Contextual Adaptation Engine

Deliverable 4.2 (version 1)

Université Lyon 1 - LIRIS

27 May 2015

Project ASAWoO
Adaptive Supervision of Avatar / Object Links for the Web of Objects
Grant Agreement: ANR-13-INFR-0012-04

Abstract

Existing adaptation solutions for Web of Things (WoT) applications are tightly coupled with application domains. In this deliverable, we propose a contextual adaptation 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.

Contents

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, among which the exposition of things as RESTful resources [10] and semantic descriptions of these resources with description logics (RDF [18], RDF-S [8], OWL2 [14]). WoT standards thus define semantic vocabularies (Thing Description[2]) and programming interfaces (servient[3], scripting API[4]) reused in projects to design actual WoT applications. In the ASAWoO project, we proposed a component-based framework to ease the development of such applications in diverse domains (industry, healthcare, agriculture...). This framework [20] 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 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 deliverable, 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 provide adaptation rules in both user-friendly and generic ways. We formally describe the theoretical adaptation framework that supports context management and details the design of adaptation rules.
This deliverable is structured as follows. In Section 2, we overview related work on adaptation classification and WoT-based solutions. In Section 3 we present a solution to illustrate our research problem. In Section 4, we present our contributions on domain independent multi-purpose context adaptation. In Section 5, we open a discussion regarding the questions brought by our solution. We conclude in Section 6.

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

[2] 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 [11] 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.

[1] https://www.w3.org/WoT/
[2] https://www.w3.org/WoT/IG/wiki/Thing_Description
[3] https://w3c.github.io/wot/architecture/wot-architecture.html#general-description-of-wot-servient
[4] https://www.w3.org/WoT/IG/wiki/Web_of_Things_scripting_API

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 [13, 2] [11, 4.4]. The following adaptation frameworks illustrate these techniques and stress their advantages and drawbacks for generic adaptation approaches.
In [4], 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) [22], 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 [16], as well as to substitution of service-based applications [30], or more generally to adaptation in dynamic, component-based middlewares.
WComp [28] 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 [23] aims to extend the Service Component Architecture (SCA) [6] 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 [19] 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 [9] "*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 [7]. Hence, it appears quite relevant to use semantic web languages (RDF, RDF-S, OWL)[5] to model contextual information, as well as to reason about contextual data to make adaptation decisions.
[17] 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 [1] 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, [29] 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. We

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

intend to reuse Web standards and provide compliance with the WoT Interest Group specifications (Thing Description, Servient), to allow for adaptive WoT applications design.

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 [15] uses semantic annotations to describe agents and behavioral tasks, to provide interoperability and reusability of these definitions. SPITFIRE [21] unites RESTful approaches using CoAP [24] with OWL/RDFS and SPARQL[6] for constraint devices. Sense2Web [3] 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 [12]. While these approaches facilitate things interoperability and WoT application development, they do not tackle the adaptation concern.

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

Our work is based on the ASAWoO project[7]. ASAWoO consists in a WoT platform where each object has its own *avatar* [20]. An avatar is a form of servient (a virtual representation of the object) that allows access to and control over an object on the Web.
In ASAWoO, we distinguish the object capabilities (*i.e.* the physical API provided by the manufacturer) from the object functionalities (*i.e.* high-level tasks). Avatars exposes the object functionalities, which can be either atomic (*i.e.* directly implemented by the object physical capabilities), or composed by lower-level

[6] https://www.w3.org/TR/rdf-sparql-query/
[7] https://liris.cnrs.fr/asawoo/

functionalities. Functionalities are semantically annotated, in order to infer both atomic and composed functionalities for each avatar.
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 [25], 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*.

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.).*
$d = \{i_d\}$, $d \in \mathscr{D}$ *where* $\mathscr{D}$ *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.*
$ap = \{i_{ap}\}$, $ap \in \mathscr{AP}$.

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

- Imp Finding the best capability candidates to implement an atomic functionality,
- $Comp$ Finding the best functionality candidates to take part in a functionality composition (either locally on the object or with 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 object during the execution of a functionality,
- CdL Finding whether to locate the application code modules on the object processing unit, on a local machine (*e.g.* a gateway) or on a cloud infrastructure.

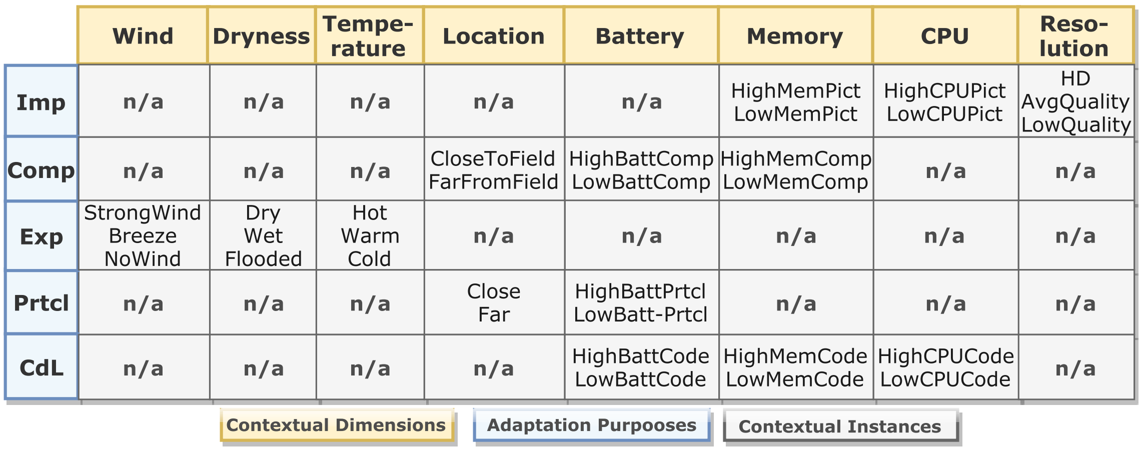
In our scenario, the "Temperature" dimension contains instances for the the Exp adaptation purpose only, as it allows deciding if the SprayWater functionality should be exposed or not.

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* $\mathscr{M}$ *is a two-dimensional set of contextual instances corresponding to both adaptation purposes* $\mathscr{AP}_{\mathscr{M}}$ *and contextual dimensions* $\mathscr{D}_{\mathscr{M}}$*. Within a given model* $\mathscr{M}$*, dimensions and adaptation purposes are respectively disjoint in* $\mathscr{D}_{\mathscr{M}}$ *and* $\mathscr{AP}_{\mathscr{M}}$*.*
$\mathscr{M} = \mathscr{AP}_{\mathscr{M}} \times \mathscr{D}_{\mathscr{M}} = \{i_{ap_{\mathscr{M}};d_{\mathscr{M}}}\}$ *where* $ap_{\mathscr{M}} \in \mathscr{AP}_{\mathscr{M}}$ *and* $d_{\mathscr{M}} \in \mathscr{D}_{\mathscr{M}}$*.*

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.



	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

Figure 1: The context model of our scenario.

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, which 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 [26] 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.

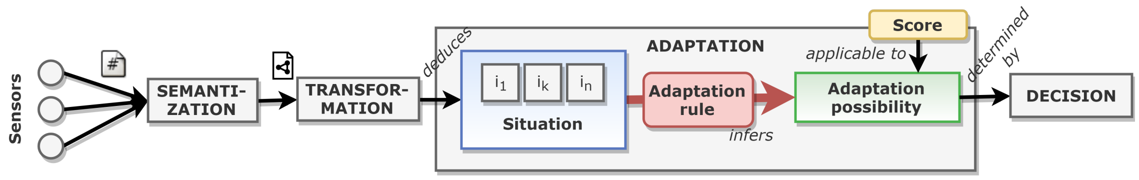


Figure 2: The contextual adaptation workflow

4.2.1 Semantization step

The contextual adaptation solution we have proposed in [27] relies on semantized contextual instances implemented as RDF triples. At runtime, sensors send data to avatars to be semantized, *i.e.* converted into RDF triples. For instance, if a thermometer sends the value “20.388”, the system will interpret this information as the following triple: `asawoo:Thermometer rdf:value “20.388”`.

4.2.2 Transformation step

As soon as numerical values are semantized as RDF triples, they are inserted into the 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, the triple `(asawoo:Thermometer rdf:value “20.388”)` 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 formalize them 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.*
$\varsigma = \{i_{j,k}\}$ *where* $j \in \mathscr{A}\mathscr{P} \cup \emptyset$ *and* $k \in \mathscr{D} \cup \emptyset$

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. Table 1 below details the possibility triple patterns for each purpose.

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* $p_{f,ap} \in P_{f,ap}$.

Table 1: Pattern of the possibility triple t 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

For instance, in our scenario, the adaptation of communication protocols for the TransferPicture functionality may have either Wifi of Bluetooth 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:* $\bigwedge_{k\leq|\varsigma|} i_k \rightarrow P$ *where* $i_k \in \varsigma$ *is a contextual instance and* P *a set of inferred adaptation possibilities* p*. Each purpose* $ap \in \mathscr{A}\mathscr{P}$ *is associated to a set* $\mathscr{R}_{ap}$ *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
$if\ \{Dry, Breeze, Warm\}\ then\ \{(SprayWaterhasExposabilityExposable)\}$

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

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

In ASAWoO, 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 1: SPARQL-based 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 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 considering the data units and ranges. Semantics also provide reusability through linked open vocabularies[8] and domain knowledge expertise ontologies[9], in an interoperable manner. Being a proven standard, OWL guarantees the inference correctness at all expressivity levels (being $\mathscr{AL}+$ 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.

5.1 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[10] that only recomputes the parts of the graph that are impacted by context

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

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

[10]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.

changes. In [27], 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[11].

5.2 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 [5] shows that discretization considerably reduces the number of rules produced for an adaptation solution. However, we could not literally reuse the same approach of targeting an ideal variant, as fuzzy logics can admit uncertainty, whereas semantic inferences cannot. This is one of the reasons why we developed this 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.

5.3 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 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.

6 Conclusion

In this deliverable, we present 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.

[11]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.

References

[1] Carlos Baladron, Javier M Aguiar, Belen Carro, Lorena Calavia, Alejandro Cadenas, and Antonio Sanchez-Esguevillas. Framework for intelligent service adaptation to user's context in next generation networks. *IEEE Communications Magazine*, 50(3), 2012.

[2] Franck Barbier, Eric Cariou, Olivier Le Goaer, and Samson Pierre. Software adaptation: Classification and a case study with state chart xml. *IEEE Software*, 32(5), 2015.

[3] Payam Barnaghi and Mirko Presser. Publishing linked sensor data. In *Proceedings of the 3rd International Conference on Semantic Sensor Networks-Volume 668*, pages 1–16. CEUR-WS. org, 2010.

[4] Jörg Becker, Patrick Delfmann, and Ralf Knackstedt. Adaptive reference modeling: integrating configurative and generic adaptation techniques for information models. In *Reference Modeling*, pages 27–58. Springer, 2007.

[5] Mounir Beggas, Lionel Médini, Frederique Laforest, and Mohamed Tayeb Laskri. Towards an ideal service qos in fuzzy logic-based adaptation planning middleware. *Journal of Systems and Software*, 92:71–81, 2014.

[6] Michael Beisiegel, Henning Blohm, Dave Booz, Jean-Jacques Dubray, Adrian Colyer Interface21, Mike Edwards, Don Ferguson, Jeff Mischkinsky, Martin Nally, and Greg Pavlik. Service component architecture. *Building systems using a Service Oriented Architecture. BEA, IBM, Interface21, IONA, Oracle, SAP, Siebel, Sybase, white paper, version*, 9, 2007.

[7] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[8] Dan Brickley, Ramanathan V Guha, and Brian McBride. Rdf schema 1.1. *W3C recommendation*, 25:2004–2014, 2014.

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

[10] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[11] Jorge Fox and Siobhán Clarke. Exploring approaches to dynamic adaptation. In *Proceedings of the 3rd International DiscCoTec Workshop on Middleware-Application Interaction*, pages 19–24. ACM, 2009.

[12] Amelie Gyrard, Soumya Kanti Datta, Christian Bonnet, and Karima 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*, pages 9–16. IEEE, 2015.

[13] Kathleen Hanney, Mark Keane, Barry Smyth, and Padraig Cunningham. Systems, tasks and adaptation knowledge: Revealing some revealing dependencies. In *International Conference on Case-Based Reasoning*, pages 461–470. Springer, 1995.

[14] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-primer/.

[15] Artem Katasonov, Olena Kaykova, Oleksiy Khriyenko, Sergiy Nikitin, and Vagan Y Terziyan. Smart semantic middleware for the internet of things. 2008.

[16] Woralak Kongdenfha, Régis Saint-Paul, Boualem Benatallah, and Fabio Casati. An aspect-oriented framework for service adaptation. In *International Conference on Service-Oriented Computing*. Springer, 2006.

[17] Sébastien Laborie, Jérôme Euzenat, and Nabil Layaïda. Semantic adaptation of multimedia documents. *Multimedia tools and applications*, 55(3):379–398, 2011.

[18] Frank Manola, Eric Miller, Brian McBride, et al. Rdf primer. *W3C recommendation*, 10(1-107):6, 2004.

[19] Marius Mikalsen, Nearchos Paspallis, Jacqueline Floch, Erlend Stav, George A Papadopoulos, and Akis Chimaris. Distributed context management in a mobility and adaptation enabling middleware (madam). In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 733–734. ACM, 2006.

[20] Michael Mrissa, Lionel Médini, Jean-Paul Jamont, Nicolas Le Sommer, and Jérôme Laplace. An avatar architecture for the web of things. *IEEE Internet Computing*, 19(2):30–38, 2015.

[21] Dennis Pfisterer, Kay Romer, Daniel Bimschas, Oliver Kleine, Richard Mietz, Cuong Truong, Henning Hasemann, Alexander Kröller, Max Pagel, Manfred Hauswirth, et al. Spitfire: toward a semantic web of things. *IEEE Communications Magazine*, 49(11):40–48, 2011.

[22] Seyed Masoud Sadjadi and Philip K McKinley. Act: An adaptive corba template to support unanticipated adaptation. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 74–83. IEEE, 2004.

[23] Lionel Seinturier, Philippe Merle, Damien Fournier, Nicolas Dolet, Valerio Schiavoni, and Jean-Bernard Stefani. Reconfigurable sca applications with the frascati platform. In *Services Computing, 2009. SCC'09. IEEE International Conference on*, pages 268–275. IEEE, 2009.

[24] Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (coap). 2014.

[25] Mehdi Terdjimi, Lionel Médini, and Michael Mrissa. Towards a meta-model for context in the web of things. In *Karlsruhe Service Summit Workshop*, 2016.

[26] Mehdi Terdjimi, Lionel Médini, Michael Mrissa, and Nicolas 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*, pages 62–67. IEEE, 2016.

[27] Mehdi Terdjimi, Lionel Médini, Michael Mrissa, and Maria Maleshkova. Multi-purpose adaptation in the web of things. In *CONTEXT-17*, 2017.

[28] Jean-Yves Tigli, Stéphane Lavirotte, Gaëtan Rey, Vincent Hourdin, Daniel Cheung-Foo-Wo, Eric Callegari, and Michel Riveill. Wcomp middleware for ubiquitous computing: Aspects and composite event-based web services. *annals of telecommunications-annales des télécommunications*, 64(3-4), 2009.

[29] Xiao Hang Wang, D Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 18–22. Ieee, 2004.

[30] Chrysostomos Zeginis and Dimitris Plexousakis. Web service adaptation: State of the art and research challenges. *Self*, 2:5, 2010.