Context-based modules deployment specification

Deliverable 5.2 (version 1)

Mehdi Terdjimi, Lionel Médini and Michael Mrissa

31 December 2015

Project ASAWoO

Adaptive Supervision of Avatar / Object Links for the Web of Objects

Grant Agreement: ANR-13-INFR-0012-04






Abstract

Web of Things applications require advanced solutions to provide adaptation to different purposes from common context models. While such models are application-specific, the adaptation itself is based on questions (i.e. concerns) that are orthogonal to application domains. In this deliverable, we present a generic solution to provide reusable and multi-purpose context-based adaptation for smart environments. We rely on semantic technologies and reason about contextual information to evaluate, at runtime, the pertinence of each adaptation possibility to adaptation questions covering various concerns. We evaluate our solution against a smart agriculture scenario using the ASAWoO platform, and discuss how to design context models and rules from "classical" information sources (e.g. domain experts, device QoS, user preferences).

Contents

1 Introduction

The Web of Things (WoT) relies on Web technologies and standards to build applications that make heterogeneous objects (i.e. things) interoperate in diverse situations. As WoT applications cover numerous domains such as healthcare, smart cities, smart factories and smart agriculture, their development has been substantially gaining interest in the past few years.
In the ASAWoO project, we have defined a virtual extension for a thing called *avatar*. Just like a servient [1], an avatar allows access to and control over a thing. An avatar is a component-based software artifact that relies on a semantic architecture to process and reason about semantically-annotated information. This way, in the ASAWoO platform, avatars can communicate with one another and expose high-level *functionalities* to form *WoT applications* involving several things that achieve a common goal. A WoT application consists of a hierarchy of functionalities: composed functionalities may require several devices to communicate and terminal ones are implemented using device *capabilities* (sensors, actuators, processing units). WoT application execution relies on several research fields [9], among which semantic web services and multi-agent systems. The functionality composition approach is not detailed here, but is precised in [11].
Avatars must also adapt their behaviors to comply with non-functional concerns such as quality of service (QoS), energy efficiency and security, with respect to several factors like environmental and natural conditions, computing resources and user preferences. To do so, they embed a generic, multi-purpose adaptation process designed to answer *adaptation questions*. Due to the set of concerns addressed in the ASAWoO project, we herein target the following adaptation questions [15]:

Q_1 *Which protocols should the application use to communicate with things?*
Q_2 *Where should the application code be executed?*
Q_3 *Which thing capability should be involved in a given terminal functionality?*
Q_4 *Which functionality should be involved in a given composed functionality?*
Q_5 *Which functionality should be exposed to clients and other avatars?*[1]

They may relate to concerns that are independent of the application domain, so that any application can be deployed on the ASAWoO WoT platform. And so must be the hardware and software architecture that provides adaptation facilities to the avatars. However, the actual data about sensors, domain concepts or users' preferences are application-dependent. As adaptation questions must be answered using these data, these questions need to be related to the application domain and actors.
In this deliverable, we provide a multi-purpose context adaptation process, able for any WoT application to simultaneously answer domain-independent adaptation questions, including modules deployment, with data originating from application-specific sources. To do so, we turn data into semantically-explicit contextual information, transform these information into *context model instances*, and infer all *adaptation possibilities*, along with their pertinence values. We detail the different types of rules involved in the adaptation process, and show how reifying RDF sets of triples that represent adaptation possibilities allows keeping the genericity of adaptation questions and preserving the domain-agnostic nature of avatar components.
Section 2 presents a smart agriculture scenario that illustrates our approach. Section 3 details our adaptation and question answering processes. Section 4 presents our implementation and evaluates it in terms of accuracy and performance. Section 5 presents related work in context description and management. Section 6 discusses our results and compares our solution to the state of the art. Section 7 provides a summary of our results and insights for future work.

2 Smart agriculture scenario

We use the following scenario from the sustainable agriculture domain. The considered WoT application aims at watering a vineyard (aka *field*) according to its local water needs. The field is divided in several *parts*, each of which can be watered separately using an irrigation network. Even though each object has its own avatar, this scenario only focuses on *drones* that are in charge of taking photos of the field parts to detect which parts lack water. The application takes into account current weather conditions and forecast,

[1] Functionalities must be exposed by avatars prior to be called by clients and executed.

in order not to water if precipitations are expected and to only provide each field part with the necessary quantity of water otherwise.
The application is designed as a hierarchy of functionalities, on top of which *ManageFieldWatering*, which is composed of several *ManagePartOfFieldWatering* functionalities, each of which requires *DetectWateringNeeds*, which implies *TakePicture*, *ProcessPicture*, *TransferPicture*, etc. Drones can collaborate to achieve a composed functionality such as DetectWateringNeeds. In this case, a drone "takes the lead" and selects its most appropriate collaborators to take pictures. In the same way, drones can serve as network relays to transmit information between drones over the field and a gateway connected to the cloud. Complex processing task can also be executed on the platform cloud.
The application is deployed in a WoT platform that comprises a cloud infrastructure and several wireless gateways around the field. The field is equiped with an anemometer and a pluviometer to sense actual weather conditions. However, it is not covered with a wide wireless network. 3 Drones can be used. They are equiped with a GPS sensor, a camera that can take photos of different qualities, two wireless network interfaces (Wifi and Bluetooth), and can sense their own current capabilities (battery, memory availability, CPU usage, storage space). Lastly, the application is connected to a weather forecast service.
As there are multiple ways to achieve these functionalities, the WoT application must adapt its behavior. To precise our scenario, we consider an example of situation in which an avatar has to answer the five adaptation questions described in Section 1. We hereafter instantiate each adaptation question and precise the elements that would allow a human operator to answer it in this situation.

Q_1 Chosing a network interface to transfer a picture depends on the distance between the drones and on the energy consumption of available interfaces.
Conditions: The drone has to transmit data to another drone located at 11 meters and has 45% of battery left.
Deduction: The distance is too high for using Bluetooth and the battery level is sufficient for both.
Decision: Use Wifi to perform TransferPicture.

Q_2 The application module that processes pictures to determine water needs may be executed either on the drone or on the cloud. It requires high CPU availability and a minimum battery level. In addition, executing it on the cloud requires high bandwidth to transfer the picture in acceptable time.
Conditions: CPU availability is 80% and battery 45%.
Deduction: The CPU level is sufficient to do the processing on the drone and the battery level is sufficient for both.
Decision: Execute the ProcessPicture code on the drone.

Q_3 Taking HD pictures is preferable when implementing the functionality DetectWateringNeeds. However, this requires a high-resolution camera and sufficient storage capacity.
Conditions: The drone has the capability to take HD pictures and 2.5Gb of free internal storage.
Deduction: The picture can be taken and stored in high definition.
Decision: Use the TakeHdPicture capability to realize the TakePicture functionality.

Q_4 Choosing the right drone to identify if part of field #1 needs to be watered depends on the remaining battery power and on distance from the part of the field of each drone.
Conditions: Drone1 (resp. drone2, resp. drone3) is at 120 (resp. 20, resp. 70) meters from part of field #1, and has 90 (resp. 45, resp. 70)% battery left.
Deduction: All drones can fulfill the functionality, but drone 2 is closer.
Decision: Choose drone 2 to perform DetetWateringNeeds on field part 1.

Q_5 Drones may deteriorate if they are exposed to strong wind or to the rain. They should not be able to go outside if the weather is inconvenient.
Conditions: No rain, but a 55km/h wind.
Deduction: Drones 1, 2 and 3 have no risk to be deteriorated by rain, but have significant risk to be damaged by wind.
Decision: Do not expose the GoOutside functionality on drones 1, 2 and 3. Consequently, do not expose the TakePicture and DetectWaterNeeds either.

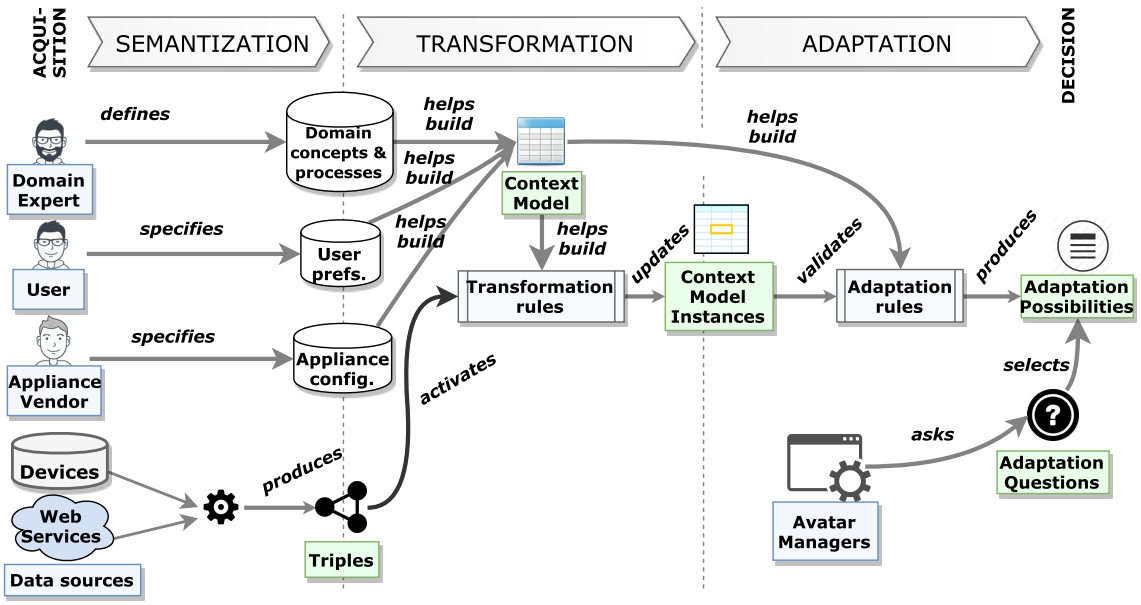


Figure 1: The adaptation process and its actors.

3 Multi-purpose adaptation in WoT applications

Multi-purpose context adaptation in WoT applications originates from several processes in which various actors take part. WoT platform designers create complex execution environments that need to handle several concerns to support a variety of use cases and applications; they document these concerns and the corresponding adaptation questions. Appliance manufacturers describe device characteristics (QoS) in their documentations. Domain experts identify application concepts and processes, along with all the environmental sensors and Web services able to provide useful contextual data. Users specify their preferences (e.g. working hours, prefered devices, privacy levels, etc.).
WoT application designers then need to interpret all those instances of contextual dimension and to integrate them in a comprehensive adaptation process. Their work consists in designing a context model and two sets of rules: *transformation* and *adaptation*. Then, they wire the model to the available data sources. The ASAWoO platform does the rest. At configuration time, static data (e.g. application context model, appliance configuration, user preferences) are stored in semantic repositories [11]. At runtime, an avatar receives raw data from various sources, including devices and Web services. These data are semantically annotated and transformed into instances of the context model, using transformation rules. Adaptation rules are then applied to the instantiated context model to infer each possible adaptation choice. When avatar components require adaptation decisions, they send adaptation questions to the avatar *context manager*, which retrieves the best candidate. The querying process is the same, regardless of whether the question relies on filtering (e.g. can I expose a given functionality) or on ranking (e.g. which communication protocol is the most suitable). The whole process is based on the management workflow described in [15], depicted in Figure 1. In this section, we focus on the following activities: identifying semantic instances of the context model, building transformation and adaptation rules, and solving adaptation questions. We apply these processes on our scenario, which involves three drones in the ManageFieldWatering functionality.

3.1 Context model instantiation

We use hereafter the following formalization.

- The context model M is structured according to contextual *dimensions* D_j [15], that correspond to the conditions described in the example in Section 2. $M = \sqcup D_j, j \in \{1,...,n\}$, where n is the number of dimensions.
- A contextual dimension D is a set of possible contextual *instances* i.

$D_j = \{i_{jk}, k = 1, ..., Card(D_j)\}$, where $Card(D_j)$ depends on the transformation rules defined in Section 3.2

- An instantiated context model is called a contextual *situation* ς. It denotes a semantized observation of all available contextual data at a given instant t, at which each dimension can be valued or empty.
  $\varsigma_t = \{i_{tj}\}, i_{tj} \in D_j \cup \varnothing, j \in \{1, ..., n\}$

In our scenario, we identify the sets of contextual instances in our WoT infrastructure below:

$D_{wind} = \{NoWind, Breeze, StrongWindForGoOutside\}$
$D_{rain} = \{NoRain, Rainy\}$
$D_{distance} = \{CloseToFieldPart1, CloseToFieldPart2, CloseToFieldPart3,$
$FarFromFieldPart1, FarFromFieldPart2, FarFromFieldPart3\}$
$D_{bandwidth} = \{HighBandwidthForTransferPicture,$
$LowBandwidthForTransferPicture\}$
$D_{storage} = \{HighCapacityForStorePicture, LowCapacityForStorePicture\}$
$D_{battery} = \{HighBatteryForMoving, LowBatteryForMoving,$
$HighBatteryForTakePicture, LowBatteryForTakePicture,$
$HighBatteryForTransferPicture, LowBatteryForTransferPicture,$
$HighBatteryForProcessPicture, LowBatteryForProcessPicture\}$
$D_{CPU} = \{HighCPUAvailabilityForProcessPicture,$
$LowCPUAvailabilityForProcessPicture\}$
$D_{protocols} = \{Wifi, Bluetooth\}$
$D_{resolution} = \{LowQuality, AverageQuality, HighQuality\}$

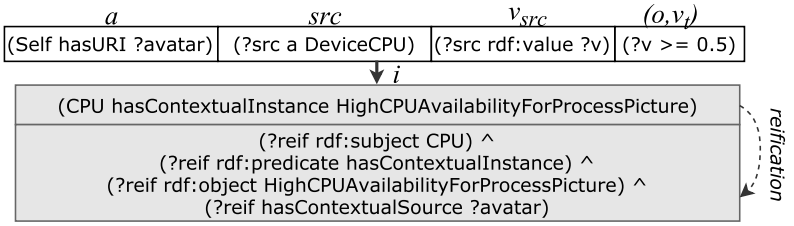
3.2 Transformation rules

Contextual Information	Data Source	Raw Value Type	Contextual Instance Thresholds
Wind	Anemometer	Integer (wind speed in km/h)	NoWind: x=0 Breeze: 20<x<50 StrongWindForGoOutside: x>=50
Rain	Web service, Pluviometer	Integer (precipitation rate in mm/s)	NoRain: x=0 Rainy: x>0
Distance	GPS	Integer (distance from a part of the field)	CloseFromField*X*: x <= 100 FarFromField*X*: x > 100
Bandwidth	Cloud Infrastructure	Integer (ping in milliseconds)	HighBandwidth ForTransferPicture: x<100 LowBandwidth ForTransferPicture: x>=100
Storage	(self) memory	Decimal (% of remaining capacity)	LowCapacity ForStorePicture: x<0.5 HighCapacity ForStorePicture: x>=0.5
Battery	(self) battery	Decimal (% of remaining power)	LowBatteryForMoving: x<0.6 LowBatteryFor(*others*): x<0.5 HighBatteryForMoving: x>=0.6 HighBatteryFor(*others*): x>=0.5
CPU	(self) CPU	Decimal (% of CPU load)	LowCPUAvailability ForProcessPicture: x<0.5 HighCPUAvailability ForProcessPicture: x>=0.5
Protocols	(self) network interfaces	Boolean (presence of the interface)	(Bluetooth/Wifi) x=1
Resolution	(self) camera	Integer (number of horizontal lines)	LowQuality: x<240 AverageQuality: 240<x<720 HighQuality: x>=720

Figure 2: The relation between contextual dimensions, data sources, raw value types and contextual instance thresholds.

Transformation rules aim at pre-processing continuous data and transforming them into discrete contextual instances, to facilitate the adaptation step. To do so, a transformation rule applies a condition on seman-

tically annotated data from a given data source. Let *src* be the data source, v_{src} its value, d a contextual dimension indicating the type of the data source, i a contextual instance, and a the avatar on which this transformation process is performed. The condition is based on a set T of contextualization thresholds $t = (o, v_t)$ with $o \in \{>, <, >=, <=\}$, $v_t \in \mathbb{R}$. Figure 2 depicts these elements for our smart agriculture scenario. A transformation rule is formed as follows:
$src \wedge v_{src} \wedge T \rightarrow [d$ *:hasContextualInstance* $i]$ *:hasContextualSource* a
The inferred graph is composed of two triples. The subject of the first one is the URI of the contextual dimension, its predicate *hasContextualInstance* and its object the URI of the contextual instance. The second one has the first as subject (by applying *triple reification* [8]), *hasContextualSource* as predicate and the avatar URI as object. An example of transformation rule is the following:



3.3 Adaptation rules

Adaptation rules infer *adaptation possibilities* from contextual instances. An adaptation possibility is a triple that is a potential answer to an adaptation question. The head of an adaptation rule is a conjunction of reified graphs $G_i, i \in \mathbb{N}$ produced using transformation rules. Its body is a new reified graph with three triples, stating the adaptation candidate triple p, the source avatar a of this candidate triple, and its contextual ranking value r. Thus, an adaptation rule has the form: $\wedge G_i \rightarrow p$, p *:hasContextualSource* a, p *rdf:value* r
The form of an adaptation possibility triple varies according to adaptation questions. Its subject is the URI of either a preferred protocol (Q_1), a code location (Q_2), a local capability (Q_3), another avatar functionality (Q_4) or a local functionality (Q_5). Its predicate is the URI of the adaptation question itself, and the object is the URI of the functionality to adapt. The ranking value determines the accuracy of a candidate in the current contextual situation. An example of adaptation rule is the following:

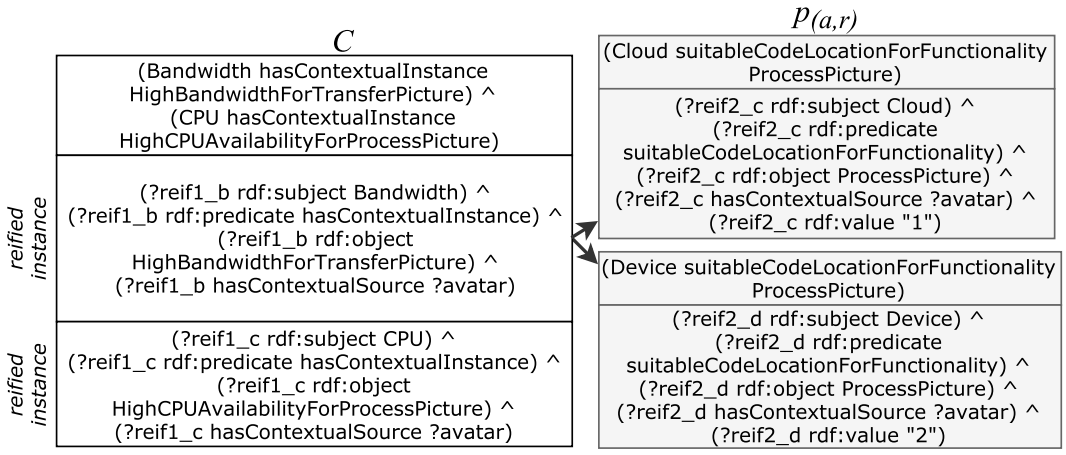


Figure 3 depicts the reifications in the semantized contextual data both on the transformation scope (i.e. ready-to-provide adaptation possibilities) and on the adaptation scope (i.e. ready-to-provide adaptation answers).

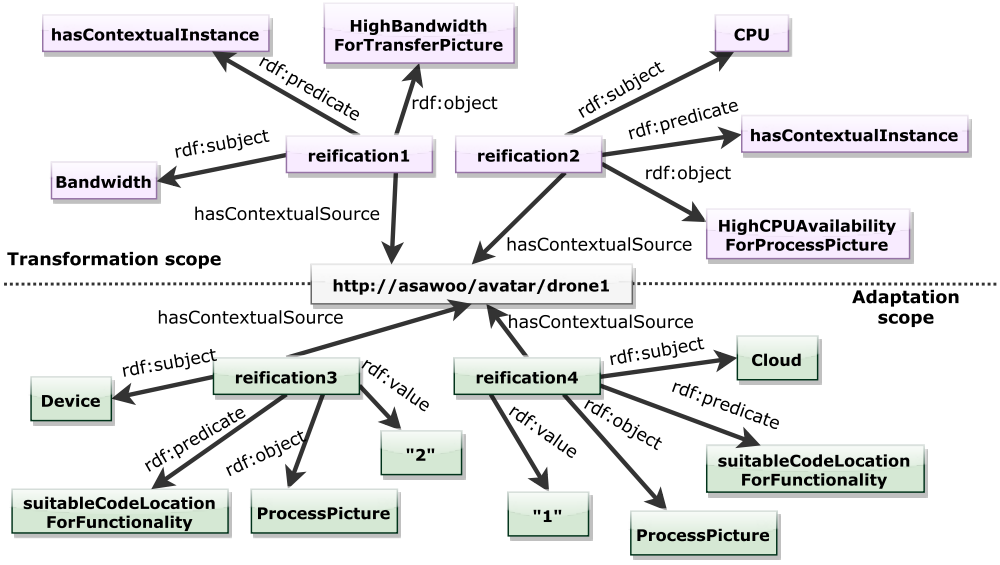


Figure 3: Linked graph of reified results of transformation and adaptation rules.

3.4 Adaptation question answering

Answering an adaptation question aims at retrieving the subject of the best valued adaptation possibility triple (along with its avatar), for a given adaptation question and regarding a given functionality. The context manager converts such questions into SPARQL SELECT queries with two variables: the adaptation question answer `?answer` and the contextual source `?ctxSrc`. The adaptation decision either relies on filtering (e.g. the functionality can be exposed or not as in [10]) or on ranking (e.g. select the appropriate features as in [4]). While Q_5 acts as a filter, answering Q_1, Q_2, Q_3 and Q_4 require adaptation possibilities to be ranked. For ranking-based adaptation questions, the SPARQL query includes an ORDER BY clause depending on the rank value `?rank`, along with a LIMIT 1 clause to only return the optimal answer. An example of adaptation question that determines the suitable location to execute ProcessPicture is the following:

```
SELECT ?answer ?ctxSrc { ?candidate rdf:subject ?answer .
?candidate rdf:predicate <suitableCodeLocationForFunctionality> .
?candidate rdf:object <ProcessPicture> .
?candidate <hasContextualSource> ?ctxSrc .
?candidate rdf:value ?rank } ORDER BY ?rank LIMIT 1
```

4 Evaluation

We base our evaluation prototype on the ASAWoO platform described in [9]. This platform provides a WoT runtime environment that instantiates and runs one avatar for each connected device, and provides facilities to store application and contextual data in semantic repositories. Avatars are implemented in Java and their components are OSGi bundles. Each avatar exposes its device functionalities and context data to the other platform elements using Web standards. Inside an avatar, components ask adaptation questions at runtime to the context manager, which delegates answering to a local semantic reasoner by interpreting them as SPARQL queries. At the same time, the context manager constantly acquires raw data from various sources and transmits them to the semantic reasoner as described in Section 3.

As, to the best of our knowledge, there exists no similar work that led to the design and implementation of a domain-agnostic, semantic-based and multi-purpose adaptation platform for the Web of Things, we could not compare our work to the state of the art. We then chose to evaluate our prototype according to two criteria based on the work of Bass et al [2]. First, we evaluate its accuracy, i.e. its ability to do the work

for which it was intended. Second, we evaluate its performance for both the data integration and query answering tasks. All experiments were performed using the HyLAR semantic reasoner [14].

4.1 Qualitative evaluation

We evaluate the accuracy of our solution by simulating the scenario from Section 2 by connecting three drones to our WoT platform, with the objective of realizing the ManageWatering functionality. For the sake of simplicity, we focused on the tasks that only required drones and did not consider other appliances such as the automatic irrigation system. The experimental setup was the following: our platform is the ASAWoO platform and runs in an Ubuntu 16.04 VM with 2 VCPUs and 4Gb of RAM, in an OpenStack cloud infrastructure. We vary the contextual parameters identified in Section 3 in a 2-days time interval. In this interval, we ask our five adaptation questions to each drone avatars, at different times $t1$, $t2$, $t3$ and $t4$, as depicted in Figure 4. We expect the answers to these questions to correspond to the adaptation rules described in 3.3.

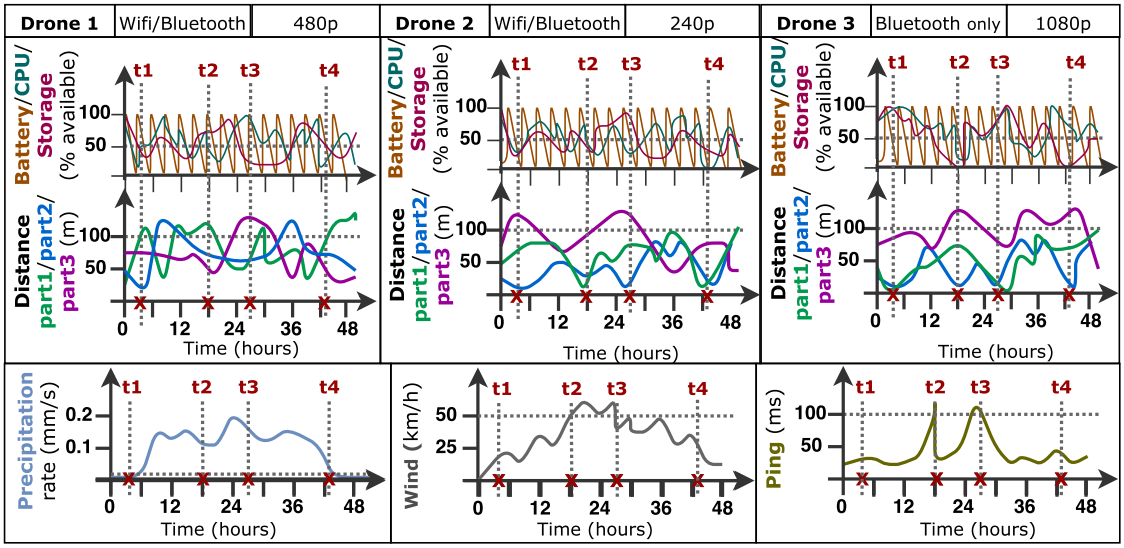


Figure 4: Variation of contextual data during a simulated 2-days time interval.

Figure 5 shows the answers returned to the adaptation questions by the five managers. At $t2$, we see that the optimal protocol for the TransferPicture functionality between drone 1 and drone 2 is Wifi as they are in different parts of the field. At $t2$ and $t3$, executing ProcessPicture in devices is preferable due to ping timeouts or long delays caused by the weather conditions. At $t2$, drone 1 capabilities are preferred as drone 3 capacity is too low. We also see that drone 2 is not a good choice for detecting parts of field to water due to its insufficient resolution. At $t1$ and $t3$, drone 3 is the optimal choice to take pictures due to its proximity to part-of-field 1. However, drone 1 is preferred at $t2$ due to its high battery level. At $t2$ and $t3$, the functionality GoOutside is not exposed as either the wind is too strong or it rains.
In all cases, we verified that all avatar context managers provided the expected answers to all adaptation questions. The correctness of our system is enforced by the use of a standard semantic reasoner, which ensures that any other rule-based solution would have given the same answers. In addition, this evaluation shows that our solution allows performing accurate multi-purpose adaptation from a common semantic contextual model.

4.2 Quantitative evaluation

In this section, we evaluate the performance of our prototype in terms of processing times. These experiments were performed on a Dell OptiPlex 780 - Core 2 Duo E8400 @ 3 GHz. As the integration (semanti-

Time	*Q1* – Protocol for TransferPicture (Drones 1 & 2)	*Q2* – Location of ProcessPicture	*Q3* – Implementation of DetectWateringNeeds	*Q4* – Composition with TakePicture (FieldPart1)	*Q5* – Exposability of GoOutside
t1	Bluetooth	Cloud	Drone 3	Drone 3	Exposable
t2	Wifi	Device	Drone 1	Drone 2	Not exposable
t3	Bluetooth	Device	Drone 3	Drone 3	Not exposable
t4	Bluetooth	Cloud	Drone 3	Drone 1	Exposable

Figure 5: Answers to five adaptation questions in four different times.

zation, transformation and application of adaptation rules) and adaptation question answering processes can run in parallel, we evaluate them in separate runs, and with different goals (i.e. maximum processing times). Contextual data integration runs as a background task, but must not monopolize all computing capabilities allocated to the avatar, especially if this avatar runs on a constraint device. Hence, it must be lightweight but does not need to be immediate. We then chose a threshold of 1 second as success for this experiment[2]. Adaptation question answering, however, is time-critical, as it is required for the current functioning of the avatar and of the WoT application. We then limit its acceptable response time to 100ms.
For both processes, we ran two experiments varying the number of rules and of triples in the knowledge base. The results of these experiments are depicted in Figure 6. They show that we reach by far our two initial goals as the respective processing times for integration and answering do not exceed 650ms and 30ms.

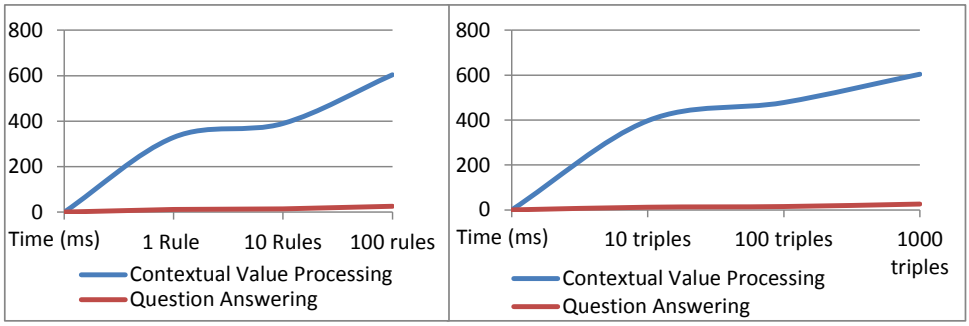


Figure 6: Context processing and question answering times on different situtations.

5 Related work

This section overviews related work on context identification, as well as on context management lifecycles in pervasive environments.
The definition of context depends on the domain concepts, the actors of the tasks, the application's infrastructure, etc. In [5], Brézillon describes the *focus* as the identification of context elements that are relevant for the current task. He separates the contextual knowledge (related to the current focus) and the external knowledge (which is not relevant to the current focus of the task operator). In [6], he identifies and represents the type of knowledge using expert maps, where the expert is the task operator. This differs from our work, where the task operator is actually the application user. Brézillon's context representation varies across operators and allows flexibility in terms of users, while our avatar-centered vision of context provides reusability of the context model amongst each object of the infrastructure and allows capitalizing on the reasoning process. In [13], Schaap et al. propose four steps that rely on stakeholders expertise and existing data about the domain. Their solution is based on specific climate risks that are linked to the adap-

[2]One second is the commonly admitted threshold upon which the user's attention stops focusing on the current task.

tation process. This specificity and the lack of context representation both make their solution difficult to reuse in general cases. We instead provide a loose coupling between adaptation possibilities and contextual information, allowing applications from any domain to fit our infrastructure.
The management of contextual information also varies across context-aware adaptation solutions. Based on the literature, Perera et al. propose in [12] a context lifecycle in four steps: context acquisition, context modeling, context reasoning, and context dissemination to other devices. The context modeling step implies structuring the context as part of the lifecycle. Our solution separates context modeling from context instantiation, as the model is fixed throughout the lifecycle of the application, while the instances vary according to a given contextual situation. The workflow proposed by Bernardos et al. [3] (acquisition, processing, reasoning and decision) separates context processing from reasoning. We instead process reasoning tasks simultaneously at context instantiation time and at adaptation possibility generation time. Thus, we provide straightforward decision-making through generic adaptation questions. Ferscha et al. propose in [7] the sensing, transformation, representation, rule-base and actuation steps. The context changes trigger this last step and propagates information to several actuators, which can be problematic if several context managers send the same information to the same actuators. We instead provide ready-to-answer adaptation possibilities that are queried only when needed.

6 Discussion on WoT application design

This section discusses the role of a WoT Application Designer (WAD) regarding the definition of the contextual management cycle in an application. It complements Section 3, which deals with automated aspects of the management process, and defines the knowledge required for a WAD to bridge functional and non-functional concerns. It provides guidelines to link "classic" sources (i.e. appliance documentation, domain expert knowledge, user preferences, etc.) with other data sources (devices, Web services) to realize the adaptation.
The choice of contextual dimensions to build the context model requires knowledge of the main concepts and processes of the domain. At this stage, the WAD asks the domain expert which factors may change the behavior of the application (environmental occurrences, defects, etc.) to provide the appropriate solutions. The WAD then interprets these factors as contextual instances to build the context model. When designing transformation rules, the WAD confronts the expert's assessments to the appliance's technical constraints. The context meta-model we have presented in [15] can validate the choice of contextual dimensions and instances to ensure they fit the infrastructure needs. Designing transformation rules requires expressing thresholds using comparison operators and associate data sources to their value using two triple patterns, where the first pattern describes the carried value and the second pattern describes the nature of the data source. The WAD also identifies possible user preferences as triple patterns that link data sources to preferential characteristics, such as service fee, type of technology, brand, etc. Designing adaptation rules consists in interpreting expert answers as adaptation possibilities for a given contextual situation. Finally, the WAD determines the rank values, based on device properties and quality-of-service agreements.
As we aim to ease joint work between WADs and domain experts, the design of the adaptation solution itself can be adapted all along the application use. In addition to easing the WAD's work by mutualising contextual data collection for various adaptation purposes (which would anyway have been collected and processed separately otherwise), our adaptation solution design cycle is iterative and incremental, similarly to agile methods used in software development. It supports changes in domain knowledge, as well as in appliances and actors' description through the use of generic semantic methods. For instance, in our scenario, if the user buys new drones with different characteristics (in terms of battery, storage or computing capabilities), the platform will seamlessly integrate them and adapt the application to these new devices, to the cost of a simple configuration task.

7 Conclusion

We propose an adaptation solution for Web of Things platforms that simultaneously supports several adaptation concerns, ensures compatibility across devices, applications and domains, and relies on extensible

sets of context dimensions. This solution is based on semantic reasoning and limits the reasoning extent to a closed world determined by semantization thresholds. After presenting the general framework of our proposition (i.e. the ASAWoO platform), we focus on the definition of the following elements: set of context model instances, transformation rules that turn semantized sensor data into actual contextual model instances, adaptation rules that infer adaptation possibilities from contextual situations, and multi-purpose adaptation questions answering to make adaptation decisions. We evaluate our solution in a smart agriculture scenario both in terms of accuracy and performance. Results validate adaptation answers and reach performance goals on context processing and adaptation question answering. We compare our approach to the literature regarding context identification and management. Finally, we describe the role of WoT application designers in the collection of contextual data and specification of application context management workflows, and open a discussion on the design of adaptive solutions that can in turn diachronically evolve while managing dynamic data that synchronically change.
Our perspectives include the automatic generation of domain-specific transformation and adaptation rules through meta-rules. We also aim at formalizing a method to rank adaptation possibilities using discretization techniques.

References

[1] Web of Things Architecture, Unofficial Draft: General Description of WoT Servient. https://w3c.github.io/wot/architecture/wot-architecture.html#general-description-of-wot-servient, 09 September 2016.

[2] Len Bass. *Software architecture in practice*. Pearson Education India, 2007.

[3] Ana M Bernardos, Paula Tarrio, and Jose R Casar. A data fusion framework for context-aware mobile services. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 606–613. IEEE, 2008.

[4] Arnaud Blouin, Brice Morin, Olivier Beaudoux, Grégory Nain, Patrick Albers, and Jean-Marc Jézéquel. Combining aspect-oriented modeling with property-based reasoning to improve user interface adaptation. In *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems*, pages 85–94. ACM, 2011.

[5] Patrick Brézillon. Task-realization models in contextual graphs. In *International and Interdisciplinary Conference on Modeling and Using Context*, pages 55–68. Springer, 2005.

[6] Patrick Brézillon. Modeling expert knowledge and reasoning in context. In *International and Interdisciplinary Conference on Modeling and Using Context*, pages 18–31. Springer, 2015.

[7] Alois Ferscha, Simon Vogl, and Wolfgang Beer. Context sensing, aggregation, representation and exploitation in wireless networks. *Scalable Computing: Practice and Experience*, 6(2), 2001.

[8] Patrick Hayes and Brian McBride. Rdf semantics. *W3C recommendation*, 10, 2004.

[9] Lionel Médini, Michael Mrissa, Mehdi Terdjimi, El-Mehdi Khalfi, Nicolas Le Sommer, Philippe Capdepuy, Jean-Paul Jamont, Michel Occello, and Lionel Touseau. Building a Web of Things with Avatars. In Lina Yao Michael Sheng, Yongrui Qin and Boualem Benatallah, editors, *Managing the Web of Things: Linking the Real World to the Web*. Morgan Kaufmann, Elsevier, October 2016. Domains (unavailable categories): Internet of Things, Web of Things.

[10] Rabeb Mizouni, Mohammad Abu Matar, Zaid Al Mahmoud, Salwa Alzahmi, and Aziz Salah. A framework for context-aware self-adaptive mobile applications spl. *Expert Systems with applications*, 41(16):7549–7564, 2014.

[11] 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.

[12] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014.

[13] Ben F Schaap, Pytrik Reidsma, Jan Verhagen, Joost Wolf, and Martin K van Ittersum. Participatory design of farm level adaptation to climate risks in an arable region in the netherlands. *European Journal of Agronomy*, 48:30–42, 2013.

[14] Mehdi Terdjimi, Lionel Médini, and Michael Mrissa. HyLAR+: Improving Hybrid Location-Agnostic Reasoning with Incremental Rule-based Update. WWW '16: 25th International World Wide Web Conference Companion, April 2016.

[15] Mehdi Terdjimi, Lionel Médini, Michael Mrissa, and Nicolas Le Sommer. An avatar-based adaptation workflow for the web of things. In *2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 62–67. IEEE, 2016.