Research online Publishing
www.ronpub.com

Open Access

Open Journal of Internet of Things (OJIOT)
Volume X, Issue X, 20XX
http://www.ronpub.com/ojiot
ISSN 2364-7108

Controlled Components for Internet of Things as-a-service

Aubonnet Tatiana 1 [A], Boubendir Amina 2 [B], Lemoine Frédéric 3 [A], Simoni Noëmie 4 [A]

[A] CEDRIC, Conservatoire National des Arts et Mtiers, 292 rue Saint-Martin, 75003 Paris, France, {tatiana.aubonnet, frederic.lemoine}@cnam.fr
[B] Télécom ParisTech, 46 rue Barrault, 75013 Paris, {amina.boubendir, noemie.simoni}@telecom-paristech.fr

Abstract

In order to facilitate developers willing to create the future Internet of Things (IoT) services incorporating the non-functional aspects, we introduce an approach and an environment based on controlled components. Our approach allows developers to design an IoT "as-a-service", to build the service composition and to manage it. This is important, because the IoT allows us to observe and understand the real world in order to have decision-making information to act on reality. It is important to be sure that all these components work in accordance with their mission, ie their Quality of Service (QoS) contract. Our environment provides the modelling, generates Architecture Description Language (ADL) formats, and uses them in the implementation phase on an open-source platform.

Type of Paper and Keywords

Regular research paper: *controlled IoT service, quality of service, management.*

1 Introduction

The world of Internet of Things (IoT), Smart Object, Smart Connected products and Cyber Physical System introduces devices of all kinds which, because of their ability to "observe" the physical world, and to "provide" decision-making information, should be part of the architecture of the future Internet [8], [34]. The questions that arise are the following: How can they be integrated in this all connected context? Can we have a homogeneous or standardised architecture?

Among the fundamental characteristics of IoT systems International Telecommunication Union (ITU-T) [12], we focus on the following characteristics: (i) things-related services, (ii) heterogeneity and (iii) interconnectivity.

(i) The IoT must provide things-related services taking into account the inherent requirements of these services. The architecture that is emerging is a service oriented architecture (SOA) in which a semantic consistency is needed between physical and virtual objects associated with them. So that such services can be provided in compliance with these requirements, used technologies will have to change.

(ii) The heterogeneity is located at several levels. There is in particular "the provided data" from very different fields. We need to understand and know the field in order to qualify the treatment. Then the devices themselves will be affected, since they do not use the same hardware platforms or the same network. We need to make sure that they are reliable and that they behave properly.

(iii) With regard to the IoT, any object can be connected to the infrastructure of information and communication. The devices must be managed.

To meet these new challenges for IoT, we need to rethink the services and ensure their behaviours. With our proposals for a cloud application to become a services

and micro services composition, and in order to reach maximum flexibility, to compose "as-a-service" with our connected objects, we propose in this paper:

(a) Service Oriented Architecture allowing the composition of IoT applications in order to integrate functional and non-functional (e.g. quality of service) aspects.

(b) Integration of IoT connectivity to its environment.

(c) Management closer to each IoT service during the operation phase.

The paper is organised as follows. We analyse related works in Section 2. We propose in section 3 an approach to compose IoT components "as-a-service" based on self-controlled components called "IoT Self-Controlled service Component" (IoTSCC). A platform for components based architecture design is then presented. Moreover, we show in Section 4, how to add security functionalities. In Section 5 we present a study case related to warehouse arrival management. Finally, a conclusion (Section 6) ends this paper.

2 RELATED WORKS

IoT may open new opportunities to create innovative applications. The Intelligent Network has introduced the concept of "Service Creation Environment" (SCE) [22]. This concept has been introduced with Intelligent Networks to ease and speed up the development and deployment of services [11]. An SCE is generally a graphical user interface for developing services using predefined components, also called building blocks (SIB). This approach has been a precursor in separating the service components and execution logic. The SIBs allows construction of telecommunication services as easy as possible. In the same way, the "as-a-service" in the IoT should allow a service composition flexibility.

IoT Service Platforms play a fundamental role for creating and managing IoT applications. It is crucial to hide the heterogeneity of hardware, software, data formats, technologies and communication characterising IoT [31]. It is responsible for abstracting all the features of objects, network, and services, and for offering a loose coupling of components. IoT platforms in [16], [19] focus on cloud computing architecture to meet the challenges of flexibility, extensibility and economic viability of IoT.

Some IoT platforms focus on the development of IoT architectures that ensure interoperability between vertical application solutions and different technologies. For example, the main goal of iCORE [15] and COMPOSE [20] is to develop an open network architecture based on objects virtualisation that encompasses the technological heterogeneity.

IBM BlueMix [7] is a platform "as-a-service" (PaaS) cloud, developed by IBM. It supports rapid development of analytic applications, visualisation dashboard, and mobile IoT applications. IBM secures the platform and infrastructure and provides you with the tools to secure your apps and connect your device data with it. IBM IoT foundation (IoTF) [21] is the hub where you can set up and manage your connected devices. A device, to be connected, will require a device management agent that is a collection of logic installed on a device that allows it to connect to the Cloud Internet of Things services as a managed device.

AWS IoT [2] is a platform that enables you to connect devices to AWS Services [1] and other devices, secure data and interactions, process and act upon device data, and enable applications to interact with devices even when they are offline. It provides secure, bidirectional communication between Internet-connected things (such as sensors, actuators, embedded devices, or smart appliances) and the Amazon Web Services (AWS) cloud. This enables you to collect telemetry data from multiple devices and store and analyse the data. The rules engine makes it possible to build IoT applications that gather, process, analyse and act on data generated by connected devices at global scale without having to manage any infrastructure.

Azure IoT Hub [9] is a fully managed service that enables reliable and secure bidirectional communications between millions of IoT devices and a solution back end. Azure IoT Hub can reliably receive process or store for the analysis of millions of events per second from devices and provides extensive monitoring for device connectivity and device identity management events.

The SPRINT project [14] provides a platform to connect the software tools used by the industrial companies within the project and allows integration of different subsystems at the design level. Other platforms as BUTLER [28] or MobilityFirst [35] aim to develop open architectures providing secure location and context-aware services.

All IoT platforms focus on the same problems such as homogenising and transforming an object so that it becomes a little smarter and can be managed by understanding the same device management commands, securing communications between devices or between devices to cloud services, obtaining diagnostic information, both for connectivity and for the devices themselves (rich device metadata, status information) and managing scaleability by sending/receiving bulk operations on/from many devices at a time.

These platforms provide undeniable help for quick implementation, but all, have they not implicit management aspects always adapted to a particular context? Are they not the result of a compromise? The architect has to be allowed to select, customise and adapt his solutions

according to behavioural progress. Theoretical models of the IoT architecture and the definition of an initial set of key building blocks are indeed key objectives of [18] and IoT-A [37], respectively. The works [13], [17] focuses on business services and on the development of SOA-based architectures and dynamic environments to semantically integrate services into IoT but without offering a loose coupling of components allowing re-composition according to behavioural changes during the session.

Furthermore, observance and integration of Quality of Service (QoS) is mandatory for IoT services and applications. The solutions provided by these platforms, as well as the standardisation works of the ITU-T, ETSI oneM2M [5][3][4][6][12] is not sufficient and not complete, in our view, for the IoT and Cloud Computing. Indeed, service components (”as-a-service”) in this IoT/Cloud environment must offer to the user, who will select them, not only a feature but well-defined behaviour (QoS) too. This led us to propose a control allowing a component to monitor his compliance with the contract.

We note that in the mobile context there is a missing feature, the continuity of service. Our motivation is to design, manage and control components, reacting throughout the life cycle and during running time. That is why we present our approach of IoT ”as-a-service” composition and control mechanisms to satisfy the continuity of service and the compliance with the contract.

3 Propositions for an as-a-service IoT design

As IoT is about smart objects [33] being first sensed then controlled and managed remotely across network infrastructure, there is a need to go towards an effective, structured and efficient realisation of such a definition. For that, we propose that IoT devices be introduced in the ”as-a-service” ecosystem of the Cloud. This is a major direction to acquire a significant role meeting the need for remote control and management. We present in this Section the approach to achieve this. We first describe the advocated approach step-by-step and highlight the features introduced at each step of the transformation approach of the smart object into a controllable IoT service component (Section 3.1). Then, we define more precisely the proposed solutions within the architectural dimension (Section 3.2), the organisational dimension (Section 3.3) and the functional dimension (Section 3.4).

3.1 From smart object to IoT service

In order to make a software component compliant with the IoT services world, we propose an approach that allows it to cover progressively the properties required for this transformation.

Our approach involves six steps.

Step 1: To structure

In an ecosystem where a service is available through a network, we need to distinguish, and thus structure, the service according to three planes: the user plane representing the offered functionality, the control plane representing the automation and policies serving the user plane, and the management plane that allows the coherence of the global system. In a fractal approach [27], a smart object is represented as a business component, which is the functional aspect of the smart object, with its usage interface as shown in Figure 1.

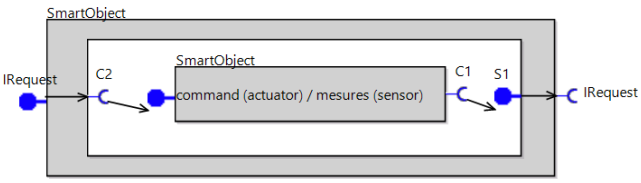

Figure 1: Representation of a Smart Object

The smart object needs to have control and management interfaces. That is why we propose to adopt the Grid Component Model (GCM) [25][30]. The resulting smart object structure at this step will be transformed to become a component including a management membrane with two interfaces, one of control and another one of management. Thus, we have the possibility of dialogue with the three planes.

Step 2: To integrate

To integrate the smart object into an IoT context, we propose to add, in the membrane, a management component named ”IoT processing” with management interfaces to allow the smart object to be invoked and managed with respect to an IoT profile.

We detail the ”IoT processing” component in this paper and define the micro-services that compose it. Thus, the smart object can be represented as an IoT service component as shown in Figure 2 and is integrated in its IoT environment with:

- Functional content (business) and external client and server interfaces for the user plane.
- A membrane for the non-functional aspects, with management and control interfaces to be connected and to communicate within the IoT environment (with other objects for example).

Step 3: To self-control

For the control aspect, we propose to embed a QoS agent in order to introduce the needed autonomic aspect

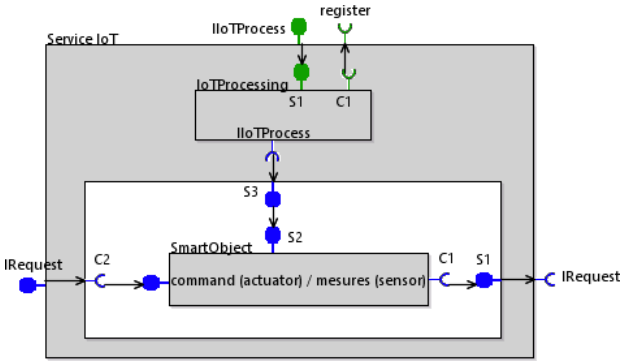

Figure 2: Representation of an IoT service

in an environment that is meant to scale and is subject to become rapidly more complex.

We have defined this auto-control for components in Cloud services [23] and we propose to extend it to IoT services. The aim of introducing autonomic control of components is to enforce monitoring mechanisms that collect information concerning the behaviour of the component in order to control the respect of Service level agreement (SLA) and QoS level and react in case of non-compliance.

Thus, at this stage, the IoT service becomes a Self-Controlled IoT service Component, we call it IoT SCC as we detail later. We rely on a recursive service architecture, where a service may comprise a set of self-controlled service components or micro-services. So the IoT SCC can be integrated within global self-controlled service architecture.

Step 4: To design as-a-service

This step aims to make sure that the offered service component can be added, removed or composed with other services, without crashing the whole organisation, i.e., the global service architecture.

The ”as-a-service” design is meant to allow the customisation, flexibility in composing service offers, adaptability of the offered services or solutions as well as an on-the-fly deployment. For this purpose, a set of properties need to be verified for designing an IoT SCC as an IoT SCC as-a-service. The elementary units constructed as the IoT service need, to rely on the following main properties of SOA and Cloud services: statelessness, autonomy and loose coupling.

Statelessness means that the service performs the same processing to all requests without keeping any information about their data or their contexts. This allows a service to always offer the same function to all clients/requests. The service component should have one type of interface. And for the service to be stateless, its operations should be conceived to perform the processing without depending on information received during a precedent invocation.

Autonomy means that a service is able to achieve its functionalities without needing another service or human intervention. In this direction, we propose to conceive a service as a black box composed of a set of operations executed in the same manner and in the same order for all requests.

Loose Coupling means that the bindings or links between service components in a service composition are unattached or even rigid, to eliminate all types of functional coupling between services. Thus, loose coupling ensures a flexible composition of service components. Service composition consists of generating a global service by composing or chaining a set of elementary service components. This composition would thus be customisable and flexible by adding, replacing, and removing service elements according to users needs.

In addition, for software engineering needs, the properties of *reusing* and *mutualisation* are strongly recommended in this approach.

Reuse is needed to simplify the software development of services that meets the new needs: IoT SCC service components would be reusable thanks to the generic character of their interfaces (usage, control and management).

Mutualisation means that the service component is a multi-tenant service element. This allows different users to invoke the service component and enforces the loose coupling property required by SOA service requirements [32].

Step 5: To describe

For a service component to be correctly visible from users and for it to be requested, there is a need for describing it and register used formal processes. That is what is performed, adopting Description and Registry properties in a web services mode.

To design application or service, the architect chooses multi-tenant IoTSCC components in the providers catalogue, based on the specified nominal/offered QoS and thresholds value. The catalogue is a showcase for reusable components. If the composition is entirely IoTSCC-composed then it can be put in a catalogue.

Step 6: To invocate

In order to be agile and not to be static and only configurable, the IoTSCC component has to be invoked through an API (Application Programming Interface). It must be standardised.

IoT service includes interfaces dedicated to QoS control or compliance, IoT service control and programming. These interfaces are classified into three groups: use, control and management (Figure 3).

This approach allows building the service components, but we also need at this stage to build the ”global system structure”. Indeed, the deployment of an IoT

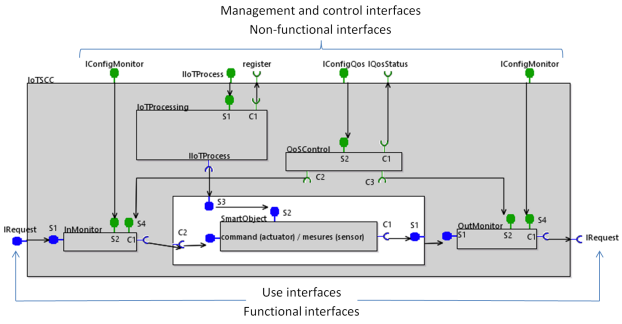

Figure 3: Controlled IoT Service

service application requires an integrated management approach. To help achieve such management in an IoT environment, we have proposed in previous papers conceptual models related to different particular five dimensions of management: architectural, organisational, functional, informational, and relational [36]. The proposition we present in this article is based on a combined use of these conceptual models in order to deploy a dynamic management of IoT service components in an IoT or Cloud system as we present in the following subsections.

3.2 Proposition for the Architectural Dimension

As you may know, the architectural dimension is the definition of the global structure. We rely on a horizontal architecture, rather than architecture in silos. We have described above the structure at the component level. It is an architect, who builds his domain from these components. For example, he can adapt the emission mode by introducing a database and a service component that emits at predefined regular period or in a continuous way. He can also introduce a gateway that allows adaptation to networking protocols and plays the role of intermediary function with Cloud access. There are two types of view: horizontal and vertical. The horizontal view of the global structure is composed of nodes (representation of processing capabilities) and links (representation of transport capabilities). And there is the vertical view where IoT objects need service components (software) at different visibility levels: Physical level and Network level then components at a Cloud (service and application) level. The architect will define the composition based on the global coherence and decisions to distribute. He will follow the proposed procedure. The architect structures the ”smart objects” and makes them ”IoT Service” by integrating our ”IoT Processing” component. Then, based on the service contracts (SLA) that the architect has to guarantee and for which he has alternative solutions based on possible malfunctions, he transforms the ”IoT service” in ”IoT SCC” in order to have QoS based decision. He composes an application based on as-a-service properties.

For the specification, verification and validation of the architecture of applications built from IoT SCC components, we use the VerCors platform from INRIA [29]. Components can be connected with other components within the same membrane or with non-functional interfaces of other components. Having a tool-supported methodology is important for the design phase, when the designer builds his application, using functional components as basic bricks, and assembling them into compositions.

VerCors helps the user to specify the architecture of an application, the interfaces, and the behaviour of assembled components. Furthermore, the tool can generate executable code containing the whole architecture description and the skeleton of the final application. Several validations are performed like structural coherency aspects of the application model for ensuring that the code generation will terminate correctly, and that the code will not fail during deployment of the application components.

A library of components integrating the non-functional aspects (IoT Processing, monitors and QoS-Control) is provided. These components would be instantiated by the application developer to deploy the architecture. VerCors is then in charge of verifying the (static) coherence of the architecture and providing a formal description of the architecture in an ADL (Architecture Description Language) file, which will be included in the specification of the application.

3.3 Proposition for the Organisational Dimension

According to the proposed architecture, different responsibilities need to be defined. Thus, the organisational dimension defines ”who does what”. We propose two kinds of scenery: With and without IoT gateways (Figure 4).

With IoT gateways: the IoT gateway is in charge of collecting information from IoT devices and making some local analysis and transmitting a report to the Cloud. IoT devices make their own report to the IoT gateway. A gateway manages a group of IoT devices. In the same way, the gateway retransmits control commands from the cloud to the IoT devices. The volume of data exchanged and thus the communication resources are extremely low since the analysis would be done on site by the gateway. Only results would be sent. It is a scenario to promote.

Without IoT gateways: IoT devices are directly connected to the Cloud and assume the same functionalities

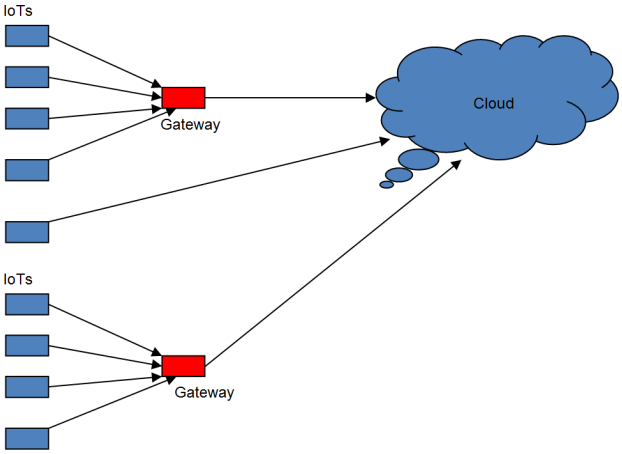

Figure 4: Organisation with Gateways

of the previously detailed gateway.

The dynamic management of an IoT system requires: local capabilities for issues that need rapid reactions, distributed capabilities for mobility-like issues, and centralised capabilities for strategic decisions.

Our proposition is that this organisational dimension relies on the QoS agent by integrating it very accurately in the architecture. It will play the role of an organisational driver as it acts in real-time during the execution to verify the SLA QoS level compliance and reacts dynamically in an autonomous manner in case of non QoS contract compliance during a user session. The QoS agent offers control and distributed management of QoS. It also plays an important role in the dynamic monitoring of the end-to-end IoT session QoS.

More precisely, through the management interface, we can assign to a managed component an autonomous role with a level of intelligence and information to enable the component to make required decisions [29].

3.4 Proposition for the Functional Dimension: IoT micro-services

In this Section, we describe in detail the components and micro-services proposed in our approach:

(A) IoT processing

(B) IoT SCC

(C) Messaging Service

(A) IoT Service integrating "IoT processing"

The IoT service component we propose is within the functional dimension. It plays an important role in the IoT Service Delivery. Thanks to the embedded "IoT processing" component, the component becomes autonomous and manageable. Managing smart objects this way requires appropriate non-functional complementary aspects that we propose to be carried by micro-services.

Micro-services, here, describe a particular way of designing software applications as suites of independently deployable "micro" services.

We define and propose a set of micro-services to be introduced according to the needs:

(i) for the management:

- Get capabilities: means to ask for the characteristics and capabilities of the smart object (screen definition, screen size, memory size, supported codecs, etc.).
- Remote configurations: means to configure the smart object remotely.
- Register service: allows the smart object to register with the gateway or the Cloud to be known, to inform about its presence and to become part of a trusted community.

(ii) for the control:

- Remote control: act remotely and control the smart object
- Time synchronisation: synchronise the set of nodes of a community

Externally communication interfaces are also defined (see Figure 2):

(i) a server interface (IoTProcess) allowing management,

(ii) a client interface allowing control.

In the following, we define the self-controlled component that we call IoT Self-Controlled service Component (IoTSCC).

(B) Controlled IoT Service (IoT SCC)

Based on our experience in the Cloud domain, we have a proposition to share regarding the control of the functionality or operation provided by the IoT service component. It appears to us that it is more accurate and reliable to connect and keep connected only objects that respect their SLA parameters values when performing their functionalities, i.e., that ensures SLA compliance. Initially, components would be chosen according to the service they offer (functional aspect: usage interface) and the QoS level they offer (non-functional aspects: management and control interface), see Figure 3. Then, once performing the service, the proposed SCC component control checks that the same QoS level promised initially is maintained during the processing of service requests. This non-functional control is integrated in the

component membrane. It relies on the triptych (trio): in monitor, out monitor and a non-functional component for the control of QoS SLA compliance that translates the behaviour (and thus the QoS level) expected initially at design-time and proposed as the offered QoS level when selecting the components. This enforces us to propose an additional but required property for this approach: the offered QoS, which selects a service component according to its functionality but also according to its promised behaviour (QoS level measured at design time). Thus, we propose an IoT service component that we call "IoT SCC" as shown in Figure 3. It is a controlled IoT service component that is enabled for self-monitoring and self-control. As we have demonstrated in OpenCloudware Project [10], the Self-Controlled service Component (SCC) controls also functional aspects.

The membrane (non-functional aspects) of the IoT SCC (controlled IoT service component) is composed of:

- An IoT processing component for the management of the smart object.
- Two monitoring mechanisms: a monitor at the entrance of the functional component called "In-Monitor" and a monitoring at the back of the functional component called "OutMonitor". They play a role of interceptors. The service requests arriving to the smart object are intercepted, and then transmitted (of course without being altered) for processing to the functional content of the smart object through the corresponding internal request interfaces. The OutMonitor intercepts the outgoing service requests or responses. They provide measurement information about the interfaces of the functional component.
- A QoS component is added to the smart object functional component. It is in charge of inspecting the respect of the service contract.

The sub-components in the membrane (monitors and QoS control agent) are active in order to perform a monitoring of the QoS level at run time (during processing of requests) and notify in case of degradation of the QoS level by comparing the measured QoS parameters at run time and the QoS parameters at design time (offered/promised QoS levels). Any detected offset between the run time QoS and the design time QoS would mean a non-respect of SLA.

IoT SCC provides the usage interface, which is a functional interface (in blue on Figure 3). It provides the processing functions performed by the smart object and which is the offered service to the users as well as non-functional interfaces (in green on Figure 3). We distinguish two types of non-functional interfaces:

- Management interface: a server interface, it contains the necessary mechanisms to manage the configuration of the non-functional components in the membrane.
- Control interface: a client interface, it contains the mechanisms controlling the service behaviour. It verifies if the non-functional behaviour of the smart object is meeting the service contract. Our IoTSCC structure makes an IoT service component homogeneous. But as modelling allows abstraction, the structure may be applied to different services, either at a device level or a (cloud) service level.

(C) Messaging Service

In addition to the logical bus acting as a hub, we propose different possibilities for sending data using a "Service Messaging" component that can be composed with the IoT SCC component. Figure 5 represents the following composition:

- An IoT SCC component (defined above).
- A database component used to store information (measurements for example) produced by the IoT SCC component.
- A "message processing" component used to consume information stored in a database component to send it to a caller in different ways:
  - *On demand* reporting service: the information is sent when a calling component requests it.
  - *Periodic* reporting Service: the information is sent recurrently at regular time intervals.
  - *Scheduled* reporting service: the information sent is planned or scheduled to be sent at defined times. Each one of these services may be self-controlled and thus becomes an SCC with the triptych (InMonitor, OutMonitor and QoS control components). If the service architect wishes to make a whole self-controlled service composition, he introduces the previous triptych again in the membrane of the highest level of the composition.

In order to be designed as-a-service, all service components should meet the required properties defined in Section 3.1. That is: statelessness autonomy, loose couplings, description, registry, invocations and management (with respect of separation of functional and non-functional aspects of the Grid Component Model). This composition is called IoTAaS (Figure 5).

To compose IoT SCC, the service composition can, of course, be extended with further components like security service, presented in the next subsection.

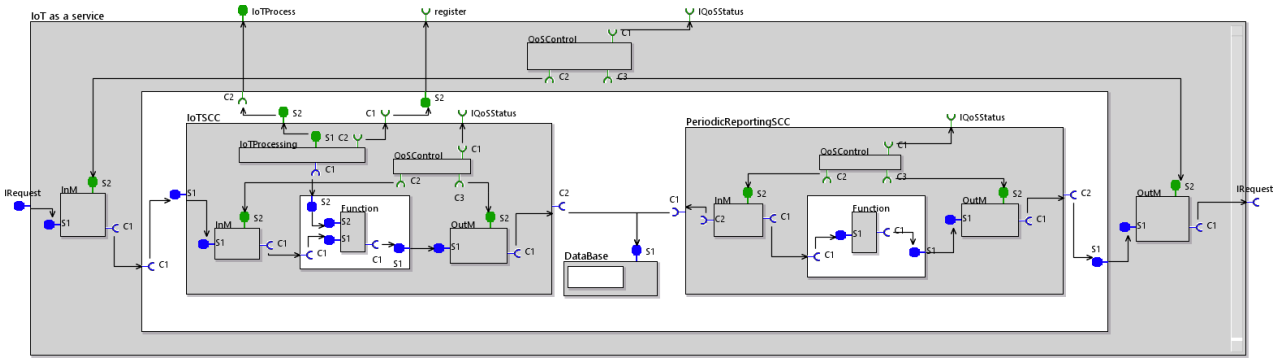

Figure 5: IoT as a service (IoTAaS)

4 SECURED IOT

Concerning the security of the IoT we note a significant upheaval. This open, heterogeneous and mobile environment is vulnerable. It presents significant risks in terms of security. Indeed, the borders of the system are more open since the system is extended: from smart object to the gateway, then to the cloud. In addition, application in IoT devices that generate information, which is accountable and billable, or IoT devices that require device integrity validation, should provide a trusted secure execution environment or trusted platform for executing high security applications. Therefore, in IoT environment security must be provided at different levels:

(i) at the smart object that processes the send/receive message within a trusted environment.

(ii) at the gateway that collects information for the connected objects (Gather Information) and sends them within a trusted environment to the Cloud.

Smart object that requires device integrity validation should provide a trusted execution environment. All data produced through the execution of functions within the trusted environment should be unknowable to unauthorised external entities. The trusted environment should perform confidential functions (such as storing secret keys and providing cryptographic calculations using those secret keys) needed to perform smart object device integrity check and device validation.

In our work, we take into account the concept of security ”as-a-service”. Thus, security is provided ”as-a-service” and it is offered as service components in accordance to the spatial and temporal needs. The user preferences and security objectives are guaranteed. To ensure its aims in IoT and Cloud environment the standard ETSI EG 202 009-2 [24] includes the following security services: authentication, authorisation, certificates, encryption, time stamping, and digital signatures.

In IoT trusted environment, the simple security level can be represented by services such as authentication, authorisation, certificate or non-repudiation. Authentication provides the assurance for the claimed identity of an entity such as a smart object. Authorisation ”as-a-service” adds the process of granting of permission based on authenticated identification. A certificate is issued by a certification body in accordance with the conditions of its accreditation. In IoT environment the certificate can be associated with identifier metadata for interoperability. Non-repudiation provides the ability to prove an action or event has taken place, so that this event or action cannot be repudiated later. Usually, non-repudiation is based on digital certificates, electronic signatures and other similar data stored safely as the proof of the occurrence of an action or event. These four security services will form an IoT trust environment allowing the different degrees of security.

Enhanced security would require the integration of other security services such as encryption, time stamping and digital signatures. Encryption ensures the reversible transformation of data by a cryptographic algorithm to produce cipher text, i.e. hides the information content of the data posted by a smart object or a gateway. Time stamping service allows a security service that attests the existence of electronic data at a precise instant of time. Time stamping services are useful and probably indispensable to support long-term validation of signatures. A digital signature allows a recipient of the data unit to prove his origin and integrity and protect the sender and the recipient against forgery by unauthorised person.

An IoT security environment cannot be guaranteed in a static and centralised manner. Henceforth, in complex and evolving situations we propose that security components to be integrated in the applications themselves. Figure 6 shows a secured IoT service with an authentication security service (IoTAaSS).

Advantages of using security services in IoT are multiple. Considering the security services, IoT providers have applications that differ by their levels of security as

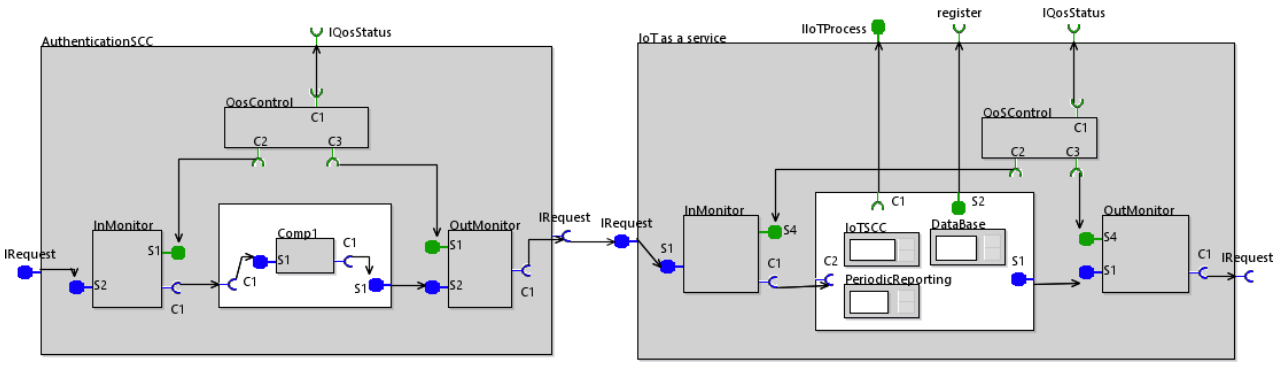

Figure 6: Secured IoT as a service (IoTAaSS)

well as by the way in which they are built, deployed and managed.

5 Design and implementation of a study case

This section presents a study case implementing our IoTAaS component. Section 5.1 describes the study case. Section 5.2 shows that the proposed service component architecture can easily be implemented to provide complex services or applications.

5.1 Study case description

We propose to build warehouse arrival management services (Figure 7). The goal is to automate and streamline the handling of trucks that arrive at a warehouse. Each truck carries dangerous products placed on a different container that are continuously monitored. The statuses of the containers are thereby sent to a cloud application. The truck is also monitored especially to know its location in real time. An employee located in the warehouse can consult the truck arrival board. He knows the estimated arrival time of each truck and can prepare its unloading.

Approaching trucks are assigned automatically to a specific dock number. The dock number is sent to the driver, together with the expected arrival window. The driver confirms or corrects the expected arrival time. Corrections may be necessary due to prescribed driving breaks. The status of approaching trucks is shown on display in the arrival hall and on the mobile phone of all warehouse employees. The display shows expected arrival time and planned dock. All communications have to be secured.

The design and implementation of the study case includes four phases:

- Diagram design on VerCors Component Editor (VCE) with classes and interfaces
- Checking of the validity of the diagram
- Generation of the ADL file and code template of classes and interfaces.
- Code implementation and execution.

5.2 Design and implementation phases

In the architect role we used VCE (Figure 8) to begin designing the architecture. We follow the steps proposed in Section 3. We have a component with a membrane separating the usage plan from the control and management plans (step 1, Figure 1). The encapsulated SmartObject can be a tilt sensor, level sensor or a force sensor located in/or each container. We add an IoTProcessing sub-component (step 2, Figure 2) to transform it in a Service IoT. We decide to control it, thus we add the QosControl and the two In and Out monitors (step 3, Figure 3). The component becomes an IoTSCC.

We wish that this component continuously reports its measurements. So, we make a composition with a database in order to store measures and a periodic reporting component. We choose to control this composition, so we add the QosControl and two monitors. The final composition is called IoT as-a-service (IoTAaS) (step 4, Figure 5). At this step, the IoTAaS could be placed in a providers catalogue for reusing (step 5). We cover the architectural, organisational, and functional dimensions previously defined.

The communication is not secured, so we make a new composition first by reusing the previously defined IoTAaS component from the providers catalogue and second by adding a securing component. The final composition forms a Secured IoT as-a-service (Figure 6). We can repeat the same process to make more complex compositions as needed.

We placed the IoTAaSS previously defined component at different locations. The chosen architecture is defined as follows:

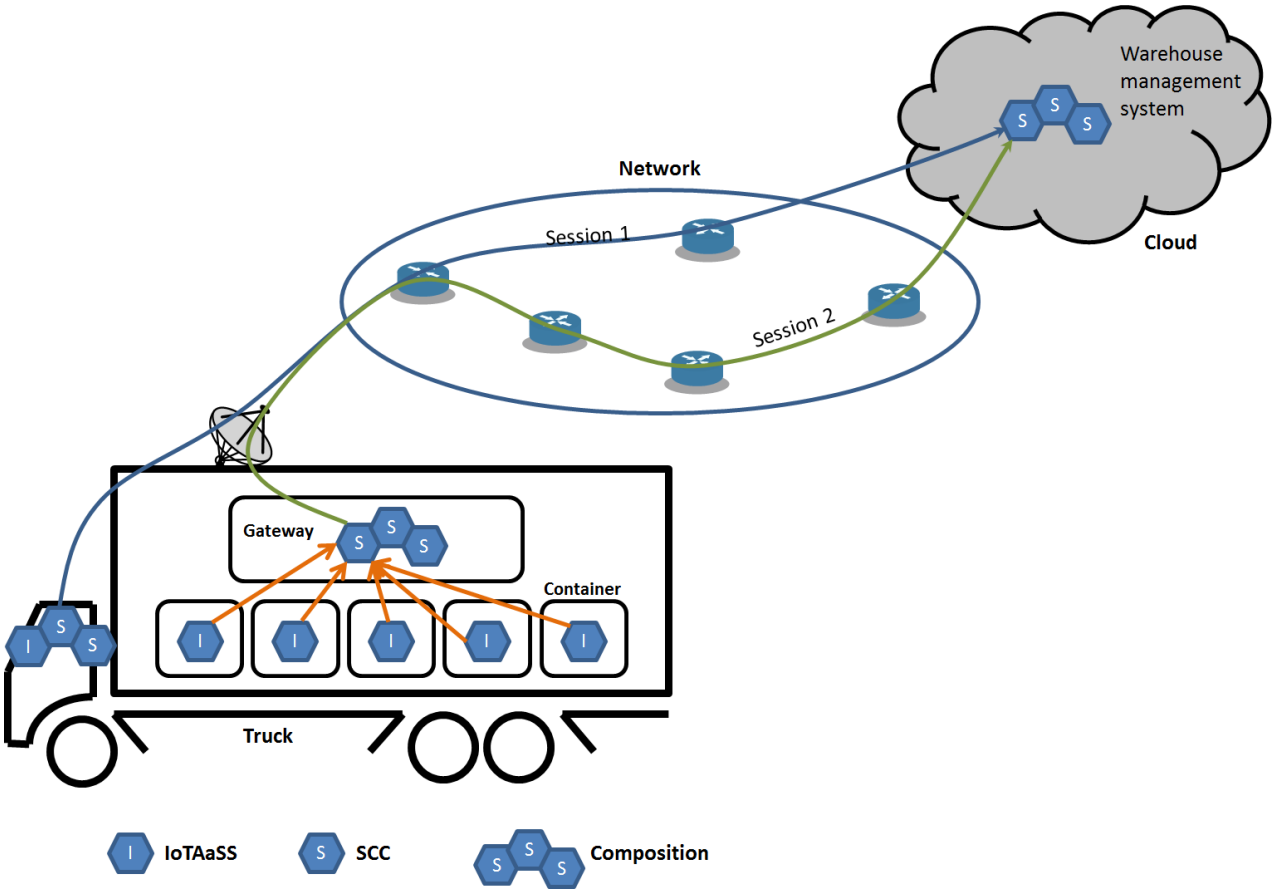

Figure 7: Warehouse arrival management based on IoTAaS

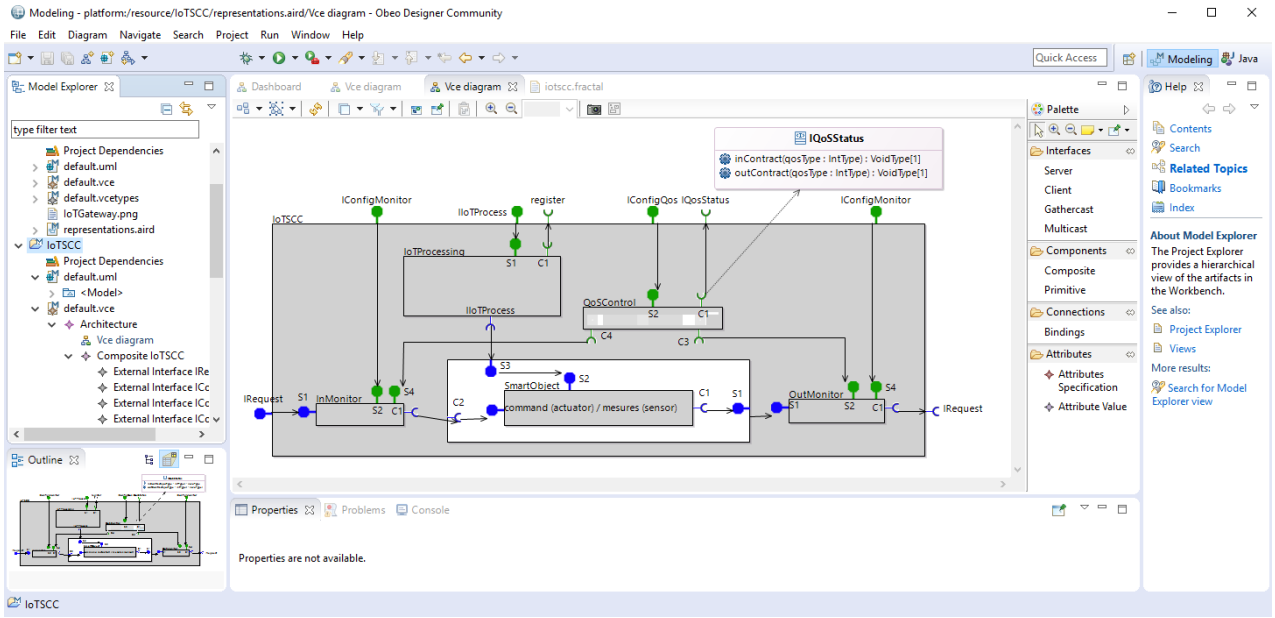

Figure 8: Interface of the VerCors Component Editor (VCE)

(i) An IoTAaSS is located on each container monitoring its movement or its status with the help of gyroscopes, accelerometers or pressure sensors.

(ii) The IoTAaSS performs a periodic and secured reporting to the IoT Gateway. This latter is responsible for gathering all the data sent by the IoTAaSS.

(iii) The IoT Gateway performs a periodic and secured reporting too to the warehouse arrival application located in the Cloud.

The truck embeds a composition responsible for notifying the dock number to the driver, for allowing the driver to confirm or correct the expected arrival time and periodically for sending the truck location to the warehouse arrival application with the help of an IoTAaSS. The data are sent securely and directly to warehouse arrival application bypassing the IoT gateway.

The Warehouse arrival application is a composition based on SCC located in the Cloud. Its function is to gather data from the fleet of trucks (containers and truck statuses), to notify the dock number to the driver, to take into account his corrections concerning the arrival time, and to create the arrival board to the intention of the warehouse employees. Figure 9 shows another example of composition located on IoT Gateway responsible for gathering information. The first component (AuthenticationProcessSCC) verifies the identity of the IoTAaSS sending the request. The GatheringSCC component stores information in a database component.

Note that there are two open sessions. The first one (blue) takes place between the truck composition and the warehouse arrival application and the second one (green) takes place between the IoT Gateway and the warehouse arrival application. The Warehouse arrival application is a central unit. Each session is seen herself as a composition. With our architecture, the architect can control the whole session if he desires it, like any composition by adding QoSControl and Monitors.

At any moment of the design, with VCE, we have the possibility to check the validity of the diagram. When a diagram is completed, VCE can generate a set of files allowing the deployment of the application like code template of classes and interfaces and Architecture Description Language (ADL) for the architecture description. An extract of an ADL file is given in Listing 1. The developer implements the missing service methods of the components in the java classes created with the generated skeleton. These files are then used to build an executable application that can be executed within the GCM/ProActive [26] execution environment (step 6).

6 Summary and conclusions

We have presented an innovative approach to domain engineering based on IoT as-a-service components, QoS control and self-management mechanisms. We have described the whole approach, step by step, in order to allow developers to design an IoT "as-a-service", to build the service composition and to manage it.

This approach has been assessed and refined in the OpenCloudware project. Our IoT service creation environment adopted service composition approach. Thus, IoT service proposed components have the properties recommended by SOA, namely: statelessness, autonomy, and loose coupling, extended with the following properties: description, registry, reuse, mutualisation, and self-management. The IoT service component is QoS based, applicable in all phases of the life cycle to satisfy the continuity of service. Our approach ensures that IoT users have QoS control on IoT services in a dynamic way. Our proposal is backed-up by a design and verification VCE platform, used to build early models of the applications, check their properties, and generate code supported by GCM/proactive open source execution environment. These environments were used in the implementation of a study-case scenario that shows the feasibility of our proposals.

Acknowledgements

The authors would like to thank Professor H. Dayan for his help and relevant remarks.

This work is supported by the OpenCloudware project. OpenCloudware is funded by the French FSN (Fond national pour la Société Numérique), and is supported by Pôles Minalogic, Systematic and SCS.

References

[1] "Amazon Web Services (AWS)." [Online]. Available: https://aws.amazon.com

[2] "AWS IoT." [Online]. Available: https://aws.amazon.com/iot/

[3] "ETSI TR 118 501 - onem2m use case collection."

[4] "ETSI TR 118 502 - analysis of the architectures proposed for consideration by onem2m."

[5] "ETSI TS 118 101 - onem2m functional architecture."

[6] "ETSI TS 118 102 - onem2m requirements."

[7] "IBM Bluemix." [Online]. Available: http://www.ibm.com/Bluemix

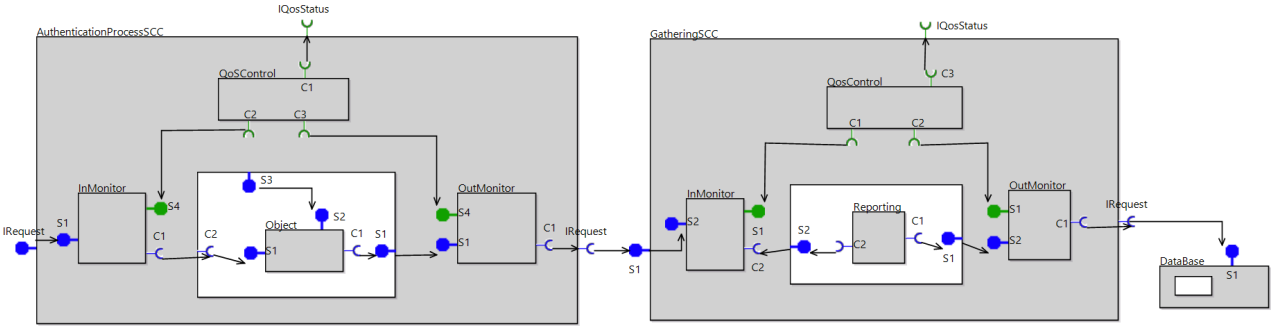

Figure 9: Composition located on IoT Gateway responsible of gathering information

[8] "Internet of Things: Science fiction or business fact?" [Online]. Available: https://hbr.org/resources/pdfs/comm/verizon/18980_HBR_Verizon_IoT_Nov_14.pdf

[9] "Microsoft Azure IoT Hub." [Online]. Available: https://azure.microsoft.com/en/services/iot-hub/

[10] "The OpenCloudware project." [Online]. Available: http://www.opencloudware.org/

[11] "Y.2001 : General overview of NGN." [Online]. Available: https://www.itu.int/rec/T-REC-Y.2001/en

[12] "Y.2060 : Overview of the Internet of things." [Online]. Available: https://www.itu.int/rec/T-REC-Y.2060-201206-I/en

[13] "FP7-ICT 257852, ebbits Enabling the Business-Based Internet of Things and Services," 2010. [Online]. Available: http://www.ebbits-project.eu

[14] "FP7-ICT 257909, SPRINT Software Platform for Integration of Engineering and Things," 2010. [Online]. Available: http://www.sprint-iot.eu/

[15] "FP7-ICT 287708, iCORE Internet Connected Objects for Reconfigurable Ecosystem," 2010. [Online]. Available: http://www.iot-icore.eu/

[16] "Nsf, FIA CNS-1040672, NEBULA a trustworthy, secure and evolvable Future Internet Architecture," 2010. [Online]. Available: http://nebula-fia.org/

[17] "FP7-ICT 288385, IoT.est Internet of Things Environment for Service Creation and Testing," 2011. [Online]. Available: http://ict-iotest.eu

[18] "National Basic Research 973 Program of China under Grant No. 2011cb302701, Basic Research on the Architecture of Internet of Things," 2011.

[19] "FP7-ICT 317674, BETaaS Building the Environment for the Things as a Service," 2012. [Online]. Available: http://www.betaas.eu/

[20] "FP7-ICT 317862, COMPOSE Collaborative Open Market to Place Objects at your SErvice," 2012. [Online]. Available: http://www.compose-project.eu/

[21] "IBM Watson IoT Platform," Sep. 2015. [Online]. Available: https://internetofthings.ibmcloud.com

[22] T. Aubonnet and N. Simoni, "PILOTE: a service creation environment in next generation networks," in *2001 IEEE Intelligent Network Workshop*, May 2001, pp. 36–40.

[23] T. Aubonnet and N. m. Simoni, "Service Creation and Self-management Mechanisms for Mobile Cloud Computing," in *Wired/Wireless Internet Communication*, ser. Lecture Notes in Computer Science, V. Tsaoussidis, A. J. Kassler, Y. Koucheryavy, and A. Mellouk, Eds. Springer Berlin Heidelberg, Jun. 2013, no. 7889, pp. 43–55, dOI: 10.1007/978-3-642-38401-1_4.

[24] T. Aubonnet, N. m. Simoni, and P.-Y. Hebert, "ETSI EG 202 009-2: User Group; Quality of telecom services; Part 2: User related parameters on a service specific basis V0.0.7," Aug. 0030, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01126482

[25] F. o. Baude, D. Caromel, C. Dalmasso, M. Danelutto, V. Getov, L. Henrio, and C. Prez, "GCM: A Grid Extension to Fractal for Autonomous Distributed Components," *annals of telecommunications - annales des tlcommunications*, 2008. [Online]. Available: https://hal.inria.fr/inria-00323919

[26] F. o. Baude, L. Henrio, and C. Ruz, "Programming distributed and adaptable autonomous componentsthe GCM/ProActive framework," *Software: Practice and Experience*, vol. 45, no. 9, pp. 1189–1227, 2015. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/spe.2270/full

[27] E. Bruneton, T. Coupaye, M. Leclercq, V. Quma, and J.-B. Stefani, "The FRACTAL component model and its support in Java," *Software: Practice*

and Experience, vol. 36, no. 11-12, pp. 1257–1284, Sep. 2006.

[28] BUTLER, "FP7-ICT 287901, BUTLER uBiquitous, secUre internet-of-things with Location and contExt-awaReness," 2011. [Online]. Available: http://www.iot-butler.eu/

[29] A. Cansado and E. Madelaine, "Specification and Verification for Grid Component-Based Applications: From Models to Tools," in *Formal Methods for Components and Objects*, ser. Lecture Notes in Computer Science, F. S. d. Boer, M. M. Bonsangue, and E. Madelaine, Eds. Springer Berlin Heidelberg, Oct. 2008, no. 5751, pp. 180–203, dOI: 10.1007/978-3-642-04167-9_10.

[30] A. Cansado, E. Madelaine, and P. Valenzuela, "VCE: A Graphical Tool for Architectural Definitions of GCM Components," Spain, 2008.

[31] M. Chen, V. C. M. Leung, R. Hjelsvold, and X. Huang, "Smart and interactive ubiquitous multimedia services," *Computer Communications*, vol. 35, no. 15, pp. 1769–1771, Sep. 2012.

[32] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services," *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp. 223–235, 2010.

[33] F. Mattern, "From smart devices to smart everyday objects," in *Proceedings of smart objects conference*, 2003, pp. 15–16. [Online]. Available: http://www.vs.inf.ethz.ch/publ/papers/Generic_106.pdf

[34] F. Mattern and C. Floerkemeier, "From the Internet of Computers to the Internet of Things," in *From Active Data Management to Event-Based Systems and More*, ser. Lecture Notes in Computer Science, K. Sachs, I. Petrov, and P. Guerrero, Eds. Springer Berlin Heidelberg, 2010, no. 6462, pp. 242–259, dOI: 10.1007/978-3-642-17226-7_15.

[35] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "MobilityFirst: A Robust and Trustworthy Mobility-centric Architecture for the Future Internet," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 16, no. 3, pp. 2–13, Dec. 2012.

[36] N. Simoni, S. Znaty, N. Perdigues, and S. Arsenis, *Gestion de rseau et de service: similitude des concepts, spcificit des solutions*. Paris, France: Interditions : Masson, 1997.

[37] slange, "FP7-ICT 257521, iot-a - Internet of Things Architecture," 2010. [Online]. Available: http://www.iot-a.eu

AUTHOR BIOGRAPHIES

Tatiana Aubonnet is assistant professor at computer science department of CNAM (Paris). She holds a PhD in computer and network science from Telecom ParisTech and an Habilitation from Pierre and Marie Curie University (University of Paris VI). Her research interests cover service creation and management in Next Generation Networks.

Amina Boubendir is a PhD Student at Orange Labs Networks France and at Tlcom Paris Tech in the Department of Networking and Computer Science. Her main research interests focus on management of network operations and services as well as the application of service-oriented and model-driven engineering to Telco Cloud and Network Functions Virtualization.

Frédéric Lemoine has an engineering degree in computer science, microelectronics and automatics. He is a research engineer and development project manager at the computer science department of CNAM (Paris). His expertise includes programming and modelling languages, heterogeneous parallel systems programming, embedded systems and mobile devices programming.

Noëmie Simoni is an Emeritus Professor of Telecom-Paristech. She was Head of Architecture and Engineering of Networks and Services (AIRS) research group at the Department of Computer Science and Network. Her research interests include QoS management and modelling of complex systems. Her expertise, gained through many academic projects and industrial contracts, covers wide range of management topics. Today, her main work is focused on network convergence and service convergence, network virtualisation and cloud computing.

Listing 1: Extract of ADL code

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE definition PUBLIC "-//objectweb.org//DTD Fractal ADL 2.0//EN"
"classpath://org/objectweb/proactive/core/component/adl/xml/proactive.dtd">
<!-- Automatically generated by Vercors, INRIA Sophia-Antipolis -->
<definition name="IoTSCC">
    <interface name="IRequest" role="server" signature="interfaces.IRequest"/>
    <interface name="IRequest" role="client" signature="interfaces.IRequest"
    contingency="optional" interceptors="OutMonitor.Interceptor"/>
    <component name="SmartObject">
        <interface name="command (actuator) / mesures (sensor)" role="server"
        signature=".interfaces.AutoGeneratedInterface"/>
        ...
        <content class="classes.Business"/>
        <controller desc="primitive"/>
    </component>
    <binding client="SmartObject.C1" server="this.IRequest"/>
    ...
    <content class=".classes.CompositeDefaultClass"/>
    <controller desc="composite">
        <interface name="IConfigMonitor-controller" role="server"
        signature="interfaces.IConfigMonitor"/>
        ...
        <component name="InMonitor">
            ...
            <content class="classes.Monitor"/>
            <controller desc="primitive"/>
        </component>
        <component name="OutMonitor">
            ...
            <content class="classes.Monitor"/>
            <controller desc="primitive"/>
        </component>
        <component name="QoSControl">
            ...
            <content class="classes.QoSControl"/>
            <controller desc="primitive"/>
        </component>
        <component name="IoTProcessing">
            ...
            <controller desc="primitive"/>
        </component>
        <binding client="this.IConfigMonitor-controller" server="InMonitor.S2"/>
        ...
    </controller>
</definitio>
```