Situation-Driven Multi-Purpose Adaptation in the Web of Things

Mehdi Terdjimi [a], Lionel Médini [b] and Michael Mrissa [c]
[a] *Aidaxis, 26 rue Louis Guérin 69100 Villeurbanne, France*
E-mail: mehdi.terdjimi@aidaxis.com
[b] *Univ Lyon, LIRIS, Université Lyon 1 - CNRS UMR5205 F-69622, France*
E-mail: lionel.medini@liris.cnrs.fr
[c] *Innorenew, Slovenia*
E-mail: michael.mrissa@innorenew.eu

Abstract. Existing adaptation solutions for Web of Things (WoT) applications are tightly coupled with application domains. We propose a generic solution that relies on Semantic Web standards to make adaptation decisions for various concerns, using contextual information. We formalize multi-purpose adaptation rules to infer and rank adaptation possibilities. We propose meta-rules to construct situation-driven sets of adaptation rules at application design time. As such, we propose a multi-layered theoretical framework that reduces the number of rules to produce coherent sets of adaptation decisions in various situations. We evaluate the correctness and coverage of this framework on a sustainable agriculture scenario.

Keywords: Web of Things, Contextual Adaptation

1. Introduction

The ambition of the Web of Things (WoT) is to provide a layer on top of the Internet of Things (IoT), where physical devices (things) can be involved in software applications in a standardized, interoperable and secure manner[1]. To do so, existing WoT specifications rely on Web standards and semantic descriptions based on description logics (RDF [1], RDF-S [2], OWL2 [3]). WoT standards at large thus define semantic vocabularies to model physical data [4] (Semantic Sensor Networks[2]) and devices (Thing Description[3]), as well as architectural elements (servient[4], scripting API[5]) reused in projects to design actual WoT applications.

In the ASAWoO project[6], we proposed a component-based framework to ease the development of such applications in diverse domains (industry, healthcare, agriculture...). This framework [5] allows semantically describing and running applications, so that their software elements (descriptions, code) are decoupled from physical (things, network) setups. The variety of technical constraints, applications and tasks imposes WoT standards and their implementations to cope with numbers of situations: an application must be able to run on different things, use different data sources and perform several tasks.

Yet, contextual adaptation in such applications is a major challenge. Existing adaptation solutions are either tightly coupled with their application domains (as they rely on domain-specific context models) or offered as standalone software components that hardly fit in Web-based and semantic architectures. The sets of adaptation goals are usually difficult to extend for newly identified purposes, as it requires redefining the

[1] https://www.w3.org/WoT/
[2] https://www.w3.org/TR/vocab-ssn/
[3] https://w3c.github.io/wot-thing-description/
[4] https://w3c.github.io/wot-architecture/#sec-servient-architecture
[5] https://w3c.github.io/wot-scripting-api/
[6] https://liris.cnrs.fr/asawoo/

whole context model and – in worst cases – reconsidering the adaptation process itself. This leads to integration, performance and maintainability problems. Hence, there is a need for an adaptation approach able to remain independent from application domains and to comply with Web standards (e.g. resource-oriented architectures, semantic Web, Web of things) by design. The ASAWoO framework proposes a context management component that copes with these constraints. It is loosely coupled with semantic descriptions of relevant resources - including the application domain - and applies generic reasoning techniques to enable domain-specific adaptation in WoT applications.

In this paper, we tackle the research question of multi-purpose contextual adaptation in WoT applications. We aim to provide an adaptive solution that brings together contextual information and adaptation purposes, to generate adaptation rules in both user-friendly and generic ways. We formally describe the theoretical adaptation framework that supports context management and detail the design of adaptation rules. Our contributions are listed as follows:

1. A theoretical adaptation framework to support domain-independent multi-purpose adaptation
2. A generic model for/set of *meta-rules* to generate adaptation rules that will in turn allow to infer ranked adaptation possibilities
3. A comprehensive solution for both designers and experts to efficiently process context models and adaptation rules using an incremental reasoning engine

This paper is structured as follows. In Section 2, we overview related work on adaptation classification and WoT-based solutions. In Section 4, we formalize our contributions on domain independent multi-purpose context adaptation. In Section 5, we describe the meta-adaptation rule engine that allow generating adaptation rules at design time. In Section 6, we evaluate our solution on a sustainable agriculture scenario, on various situations. In Section 7, we open a discussion regarding the questions brought by our solution. We conclude this paper in Section 8.

2. Related work: semantic approaches for adaptation in the Web of Things

[6] classify adaptation approaches according to several characteristics: functional *vs.* nonfunctional, designed *vs.* unanticipated, predictable *vs.* unpredictable, third-party *vs.* self-adaptive (*i.e.* provoked by the adaptation software itself) and business-specific *vs.* generic. Another adaptation classification from [7] includes the notions of anticipation (as the degree of anticipation to changes) and domain-specificity (as the level of parametrization of the adaptation solution) as well. It also introduces two additional characteristics to position adaptation solutions: tools (denotes whether the adaptation solution comes with a dedicated development environment or a runtime monitoring environment) and scope (describes the extent of the adaptation process over the application components and services). In this work, we target generic and largely scoped contextual adaptation techniques. In this section, we position our research with respect to existing work on generic adaptation, semantics-based adaptation, semantics-based WoT solutions, and adaptation rule generation.

2.1. Generic adaptation frameworks

Several techniques can be used to adapt a software system to contextual parameters: service configuration, service substitution and adaptation planning [6, 8] [7, 4.4]. The following adaptation frameworks illustrate these techniques and stress their advantages and drawbacks for generic adaptation approaches.

In [9], the authors provide a service configuration adaptation approach that is generic and based on reference models. It aims to ease the design of adaptation solutions for business processes that are shared by various participants, based on a multidimensional context model and complex transformation rules. The fact that a service configuration engine has to generate numerous parameters makes such an approach strongly parametric and does not guarantee optimal adaptation decisions. Moreover, it is more difficult for a domain expert to completely formalize the transformation function than to express its behavior in natural language.

In the Adaptive CORBA Template (ACT) [10], the authors propose a service substitution approach for dynamic adaptation based on CORBA middleware interceptors. Such interceptors can be registered, unregistered and enhanced at runtime, thus producing adaptation cases that are not known in advance. This framework is domain-independent, partially anticipatory as it preconfigures so-called *adaptive CORBA templates* and its adaptation scope can address different tasks through the use of rule-based interceptors. In this sense – and even though it requires the CORBA middleware,

which is neither tailored for resource-limited devices and nor compliant with Web standards - it is close to more recent approaches such as aspect-oriented adaptation [11], as well as to substitution of service-based applications [12], or more generally to adaptation in dynamic, component-based middlewares.

WComp [13] is an aspect-oriented, Web service-based middleware that relies on the Aspect of Assembly (AA) aproach to provide compositional adaptation of event-based services, depending on context changes. The adaptation decision relies on aspect-oriented concepts, such as joint points, pointcuts, advices, etc.

The FraSCAti platform [14] aims to extend the Service Component Architecture (SCA) [15] by providing reflexive behavior for service oriented architecture components. To do so, each component is associated to a generic container (at a *meta* level of the architecture) that includes several services such as component identity, lifecycle, hierarchy, wiring, etc.

The Mobility and ADaptation enAbling Middleware (MADAM) framework [16] applies an adaptation planning approach to ease the development of adaptive solutions. At request time, it chooses the combination of available services to compose a response to a particular user's need. To do so, it provides several managers (context, configuration and adaptation managers) that allow for decoupled adaptation processes. Moreover, it relies on "context reasoners", which are generic means to locate sensor data processing in the adaptation workflow. In the perspectives, the authors foresee to reuse this approach using Web services and semantic Web technologies.

2.2. Semantics-based adaptation approaches

By essence, context is metadata. Behind the well-known definition of context by Anind Dey [17] "*Context is any information that can be used to characterize the situation of an entity[...]*" emerges the notion of information annotation, and therefore of generating graphs of interrelated pieces of data. The semantic Web was built to ease and to standardize the creation, handling, querying and transformation of such graphs [18]. Hence, it appears quite relevant to use semantic web languages (RDF, RDF-S, OWL)[7] to model contextual information, as well as to reason about contextual data to make adaptation decisions.

[19] present a semantic adaptation framework for multimedia documents in which they describe documents as sets of parts related by constraints. Although they only apply adaptation to this domain, their formalization makes their approach quite reusable, even though they use an on-purpose graph matching algorithm rather than classical reasoning tools. The work presented in [20] proposes a generic approach based on the semantization of sensor data and a *context intelligence module*, that partly relies on a semantic reasoner to provide adaptation propositions. However, to the best of our knowledge, [21] describes the first work to use both rule-based reasoning for context adaptation, as well as the expressivity of high-level description logics constructs available in OWL relations.

Hence, the potential offered by Web-based reasoning and semantic technologies could provide adaptation in WoT applications in a generic, flexible and reusable manner, for multiple and high-level purposes. In this view, we intend to comply with Web standards for data exchange (SOSA/SSN vocabularies, Streams API[8]) and with the W3C WoT Working Group specifications (Thing Description, Servient), while designing our adaptation solution for WoT applications.

2.3. Semantics-based WoT platforms

To reason about contextual data, semantic adaptation tools must first gather semantized data. In the WoT application field, not all platforms provide semantic data to their components. Actually, only recent advances in the WoT aim to bring together the Semantic Web with Web standards, on top of the Internet of Things. UBIWARE [22] uses semantic annotations to describe agents and behavioral tasks, to provide interoperability and reusability of these definitions. SPITFIRE [23] unites RESTful approaches using CoAP [24] with OWL/RDFS and SPARQL[9] for constraint devices. Sense2Web [25] allows publishing sensor data and measurements on the Web through a SPARQL endpoint, but still has to be coupled with a functional solution to provide a complete WoT application capable of managing any type of thing (sensor or actuator). The M3 framework provides a more comprehensive and domain-independent approach through a dedicated vocabulary to describe sensors, together with the tools to reason about these descriptions and deduce application templates [26]. While these ap-

[7] https://www.w3.org/2001/sw/interest/

[8] https://www.w3.org/TR/streams-api/

[9] https://www.w3.org/TR/rdf-sparql-query/

proaches facilitate things interoperability and WoT application development, they do not tackle the adaptation concern.

2.4. Automatic generation of rules

Dynamic context-aware environments involve a large number of parameters (system resources, perceived environment, user preferences, etc.). Defining an analytical workflow starting from all these parameters and resulting in an adaptation decision is a complex task [27–29] that is often achieved using rules. This large number of parameters causes a "rule-explosion" problem [27] that would, in theory, force application developers to state all possible parameter combinations to make sure that every situation will meet an adaptation plan. In this section, we present some approaches that prevent developers from issuing all these rules. The first approach is based on reutilization. S-LOR [30] allows sharing rules as Linked Data using a Linked Open Data vocabulary, and provides means to integrate those rules in semantic reasoners through the Jena API. Though this approach is promising, the rules that have been shared yet are designed for the interpretation of sensor data as semantic information rather than for adaptation. Another approach is to reduce the coupling between applications and adaptation solutions by automatically producing domain-specific rules from generic representations. Still, challenges remain in the domain of rules generation. [31] propose a business rule design framework that relies on the Semantics of Business Vocabulary and Business Rules (SBVR) formalism[10]. The method described in [32] generates such rules in the same manner, while providing a semantic formalization of them as SPARQL queries. However, both methods requires application users and domain experts to express each rule in natural language and are intended for functional purposes only.

The reflexive method presented in [33] allows generating on-the-fly adaptation rules based to support dynamic changes on a system. However, this adaptation solution is exclusively nonfunctional, centralized, and may not scale in environments with numerous connected objects. This method also does not provide adaptation possibilities ranking, and therefore cannot ensure optimal adaptation decisions. The solution presented in [34] infers rules with respect to contextual information, referred as the user's behavior. This however requires sufficient data to provide significant accuracy. Such method is appropriate for recommender systems, but not for adaptive systems on critical domains (such as security, healthcare, driving, etc.), as they require highly accurate adaptation rules as soon as the application starts.

[10] http://www.omg.org/spec/SBVR/

3. Scenario

To illustrate our approach and contributions, we consider a vineyard-watering application. This application detects parts of the field that need to be watered, while taking environmental conditions into account. A WoT infrastructure hosts this application, which includes a cloud infrastructure, several wireless gateways as well as an irrigation system composed of geolocated watering agribots and drones that embed a GPS sensor and a thermal camera. Drones and agribots have the ability to move over/across the field to detect/water given parts of the field.

While the application is active, several sensors (which are placed on each part of the field) regularly send data to agribots to adapt their behavior with respect to the information given (*e.g.* the presence of entities would disable their moving and watering functionalities). Drones possess a thermo-sensing camera along with a CPU to process field images. If a drone cannot process a picture (due to limited memory or high CPU usage), it sends the picture to another drone to fulfill the task. The drone then communicates with the suitable agribots to take care of the parts that lack watering. Other devices consist in a desktop computer to host the WoT infrastructure, as well as a tablet allowing users to remotely monitor the system.

In this scenario, the WoT application is designed as a hierarchy of functionalities, with the ManageWatering functionality on top of it. The latter is composed of several functionalities, among which SprayWater and DetectWateringNeeds, which implies TakeHdPicture, ProcessPicture and TransferPicture. Complex processing task can be be executed either on the cloud platform, or on the device itself. We consider a field equiped with an anemometer, a thermometer and a pluviometer to sense actual weather conditions. Drones are equiped with a GPS sensor, a thermal camera, a Wifi and a Bluetooth network interfaces, and are able to sense their hardware status (battery level, storage capacity, CPU usage, storage space).

4. Contextual Adaptation in the WoT

The ASAWoO project issued a freely available WoT platform[11]. This platform is a runtime environment for WoT applications that are composed of high-level *functionalities*, constructed on top of the *capabilities* available through the things APIs and provided by their manufacturers.

To build such applications, the ASAWoO approach states that each physical thing has its own *avatar* [5]. An avatar is a form of servient implementation: it provides a virtual representation of this thing as a Web resource, monitors and controls the capabilities of this thing, and augments its interaction patterns by exposing application functionalities. Avatars expose functionalities that can either be atomic (*i.e.* directly implemented by the thing physical capabilities), or composed by lower-level functionalities.

Within the ASAWoO platform, an avatar has a component-based architecture that allows its elements (aka "managers") to exchange semantic data. Managers are designed to tackle different concerns: network disconnections, social interactions, and naturally dynamic adaptation. In particular, managers allow the implementation, composition, exposition, communication with and location of the object functionalities. Hence, each of them deals with a specific purpose, and these purposes may require adaptation to the context.

4.1. Contextual Modeling

In our previous work from [35], we have shown that contextual information can be diverse amongst application domains. Most of the literature group information thematically. We hereafter formalize concepts related to the context and its adaptation.

A *contextual instance* is a high-level piece of contextual information. In our work, contextual instances can be either inserted at design time as semantic information by WoT application designers (*e.g.* user preferences, regional settings, device static information, etc.) or inferred at design time from raw sensor data using rule-based semantic reasoning. We group these contextual instances in thematic sets called *contextual dimensions*.

[11] https://asawoo.gitlab.io/

4.1.1. Contextual Dimension

Let i be a contextual instance, *i.e.* a high-level, semantized piece of contextual data. We define contextual dimensions as follows:

Definition 1 (Contextual dimension)**.** *A contextual dimension d represents the set of contextual instances needed for any adaptation purpose, regarding a given type of observation (temperature, location, etc.).*

$d = \{i_d\}$, $d \in \mathcal{D}$ *where* $\mathcal{D}$ *is the set of available observations that are relevant for the application.*

For instance, in our scenario, we use "Temperature" as a contextual dimension, which groups the following contextual instances: $\{Hot, Warm, Cold\}$.

4.1.2. Adaptation Purpose

WoT application execution can rely on different types of adaptation we call *adaptation purposes*. To avoid the redefinition of specific contextual dimensions and allow the instantiation of reusable domain-specific context models, we have propose a meta-model for context in WoT application. This meta-model relies on cross-domain adaptation purposes and promotes the usage of identical reasoning mechanisms for any application domain, through common ontological concepts. We formally define an adaptation purpose as follows:

Definition 2 (Adaptation purpose)**.** *An adaptation purpose ap represents the set of contextual instances related to a certain type of adaptation, as described above.*

$ap = \{i_{ap}\}$, $ap \in \mathcal{AP}$.

In the ASAWoO platform, the different managers that compose the avatar architecture require the set $\mathcal{AP} = \{Imp, Comp, Exp, Prtcl, CdL\}$ of adaptation purposes explained below:

- *Imp* Finding the best capability to implement an atomic functionality,
- *Comp* Finding the best lower-level functionalities to take part in a functionality composition (exposed by either the local avatar or other avatars),
- *Exp* Deciding whether to expose or not a functionality to other avatars and users,
- *Prtcl* Finding the most suitable network protocol to communicate with the thing during the execution of a functionality,
- *CdL* Finding the optimal location for application code modules (*e.g.* on the thing processing unit, on a closeby gateway or on a cloud infrastructure).

While for the *Exp* purpose, the adaptation possibilities are either to expose a functionality or not, the possibilities for the other purposes may be multiple.

4.1.3. Multi-Purpose Context Model

The meta-model we have proposed allows to build multi-purpose context models using both contextual dimensions and adaptation purposes. At adaptation solution design time, WoT application designers discuss with domain experts to determine the appropriate contextual dimensions and adaptation purposes composing the context model, as well as the contextual instances that will populate the model at runtime. We define such models as follows:

Definition 3 (Context model)**.** *A context model $\mathcal{M}$ is a two-dimensional set of contextual instances corresponding to both adaptation purposes $\mathcal{AP}_\mathcal{M}$ and contextual dimensions $\mathcal{D}_\mathcal{M}$. Within a given model $\mathcal{M}$, dimensions and adaptation purposes are respectively disjoint in $\mathcal{D}_\mathcal{M}$ and $\mathcal{AP}_\mathcal{M}$.*

$\mathcal{M} = \mathcal{AP}_\mathcal{M} \times \mathcal{D}_\mathcal{M} = \{i_{ap_\mathcal{M}, d_\mathcal{M}}\}$ where $ap_\mathcal{M} \in \mathcal{AP}_\mathcal{M}$ and $d_\mathcal{M} \in \mathcal{D}_\mathcal{M}$.

The context model for our scenario is composed of the following dimensions: Wind, Dryness, Temperature, Location, Battery, Memory, CPU and Resolution. For each purpose, we have different sets of contextual instances, depending of the adaptation needs. These instances are depicted in Figure 1. Sometimes, a dimension may not be useful for a given purpose. In that case, its list of instances is marked as *not applicable* (n/a) on the figure.

We justify our adaptation purposes association with dimensions as follows:

- *Imp* Choice of capability to implement TakeHdPicture depending on the available memory and CPU on the device, as well as on its camera resolution.
- *Comp* Composition of DetectWateringNeeds depending on the drone location and battery level (sufficient to move towards the field to take pictures from), as well as on its available storage space (to save pictures taken).
- *Exp* Exposition of the functionality SprayWater requiring acceptable weather conditions.
- *Prtcl* Choice of protocol to send pictures with TransferPicture depending on the device location and on its battery level (as some protocols are more energy consuming).
- *CdL* Location of the ProcessPicture functionality code depending on available memory, battery and CPU on the device. The code can be deployed either on the device itself or on the cloud.

4.2. Contextual Adaptation Workflow

The solution we propose relies on adaptation planning. In our previous work [36], we have presented a contextual adaptation solution that infers ready-to-query *adaptation possibilities* for multiple *adaptation purposes* from both static information and raw sensor data using semantic reasoning. The work we have proposed in [37] details the runtime process: when another manager needs to make an adaptation decision concerning a particular purpose, it sends an adaptation question to the context manager. Processing such questions is reduced to a simple purpose-dependent SPARQL SELECT query to the endpoint. The steps of this workflow are depicted in Figure 2 and detailed in the subsections below.

4.2.1. Semantization step

The contextual adaptation solution we have proposed in [36] relies on semantized contextual instances implemented as RDF triples. At runtime, sensor data are sent to avatars and are converted into RDF triples by the appropriate manager. For instance, a such manager could generate the graph shown in Listing 1:

Listing 1: Example of a semantized temperature observation based on SOSA and SSN (in N3).

```
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix qudt-1-1: <http://qudt.org/1.1/schema/qudt#> .
@prefix qudt-unit-1-1: <http://qudt.org/1.1/vocab/unit#> .

<observation/1087> rdf:type sosa:Observation ;
  sosa:resultTime "2018-12-14T12:00:00Z"^^xsd:dateTime ;
  sosa:hasFeatureOfInterest <Air?lat=45.80902&long=4.94993> ;
  sosa:hasResult [
    a qudt-1-1:QuantityValue ;
    qudt-1-1:unit qudt-unit-1-1:DegreeCelsius ;
    qudt-1-1:numericValue "20.388"^^xsd:double ] .
```

	Wind	Dryness	Tempe-rature	Location	Battery	Memory	CPU	Reso-lution
Imp	n/a	n/a	n/a	n/a	n/a	HighMemPict LowMemPict	HighCPUPict LowCPUPict	HD AvgQuality LowQuality
Comp	n/a	n/a	n/a	CloseToField FarFromField	HighBattComp LowBattComp	HighMemComp LowMemComp	n/a	n/a
Exp	StrongWind Breeze NoWind	Dry Wet Flooded	Hot Warm Cold	n/a	n/a	n/a	n/a	n/a
Prtcl	n/a	n/a	n/a	Close Far	HighBattPrtcl LowBatt-Prtcl	n/a	n/a	n/a
CdL	n/a	n/a	n/a	n/a	HighBattCode LowBattCode	HighMemCode LowMemCode	HighCPUCode LowCPUCode	n/a

Contextual Dimensions | Adaptation Purposes | Contextual Instances

Fig. 1. The context model of our scenario.

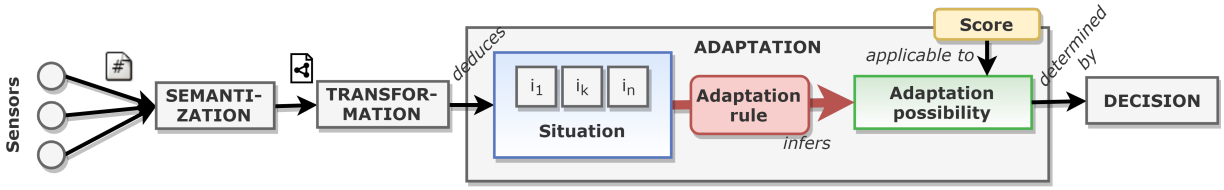

Fig. 2. The contextual adaptation workflow

4.2.2. *Transformation step*

As soon as numerical values are semantized as RDF triples, they are inserted into the avatar internal semantic reasoner to be transformed into contextual instances using *transformation rules*. Transformation rules relies on thresholds and upper limits provided by experts to infer appropriate contextual instances. For example, inserting the observation in Listing 1 will infer the contextual instance *Warm*, considering the threshold between *Cold* and *Warm* is 20 degrees.

At runtime, contextual instances are inserted in and deleted from the semantic repository, which is equipped with both a SPARQL endpoint and an OWL2 RL incremental reasoner. Context models are instantiated this way; each cell of the matrix contain one contextual instance, or is empty in the case of a contextual dimension is not applicable for a given adaptation purpose (*e.g.* the dimension "Location" and *Imp* in our scenario).

We define subsets of instantiated contextual models as *contextual situations*. A contextual situation is identified by domain experts to design appropriate adaptation rules in response of these situations. We define it as follows:

Definition 4 (Contextual situation)**.** *A contextual situation ς is a subset of an instantiated context model that characterizes a salient situation identified by domain experts.*

$$\varsigma = \{i_{j,k}\} \text{ where } j \in \mathcal{AP} \cup \emptyset \text{ and } k \in \mathcal{D} \cup \emptyset$$

For example, in our scenario, the experts refer to a situation "FarDroneWithLowBatteryForTransfer" which consists in the set of contextual instances $\{Far, LowBatteryPrtcl\}$.

4.2.3. *Adaptation step*

Adaptation possibilities are triples inferred using business-specific *adaptation rules* at the insertion of contextual instances, and removed at their deletion using incremental reasoning. Their subject is the functionality to be adapted in the application, the predicate is related to the purpose, and the object is the adaptation candidate[12]. Table 1 below shows the possibility triple patterns for each purpose.

[12] An adaptation candidate is an available resource that can satisfy the given adaptation purpose, *i.e.* of the type corresponding to this purpose. For instance, for the *Prtcl* purpose, it must be a protocol implemented on the device network interface.

Table 1
Patterns of possibility triples for each adaptation purpose.

Purpose	Subject Type	Predicate	Object Type
Imp	Functionality	*hasSuitableCapabilityForImplementation*	Capability
Comp	Functionality	*hasSuitableFunctionalityToTakePart InComposition*	Functionality
Exp	Functionality	*hasExposability*	Exposability
Prtcl	Functionality	*hasSuitableProtocol*	Protocol
CdL	Functionality	*hasSuitableCodeLocation*	CodeLocation

Definition 5 (Adaptation possibility)**.** *An adaptation possibility p associates an adaptation candidate to a functionality to be adapted, with respect to an adaptation purpose. The set of possibilities for the same adaptation purpose is denoted P.*

$p_{f,ap} : f, ap \rightarrow c$ *where* f *is a functionality,* c *is a candidate and* $p_{f,ap} \in P_{f,ap}$.

For instance, in our scenario, the adaptation of communication protocols for the TransferPicture functionality may have either Wifi of Bluetooth protocols as adaptation candidates.

Adaptation possibilities are inferred from contextual instances using *adaptation rules*. Adaptation rules have similar patterns, regardless of their adaptation purposes: the body of the rule is a conjunction of contextual instances, *i.e.* a contextual situation, and the head of the rule is a set of adaptation possibilities.

Definition 6 (Adaptation rule)**.** *An adaptation rule is a conjunctive rule in the form:* $\bigwedge_{k \leqslant |\varsigma|} i_k \rightarrow P$ *where* $i_k \in \varsigma$ *is a contextual instance and* P *a set of inferred adaptation possibilities p. Each purpose* $ap \in \mathcal{AP}$ *is associated to a set* $\mathcal{R}_{ap}$ *of adaptation rules.*

In ASAWoO, contextual instances that trigger adaptation possibilities are respectively associated to capability instances for *Imp*, and to functionality instances for *Comp*. An example of adaptation rule in our scenario is
if {Dry, Breeze, Warm}
then {(SprayWater hasExposability Exposable)}

4.2.4. Scoring

To allow an optimal decision amongst several adaptation possibilities regarding a particular contextual situation at runtime, our solution relies on *scoring*. This score is determined as follows. At design time, each possible contextual situation is presented to the domain expert. The expert then suggests an appropriate response to this situation, to allow the WoT application designer to determine which capabilities/functionalities are the most/least appropriate to implement/compose a functionality, which functionalities should not be executed (*i.e.* should not be exposed to clients), which protocols shoud be used, or where the functionality code should be located. The "most/least" degree is thereafter interpreted as a score for this adaptation possibility with respect to the observed situation.

Making binary decisions then consists in selecting an adaptation possibility if its score equals 1. Finding the best candidate for another purpose consists in selecting the possibility with the highest score.

Definition 7 (Scoring function)**.** *Each instance of a context model is associated to a scoring function* sf *that depends on its situation and allows to weight several adaptation possibilities. As scoring function results will be summed for each adaptation purpose, their results are normalized by the number of considered dimensions.*

$sf : i_{ap_{\mathcal{M}},d_{\mathcal{M}}}, \varsigma \rightarrow \{s_{p_n}\}, \forall p_n \in P_{f,ap_{\mathcal{M}}}$ *with* $s_{p_n} \in [0; \frac{1}{|\mathcal{D}_{\mathcal{M}}|}]$.

Definition 8 (Adaptation score)**.** *An adaptation score s is a numeric value that allows to weight an adaptation possibility for an adaptation purpose. Scores are normalized, situation-dependent and obtained using scoring functions.*

$s_{\varsigma,f,ap,p} = \sum_{k=1}^{|\mathcal{D}_{\mathcal{M}}|} \Pi_p(sf(i_{ap_{\mathcal{M}},k}, \varsigma))$ *with* Π_p *the projection of the result set of* sf *on the possibility* p *and* $0 \leqslant s_{\varsigma,ap,p} \leqslant 1$.

To determine the score of a composition candidate for the *Cmp* purpose, we calculate the average of each functionality scores that are part of the composition. Hence, if a functionality can be composed in several ways, the composition with the highest score would be chosen.

4.2.5. Decision step

Adaptation possibilities are queried at runtime, using adaptation questions. These questions are formulated as SPARQL SELECT queries. Their pattern is based on the vocabulary described in Definition 5: subjects are the components to adapt (a capability or a functionality), predicates are based on the adaptation purpose, and objects are the adaptation candidates. Adaptation possibilities are linked with their purpose-based score through blank node *reification*, so that each possibility is ranked using an ORDER BY clause on the scores. The pattern of a generic SPARQL-based adaptation question is the following:

Listing 2: General SPARQL expression of an adaptation question.

```
PREFIX rdf: <http://www.w3.org/1999/02/22
    -rdf-syntax-ns#>
SELECT ?adaptationPossibility ?score {
 [] rdf:subject ?componentToBeAdapted ;
    rdf:predicate ?adaptationPurpose ;
    rdf:object ?adaptationPossibility ;
    rdf:value ?score . }
ORDER BY ?score
```

The adaptation question for the *Exp* purpose requires an additional FILTER clause on the score (this score must be strictly positive to allow exposability). In the next section, we propose a generic way to design adaptation rules and determine scores by relying on *meta adaptation rules*.

5. Advanced adaptation rule management

Providing fully automatic generation of adaptation rules without any interaction from users, experts or application designers is a very long term challenge. Still, we believe that the involvement of each participant in the WoT application design process can be simplified and reduced. In this work, we improve the declarativity of this process by proposing a method to generate rules at application design time. Moreover, we are aware that under certain circumstances (such as crisis), applications should behave in different manners (*e.g.* adopt a degraded mode to ensure users' safety). Hence, we also introduce the notion of situation-based rule generation, which also reduces the number of adaptation rules.

In this section, we present how we build adaptation rules at design time and detail our meta adaptation rule engine that generates scored adaptation rules. This engine 1) generates contextual situations, 2) generates adaptation possibilities, 3) combines these elements with scoring functions to apply a score on each adaptation possibility, and 4) outputs a set of adaptation rules using the situations and the scored adaptation possibilities.

5.1. Meta adaptation rule engine main components

The meta adaptation rule engine generates a set of adaptation rules at design time. This process can be executed as soon as the context model is designed, yet before starting the adaptive solution (as it actually requires adaptation rules to work). This engine relies on a triplestore that supports SPARQL SELECT and CONSTRUCT queries, and takes semantically annotated information as input. The architecture of the meta adaptation rule engine is depicted in Figure 3 and is detailed in the subsections below.

5.2. Situation Generation

At first, the context model is loaded into the engine triple store by the solution designer. The engine takes the contextual dimensions, the contextual instances and the adaptation purposes from the loaded model, as input. An example of imported purpose, dimension, and instance is listed in Listing 3 below. We use the semantic relationships between contextual instances i, dimensions $d \in \mathcal{D}_{\mathcal{M}}$ and purposes $ap \in \mathcal{AP}_{\mathcal{M}}$ from Definitions 1 to 3.

Listing 3: Example of data inserted onto the triplestore to generate situations (in N3).

```
@prefix rdf: <http://www.w3.org
    /1999/02/22-rdf-syntax-ns#>.
@prefix asawoo-ctx: <http://liris.cnrs.fr
    /asawoo/vocab/context.owl#>.
@prefix asawoo-wtr: <http://liris.cnrs.fr
    /asawoo/vocab/watering-application.
    owl#>.

# Exposition Purpose
asawoo-wtr:Exp rdf:type
    asawoo-ctx:AdaptationPurpose ;
    asawoo-ctx:purposePredicate
        asawoo-ctx:hasExposability .

# Wind Dimension
asawoo-wtr:Wind rdf:type
    asawoo-ctx:ContextualDimension .

```

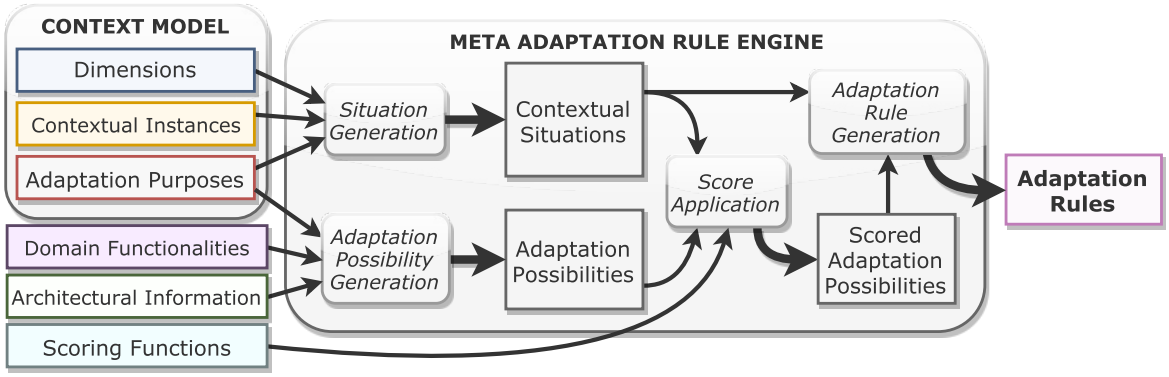

Fig. 3. The meta adaptation rule engine, its components and inputs.

```
12  # NoWind Instance
13  asawoo-wtr:NoWind rdf:type
        asawoo-ctx:ContextualInstance ;
14      asawoo-ctx:instanceFromDimension
            asawoo-wtr:Wind ;
15      asawoo-ctx:instanceForPurpose
            asawoo-wtr:Exp .
```

Then, the *situation generation* module sends a SPARQL query to the triplestore to determine each possible contextual situation, for each purpose. Algorithm 1 below generates such query. As stated in Definition 4, each contextual situation ς is a set of distinct contextual instances $i_{j,k}$, where $j \in \mathcal{AP}$ and $k \in \mathcal{D}$.

ALGORITHM 1: SPARQL situation query generation

Data: An adaptation purpose ap and a set $\mathcal{D}$ of contextual dimensions that compose the context model.
Result: A SPARQL query that determines each possible contextual situation for ap.

```
query ← generateSelectQuery(),
 instancesVariables ← [ ]
foreach D ∈ 𝒟 do
    i ← generateNewInstanceVariable()
    instancesVariables.push(i)
    query.appendBody("OPTIONAL { ")
    query.appendBody("i rdf:type
      ContextualInstance")
    query.appendBody("i instanceForPurpose ap")
    query.appendBody("i instanceFromDimension d")
    query.appendBody(" }")
query.setDistinctVariables(instancesVariables)
return query
```

In Algorithm 1, the function *generateNewInstanceVariable*() is used to create a new variable to be selected, the function *appendBody*() adds a triple pattern (or OPTIONAL clause) in the body of the query, and the function *setDistinctVariables*() adds a DISTINCT clause with the corresponding variables to be selected.

5.3. Adaptation Possibility Generation

The *adaptation possibility generation* step produces adaptation possibilities using the application domain functionalities, the architecture information related to candidates (protocols, code hosts), and the adaptation purposes. To do so, it populates the triplestore with these semantically-annotated information and generates a graph of adaptation possibilities for each purpose, using the following query:

Listing 4: CONSTRUCT query to retrieve the graph of adaptation possibilities.

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22
       -rdf-syntax-ns#>
2  PREFIX rdfs: <http://www.w3.org/2000/01/
       rdf-schema#>
3  PREFIX asawoo-ctx: <http://liris.cnrs.fr/
       asawoo/vocab/context.owl#>
4  CONSTRUCT {
5    ?adapted ?possibilityPred ?candidate
6  }
7  WHERE {
8    ?purpose asawoo-ctx:purposePredicate
9      ?possibilityPred .
10   ?possibilityPred rdfs:domain
11     ?adaptedClass .
12   ?possibilityPred rdfs:range
13     ?candidateClass .
14
15   ?adapted rdf:type ?adaptedClass .
16   ?candidate rdf:type ?candidateClass .
17 }
```

In Listing 4, the triple patterns at lines 6-8 allow retrieving the adaptation purposes domains and ranges, so that each possibility is generated through the CONSTRUCT pattern (line 4). The *adapted/candidate* and their classes from lines 9 and 10 refer to the subjects and objects enumerated in Section 4 – Table 1. The set P of adaptation possibilities generated in this process is based on Definition 5.

5.4. Score management

To calculate possibility scores for each situation, the *score application* step combines contextual situations with adaptation possibilities and their scoring functions. Scoring functions are represented as triples, as follows:

- their *rdf:type* is *asawoo-ctx:ScoringFunction*,
- they are linked respectively to a dimension and to an adaptation purpose through the object properties *asawoo-ctx:applicableTo* and *asawoo-ctx:forDimension*,
- they are reified using the object property *asawoo-ctx:scores*,
- this reification includes four object properties, based on Definition 8
  - *asawoo-ctx:forCandidate* for the adaptation possibility candidate p,
  - *asawoo-ctx:forAdapted* for the adapted functionality f,
  - *asawoo-ctx:withInstance* for the contextual instance i,
  - *rdf:value* for the score s.

These relationships are based on the formalization of the scoring function sf detailed in Definition 7. Below is an example of scoring function for the Bluetooth candidate, to adapt the communication protocol to transfer a picture (in N3 syntax):

Listing 5: N3 description of a scoring function (Battery dimension and Prtcl purpose).

```
1  @prefix rdf: <http://www.w3.org
       /1999/02/22-rdf-syntax-ns#>.
2  @prefix asawoo-ctx: <http://liris.cnrs.fr
       /asawoo/vocab/context.owl#>.
3  @prefix asawoo-wtr: <http://liris.cnrs.fr
       /asawoo/vocab/watering-application.
       owl#>.
4
5  asawoo-wtr:F1 rdf:type
       asawoo-ctx:ScoringFunction .
6  asawoo-wtr:F1 asawoo-ctx:applicableTo
       asawoo-wtr:Prtcl .
7  asawoo-wtr:F1 asawoo-ctx:forDimension
       asawoo-wtr:Battery .
8
9  asawoo-wtr:F1 asawoo-ctx:scores [
10   asawoo-ctx:forCandidate
         asawoo-wtr:Bluetooth ;
11   asawoo-ctx:forAdapted
         asawoo-wtr:TransferPicture ;
12   asawoo-ctx:withInstance
         asawoo-wtr:HighBattery ;
13   rdf:value "0.5"^^xsd:double ] .
```

After importing instances, scoring functions and possibilities, the SPARQL query from Listing 6 below is processed, to calculate scores for each group (contextual situation, adapted functionality, possibility candidate). Line 4-6 select the atoms that will compose each respective adaptation rule, compute the total score of the possibility candidate with the SUM aggregate (line 5), and retrieve each contextual instance from the current contextual situation using the GROUP_CONCAT aggregate at line 6 (these two aggregates come with the corresponding GROUP BY at line 19). Lines 8-10 link the adaptation possibility with the contextual situation. Line 12 retrieves the scoring functions applicable to the current purpose. Lines 13-17 then retrieve the score of each contextual instance that belongs to the current contextual situation, by reification.

Listing 6: SPARQL query that retrieves each group of atoms composing the scored adaptation rules.

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22
       -rdf-syntax-ns#>
2  PREFIX asawoo-ctx: <http://liris.cnrs.fr/
       asawoo/vocab/context.owl#>
3
4  SELECT DISTINCT ?adapted ?purposePred ?
       candidate
5    (SUM(?score) AS ?candidateScore)
6    (GROUP_CONCAT(?contextInstance) AS ?
         instances) {
7      ?adapted ?purposePred ?candidate .
8      ?purpose asawoo-ctx:purposePredicate
           ?purposePred .
9      ?contextSituation
           asawoo-ctx:containsInstance ?
           contextInstance .
10
11     ?scoringFunction
           asawoo-ctx:applicableTo ?purpose
           .
```

```
    ?scoringFunction asawoo-ctx:scores [
      asawoo-ctx:withInstance ?
          contextInstance ;
      asawoo-ctx:forCandidate ?candidate
          ;
      asawoo-ctx:forAdapted ?adapted
      rdf:value ?score ]
} GROUP BY ?adapted ?purposePred
          ?candidate ?contextSituation
```

Afterwards, each score is divided by the number of effective contextual instances from each contextual situation, to avoid invalid score computation if a contextual instance is missing (*e.g.* when a sensor is unable to send data).

5.5. Adaptation Rule Generation

The *adaptive rule generation* step produces a set of adaptation rules (in the form detailed in Definition 6) using the SELECT bindings returned by the scoring application step. Algorithm 2 below details this process.

ALGORITHM 2: Scored adaptation rule generation

Data: A set B of SPARQL bindings with the variables { *adapted*, *purposePredicate*, *candidate*, *candidateScore*, *instances* } corresponding to the scoring application output.
Result: A set R of conjunctive adaptation rules.
$R \leftarrow [\,]$
foreach $b \in B$ **do**
  $cause \leftarrow new\ Conjunction()$
  foreach $atom \in b.instances$ **do**
    $cause.addConjunctiveAtom$(" *atom* asawoo-ctx:isEffectiveInstance xsd:true")
  $consequence \leftarrow new\ Conjunction()$
  $consequence.addConjunctiveAtom$(" _:blankNode rdf:subject $b.adapted$")
  $consequence.addConjunctiveAtom$(" _:blankNode rdf:predicate $b.purposePredicate$")
  $consequence.addConjunctiveAtom$(" _:blankNode rdf:object $b.candidate$")
  $consequence.addConjunctiveAtom$(" _:blankNode rdf:value $b.candidateScore$")
  $tmpRule \leftarrow R.getRuleWithCause(cause)$
  if $tmpRule\ != null$ **then**
    $tmpRule.addConsequence(consequence)$;
  else $R.push(new\ Rule(cause, consequence))$;
return R

In Algorithm 2, causes and consequences are generated using the function $addConjunctiveAtom()$, which adds an atom (a triple) in the conjunction. The cause is a conjunction of contextual instances, and the consequence is a set of scored adaptation possibility (*i.e.* a set of reified blank nodes, as described in the Definition 5 from Section 4). The predicate `asawoo-ctx:isEffectiveInstance` denotes the current presence of contextual instances in the model, which would trigger the suitable adaptation rules. At each loop, if a rule with the generated cause already exists in the set ($getRuleWithCause()$ function), the generated consequence is added this already existing rule ($addConsequence()$ function). Otherwise, a new adaptation rule is created and added to the set of adaptation rules. After this step, the meta adaptation rule engine finally outputs the set of adaptation rules, to be integrated onto each avatar reasoner.

6. Evaluation

In this section, we present two different kinds of evaluations of our situation-based approach. We first evaluate its correctness by ensuring the adaptation results correspond to the expected adaptation results for different situations. Second, we verify that our approach covers every kind of adaptation question for every situation. In both cases, we evaluate our theoretical solution through the sustainable agriculture scenario described in Section 3. We ran all experiments on a Dell OptiPlex 780 - Core 2 Duo E8400 @ 3 GHz and used Stardog 4.2.3[13] as a triplestore.

6.1. Situation-based adaptation correctness

We evaluate the correctness of our solution, *i.e.* we check that the adaptation rules generated by our solution produces the expected scored adaptation possibilities, with respect to the purposes described in the previous subsection. We consider the following situations: 1a) and 1b) are two different situations for the implementation of TakeHdPicture by different drones (distinct storage capacities and camera resolutions); 2a) and 2b) are two different drone situations for the composition of DetectWateringNeeds (distinct locations and storage capacities); 3) queries the SprayWater exposability during a flood; 4) requires the TransferPicture to choose a protocol in low-battery conditions for a far drone; and 5) require to find the accurate ProcessPicture code location in a situation with a low CPU availability and a low battery level.

[13] http://stardog.com/

6.1.1. Situations 1a and 1b: Heterogeneous drones in terms of storage and camera resolution

The functionality TakeHdPicture can be implemented by two different drones with distinct situations. We expect an optimal decision, otherwise the watering needs detection would fail. We consider drone 1, which has high storage capacity but has a low quality camera, and drone 2, which has limited storage capacity but is equipped with a HD camera.

Table 2
Situations 1a/1b implementation scores.

Drone	CPU availability	Score
#1	High	0.6
#1	Low	0.3
#2	High	0.7
#2	Low	0.4

Table 2 shows the scores obtained for the implementation of TakeHdPicture for each drone, with varying CPU availabilities. Results shows that the camera quality is the most impacting contextual information to determine the implementation of TakeHdPicture. However, a low quality drone will always be preferred over a HD one if it has both higher storage capacity and high CPU availability. This means that taking a high definition picture when having low storage capacity with a busy CPU does not ensure acceptable response times, and can even cause disruptions in worst cases.

6.1.2. Situations 2a and 2b: Heterogeneous drones in terms of storage, located over different parts of the field

Table 3
Situations 2a/2b composition scores.

Drone	Battery level	Score
#3	High	0.6
#3	Low	0.3
#4	High	0.7
#4	Low	0.4

Two drones can take pictures to detect the watering needs. As these drones experience different situations, an optimal choice is expected to limit the energy and time consumptions on drones. We consider drone 3, which is far from the field but has high storage capacity, and drone 4, which is closer from the field but has a lower storage capacity.

We choose to vary the battery levels of drones. Table 3 details the scores obtained for the composition of DetectWateringNeeds for each drone. As expected, results shows that drones closer from the field are always preferred to take pictures. Further more, on this adaptation configuration, having a low battery level on a drone significantly lowers its composition score (regardless of its location).

6.1.3. Situation 3: Flood

The system must determine if the weather condition is satisfiable to expose the SprayWater functionality. In a flood situation, the Temperature and Wind dimensions have no impact; as long as the contextual instance "Flooded" is present, the score of the possibility $(SprayWater\ hasExposability\ Exposable)$ is always 0. Thus, the functionality of SprayWater is never exposable for this situation.

Table 4
Situation 3 exposability scores.

Wind	Temp.	Score
StrongWind	Hot	0.4
StrongWind	Warm	0.4
StrongWind	Cold	0.4
Breeze	Hot	0.7
Breeze	Warm	0.7
Breeze	Cold	0.7
NoWind	Hot	0.7
NoWind	Warm	0.7
NoWind	Cold	0.7

The scores obtained in Table 4 also show that the contextual dimension Temperature has no impact on scores whatsoever, and that the Breeze and NoWind instances from the Wind dimension produce identical scores. Thus, this adaptation configuration also takes into account stormy weather situations.

6.1.4. Situation 4: Picture transmission to a far drone with low battery

In this situation, a drone must send a picture of the field to an another one in order to process it. However, the only drone able to receive is far, and the one that sends the picture has a low battery level. The system must choose the best protocol to use, to save as much battery level as possible and to send the picture with acceptable delays.

Table 5 shows that the adaptation configuration strongly favors Wifi if the picture receiver is too far.

Table 5
Situation 4 protocols scores.

Protocol (far)	**Score**
Bluetooth	0.1
Wifi	0.6
Protocol (close)	**Score**
Bluetooth	0.7
Wifi	0.6

Indeed, Bluetooth has a limited range. However, Bluetooth has a slightly higher score if drones are close from each other (regardless of the battery level). Still, as it consumes more battery than Wifi, using Bluetooth is also viable.

6.1.5. Situation 5: Code host and execution for a drone with limited capacities

In this situation, the system must determine the best solution to provide picture processing on a drone with limited capacities. It must determine the best location to host and execute this functionality, in order to process pictures as fast as possible.

Table 6
Situation 5 code locations scores.

Storage capacity	**Drone location score**	**Cloud location score**
Low	0	1
High	0.3	0.9

Results from Table 5 show that the adaptation configuration prevents from locating the ProcessPicture functionality code on the device. It also shows that a high storage capacity is not enough for the device to be the optimal code location. In this configuration, we expect at least one more resource (either CPU or battery level) to be "High", to allow a score $>= 0.7$ for locating the code on the device. For this reason, the current situation always favors the cloud as the code location.

6.2. Adaptation coverage

The adaptation coverage means that each possible contextual situation (i.e. each identified contextual instances conjunctions) necessarily produces an adaptation possibility. Our solution provides adaptation coverage by design, *i.e.* as long as the contextual dimension and its instances cover the necessary range for a given contextual information. This coverage applies to each adaptation purpose, independently of the domain. The number of dimensions is not limited, and can be projected on each adaptation purpose. We define the adaptation solution coverage for a given application as follows.

Definition 9 (Adaptation Solution Coverage)**.** *Let $\mathcal{AP}$ be the set of adaptation purposes for an adaptation solution Δ. Let $ap \in \mathcal{AP}$ be an adaptation purpose, and $S_{ap} = \{\varsigma_1, ..., \varsigma_n\}$ a set composed of each possible contextual situations for ap. Let R be the set of adaptation rules for Δ. We assume that each rule in R has distinct causes. The number of possible situations for a purpose ap is $|S_{ap}| = |\varsigma_1 \times ... \times \varsigma_n| = \prod_{i=1}^{n} |\varsigma_n|$. Thus, the coverage of Δ corresponds to the number of distinct causes divided by the sum of all possible situations for each adaptation purpose, i.e.*

$$Cov(\Delta) = \frac{|R|}{\sum_{j=1}^{m} |S(ap_m)|}, m = |\mathcal{AP}|$$

For the evaluated scenario, our adaptation solution Δ outputs 59 rules with unique causes ($|R| = 59$). For each purpose, we have the following number of situations:

$$|S(Exp)| = 3 \times 3 \times 3 = 27 \quad (1)$$

$$|S(Imp)| = 2 \times 2 \times 3 = 12 \quad (2)$$

$$|S(Comp)| = 2 \times 2 \times 2 \quad (3)$$

$$|S(Prtcl)| = 2 \times 2 = 4 \quad (4)$$

$$|S(CdL)| = 2 \times 2 \times 2 = 4 \quad (5)$$

The total number of situations is therefore
$S(\mathcal{AP})| = |S(Exp)| + |S(Imp)| + |S(Comp)| + |S(Prtcl)| + |S(CdL)| = 27 + 12 + 8 + 4 + 8 = 59$
Thus, we ensured a total coverage as
$Cov(\Delta) = \frac{|R|}{|S(\mathcal{AP})|} = \frac{59}{59} = 1.$

7. Discussion

In this work, we chose to rely on a semantic infrastructure to perform adaptation. The choice of using semantic reasoning is questionable. Indeed, our whole adaptive solution takes raw sensor data – a number – as input, infers contextual instances from these data, and attributes a score to adaptation possibilities regarding a set of contextual instances, which is also a number. However, the meaning of the score number is not the same. In a sense, scoring allows each type of information to be compared to each other, without consider-

ing the data units and ranges. Semantics also provide reusability through linked open vocabularies[14] and domain knowledge expertise ontologies[15], in an interoperable manner. Being a proven standard, OWL guarantees the inference correctness at all expressivity levels (being $\mathcal{AL}+$ in our solution). Thus, having a *correct* adaptation requires consistent rules and scores, which depends on the design of the rules and the scoring functions rather than on the adaptation solution. To ensure this, we always keep the correct orders between adaptation possibility scores during the rule engine transformation processes, as we always compute the average amongst each individual contextual instance score related to a given possibility.

7.1. Runtime performance

In this work, we propose a semi-anticipated adaptation planning approach. At design time, the application designer and experts pave the way for generating adaptation possibilities and situation-based adaptation rules. These possibilities and rules are inferred at runtime and triggered by context change events. In this sense, this approach can be considered anticipated, as it pre-processes some (most of the) necessary element on which the adaptation planning is based. However, plans are not yet available at context change. Actual adaptation planning corresponds to the selection of an adaptation possibility (if any), which depends on the scores of these possibilities that are computed at request time (*i.e.* unanticipatedly). This strategy allows reconciling the cost and benefits of relying on an inference engine. Indeed, while inferences are processed at runtime, they are not triggered when an adaptation request arises, but can be processed in parallel. The inference results are only integrated in the context graph when they have all been inferred, so that the graph that is queried for adaptation always keep consistent. Moreover, performance also comes from the fact that we use an incremental reasoner[16] that only recomputes the parts of the graph that are impacted by context changes. In [36], we have evaluated the time performance of the adaptation process. The processing times were respectively less than 650ms and 30ms for the contextual data integration and the adaptation decision processes[17].

7.2. Optimization of adaptation rule set

The choice of dimensions and instances is also crucial to pre-reduce the set of adaptation rules. The number of instances per dimension is fixed by semantization thresholds, which allow inferring contextual instances regarding a raw value range. The fuzzy logic-based solution proposed in [38] shows that discretization considerably reduces the number of rules produced for an adaptation solution. This is one of the reasons we developed the concept of context situation, which also aims at identifying and targeting as soon as possible a given "type of adaptation", and at optimizing the rule according to this target. Moreover, from a human-centered point of view, the notion context situation is also easy to handle by domain experts as it copes with the way they model application domains. As such, it is provided as a tool to facilitate the interactions between domain experts and application designers.

7.3. Genericity and adaptability of our approach

In our approach, adaptation rules always infer "positive" answers, *i.e.* our adaptation solution is not built upon negative assumptions. This way, the system reasons about contextual information in a open-world assumption: the absence of data does not imply that the contextual information is false or invalid, but rather is unknown. By relying on the open-world assumption characteristic of OWL, we avoid to unexpectedly block application functionalities because a data is missing, or not available, or if it takes longer to transfer. Moreover, the scoring functions can take into account this imprecision by normalizing scores according to the number of actually available observations (aka dimensions). This makes our approach dynamically adaptive to context changes.

On the long run, our approach can also be adapted throughout projects and platforms. dimensions can be removed or added at design time. In turn, adaptation

[14] Linked Open Vocabularies (LOV) – http://lov.okfn.org/dataset/lov/

[15] Linked Open Vocabularies for Internet of Things (LOV4IoT) – http://sensormeasurement.appspot.com/?p=ontologies

[16] Incremental reasoning allows capitalizing on previous reasoning process. The inferred information is continuously maintained in a graph, and the algorithm only processes the knowledge impacted by new insertions and deletions. In this work we used the HyLAR (https://github.com/ucbl/HyLAR-Reasoner/) reasoner.

[17] For data integration, we have assumed one second as the commonly admitted threshold upon which the user's attention stops focusing on the current task. The decision step is however time critical; we have fixed its acceptable response time to 100ms.

purposes can vary according to the platform needs. For instance, at some point of the software maintenance cycle, the adaptation solution may require an additional adaptation purpose (corresponding to a new identified adaptation need). In that case, the application designers have to identify the possible candidates for this purpose as well as their nature (identified as objects in Section 4 – Table 1). They must also identify the contextual dimensions and their set of contextual instances, as well as the scoring functions required to provide adaptation for this purpose.

8. Conclusion

In this paper, we propose a solution to provide multi-purpose context adaptation in WoT applications. Our solution relies on a semantic WoT platform, able to incrementally reason about contextual information to support situation-based adaptation for multiple purposes. We formalize the notions of multi-purpose and situation-based adaptations, and detail how this process generates adaptation rules in a declarative way, through the use of a meta adaptation rule engine. Our approach defines sets of adaptation possibilities for each adaptation purpose, that are scored using scoring functions. We present our implementation of this engine and evaluate it on a sustainable agriculture scenario, for several situations and on different criteria (corectness and coverage). We discuss the positioning of our semantic approach and its performance (in terms of computation times) and adaptability.

Our perspectives include dynamically generating scoring functions to reduce the design time of application design. We also aim at integrating semantization rules as in S-LOR, to generate contextual instances in a reusable manner, and to combine the deterministic inferences used to make adaptation decisions with probabilistic feedback learned from of previous adaptation decisions using probabilistic ontologies using for instance PR-OWL [39].

Acknowledgements

This work is supported by the French ANR (Agence Nationale de la Recherche) under the grant number <ANR-13-INFR-012>.

References

[1] F. Manola, E. Miller, B. McBride et al., RDF primer, *W3C recommendation* **10**(1–107) (2004), 6.

[2] D. Brickley, R.V. Guha and B. McBride, RDF Schema 1.1, *W3C recommendation* **25** (2014), 2004–2014.

[3] P. Hitzler, M. Krötzsch, B. Parsia, P.F. Patel-Schneider and S. Rudolph (eds), *OWL 2 Web Ontology Language: Primer*, W3C Recommendation, 27 October 2009, Available at http://www.w3.org/TR/owl2-primer/.

[4] A. Haller, K. Janowicz, S.J. Cox, M. Lefrançois, K. Taylor, D. Le Phuoc, J. Lieberman, R. García-Castro, R. Atkinson and C. Stadler, The SOSA/SSN ontology: a joint WeC and OGC standard specifying the semantics of sensors observations actuation and sampling, in: *Semantic Web*, Vol. 1, IOS Press, 2018, pp. 1–19.

[5] M. Mrissa, L. Médini, J.-P. Jamont, N. Le Sommer and J. Laplace, An avatar architecture for the web of things, *IEEE Internet Computing* **19**(2) (2015), 30–38.

[6] F. Barbier, E. Cariou, O.L. Goaer and S. Pierre, Software Adaptation: Classification and a Case Study with State Chart XML., *IEEE Software* **32**(5) (2015).

[7] J. Fox and S. Clarke, Exploring approaches to dynamic adaptation, in: *Proceedings of the 3rd International DiscCoTec Workshop on Middleware-Application Interaction*, ACM, 2009, pp. 19–24.

[8] K. Hanney, M. Keane, B. Smyth and P. Cunningham, Systems, tasks and adaptation knowledge: Revealing some revealing dependencies, in: *International Conference on Case-Based Reasoning*, Springer, 1995, pp. 461–470.

[9] J. Becker, P. Delfmann and R. Knackstedt, Adaptive reference modeling: integrating configurative and generic adaptation techniques for information models, in: *Reference Modeling*, Springer, 2007, pp. 27–58.

[10] S.M. Sadjadi and P.K. McKinley, ACT: An adaptive CORBA template to support unanticipated adaptation, in: *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, IEEE, 2004, pp. 74–83.

[11] W. Kongdenfha, R. Saint-Paul, B. Benatallah and F. Casati, An aspect-oriented framework for service adaptation, in: *International Conference on Service-Oriented Computing*, Springer, 2006.

[12] C. Zeginis and D. Plexousakis, Web service adaptation: State of the art and research challenges, *Self* **2** (2010), 5.

[13] J.-Y. Tigli, S. Lavirotte, G. Rey, V. Hourdin, D. Cheung-Foo-Wo, E. Callegari and M. Riveill, WComp middleware for ubiquitous computing: Aspects and composite event-based Web services, *annals of telecommunications-annales des télécommunications* **64**(3–4) (2009).

[14] L. Seinturier, P. Merle, D. Fournier, N. Dolet, V. Schiavoni and J.-B. Stefani, Reconfigurable sca applications with the frascati platform, in: *Services Computing, 2009. SCC'09. IEEE International Conference on*, IEEE, 2009, pp. 268–275.

[15] M. Beisiegel, H. Blohm, D. Booz, J.-J. Dubray, A.C. Interface21, M. Edwards, D. Ferguson, J. Mischkinsky, M. Nally and G. Pavlik, Service component architecture, *Building systems using a Service Oriented Architecture. BEA, IBM, Interface21, IONA, Oracle, SAP, Siebel, Sybase, white paper, version* **9** (2007).

[16] M. Mikalsen, N. Paspallis, J. Floch, E. Stav, G.A. Papadopoulos and A. Chimaris, Distributed context management in a mobility and adaptation enabling middleware (madam), in: *Proceedings of the 2006 ACM symposium on Applied computing*, ACM, 2006, pp. 733–734.

[17] A.K. Dey, Understanding and using context, *Personal and ubiquitous computing* **5**(1) (2001), 4–7.

[18] T. Berners-Lee, J. Hendler, O. Lassila et al., The semantic web, *Scientific american* **284**(5) (2001), 28–37.

[19] S. Laborie, J. Euzenat and N. Layaïda, Semantic adaptation of multimedia documents, *Multimedia tools and applications* **55**(3) (2011), 379–398.

[20] C. Baladron, J.M. Aguiar, B. Carro, L. Calavia, A. Cadenas and A. Sanchez-Esguevillas, Framework for intelligent service adaptation to user's context in next generation networks, *IEEE Communications Magazine* **50**(3) (2012).

[21] X.H. Wang, D.Q. Zhang, T. Gu and H.K. Pung, Ontology based context modeling and reasoning using OWL, in: *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, Ieee, 2004, pp. 18–22.

[22] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin and V.Y. Terziyan, Smart Semantic Middleware for the Internet of Things. (2008).

[23] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kröller, M. Pagel, M. Hauswirth et al., SPITFIRE: toward a semantic web of things, *IEEE Communications Magazine* **49**(11) (2011), 40–48.

[24] Z. Shelby, K. Hartke and C. Bormann, The constrained application protocol (CoAP) (2014).

[25] P. Barnaghi and M. Presser, Publishing linked sensor data, in: *Proceedings of the 3rd International Conference on Semantic Sensor Networks-Volume 668*, CEUR-WS. org, 2010, pp. 1–16.

[26] A. Gyrard, S.K. Datta, C. Bonnet and K. Boudaoud, Cross-domain Internet of Things application development: M3 framework and evaluation, in: *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, IEEE, 2015, pp. 9–16.

[27] S.-N. Chuang and A.T. Chan, Dynamic QoS adaptation for mobile middleware, *IEEE Transactions on Software Engineering* **34**(6) (2008), 738–752.

[28] K. Kakousis, N. Paspallis and G.A. Papadopoulos, A survey of software adaptation in mobile and ubiquitous computing, *Enterprise Information Systems* **4**(4) (2010), 355–389.

[29] J. Floch, S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund and E. Gjorven, Using architecture models for runtime adaptability, *IEEE software* **23**(2) (2006), 62–70.

[30] A. Gyrard, M. Serrano, J.B. Jares, S.K. Datta and M.I. Ali, Sensor-based Linked Open Rules (S-LOR): An Automated Rule Discovery Approach for IoT Applications and its use in Smart Cities, in: *Proceedings of the 26th International Conference Companion on World Wide Web*, International World Wide Web Conferences Steering Committee, 2017, p. .

[31] A. Boussadi, C. Bousquet, B. Sabatier, T. Caruba, P. Durieux, P. Degoulet et al., A business rules design framework for a pharmaceutical validation and alert system, *Methods of information in medicine* **50**(1) (2011), 36.

[32] C.K. Emani, Automatic detection and semantic formalisation of business rules, in: *European Semantic Web Conference*, Springer, 2014, pp. 834–844.

[33] J. Andersson, M. Ericsson and W. Lowe, Automatic rule derivation for adaptive architectures, in: *Software Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference on*, IEEE, 2008, pp. 323–326.

[34] J. Hong, E.-H. Suh, J. Kim and S. Kim, Context-aware system for proactive personalized service based on context history, *Expert Systems with Applications* **36**(4) (2009), 7448–7457.

[35] M. Terdjimi, L. Médini and M. Mrissa, Towards a Meta-model for Context in the Web of Things, in: *Karlsruhe Service Summit Workshop*, 2016.

[36] M. Terdjimi, L. Médini, M. Mrissa and M. Maleshkova, Multi-purpose Adaptation in the Web of Things, in: *CONTEXT-17*, 2017.

[37] M. Terdjimi, L. Médini, M. Mrissa and N. Le Sommer, An avatar-based adaptation workflow for the web of things, in: *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2016 IEEE 25th International Conference on*, IEEE, 2016, pp. 62–67.

[38] M. Beggas, L. Médini, F. Laforest and M.T. Laskri, Towards an ideal service QoS in fuzzy logic-based adaptation planning middleware, *Journal of Systems and Software* **92** (2014), 71–81.

[39] P.C.G. Da Costa, K.B. Laskey and K.J. Laskey, PR-OWL: A Bayesian ontology language for the semantic web, in: *Uncertainty Reasoning for the Semantic Web I*, Springer, 2006, pp. 88–107.

[40] M. Salehie and L. Tahvildari, Self-adaptive software: Landscape and research challenges, *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* **4**(2) (2009), 14.

[41] R.T. Fielding, Architectural styles and the design of network-based software architectures, PhD thesis, University of California, Irvine, 2000.

[42] A. Gyrard, C. Bonnet, K. Boudaoud and M. Serrano, Assisting iot projects and developers in designing interoperable semantic web of things applications, in: *Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on*, IEEE, 2015, pp. 659–666.

[43] F.D. Macías-Escrivá, R. Haber, R. del Toro and V. Hernandez, Self-adaptive systems: A survey of current approaches, research challenges and applications, *Expert Systems with Applications* **40**(18) (2013), 7267–7279.

[44] J.-P. Jamont, L. Médini and M. Mrissa, A web-based agent-oriented approach to address heterogeneity in cooperative embedded systems, in: *Trends in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*, Springer, 2014, pp. 45–52.

[45] D. Zeng, S. Guo and Z. Cheng, The web of things: A survey, *JCM* **6**(6) (2011), 424–438.

[46] G. Montenegro, N. Kushalnagar, J. Hui and D. Culler, Transmission of IPv6 packets over IEEE 802.15. 4 networks, Technical Report, 2007.

[47] F. Scioscia and M. Ruta, Building a Semantic Web of Things: issues and perspectives in information compression, in: *Semantic Computing, 2009. ICSC'09. IEEE International Conference on*, IEEE, 2009, pp. 589–594.

[48] M. Ruta, F. Scioscia and E. Di Sciascio, Enabling the semantic web of things: Framework and architecture, in: *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, IEEE, 2012, pp. 345–347.

[49] F. Baader, *The description logic handbook: Theory, implementation and applications*, Cambridge university press, 2003.

[50] A. Gyrard, C. Bonnet and K. Boudaoud, Helping IoT application developers with sensor-based linked open rules, in: *SSN 2014, 7th International Workshop on Semantic Sensor Networks in conjunction with the 13th International Semantic Web Conference (ISWC 2014), 19-23 October 2014, Riva Del Garda, Italy*, Vol. 10, 2014.

[51] C. Yu and L. Popa, Semantic adaptation of schema mappings when schemas evolve, in: *Proceedings of the 31st international conference on Very large data bases*, VLDB Endowment, 2005, pp. 1006–1017.

[52] M.K. Asadi and J.-C. Dufourd, Context-aware semantic adaptation of multimedia presentations, in: *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, IEEE, 2005, pp. 362–365.

[53] F. Boulanger, C. Hardebolle, C. Jacquet and D. Marcadet, Semantic adaptation for models of computation, in: *Application of Concurrency to System Design (ACSD), 2011 11th International Conference on*, IEEE, 2011, pp. 153–162.

[54] M. Baldoni, C. Baroglio, V. Patti and L. Torasso, Reasoning about learning object metadata for adapting SCORM courseware, *Engineering the Adaptive Web., CS-Report* (2004), 04–18.

[55] M. Zufferey and H. Kosch, Semantic adaptation of multimedia content, in: *Multimedia Signal Processing and Communications, 48th International Symposium ELMAR-2006 focused on*, IEEE, 2006, pp. 319–322.

[56] D.G. Sampson, M.D. Lytras, G. Wagner and P. Diaz, Ontologies and the Semantic Web for E-learning, *Educational Technology & Society* **7**(4) (2004), 26–28.

[57] M. Bertini, R. Cucchiara, A. Bimbo and A. Prati, Semantic adaptation of sport videos with user-centred performance analysis, *IEEE Transactions on Multimedia* **8**(3) (2006), 433–443.

[58] D. Ejigu, M. Scuturici and L. Brunie, An ontology-based approach to context modeling and reasoning in pervasive computing, in: *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, IEEE, 2007, pp. 14–19.