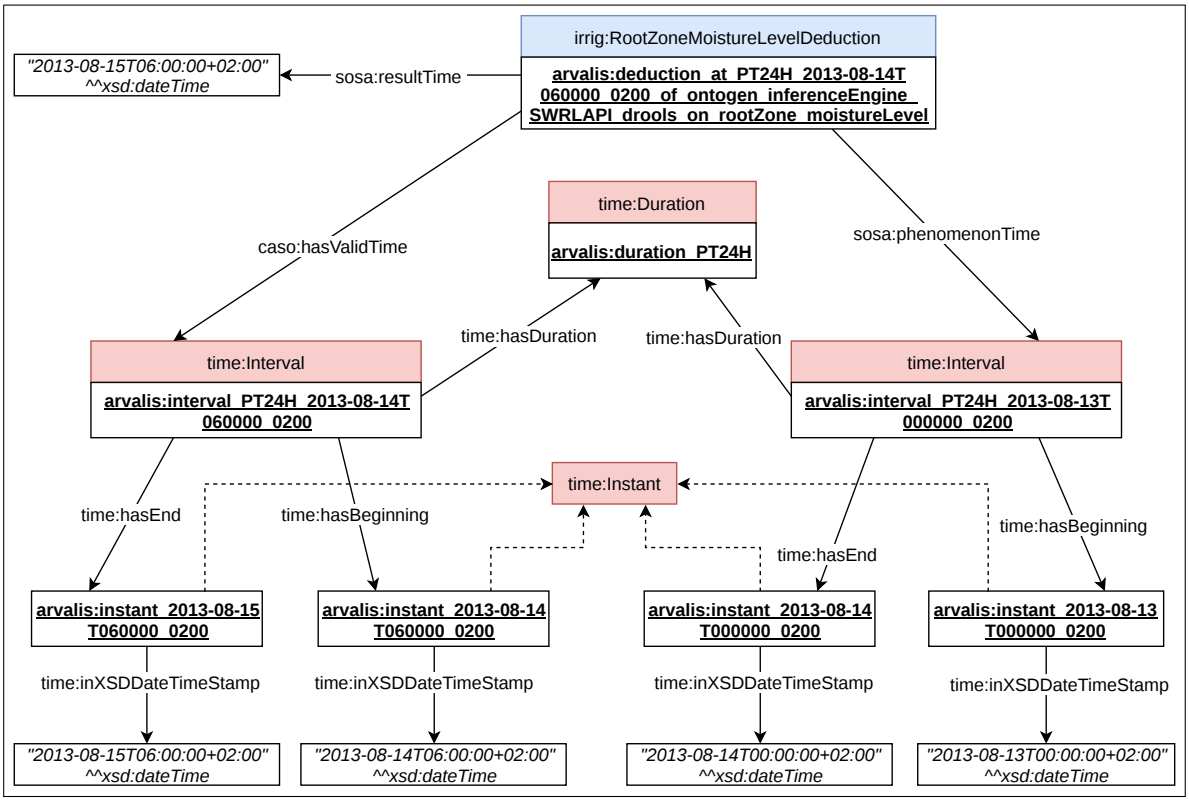

Figure 15. Example of the deduction of root zone moisture level (part 2).

An instance of the class **irrig:RootZoneMoistureLevelDeduction** represents a deduction process for the RootZoneMoistureLevel property, as shown in Figure 14. By definition, this individual is linked to the individual irrig:observedProperty_rootZone_moistureLevel, an instance of the class **irrig:MoistureProperty**, and to the individual irrig:featureOfInterest_rootZone.

The deduction individual is linked to an actor that produces the result. The actor is an SWRLAPI Drools Engine that is represented by the individual arvalis:ontogen_inferenceEngine_SWRLAPI_drools, an instance of the class **sosa:Sensor**.

The deduction is linked to the sensor via the `sosa:madeBySensor` object property, as shown in Figure 14.

A RootZoneMoistureLevel deduction is based on an observation individual that represents a computation of the function getRootZoneDailyAverageMoisture(d). The deduction instance of Figure 14 is based on the observation individual presented in Figure 13. Thus, the deduction individual is linked to the instance of the class **irrig:RootZoneDailyAverageMoistureObservation** via the `prov:wasInformedBy` object property.

The result of a RootZoneMoistureLevel deduction is an instance of the class **irrig:RootZoneMoistureLevelState**. The deduction individual is linked to its state result by the `caso:hasResultState` object property, as shown in Figure 14. The possible states are shown in Figure 16.

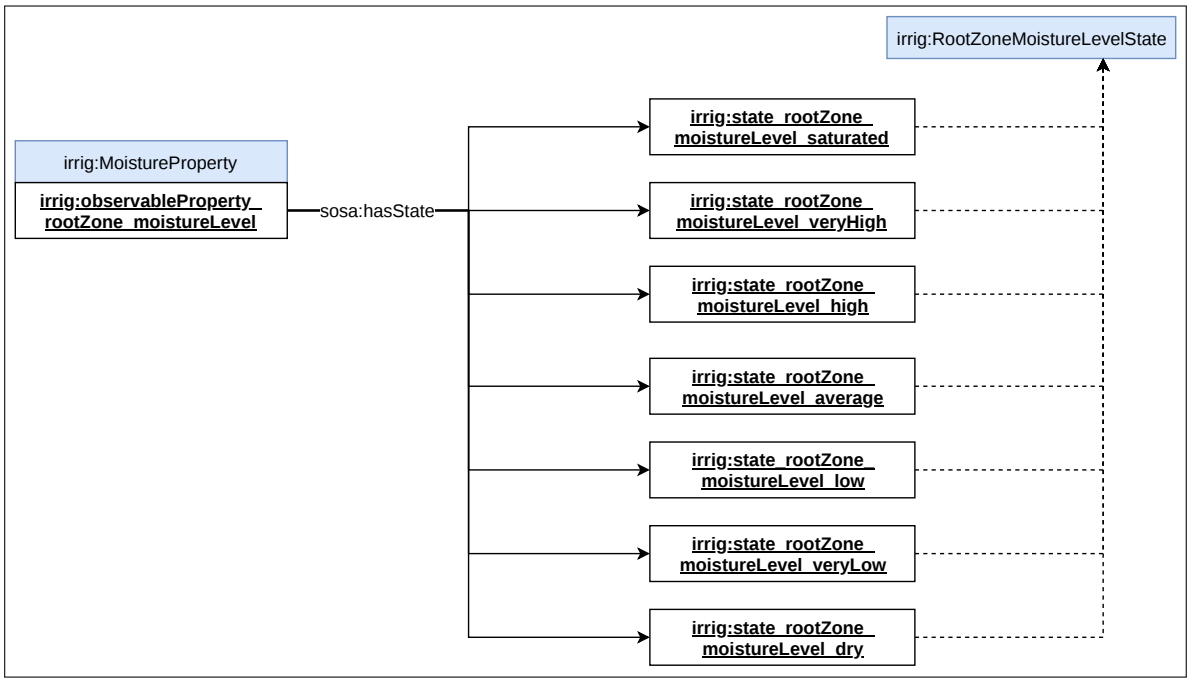

Figure 16. States of root zone moisture level property.

Each day, our system makes decisions about the value of certain properties, such as crop growth, rain intensity and root zone moisture level. These decisions are based on several sensor observations performed during day d. Day d is represented by an interval of 24 h: `[d 00:00:00, d+1 00:00:00[`. For the synchronization of computations, the frequency of such decisions should be fixed to one precise time. According to Météo-France, the record of rain quantity per day is made between day d at 06:00:00 to day d+1 at 06:00:00 [31]. This project considers that all the computations should be available at a precise time, fixed at `[d+1 06:00:00]`. Each decision regarding day d will be valid for an interval of 24 h: `[d+1 06:00:00, d+2 06:00:00[`.

Figure 15 illustrates the time period regarding a RootZoneMoistureLevel deduction. The phenomenon time represents day d. It is represented by a 24-h interval: `[d 00:00:00, d+1 00:00:00[`. In Figure 15, the time when the result of the deduction is available is named the result time. It is fixed at `[d+1 06:00:00]`. This result is valid during a period of 24 h named valid time: `[d+1 06:00:00, d+2 06:00:00[`.

To store the time of the deduction, we use the *sosa:resultTime* datatype property to links the deduction individual to the *xsd:dateTime* corresponding to the result time at `[d+1 06:00:00]`. This is presented in Figure 15.

To represent the fact that a deduction concerns day d, we use the object property `sosa:phenomenonTime`. Day d is represented by an instance of the class **time:Interval**, which is described by two instances of the class **time:Instant**:

- One instance represents the beginning of the interval: `[d 00:00:00]`. The object property `time:hasBeginning` links the instance of **time:Interval** to the beginning **time:Instant** instance.
- One instance represents the end of the interval: `[d+1 00:00:00]`. The instance of **time:Interval** is linked to the ending **time:Instant** by the object property `time:hasEnding`.

Each instance of **time:Instant** has a *time:inXSDDateTimeStamp* data value, which precisely indicates the date and time of the instant.

The deduction result for day d is valid for 24 h: `[d+1 06:00:00, d+2 06:00:00[`, i.e., no new deduction will be performed during this valid time. The valid time is represented by an instance of the class **time:Interval**, which is associated with two instances of the class **time:Instant** as presented above. The `caso:hasValidTime` object property, as illustrated in Figure 15, links the deduction individual to the valid time.

Figure 17 supports to clarify the synchronization of all the computations.

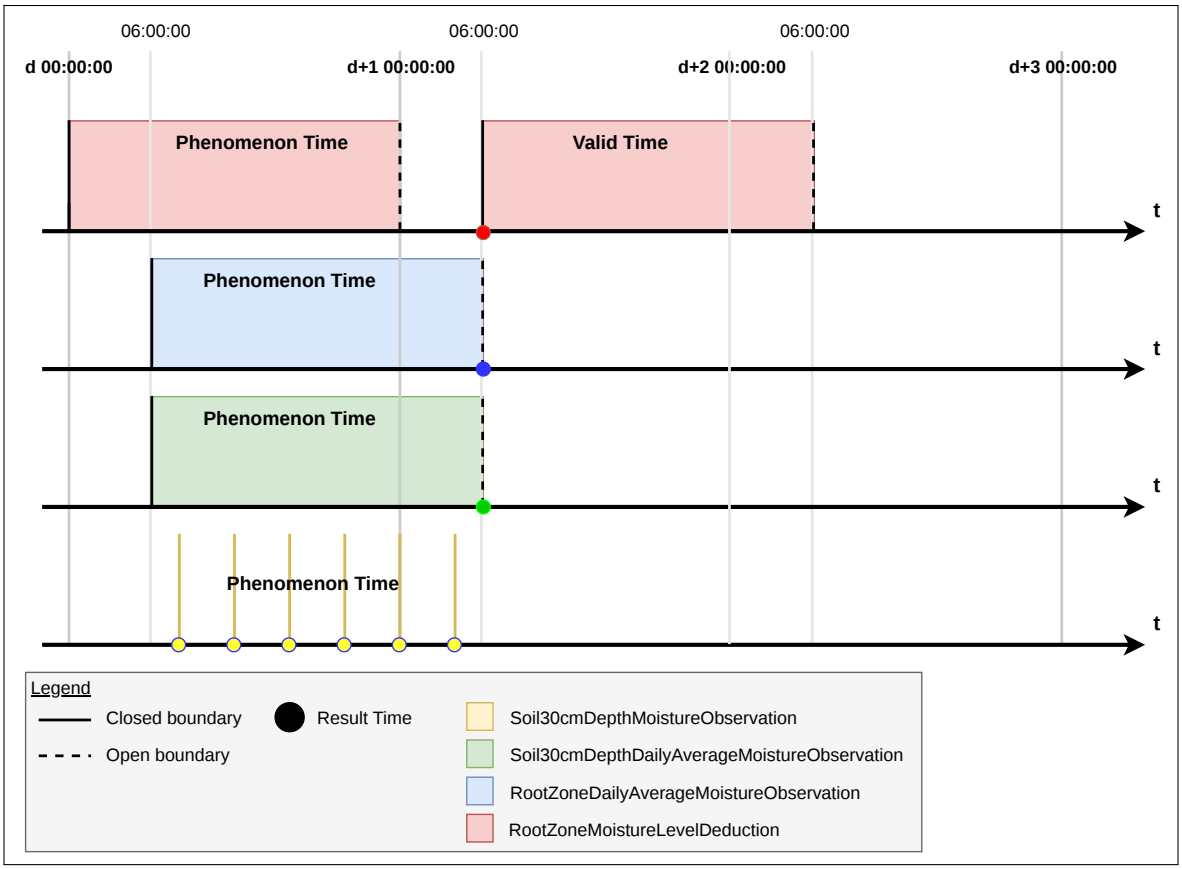

Figure 17. Time scales of the soil moisture workflow.

5.3. Implementation of the Reasoning Process in the Soil Moisture Workflow

In this case, only one rule is required to determine the state of the RootZoneMoistureLevel property. The reason is that all states of the RootZoneMoistureLevel property are associated with an open upper boundary and a closed lower boundary, as shown in Figure 9. In other words, Figure 18 shows that all transitions follow the same pattern based on an upper and lower boundary: "When (lowerBoundary <= RootZoneMoistureLevelPerDay(d) < upperBoundary)". The relation of a state and its boundaries is also modeled in the IRRIG ontology. For example, Figure 18 shows that the individual irrig:state_rootZone_moistureLevel_veryLow, an instance of the class

irrig:RootZoneMoistureLevelState, is associated with two instances of the class **caso:Boundary** titled irrig:boundary_rootZone_tension_veryLow via the object property `caso:hasOpenUpperBoundary`, and irrig:boundary_rootZone_tension_low via the object property `caso:hasCloseLowerBoundary`. The individual irrig:boundary_rootZone_tension_veryLow represents a threshold of 1600.0 mbar. The individual irrig:boundary_rootZone_tension_low represents a threshold of 1500.0 mbar. The state corresponding to the individual irrig:state_rootZone_moistureLevel_veryLow is reached when the root zone average moisture per day is inferior to the value of the individual irrig:boundary_rootZone_tension_veryLow and is superior or equal to the value of the individual irrig:boundary_rootZone_tension_low.

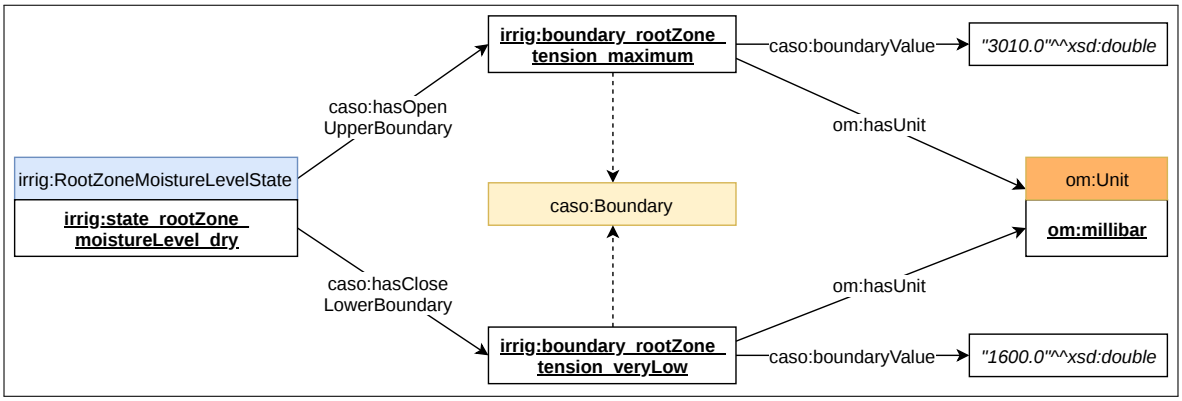

Figure 18. Boundaries of the state RootZoneMoistureLevel.Dry.

Table 5 shows the fields of the rule for the state of RootZoneMoistureLevel deduction. All this information is stored in a file of the rules-base managed by Ontogen. The "short name" field contains the code name of the rule. The "full name" field contains a full version name of the rule that is easier to understand. The description of the rule is in the "description" field. The content of the rule encoded in SWRL that is used by the SWRLAPI Drools inference engine is in the "Rule in SWRL" field.

5.4. Testing of the Reasoning Process in the Soil Moisture Workflow

This section focuses on the testing results. In each test scenario, the system takes an input value and produces an output result. It is possible to evaluate the system by comparing the inferred output results to the expected outputs. There are a total of 21 test scenarios:

- 14 unit tests: Input values are boundary values given by the testers around the threshold. Expected outputs are manually calculated by the testers using the IRRINOV® method.
- 7 sample tests: Input values are samples chosen in the Arvalis dataset. Expected outputs are manually calculated by the testers using the IRRINOV® method.

Table 6 shows the results of the 14 unit test scenarios. This table includes the four following columns.

- ID test: the code name of the test scenario.
- Observation: the input is the observation value, which is the value of the function getRootZoneDailyAverageMoisture(d).
- Expected deduction result: the state of RootZoneMoistureLevel calculated by the testers.
- Deduction result from the DSS: the state of RootZoneMoistureLevel inferred by the DSS.

Please note that in this kind of test, the time of observation has no impact on the result of the deduction. Thus, this testing ignores this kind of input.

Table 5. Rule for RootZoneMoistureLevel property.

Short name:	**Full name:**
ADv122019-RM	ArvalisData-IrrigVersion122019-RootZoneMoistureLevel

Description:
The goal of this rule is to determine the state of RootZoneMoistureLevel. The rule implements the transition from the Init state to one of the states VeryLow, Low, Average, High, VeryHigh, Saturated (see Figure 10). The input of this rule is the root zone daily moisture (?result_observation_rootzone_moisture). The output of this rule is the Root Zone Moisture Level deduction (?deduction_rootzone_moisturelevel). The mechanism of this rule checks if the value of the root zone moisture is in the value domain of two thresholds; it then concludes the state correspondingly.

Rule in SWRL:

```
irrig:RootZoneMoistureLevelDeduction(?deduction_rootzone_moisturelevel) ^

prov:used(?deduction_rootzone_moisturelevel, ?result_observation_rootzone_moisture) ^
om:hasNumericalValue(?result_observation_rootzone_moisture,?value_result_observation) ^

caso:hasState(irrig:observableProperty_rootZone_moistureLevell,?state_rootzone_moisturelevel) ^
caso:hasOpenUpperBoundary(?state_rootzone_moisturelevel,?boundary_upper) ^
caso:boundaryValue(?boundary_upper,?value_boundary_upper) ^
caso:hasClosedLowerBoundary(?state_rootzone_moisturelevel,?boundary_lower) ^
caso:boundaryValue(?boundary_lower,?value_boundary_lower) ^

swrlb:lessThan(?value_result_observation,?value_boundary_upper) ^
swrlb:greaterThanOrEqual(?value_result_observation,?value_boundary_lower)

->
caso:hasResultState(?deduction_rootzone_moisturelevel,?state_rootzone_moisturelevel)
```

Table 6. Unit tests of rules for the state of RootZoneMoistureLevel deduction.

ID Test	Observation (Input)	Expected Deduction Result	Deduction Result from the DSS
moistureLevel-Saturated-0	0 cbar	Saturated	Saturated
moistureLevel-Saturated-9	9 cbar	Saturated	Saturated
moistureLevel-VeryHigh-10	10 cbar	VeryHigh	VeryHigh
moistureLevel-VeryHigh-69	69 cbar	VeryHigh	Very High
moistureLevel-High-70	70 cbar	High	High
moistureLevel-High-119	119 cbar	High	High
moistureLevel-Average-120	120 cbar	Average	Average
moistureLevel-Average-139	139 cbar	Average	Average
moistureLevel-Low-140	140 cbar	Low	Low
moistureLevel-Low-149	149 cbar	Low	Low
moistureLevel-VeryLow-150	150 cbar	VeryLow	VeryLow
moistureLevel-VeryLow-159	159 cbar	VeryLow	VeryLow
moistureLevel-Dry-160	160 cbar	Dry	Dry

Table 7 shows the results of the seven sample test scenarios. This table includes four columns as follows.

- ID test: the code name of the test scenario.

- Date: the input value is the date of the observation. From this date, there is a corresponding observation value. The observation is the value of the function getRootZoneDailyAverageMoisture(d).
- Expected deduction result: the state of RootZoneMoistureLevel calculated by the testers.
- Deduction result from the DSS: the state of RootZoneMoistureLevel inferred by the DSS.

Table 7. Sample tests of rules for the state of RootZoneMoistureLevel deduction

ID Test	Date (Input)	Expected Deduction Result	Deduction Result from the DSS
moistureLevel-Saturated-26062013	26/06/2013	Saturated	Saturated
moistureLevel-VeryHigh-16072013	16/07/2013	VeryHigh	VeryHigh
moistureLevel-Low-21072013	21/07/2013	Low	Low
moistureLevel-High-26072013	26/07/2013	High	High
moistureLevel-VeryLow-31072013	31/07/2013	VeryLow	VeryLow
moistureLevel-Average-03082013	03/08/2013	Average	Average
moistureLevel-Dry-14082013	14/08/2013	Dry	Dry

The deduction result and the data related to each test are coded in Turtle format and stored in a file with a ".owl" extension. The test files are available on the GitLab repository of INRAE via the following link https://gitlab.irstea.fr/irrig/public/tree/master/Testing.

6. Experimentation

The DSS or decision support application is an essential part of the smart irrigation CAS. Each day, the DSS produces an irrigation decision as a suggestion for farmers. Comparing the results of these automated decisions with human decisions enable us to evaluate the DSS. The human decisions made by irrigation experts are available in the experimental dataset provided by Arvalis and INRAE, as presented in Section 4.1.

This section first describes the methodology used to evaluate this DSS. Moreover, a discussion on the limitations of the evaluation is raised at the end of this section to highlight the advantages and drawbacks of the DSS.

6.1. Evaluation Methodology

To evaluate our system, we selected several periods from the original dataset. Those periods should have different events that imply state changes in the observed properties. The selected periods were as follows:

- `[14/08/2013, 21/08/2013]`: The period has one light rain event.
- `[21/08/2013, 03/09/2013]`: The period has one moderate rain event.
- `[31/07/2013, 14/08/2013]`: The period has two consecutive moderate rain events.
- `[22/07/2013, 31/07/2013]`: The period has a crop state change and two rain events.

Each period contains two consecutive watering days which are the first day and the last day in the period. For example, in the first period `[14/08/2013, 21/08/2013]`, the watering days are 14/08/2013 and 21/08/2013.

Then, we compared the day of the watering decision made by our DSS from the day of the watering decision made by the farmer. The comparison relies on the fact that when both the farmer and the DSS follow the same rules of the IRRINOV® method, then the watering decisions on the same

conditions should be equivalent. Thus, when both the farmer and the DSS decide to do a watering on the first day of each period, the next watering day deduced by the DSS should be the last day of the period, in line with the decision of the farmer.

Please note that with each period, the DSS takes the human decision of the first day as an input. It produces its decision from the second day of the period. For example, with the period `[14/08/2013, 21/08/2013]`, the DSS takes the watering decision on day 14/08/2013. It deduces the decisions from day 15/08/2013 to day 21/08/2013.

6.2. Result and Analysis

The results of the experiment over the four mentioned periods are shown in the four data tables below. Each table includes 16 columns and several rows. The first row of the table contains the labels of the type of data in the column. Each one from the second row contains the data corresponding to a day of the period. Details of the first row: The first 13 cells are filled in gray to indicate that the data in the columns corresponding to these cells are input data; the 14th and 15th cells are in red to indicate that the data in the columns corresponding to these cells are inferred results. The last cell is in green to indicate that the data in the 16th column are the decisions of humans in reality. The meaning of each column is as follows.

- The first three columns show the information of the day, month, and year of the date.
- The fourth column shows the total rain quantity of a day. Its unit is mm. When the rain quantity is greater than 0, the cell is filled with blue.
- The fifth column shows the number of delay days. These data in this cell are calculated from the data in the fourth column. Its unit is the day. When the number of delay days is greater than 0, the cell is filled with blue.
- The next six cells correspond to six soil tensiometers. These indicate the average soil moisture of a day as measured by a soil tensiometer. Its unit is cbar.
- The 12th cell shows the average root zone moisture of a day. The data in this cell are calculated based on the data in the six previous cells. Its unit is also cbar.
- The 13th cell shows the crop state observed by a human. If farmers recognize a new crop state and put it in the dataset, then this state is written by the name code, and the cell is filled with yellow. Otherwise, the text entered in the cell is “null”.
- The 14th cell shows the value SleepingDuration inferred by the DSS. Its unit is the day.
- The 15th cell shows the state of the CropWaterNeed. When CropWaterNeed reaches state Yes, the text in the cell is in uppercase, and the cell is filled with red.
- The 16th cell shows the decision of farmers recorded in the dataset. The day that farmers decide to water the plot, the corresponding cell is made green, and the text inside the cell is “YES”. Otherwise, there is no text.

Figure 19 shows the results of the experimental period from 22/07/2013 to 31/07/2013. The first irrigation is performed on 22/07/2013. The farmer decides to schedule the second irrigation on 31/07/2013; however, the DSS suggests that the farmer does nothing on the same day. The farmer disobeys the IRRINOV® method in this case, and he/she has a good reason. When the value of the average root zone moisture is 152 cbar, the root zone moisture level is low. If the root zone moisture level is low and the crop growth is R1, then the crop needs water. If following the IRRINOV® methods, the farmer should wait at least two more days because a rain event postpones the irrigation by four days. The data of SleepingDuration also reflect this calculation. The result of this experiment proves that the farmer decision may be different from the IRRINOV® method.

Day	Month	Year	Rain	DelayDuration	WM-01	WM-02	WM-03	WM-04	WM-05	WM-06	RzM	CropState	SleepingDuration	Ontogen-CWN	Arvalis-CWN
22	7	2013	0	0	135	111	81	106	22	60	171	null	6	YES	YES
23	7	2013	6	0	70	31	1	18	0	0	31	null	5	unavailable	
24	7	2013	0	0	84	50	3	44	0	0	50	null	4	unavailable	
25	7	2013	0	0	92	56	12	112	12	0	68	null	3	unavailable	
26	7	2013	0	0	107	67	38	144	28	36	103	null	2	unavailable	
27	7	2013	0	0	126	83	64	158	61	69	152	R1	1	unavailable	
28	7	2013	20	4	141	67	4	43	3	0	70	null	4	unavailable	
29	7	2013	0	0	144	89	16	112	10	1	99	null	3	unavailable	
30	7	2013	0	0	147	97	29	150	19	10	116	null	2	unavailable	
31	7	2013	0	0	149	101	46	169	47	51	152	null	1	unavailable	YES

Figure 19. DSS experiment during the period from 22/07/2013 to 31/07/2013.

Figure 20 shows the results of the experimental period from 31/07/2013 to 14/08/2013. The first irrigation is performed on 31/07/2013. The second irrigation is scheduled by both the farmer and the DSS is on 31/07/2013. This example shows that the farmer follows the IRRINOV® method and the DSS gives a correct result based on this method. The DSS result is correct in this case because there are rains on two consecutive days. The two rainy days are 06/08/2013 and 07/08/2013. These two rainy days produce seven delay days.

Day	Month	Year	Rain	DelayDuration	WM-01	WM-02	WM-03	WM-04	WM-05	WM-06	RzM	CropState	SleepingDuration	Ontogen-CWN	Arvalis-CWN
31	7	2013	0	0	149	101	46	169	47	51	152	null	6	YES	YES
1	8	2013	0	0	50	0	1	8	16	0	9	null	5	unavailable	
2	8	2013	0	0	75	0	10	90	54	0	64	null	4	unavailable	
3	8	2013	0	0	91	19	37	147	98	0	135	null	3	unavailable	
4	8	2013	0	0	103	29	56	162	129	1	185	null	2	unavailable	
5	8	2013	0	0	116	41	78	167	137	27	215	null	1	unavailable	
6	8	2013	27	5	133	59	104	185	151	88	255	null	5	unavailable	
7	8	2013	11	2	136	0	2	20	11	0	13	null	6	unavailable	
8	8	2013	0	0	123	0	0	69	5	0	5	null	5	unavailable	
9	8	2013	0	0	90	0	0	160	14	0	14	null	4	unavailable	
10	8	2013	0	0	75	0	0	189	32	0	32	null	3	unavailable	
11	8	2013	0	0	71	9	14	203	81	0	95	null	2	unavailable	
12	8	2013	0	0	69	19	26	199	128	0	154	null	1	unavailable	
13	8	2013	0	0	72	28	42	201	156	0	198	null	0	unavailable	
14	8	2013	0	0	78	39	59	199	177	4	236	null	6	YES	YES

Figure 20. DSS experiment during the period from 31/07/2013 to 14/08/2013.

Figure 21 shows the results of the experimental period from 14/08/2013 to 21/08/2013. The first irrigation is performed on 14/08/2013. The second irrigation is scheduled by both the farmer and the DSS is on 21/08/2013. This example shows that the farmer follows the IRRINOV® method and the DSS gives a correct result based on this method. The DSS result is correct in this case because of the light rain on 19/08/2013. There are no delay days in this period.

Day	Month	Year	Rain	DelayDuration	WM-01	WM-02	WM-03	WM-04	WM-05	WM-06	RzM	CropState	SleepingDuration	Ontogen-CWN	Arvalis-CWN
14	8	2013	0	0	78	39	59	199	177	4	236	null	6	YES	YES
15	8	2013	0	0	36	1	1	31	15	0	16	null	5	unavailable	
16	8	2013	0	0	43	0	0	136	72	0	72	null	4	unavailable	
17	8	2013	0	0	47	1	16	207	154	0	170	null	3	unavailable	
18	8	2013	0	0	49	16	27	210	174	0	201	null	2	unavailable	
19	8	2013	2	0	54	26	43	211	185	0	228	null	1	unavailable	
20	8	2013	0	0	56	31	45	207	190	1	235	null	0	unavailable	
21	8	2013	0	0	60	40	59	188	220	19	247	null	6	YES	YES

Figure 21. DSS experiment during the period from 14/08/2013 to 21/08/2013.

Figure 22 shows the results of the experimental period from 21/08/2013 to 03/09/2013. The first irrigation is performed on 21/08/2013. The farmer decides to schedule the second irrigation on 21/08/2013. However, the second irrigation is scheduled by the DSS is two days sooner. From this experiment, it is possible to conclude that the farmer disobeys the IRRINOV® method one more

time. However, this time, the reason for the farmer is unclear. From the data in the column of average root zone moisture, the root zone is at a dry state from 29/08/2013. Therefore, the decision following IRRINOV® method is better than the decision made by the farmer, since the crops in the latter case need to wait for water more than those in the former case.

Day	Month	Year	Rain	DelayDuration	WM-01	WM-02	WM-03	WM-04	WM-05	WM-06	RzM	CropState	SleepingDuration	Ontogen-CWN	Arvalis-CWN
21	8	2013	0	0	60	40	59	188	220	19	247	null	6	YES	YES
22	8	2013	0	0	24	2	1	28	109	0	30	null	5	unavailable	
23	8	2013	0	0	36	0	3	147	188	0	150	null	4	unavailable	
24	8	2013	0	0	40	5	19	199	203	0	218	null	3	unavailable	
25	8	2013	22	4	42	16	25	212	200	0	225	null	6	unavailable	
26	8	2013	4.5	0	45	10	3	51	142	0	61	null	5	unavailable	
27	8	2013	0	0	1	2	18	23	0	255	25	null	4	unavailable	
28	8	2013	0	0	0	1	51	89	0	255	90	null	3	unavailable	
29	8	2013	0	0	0	1	132	172	0	255	173	null	2	unavailable	
30	8	2013	0	0	3	9	182	202	0	255	211	null	1	unavailable	
31	8	2013	0	0	15	18	209	239	0	255	257	null	0	unavailable	
1	9	2013	0	0	24	29	229	239	0	255	268	null	6	YES	
2	9	2013	0	0	39	56	239	239	46	255	239	null	5	unavailable	
3	9	2013	0	0	0	3	33	36	0	255	36	null	4	unavailable	YES

Figure 22. DSS experiment during the period from 21/08/2013 to 03/09/2013.

6.3. Discussion

This section discusses the limitations of the IRRINOV® method, the limitations of the Arvalis dataset and the limitations of the DSS system.

First, the IRRINOV® method is not a complete method for irrigation decisions. IRRINOV® provides guidelines for irrigation decisions based on crop variety, soil and region in metropolitan France. This method is only dedicated to climate events that have a greater chance of occurring in metropolitan France. For example, the IRRINOV® method ignores the fact that in summer, some daily rain events with high intensity may occur for a long period, such as in tropical climates. Moreover, the IRRINOV® method provides no guide about the amount of water that must be used for irrigation. It does not take into account that a farmer may have some water restriction usages. For example, one farm may be allowed to water a plot only each week on Monday morning.

Second, the irrigation decision in the Arvalis dataset is a human decision, and humans can make errors. For example, the first watering decision in the Arvalis dataset occurs on 08/07/2013. Following the IRRINOV® method, farmers should not water the crop on this day since the crop is at the V7 state, and moisture sensor measurements do not reach the threshold. Maybe the farmer has made an error, or maybe he has decided to water the plot sooner because he knows that he will not be able to water the plot in subsequent days due to water restrictions.

Third, the DSS provides automatic results that seem to contain no errors. Unfortunately, the DSS is limited. The IRRIG ontology and rules decision is dedicated to a maize crop on a specific type of soil. If the irrigation decision concerns another crop or other soil, all the thresholds for crop growth, root zone moisture level and crop watering property should be changed. However, the rain intensity rule is the same for all the IRRINOV® guidelines regardless of the observed crop and type of soil. Thus, rain modeling and associated rules can be used. Fortunately, CASO contains the threshold definitions associated with the state of the property. This means that a new ontology should be defined for each type of crop and soil as long as some threshold values change. However, our rules are generic, and their updates should be easy to implement. Please note that in other systems, such as PLANTS, the thresholds are declared inside the rules. Therefore, it is more difficult to modify them.

7. Conclusions

Precision agriculture needs CASs. A CAS that contains a WSN and a DSS can reduce the work of farmers but also improve precision in farming activities. This paper presents our experience in developing a smart irrigation CAS. This system has three advantages: (1) it automates a manual

irrigation method based on the support of specialists and a real experimental dataset, (2) it uses two new ontologies called CASO and IRRIG that extend other well-known ontologies such as SSN and SAREF, and (3) it provides several rules that can work for other similar systems. This paper first describes the methodology of CAS development based on ontologies and details each step of the proposed methodology. Second, this paper also shares an example of development. Finally, an evaluation of the DSS in this system is also published.

A smart irrigation CAS is a complex system. This system requires different roles to participate and contribute to development, such as system developers, ontologists, farmers and agronomists. In developing the DSS, we encountered several challenges. First, because of the participation of different roles from different domains, sharing the same vocabulary and understanding the agronomist model correctly is complicated. Thanks to the SSN ontology and its precise modeling pattern, the design activity produced a clear understanding of the output of each aggregation process.

Second, the development of a DSS requires the development of ontologies. There are many methodologies used to develop an information system and other methodologies used to develop an ontology. To the best of our knowledge, this work is the first to combine the two types of methodologies. In the case of CASO and IRRIG, reusing standard and well-known ontologies was a decision made to increase interoperability with other systems and datasets. However, in practice, this activity was actually time-consuming. When reusing ontologies, one truly needs to understand the semantics and consequences behind every axiom and definition. It becomes even more complicated when several ontologies need to be combined, such as SSN/SOSA and SAREF. Based on this experience, it can be stated that building an ad hoc model from scratch would require less time and consume fewer resources; therefore, the decision of reusing ontologies is based completely on whether interoperability is a requirement. In other words, should the sensor streams be reused by several DSSs?

Third, the dataset provided by Arvalis is limited. Thus, to test the correctness and robustness of the DSS, more datasets should be used. As soon as the sensor network can be deployed on some maize plots, our DSS will be tested on this new dataset. We plan to perform more experiments in 2021.

Finally, the IRRINOV® method itself cannot cover all situations; for example, it does not define the action when there are consecutive multiple rainy days. However, the automatic irrigation system needs a mechanism to address this issue. One solution is to use machine learning to train the system over data of different scenarios. If the training is sufficient, the system can learn the action based on the scale of the IRRINOV® method.

The DSS is a part of a smart irrigation CAS that will be deployed at the experimental INRAE farm (The experimental farm is a part of the AgrotechnoPôle, an ecosystem for agricultural machines and digital information systems developed by an alliance of several organizations and institutes in Europe. Two members of the alliance dedicated to develop AgrotechnoPôle are INRAE and LIMOS.) located at Montoldre in France. In the future, the DSS system will be combined with other components, such as networks, sensors and actuators, to form a complete smart irrigation system.

Author Contributions: Conceptualization, Q.-D.N., C.R. and M.P.-V.; Project administration, J.-P.C.; Software, Q.-D.N.; Validation, Q.-D.N., C.R. and M.P.-V.; Writing—original draft, Q.-D.N., C.R. and M.P.-V.; Writing—review & editing, C.d.V. and J.-P.C. All authors have read and agreed to the published version of the manuscript.

Funding: The project "ConnecSens" and the project "I-Site Clermont - Programme WOW! Wide Open to the World - CAP20-25" (16-IDEX-0001) has funded this research. The CPER research project "ConnecSens" is co-financed by the Auvergne Rhône-Alpes region in France via the program n° 1130 and by the European Union via FEDER funds.

Acknowledgments: We would like to thank Sophie Gendre from Arvalis and Jacques-Eric Bergez from INRAE for their helpful support of this project. We also acknowledge the contributions of the "ETSI Specialist Task Force 534: SAREF extensions" project.

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

References

1. Abowd, G.D.; Dey, A.K.; Brown, P.J.; Davies, N.; Smith, M.; Steggles, P. Towards a Better Understanding of Context and Context-Awareness. In *Handheld and Ubiquitous Computing*; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1707, pp. 304–307.
2. Sun, J.; De Sousa, G.; Roussey, C.; Chanet, J.P.; Pinet, F.; Hou, K.M. A new formalisation for wireless sensor network adaptive context-aware system: Application to an environmental use case. In Proceedings of the Tenth International Conference on Sensor Technologies and Applications SENSORCOMM 2016, Nice, France, 24–28 July 2016; pp. 49–55.
3. Janowicz, K.; Haller, A.; Cox, S.J.; Le Phuoc, D.; Lefrançois, M. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *J. Web Semant.* **2018**, *56*, 1. [CrossRef]
4. ETSI TS 103 264 - v2.1.1. *SmartM2M; Smart Appliances; Reference Ontology and oneM2M Mapping*; Technical report; ETSI: Sophia-Antipolis, France, 2017.
5. Nada, A.; Nasr, M.; Hazman, M. Irrigation expert system for trees. *Int. J. Eng. Innov. Technol. (IJEIT)* **2014**, *3*, 170–175.
6. Kamienski, C.; Soininen, J.P.; Taumberger, M.; Dantas, R.; Toscano, A.; Salmon Cinotti, T.; Filev Maia, R.; Torre Neto, A. Smart Water Management Platform: IoT-Based Precision Irrigation for Agriculture. *Sensors* **2019**, *19*, 276. [CrossRef] [PubMed]
7. Wang, Y.; Wang, Y.; Wang, J.; Yuan, Y.; Zhang, Z. An ontology-based approach to integration of hilly citrus production knowledge. *Comput. Electron. Agric.* **2015**, *113*, 24–43. [CrossRef]
8. Wang, Y.; Wang, Y. Citrus ontology development based on the eight-point charter of agriculture. *Comput. Electron. Agric.* **2018**, *155*, 359–370. [CrossRef]
9. Goumopoulos, C.; Kameas, A.D.; Cassells, A. An Ontology-Driven System Architecture for Precision Agriculture Applications. *Int. J. Metadata Semant. Ontol.* **2009**, *4*, 72–84. [CrossRef]
10. Goumopoulos, C.; O'Flynn, B.; Kameas, A. Automated zone-specific irrigation with wireless sensor/actuator network and adaptable decision support. *Comput. Electron. Agric.* **2014**, *105*, 20–33. [CrossRef]
11. Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. Context aware computing for the internet of things: A survey. *IEEE Commun. Surv. Tutor.* **2013**, *16*, 414–454. [CrossRef]
12. Poveda Villalón, M.; Nguyen, Q.D.; Roussey, C.; de Vaulx, C.; Chanet, J.P. Ontological Requirement Specification for Smart Irrigation Systems: A SOSA/SSN and SAREF Comparison. In Proceedings of the 9th International Semantic Sensor Networks Workshop, International Semantic Web Conference , CEUR Workshop Proceedings, Monterey, CA, USA, 9 October 2018; Volume 2213, pp. 1–16.
13. Arvalis; INRA; Chambre d'Agriculture. *Guide de l'utilisateur, Carnet de terrain: Piloter l'irrigation avec la méthode IRRINOV*; Technical report 07X04; Arvalis: Midy-Pyrénées, France, 2007.
14. Abendroth, L.J.; Elmore, R.W.; Boyer, M.J.; Marlay, S.K. *Corn Growth and Development*; Iowa State University: Ames, IA, USA, 2011.
15. Muller, P.A.; Gaertner, N. *Modélisation objet avec UML*; Eyrolles: Paris, France, 2004.
16. Poveda-Villalón, M. A Reuse-Based Lightweight Method for Developing Linked Data Ontologies and Vocabularies. In *The Semantic Web: Research and Applications*; Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 833–837.
17. García-Castro, R.; Fernández-Izquierdo, A.; Heinz, C.; Kostelnik, P.; Poveda-Villalón, M.; Serena, F. *D2.2 Detailed Specification of the Semantic Model*; Technical report; Universidad Politécnica de Madrid (UPM): Madrid, Spain, 2017.
18. Suárez-Figueroa, M.C.; Gómez-Pérez, A.; Fernandez-Lopez, M. The NeOn Methodology framework: A scenario-based methodology for ontology development. *Appl. Ontol.* **2015**, *10*, 107–145. [CrossRef]
19. Lebo, T.; Sahoo, S.; McGuinness, D.; Belhajjame, K.; Cheney, J.; Corsar, D.; Garijo, D.; Soiland-Reyes, S.; Zednik, S.; Zhao, J. Prov-o: The prov ontology. *W3C Recommendation*, 30 April 2013.
20. Bechhofer, S.; Miles, A. Skos simple knowledge organization system reference. *W3C Recommendation*, 18 August 2009 .
21. Rijgersberg, H.; Van Assem, M.; Top, J. Ontology of units of measure and related concepts. *Semant. Web* **2013**, *4*, 3–13. [CrossRef]
22. W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview*, 2nd ed.; W3c Recommendation W3C: Cambridge, MA, USA, 2012.

23. Musen, M.A. The protégé project: A look back and a look forward. *AI Matters* **2015**, *1*, 4–12. [CrossRef] [PubMed]
24. Garijo, D.; Poveda Villalon, M. *A Checklist for Complete Vocabulary Metadata*; Technical report; WIDOCO: San Francisco, CA, USA, 2017.
25. Poveda-Villalón, M.; Gómez-Pérez, A.; Suárez-Figueroa, M.C. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *Int. J. Semant. Web Inf. Syst. (IJSWIS)* **2014**, *10*, 7–34. [CrossRef]
26. Alobaid, A.; Garijo, D.; Poveda-Villalón, M.; Santana-Perez, I.; Fernández-Izquierdo, A.; Corcho, O. Automating ontology engineering support activities with OnToology. *J. Web Semant.* **2018**, *57*, 100472. [CrossRef]
27. Garijo, D. WIDOCO: A wizard for documenting ontologies. In *Proceedings of the International Semantic Web Conference*; Springer: Berlin, Germany, 2017, pp. 94–102.
28. Migliaccio, K.W.; Olczyk, T.; Li, Y.; Muñoz-Carpena, R.; Dispenza, T. *Using Tensiometers for Vegetable Irrigation Scheduling in Miami-Dade County*; Technical report ABE326, University of Florida: Gainesville, FL, USA, 2002.
29. International Atomic Energy Agency (IAEA). A Practical Guide To Methods, Instrumentation and Sensor Technology. In *Field Estimation of Soil Water Content*; Training course series 30, IAEA: Vienna, Austria, 2008; p. 141.
30. Brouwer, C.; Prins, K.; Heibloem, M. Irrigation water management: irrigation scheduling. *Train. Man.* **1989**, *4*, 66.
31. Météo France. *Durées de Retour de Précipitations Journalier*; Météo France: Paris, France, 2019; pp. 1–10.