Cloud4Things

Marcus Vinícius Paulino Gomes

Instituto Superior Técnico, Universidade de Lisboa
marcus.paulino.gomes@tecnico.ulisboa.pt

Table of Contents

Abstract. In this document we present Cloud4Things, a solution that proposes to make the deployment and management of IoT applications for smart places. Due to the heterogeneity of IoT applications, management tasks such as deployment are complex tasks that require advanced technical knowledge. Cloud4Things proposes to decrease the complexity of these tasks by adopting a high-level perspective to perform the deployment and management operations of IoT applications. To simplify the deployment operation at the Cloud, Cloud4Things will rely on cloud orchestration tools to execute such task. That allow the specification of the application structure and allow the control the life-cycle of the application. Cloud4Things will allow the management and monitoring of the provisioned resources at the Cloud by a smart place in a high abstraction perspective. This document also presents the state-of-the-art solution as well an overview about the proposed approaches and evaluation methods.

Keywords: Internet of Things, Cloud Computing, Smart Places, Service Level Agreements, Orchestration Tools, Automated Deployment

1 Introduction

In recent years computing is becoming more ubiquitous in the physical world. This notion of ubicomp was introduced by Weiser [1], where the concept of smart environment was defined as "*a physical world richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives and connected through a continuous network*". Through the years, technology advances such as the creation of the Internet contributes to achieving the ubicomp's vision which enables individual devices to communicate between themselves from any part of the world [2]. In Weiser's vision of ubiquitous computing, information seamlessly moves in and out of attention as automation gives way to human interaction [3].

The progress towards ubiquitous computing has been slower than expected [4], another approaches for ubiquitous computing that constrasts with Weiser vision becomes relevant. Rogers proposes an approach which focuses in designing technologies for engaging user experiences that, in a creative and constructive way, extend the peoples' capabilities [5]. Rogers points that ubiquitous technologies can be developed not only for individuals, but for particular domains that can be set up and customized by an organization according its needs, such as agriculture, retailing and logistics.

The main difference between Weiser's and Rogers vision concerns with the user experience. In Weiser point of view computers take the initiative to act on people's behalf [6], while in Rogers vision is not the computer that is proactive, the user assumes that role.

Actually, Weiser vision is close to becoming reality thanks to the Internet of Things and Cloud Computing [7]. This world where things are connected through a continuous network is a vision thats represents the Internet of Things (IoT). In this vision, physical items are continuously connected to the virtual world and can act as remotely physical access points to Internet Services. The Internet of Things would make computing truly ubiquitous [8].

A common scenario where the Internet of Things paradigm is applied are smart environments [9], which in this document are designated as smart places. In the context of this work, a smart place can be defined as an ecosystem composed by smart objects such as RFID tags and sensors, that are able to acquire knowledge about this environment and also to adapt to the peoples that live in it in order to improve their experience in that environment [10].

An example of a smart place is a smart warehouse. In the smart warehouse, products and objects are identified with RFID tags. These tags can be used to tracking the objects inside the warehouse. To perform the tracking of the objects, the smart warehouse must have an IoT application that process the data generated by the objects. This application also allow that managers monitor the

warehouse. For instance, is possible to monitor the objects that enters and leaves the warehouse. To monitoring the objects in the warehouse, the IoT application must be able to identify the tagged objects. In the case of RFID tagged objects, the EPC Framework provides a complete support to perform the traceability of objects, as well the awareness of its status and current location [9].

In particular, the deployment of an IoT application in the smart warehouse is a challenge today due the heterogeneity of the smart objects that are present in the warehouse and also because the required infrastructure. In order to decrease the complexity of the deployment of an IoT application in a smart warehouse, the Cloud computing paradigm allows the virtualization of the required physical infrastructure. Nowadays the server infrastructure needed by IoT applications can be virtualized by Cloud providers, which gives more flexibility and reduces the costs in the deployment of an IoT application.

Therefore, another challenge concerns the heterogeneity of a smart warehouse, more precisely the variety of smart objects that can be inside of a given warehouse. Many of these objects uses different communication protocols and drivers to communicate with devices, thus the configuration of this objects must be manually handled, which makes the integration of these objects in the deployment of IoT applications in a smart warehouse an inefficient process.

1.1 Internet of Things

The Internet of Things is a concept in which the virtual world of information technology integrates seamlessly with the real world of physical things [11]. Through the amount of computer and network devices available nowadays, facts about the real world becomes accessible to business and everyday use cases.

A more precise definition of the Internet of Things was formulated in the Strategic Research Agenda of the Cluster of European Research Projects on the Internet of Things (CERP-IoT 2009):

"*In the IoT, 'things' are expected to become active participants in business, information and social processes where they are enabled (...) by exchanging data and information sensed about the environment, while reacting autonomously to the 'real/physical world' events and influencing it by running processes that trigger actions and create services with or without direct human intervention. Interfaces in the form of services facilitate interactions with these 'smart things' over the Internet (...) taking into account security and privacy issues.*"

Internet of Things will enable a more convenient life for people. One of the expectations of IoT concerns with providing services to the end-users through common objects, in that way products will be able to provide recommendations for use and maintenance instructions, supply warranty information or point to

related products.

Recently, IoT became relevant to industry and end-users [11], mainly because the reduction of cost and miniaturization of technologies used in IoT applications, such as RFID, sensor networks, NFC and wireless communication. Technologies like these allows the detection of status of things, which together with collection and processing of detailed data, allows the creation of an interactive and responsive network with huge potential for citizens, consumers, business to gather immediate responses to events that occurs in the real world [11].

Using these technologies, IoT can provide access to more detailed information, which results that the level of analysis can be performed in a large-scale perspective, as well in a small-scale perspective. But IoT is more than a tool for managing processes more efficiently and more effectively. Another expectation of IoT concerns with real-world awareness provided to information systems [8]. An example is logistics applications, by the use of RFID, companies can react promptly to relevant physical events, and then manage their processes in a better way, typically increasing efficiency and reducing costs.

1.2 Cloud Computing

Cloud Computing is a paradigm where a computing infrastructure is designated as a *"Cloud"* from which its resources are available to businesses and users from anywhere in the world on demand [12]. This paradigm enables rapid service delivery in a dynamically scalable and virtualized manner. Actually, clouds are built on top of modern data centers, that incorporated Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [13], as illustrated on Figure 1.

Cloud providers, such as Amazon Web Services[1], Google Cloud Platform[2] and Microsoft Azure[3] offer these services as utilities in a pay-per-use model in which the customer can indicate the desired service level that is delivered by the providers.

Data Centers allocates all the hardware that supports the cloud. They consist of a large number of physical machines that provide redundancy and ensure reliability in case failures. By using the IaaS pattern, Cloud Computing is able to virtualize resources as computing power, storage and network connectivity of the data centers. These computational resources are provisioned on demand in a form of Virtual Machines (VMs) deployed in a cloud provider data-center [14]. The Cloud can also be used to assist application design, development, application hosting and testing by providing a development platform, a service pattern

[1] http://www.aws.amazon.com
[2] http://cloud.google.com
[3] http://www.azure.microsoft.com/

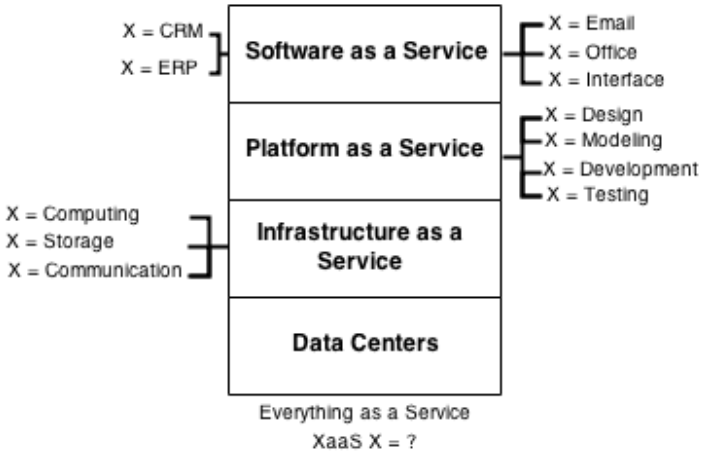

Fig. 1: Hierarchical view of Cloud Computing.

is known as PaaS. Another pattern supported by Cloud Computing is SaaS. That allows that a single piece of software that are available to millions of users through a browser [15]. This pattern is beneficial to both users and developers: the developers only needs to maintain a single program while the users can save costs in servers and software.

In a global analysis Cloud computing offers several benefits such as high-availability, high-scalability, flexibility, on-demand service provisioning and reduction of costs. By converging IoT applications and Cloud computing, it should be possible to take advantage of all these benefits.

Cloud Orchestration Due the heterogeneity of IoT applications infrastructure, service management tasks such as deployment, driver installation and gateway configuration are still handled manually for each case. In order to reduce the complexity of these tasks, Cloud Orchestration tools can be used to automate these management tasks. The orchestration can wove the software components of the application in one piece that can be managed more effectively. Cloud Orchestration helps to take advantage of the full benefits of Cloud computing by providing features that can include:

- Integration of cloud capabilities [16] such as *On-demand self-service*, *Resource Pooling* and *Measured Service* across heterogeneous environments and infrastructure, to simplify, automate and optimize service deployment;
- Automated high-scale provisioning and de-provisioning of resources with policy-based tools;
- Real-time monitoring of physical resources, virtual resources and accounting chargeback capabilities to track and optimize system usage;

In addition, orchestration allows significantly reduction of the costs related with labor and resources, since that manual intervention and management of varied IT services and resources are not needed. Currently, several tools to performing Cloud Orchestration are available in the market such as IBM Cloud Orchestrator[4], HP Operations Orchestration[5] and GigaSpaces Cloudify[6] and some open-source tools like Ubuntu Juju[7] and OpenTOSCA[8].

1.3 Quality of Service of Cloud Applications

As mentioned before, customers can define the service level that is intended to be delivered by Cloud providers. The desired service level is specified by customers through of Quality of Service *QoS* parameters [17]. QoS is a concept that encompasses a number of nonfunctional properties such as availability, reliability, price and reputation [18].

To guarantee the desired service level, Cloud providers need to consider and meet these QoS parameters for each costumer [17]. To achieve that, Cloud providers establish a bilateral agreement with the customers that is known as Service Level Agreement (SLA). The SLA defines the terms and conditions of service quality that a Cloud provider delivers to service requesters based on the desired *QoS* [19].

The definition of these SLAs establish a contract between the Cloud provider and the customer. This contract contains the specification of the QoS parameters agreed by all parties, as well the penalties for meeting and violating the expectations [17].

1.4 Smart Places Architecture

As described before, a smart place is composed by several smart objects that generate data that is processed by the IoT application. However, to process all that data an infrastructure is required to support the IoT application. These infrastructure is composed by RFID tags, sensors, readers and servers, as illustrated in Figure 2. Although this infrastructure can support the IoT application in the smart place, this architecture presents several issues regarding the deployment of the application, the low scalability, the costs of infrastructure and its maintenance.

[4] http://www.ibm.com/software/products/en/ibm-cloud-orchestrator
[5] http://www8.hp.com/us/en/software-solutions/operations-orchestration-it-process-automation
[6] http://www.gigaspaces.com/cloudify-cloud-orchestration/overview
[7] http://www.juju.ubuntu.com
[8] http://www.iaas.uni-stuttgart.de/OpenTOSCA/indexE.php

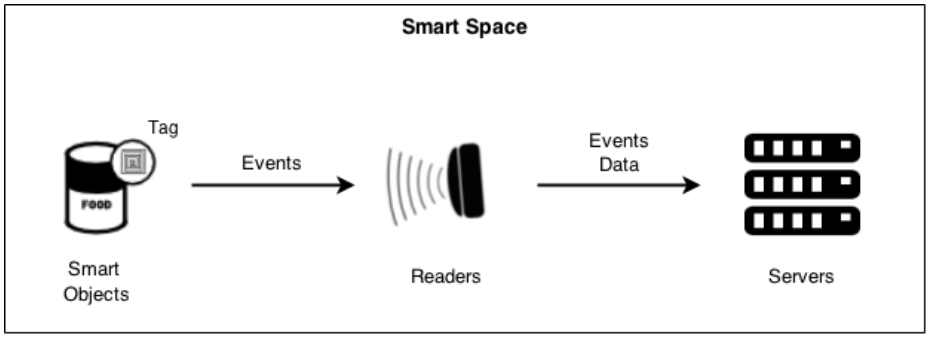

Fig. 2: Typical smart place scenario, using RFID technology.

By converging the IoT applications with the Cloud computing paradigm the objective is simplify the smart space architecture by leveraging the required infrastructure by these applications to the Cloud providers, as illustrated in Figure 3, allowing to take advantage of the benefits offered by Cloud computing (as referenced in 1.2).

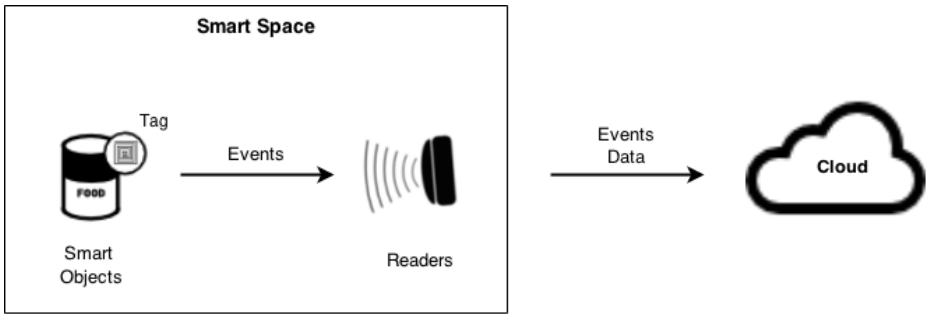

Fig. 3: Cloud-based smart place.

The convergence between the IoT applications and the Cloud allow to simplify the architecture of the smart place in a significantly way. However, as earlier mentioned these solution still presents some issues to be resolved, namely the deployment of the application.

1.5 RFID Smart Place

For our work we will focus on software for an RFID smart place. The software to use will be based on the GS1 EPCglobal standards for RFID.

Fosstrak is an open-source RFID project[9] that implements the GS1 EPC (Electronic Product Code) Network ALE, IS and LLRP standards. In Figure 4 a simplified version of the EPC Network architecture is illustrated.

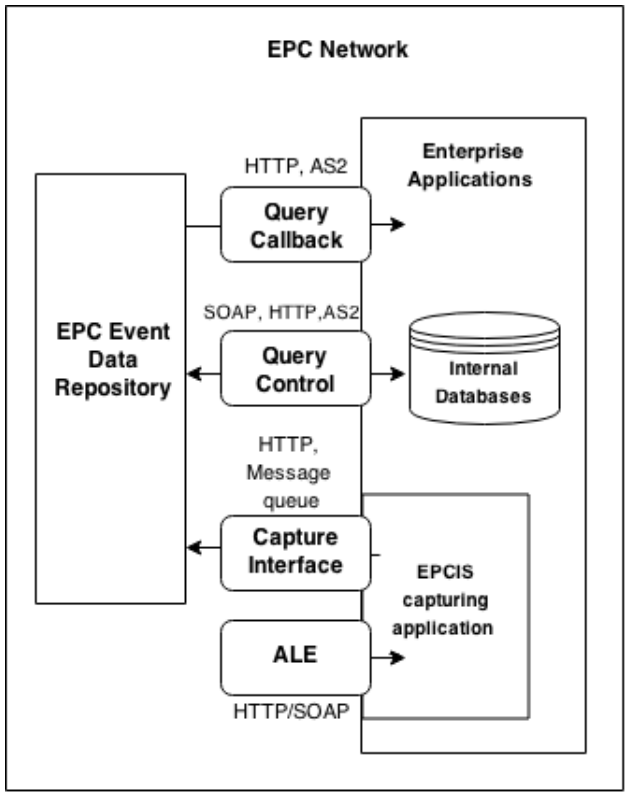

Fig. 4: Simplified view of EPC network.

The EPC Application Level Events (ALE) provides interfaces to filter and collect data. The EPC Information Services (IS) provides interfaces to interpret, store persistently and query captured EPC event data. The EPC Low-Level Reader Protocol (LLRP), that is not demonstrated in this architecture, provides an interface to configure and control RFID readers.

Fosstrak provides an entire RFID infrastructure that accelerates the development of IoT applications and allows to users to gain hands-on experience with the EPC Network.

[9] https://code.google.com/p/fosstrak/

1.6 Overview

The remainder of this document is organized as the follows. In Section 2 we describe the problem to be solved as well the main objectives of this work. Section 3 presents the related work in the research area. Then in Section 4 we present a brief description of the state of art solution and propose the architecture for our solution. In Section 5 the evaluation methodology is described. Section 6 presents the conclusion. In the appendix a schedule for the future work is presented and a methodology to evaluate the RFID event handling.

2 Objectives

The life-cycle of a Cloud-based IoT application is composed by several stages, as illustrated in the Figure 5. Some of the stages are placed in the smart place while others are placed in the Cloud.

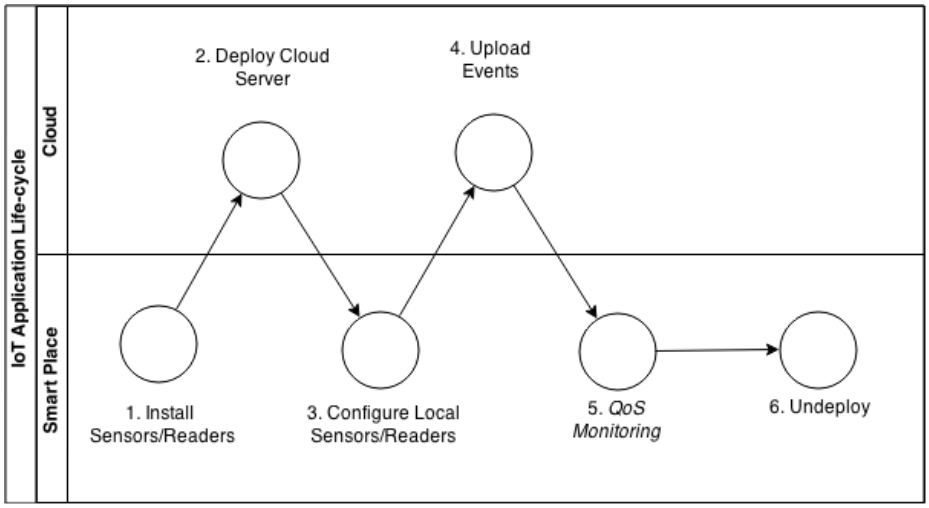

Fig. 5: IoT application life-cycle.

The life-cycle of an IoT application starts with the installation of the sensors and readers in the smart place (step 1). The next step consists in preparation to process the events sent by the smart objects. First the application must be deployed at the Cloud in order to be able to receive the events (step 2). After that the installed sensors and readers must be configured to send the generated events to the application that is running in the Cloud (step 3). At this point the smart place is already configured to support the processing of events generated by smart objects (step 4). The monitoring of the the performance of the application in the smart place is performed to determine if the application is running with the desired *QoS* (step 5). The final stage that an IoT application can reach is its undeployment, which means that the smart place is deactivated (step 6).

The main objective of Cloud4Things is to decrease the complexity of the deployment of Cloud-based IoT applications in a smart place. Usually the deployment of such applications require that all the components of the application are manually configured and installed. In order to reduce the complexity of this process, Cloud4Things will automate the deployment process of Cloud-based IoT applications. To automate the deployment process of the applications Cloud4Things will rely in Cloud Orchestration tools to execute such operation. These tools allows the specification of the components and the relation between

them in a high-level abstraction, provisioning the necessary resources at the Cloud in a effective way and monitoring the application during its life-cycle.

However, orchestration tools do not solve all problems. In particular, RFID applications require a software stack that usually is composed by a database, a web server and the software components that implements the EPC Network standards. But these tools lack the integration of the EPC Network software with the other components. Thus, to support the deployment of IoT applications that are based in RFID technologies, the orchestration tool that will perform the deployment of the application need to be extend to support the integration of the EPC software components with the other software components that are required by the application.

Another aspect that is important is guarantee that Cloud providers are delivering the desired service level. As sub objective, Cloud4Things will allow to define the QoS parameters that specify the desired service level through the orchestration tool. These QoS parameters will be expressed in terms of SLAs that will be negotiated between the Cloud provider and the customer. The definition of these QoS parameters will allow monitoring the performance of the application and verify if the delivered service level meet with the expectations. In order to monitoring the performance of the application, Cloud4Things will need to monitor the service delivered by Cloud providers and determine if the level of this service guarantee an acceptable QoS for the application.

3 Related Work

IoT applications often encapsulate several relatively complex protocols and involves different software components. Moreover, they require a significant investment in infrastructure, besides that the system administrators and users spend time with large client and server installations, setups, or software updates. As most of the computing resources are allocated on the Internet on servers in Cloud computing, integrating these paradigms in a Cloud-based model results in a solution with more flexibility of implementation, high scalability and high availability, and with a reduced upfront investment.

3.1 Internet of Things and Cloud Computing

In RFID-based IoT applications, Guinard et al. [20] point out that the deployment of RFID applications are cost-intensive mostly because they involve the deployment of often rather large and heterogeneous distributed systems. As a consequence, these systems are often only suitable for big corporations and large implementations and do not fit the limited resources of small to mid-size businesses and small scale applications both in terms of required skill-set and costs. To address this problem, Guinard et al. propose a Cloud-based solution that integrates virtualization technologies and the architecture of the Web and its services. The case of study consists in a IoT application that uses RFID technology to substitute existing Electronic Article Surveillance (EAS) technology, such as those used in clothing stores to track the products. In this scenario they applied the Utility Computing blueprint to the software stack required by the application using the AWS platform and the EC2 service. To evaluate the Cloud-based solution, two prototypes was successfully implemented to prove that the pain points of the RFID applications can be relaxed by adopting the proposed solution.

Distefano [21] et al. proposed a high-level modular architecture to implement the Cloud of Things. According to the authors, things not only can be discovered and aggregated, but also provided as a service, dynamically, applying the Cloud provisioning model to satisfy the agreed user requirements and therefore establishing "Things as a Service" providers. The "*Things as a Service*" (TaaS) paradigm envisages new scenarios and innovative, pervasive, value-added applications, disclosing the Cloud of Things world to customers and providers as well, thus enabling an open marketplace of "things". To address these issues, an ad-hoc infrastructure is required to deal with the management of sensing an actuation, resources provided by heterogeneous Clouds and and things. The proposed architecture provides blocks to deal with all the related issues, while aiming to provide things according to a service oriented paradigm. Altough the presented concept is very innovative, the proposed solution could not be evaluated since that the the solution was not implemented.

CloudThings [22] is an architecture that uses a common approach to integrate Internet of Things and Cloud Computing. The proposed architecture is an online platform which accommodates IaaS, Paas, and SaaS and allows system integrators and solution providers to leverage a complete IoT application infrastructure for developing, operating and composing IoT applications and services. The applications consists of three major modules, the CloudThings service platform, that is a set of Cloud services (IaaS), allowing users to run any applications on Cloud hardware. This service platform dramatically simplifies the application development, eliminates need for infrastructure development and reduces management and maintenance costs. The CloudThings Developer Suite is a set of Cloud service tools (PaaS) for application development, such as Web service APIs, which provide complete development and deployment capabilities to developers. The CloudThings Operating Portal is a set of Cloud services (SaaS) that support deployment and handle or support specialized processing services. To evaluate CloudThings, a smart home application based on a Cloud infrastructure was implemented. In the application, the sensors read the home temperature and luminosity and the Cloud stored and visualized the data, so that the user can view the smart home temperature and luminosity anywhere. In particular, in these implementation the Cloud architecture was extended by inserting a special layer for dynamic service composition. This middleware encapsulates sets of fundamental services for executing the users service requests and performing service composition, such as process planning, service discovery, process generation, process execution, and monitoring. This first implementation also demonstrates that this middleware as a service releases the burden of costs and risks for users and providers in using and managing those components.

3.2 Automated Deployment of Cloud Applications

The effort put in the research to integrate the paradigms of Cloud Computing and Internet of Things resulted in a essential contribution. The integration of these two paradigms allow that IoT applications to have higher availability, scalability and to reduce the costs associated with the maintenance of the application. But there are several issues regarding to the integration between Cloud Computing and Internet of Things that must be addressed. In particular, due of the heterogeneity of the IoT applications environments, it is hard for solution providers to efficiently deploy and configure applications for a large number of users. Thus, automation of the management tasks required by IoT applications is a key issue to be explored.

TOSCA (Topology and Orchestration Specification for Cloud Applications) [23] is proposed in order to improve the reusability of service management processes and automate IoT application ratified by OASIS in [] deployment in heterogeneous environments. TOSCA is a new cloud standard to formally describe the internal topology of application components and the deployment process of Cloud applications. The structure and management of IT services is specified by

a meta-model, which consists of a *Topology Template*, that is responsible for describing the structure of a service, then there are the *Artifacts*, that describe the files, scripts and software components necessary to be deployed in order to run the application, and finally the *Plans*, that defined the management process of creating, deploying and terminating a service. The correct topology and management procedure can be inferred by a TOSCA environment just by interpreting the topology template, this is known as "declarative" approach. Plans realize an "imperative" approach that explicitly specifies how each management process should be done. The topology templates, plans and artifacts of an application are packaged in a Cloud Service Archive (.csar file) and deployed in a TOSCA environment, which is able to interpret the models and perform the specified management operations. These .csar files are portable across different Cloud providers, which is a great benefit in terms of deployment flexibility. To evaluate its feasibility, TOSCA was used in the specification of a typical application in building automation. an application to control an Air Handling Unit (AHU). The common IoT components, such as gateways and drivers will be modeled, and the gateway-specific artifacts that are necessary for application deployment will also be specified. By archiving the previous specifications and corresponding artifacts into a .csar file, and deploying it in a TOSCA environment, the deployment of AHU application onto various gateways can be automated. As a newly established standard to counter growing complexity and isolation in cloud applications environments, TOSCA is gaining momentum in industrial adoption as well for academic interest.

Breitenbücher et al. [24] proposed to combine the two flavors of management supported by TOSCA, *declarative processing* and *imperative processing*, in order to create provisioning plans based on TOSCA topology models. The combination of both flavors would enable applications developers to benefit from automatically provisioning logic based on declarative processing and individual customization opportunities provided by adapting imperative plans. These provisioning plans are workflows that can be executed fully automatically and may be customized by application developers after initial generation. The approach enables benefits from both flavors that may leads to economical advantages when developing applications with TOSCA. The motivating scenario that is used to evaluate this approach consists in a LAMP-based[10] TOSCA application to be provisioned. The application implements a Web-shop in PHP that uses a MySQL database to store product and customer data. The application consists of two application stacks, one provides the infrastructure for the application logic and the other hosts the database. Both ran on Amazon's public IaaS offering Amazon EC2. To measure the performance of the deployment using the two-flavor approach, the time spent to generate provisioning plans regarding the number of templates required by the application was measured. The results indicate that

[10] LAMP is a software stack that originally consists of the Linux system, the Apache HTTP Server, the MySql relational database and the PHP programming language.

the required time increases linearly with the number of templates.

Recently a growing number of organizations are developing Orchestrators, Design Tools and Cloud Managers based on TOSCA. Juju is an Open Source TOSCA Orchestrator that can deploy workloads across public, private clouds, and in physical machines. HP Cloud Service Automation is cloud management solution that supports declaratives services design that are aligned with TOSCA modeling principles. GigaSpaces Cloudify orchestrates TOSCA Service Templates using workflows to automate deployments and other DevOps automation processes. IBM Cloud Orchestrator provides integrated tooling to create TOSCA applications, deploy them with custom polices, monitoring and scale them in cloud deployments.

3.3 Service Level Agreements in Cloud Applications

Initially the objective of this work was focused in improve the deployment operation. However, during the research of this work it became clear that the automation of the deployment of IoT applications in smart places was an area that has a notorious progress in the developed work, standing out the development of Cloud Orchestration tools based on TOSCA. Therefore, it was realized that the focus of this work could be redirected to explore a field that still needs large research effort and is fundamental for the IoT success [9], *QoS* support. In order to guarantee an acceptable *QoS* for Cloud-based IoT applications, SLAs must be negotiated between service customers and services providers in order to guarantee an acceptable *QoS* for the service. Recently, *QoS* support has been a field where researchers are proposing new approches regarding policy-aware IoT applications based on TOSCA.

Waizenegger et al. [25] proposed a mechanism to demonstrate how non-functional requirements are defined in TOSCA using policies. TOSCA allows the specification of non-functional requirements like cost, security, and environmental issues. However, TOSCA lacks detailed description of how to apply, design, and implement policies. Waizenegger et al. propose two mechanisms for processing policies during the deployment and management of Cloud services in TOSCA. The policies are described according a taxonomy where a Cloud service policy is a tuple of elements that describe its behavior and effect. To perform the implementation of this policy-specific logic, two approches were proposed: an Implementation Artifact Based Policy Enforcement (IA-Approach) and a Plan-Based Policy Enforcement (P-Approach) [25]. In the IA-Approach the existing Implementation Artifacts - responsible to perform service management operations - are extended to support policy enforcement implementations, thus these artifacts are comprised of two alternative implementations for each service management operation: one that is policy enforcing and one that is not. In the P-Approach the policies are enforced in an imperative way, for each policy the plan execute the appropriate steps to enforcing the policy. Although both of

these approaches allow the enforcement of policies, the approach used for realizing this should be chosen with caution. The IA-Approach is more suitable for less complex scenarios and the policies implemented with this approach can be reused very easily. On the other hand, the enforcement of policies in more complex scenarios should be performed according the P-Approach, that allows to have a global control of the triggered enforcement actions. The proposed solution could not be evaluated since that no implementation was performed.

Breitenbücher et al. [26] proposed an approach that enables automated provisioning and management of composite Cloud applications in compliance with non-functional security requirements defined by policies. In this approach the Management Planlet Framework [27] was extended to support policy-aware provisioning and management based in Management Policies that bind non-functional security requirements to the management tasks that enforce them. This paper also introduces the concept of Management Annotation Policies, which defines the semantics and how the policy must be processed. The automation of provisioning and management tasks are performed by Policy-aware Management Planlets, that execute these tasks considering the Annotation Policies attached to each operation. The Framework also allows the Cloud providers and application developers to specify their own policy-aware management logic in a flexible and reusable manner independently from individual applications. The management logic can be defined in two ways, by applying Automated Management Patterns or by manually creating Desired Application State Models. The Framework was evaluated in terms of feasibility, performance, economics, limitations and extensibility. A prototype was successfully implemented based to validate the concept technically based on a previous implementation of the Management Planlet Framework.

4 Solution Architecture

4.1 State of the Art

Cloud-based IoT applications are the state-of-the-art of this kind of solutions. By leveraging the infrastructure to the Cloud providers, these applications have a high-availabity, can dynamically scale while spending a fraction compared with the traditional solutions. However, as earlier mentioned the deployment of such applications still is an issue, due its complexity and required manual intervention. The deployment of Cloud-based IoT applications usually are performed through IT automation tools, such as Chef[11] and Puppet[12]. These automate the deployment of applications. However, the components of the application and the relation between themselves must be specified manually, which requires considerable manual work and expertise by the person that is performing the deployment. The deployment process of these solutions is not the only existing issue. Monitoring the application life-cycle is a task that requires a lot of effort and expertise by the system administrators.

In order to solve this problem, the adopted approach relies on using Cloud Orchestrator tools perform the deployment of these applications. As mentioned 1.2, these tools allows the specification of the application components and their relations in a high-level perspective and to execute the management tasks required by the application during its life-cycle. As a matter of fact, in a low-level perspective cloud orchestration tools express the high-level perspective defined by the user into scripts that latter are executed using IT automation tools such as Puppet and Chef.

4.2 Cloud4Things Architecture

The architecture of Cloud4Things is based in the perspectives that are used to describe an information system according the Zachman framework [28]. In this framework Zachman defines the *Conceptual*, *Logical* and *Technological* perspectives that allows to describe an information system from a more abstract to a more detailed view.

Conceptual Architecture The *Conceptual* architecture describe Cloud4Things in perspective of the business entities and processes that compose the system and how they are related, as illustrated in Figure 6.

[11] http://www.chef.io
[12] http://www.puppetlabs.com

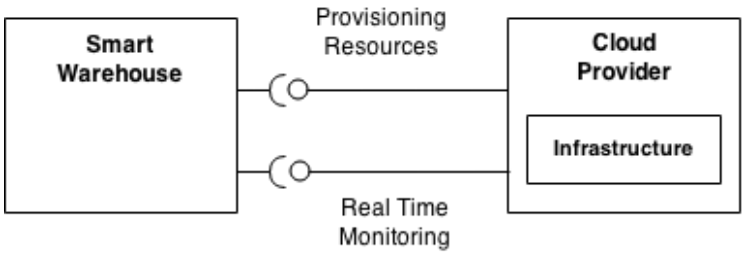

Fig. 6: Cloud4Things Conceptual architecture.

The business entities that composes Cloud4Things are the smart warehouse and the Cloud provider. The warehouse uses an Internet connection to access the resources that are allocated in the Cloud infrastructure. The Cloud providers allow the smart warehouse to perform real-time monitoring of the resources that are consumed.

Logical Architecture The *Logical* architecture describe Cloud4Things in perspective of the data elements, logical process flows and functions that represent business entities and processes, as illustrated in Figure 7.

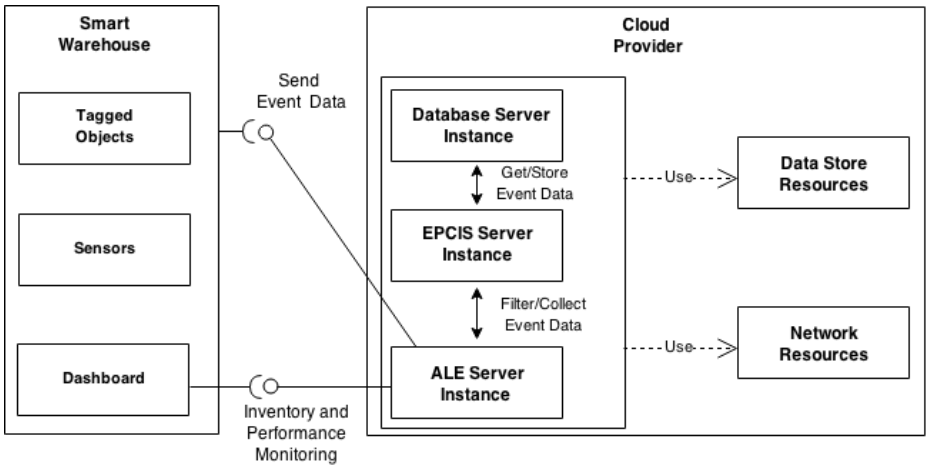

Fig. 7: Cloud4Things Logical architecture.

The smart warehouse is composed by sensors and tagged objects. These sensors and tagged objects are the physical components of the IoT application that is running in the Cloud. The tagged objects produces events that are captured by

the sensors and sent to the application in the Cloud.

The Cloud provide computing resources (ex. Network Bandwidth, Data Storage) that are used by the servers where the application is running. In case of an IoT application that is based in the EPC Network standards, the application is composed by layers that are able to collect and filter the data from the sensors (ALE) through the network, to interpret and query (EPCIS) the filtered data from the ALE layer and to store persistently (Database) the data that is captured by the EPCIS layer. Each one of these layers must be implemented by a different server in the Cloud.

The monitoring of the application is performed through a dashboard. This dashboard provides the information about the current performance of the application, allows to know what objects that enter and leave the smart warehouse and what objects are currently inside the smart warehouse.

Technological Architecture The *Technological* architecture describe Cloud4Things in perspective of the specific technology that is used to implement the *Logical* perspective, as illustrated in Figure 8.

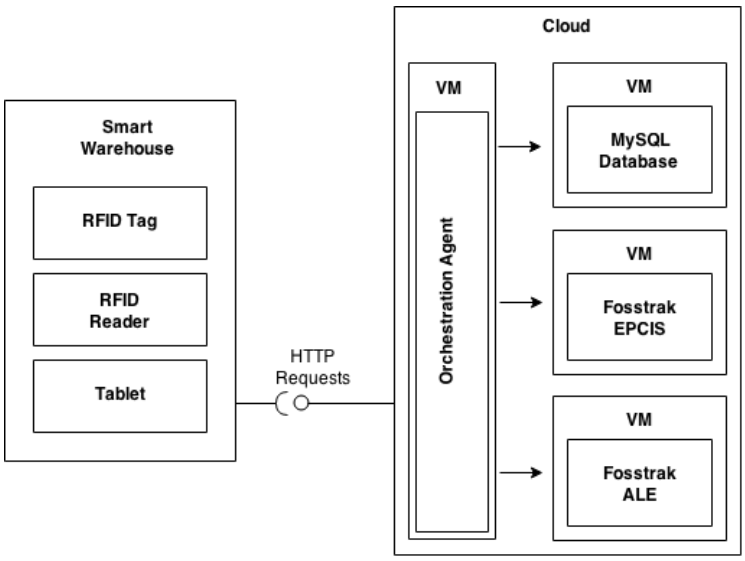

Fig. 8: Cloud4Things Technological architecture

In the smart warehouse objects are tagged with RFID tags. The RFID readers that are installed in the warehouse captures the data that is produced by these tags and send to the Cloud application through a protocol to exchange data,

usually the HTTP protocol.

Before the application start to process the data that is produced in the warehouse, the application must be deployed in the Cloud. The deployment of the application is performed through orchestration tools. These tools provides an interface that can be accessed through a device that is connected to the Internet (ex. a Tablet). This interface allow the specification of the components of the application, the execution of the deployment operation as well to monitoring the application.

In the Cloud, the Orchestration Agent provisioning the VMs where the software stack required by the application will be installed and configured. The software stack required by the RFID-based IoT application is composed by a MySQL Database and the Fosstrak framework, namely by the Fosstrak EPCIS and Fosstrak ALE modules.

QoS Monitor Architecture As mentioned in 2, a sub objective of Cloud4Things is to allow the definition of the *QoS* parameters that specify the desired service level for the application. These QoS Agent will be able to monitor the application according to the *QoS* parameters that are defined for the application, as illustrated in Figure 9.
The presented architecture is based on the LoM2HiS Framework [29] architecture. The *Host Monitor* processes the monitored values of the hardware and network resources that are delivered by monitoring agents that are embedded in the infrastructure. The *Run-time Monitor* is responsible to monitor the customer application (*Services*) status and performance based on the negotiated and agreed SLAs.

The monitoring process starts after the Cloud provider agrees on the SLA terms. The agreed SLAs are stored in the repository for service provisioning and the following steps are executed:

1. The Cloud provider creates rules for the framework mappings of low-level metrics (QoS parameters) using Domain Specific Languages[13] (DSLs).
2. The customer requests the provisioning of an agreed service.
3. Once the request is received, the run-time monitor loads the service SLA from the agreed SLA repository.
4. The resource metrics are measured by monitoring agents, these metrics are stored in a raw format that later are accessed by the host monitor.
5. The host monitor extracts metric-value pairs from the raw metrics and them transmits them periodically to the run-time monitor.

[13] Domain Specific Languages are small languages that normally are tailored to a specific problem domain.

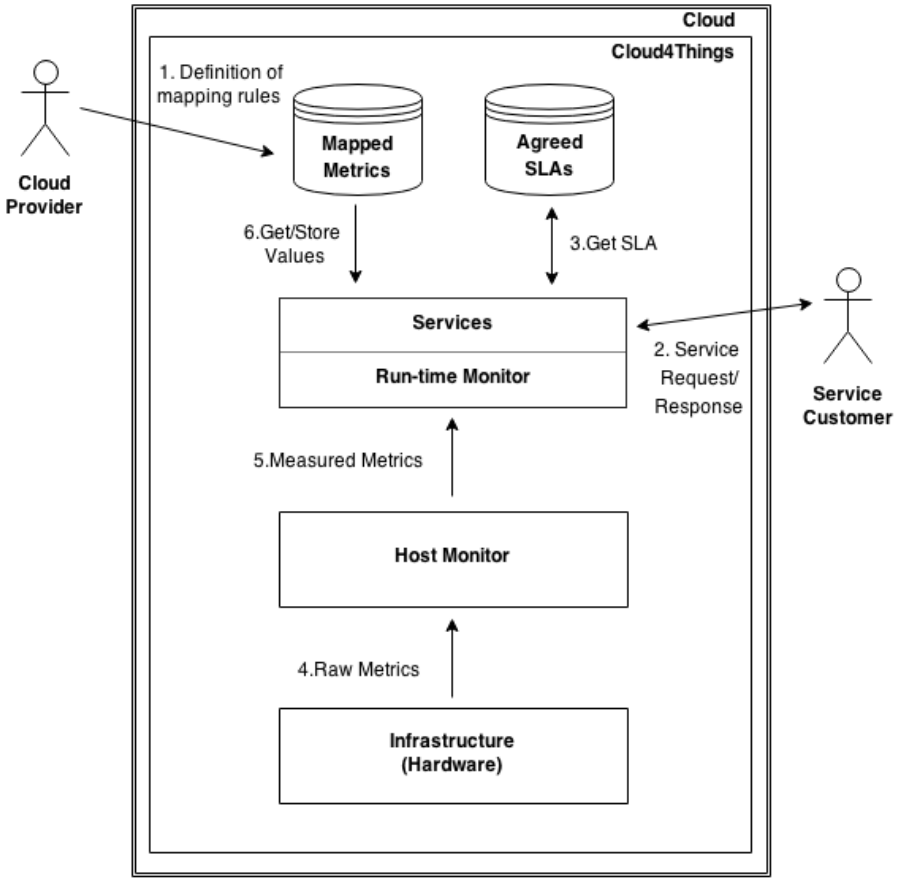

Fig. 9: QoS Monitor Architecture.

6. After receiving the low-level metrics, the run-time monitor uses predefined mapping rules to map the low-level metrics into an equivalent form of the agreed SLA and them the resulting map is stored in the mapped metrics repository.

In this architecture, the *Run-time monitor* uses the mapped values to monitor the status of the deployed services. Once it detects that a SLA is violated, the *Run-time monitor* must alert the customer of the violated SLA.

5 Evaluation Methodology

The evaluation of the solution will be performed according to two perspectives: one consists in evaluating the performance of the Cloud and the other consists in evaluate the performance of the deployment of the application.

5.1 Application Deployment Evaluation

The performance of the application deployment is an important aspect to be evaluated. The deployment operation can be evaluated in terms of the metrics of *Network Bandwith*, *Data Volume* and *Latency* of the process. These values determine the efficiency of the deployment operation. For instance, if we have a high *Data Volume* and a low *Network Bandwith* available, the deployment process will have a high *Latency*.

5.2 Cloud Performance Evaluation

An aspect that is important concerns with the performance of the Cloud regarding the amount of data generated by the events. Cloud computing creates an illusion that available computing resources on demand are limitless [30]. However, with the increase of the amount of events, we must measure these computing resources in order to determine if the system is fulfilling the *QoS* requirements. To perform the evaluation of the behaviour and performance of the Cloud we need to measure some system metrics such as:

- *CPU Utilization:* indicates the percentage of time that the CPU was working at the instances in the Cloud. Normally, this metric is available through the Cloud providers. Usually, the range of this metrics is given in percentage that can vary between 0-100%.
- *Memory Usage:* indicates the amount of memory that is consumed by the system in a given period of time. The unit of this metric is given in MBytes.
- *System Load:* is a metric that indicates the general state of the system. This metric estimates the general performance of the system by measuring the number of received events. The range of this metric varies between 0 and 1. When this metric has a value of 0, it means that the system is not receiving any events at the time. If this metric has a value of 1, it means that the system is overloaded, consequently the CPU Utilization and Memory Usage metrics are close to the maximum value.

6 Conclusion

The Internet of Things is a paradigm that revolutionizes the way in that common objects interact with the environment. In this document we introduce the concept of smart places as a specific IoT application. A smart place is an ecosystem that is composed by smart objects, in this case RFID tagged objects, that are interconnected to the Internet infrastructure. However, due to the diversity of smart objects and its different communication protocols, these smart places are characterized for its heterogeneity which increases the complexity the execution of management tasks such as deployment and monitoring of the IoT application that is running in this smart place. The state of the art solutions reduce the complexity of such tasks by automating the deployment process, but as mentioned in Section 2 these solution require a high level of expertise in order to perform these tasks.

Therefore, with this work we intend to decrease the complexity of the execution of these management tasks in two ways. The main objective if this work is to automate the deployment of IoT applications in smart places by using Cloud orchestration tools, that allow to describe the application structure in a high-level perspective, thus modelling the applications structure in terms of the application logic. We also intend to guarantee that the Cloud providers are delivering a service level that meets the expectations. To guarantee that, Cloud4Things will enable the definition of *QoS* parameters that will be used to establish Service Level Agreements between the Cloud providers and the customers.

In the evaluation of the developed solution, we will evaluate the performance of the application deployment and the performance of the application that is running in the smart place. An important point of the evaluation is to demonstrate the possible scenarios where the smart place generates more data then the Cloud can process and also the opposite scenario where the Cloud is overpowered. Ultimately, we intend that this work contributes to improving the developed work in the field of smart places, focusing in *QoS* monitoring that is essential to guarantee its correct operation.

A Appendix

A.1 Work Schedule

In this section we propose a schedule that estimates the required time in future work that will be realized.

Milestone	Dates	Total Days
Create the model of the solution's structure	10/02/2015 - 17/02/2015	7
Implement a first release of the solution that delivers a useful feature.	18/02/2015 - 04/03/2015	14
Perform the tests and the evaluation.	04/03/2015 - 06/03/2015	2
Define what will be implemented in the next releases.	07/03/2015 - 09/03/2015	2
Implement the next releases.	10/03/2015 - 11/04/2015	30
Perform the tests and the evaluation.	12/04/2015 - 14/05/2015	2
Write the dissertation document	15/05/2015 - 15/06/2015	30
Deliver the final version of the dissertation document	16/06/2015	1

Table 1: Milestones and corresponding dates for the work of Master Thesis.

A.2 RFID Event Handling Evaluation

RFID has its particular characteristics, which means that traditional event processing systems cannot support them. Furthermore, RFID events are temporal constrained [31]. Temporal constraints as the time interval between two events and the time interval for a single event are critical to event detection. In order to assure the correctness of these received events, some conditions must be defined regarding the time interval between the events. Temporal constraints are not the only cause that can compromise the correctness of received the RFID events such as collisions on the air interface, tag detuning and tag misalignement [32].

The metric used to evaluate the RFID performance is proposed by Correia [33]. The *PresenceRate* is a metric that takes in account the reported time of reading a RFID tag. This metric measures the ratio between the time that a given object spent in a given area and the expected spent time. Correia [33] defined that the expected behaviour of this metric is given by the following values:

- $PresenceRate = 0$: the object was not detected by the system;
- $0 < PresenceRate < 1$: there are false negative readings and they were reported;
- $PresenceRate = 1$: ideal scenario where the reported time is exactly the same as expected;
- $PresenceRate > 1$: there are false positive readings over-estimating the time spent by the object in the read area;

In order to perform a general evaluation of the system, we will establish a relation between the *PresenceRate* and the metrics of Cloud performance. The scenario where the *PresenceRate* is larger then 1 can be used to determine if the system is overloaded. In this scenario the metrics of *PresenceRate* and *SystemLoad* must be compared to verify this condition. If the *SystemLoad* value is equal to 1, it means that the system is operating in it maximum capacity and the amount of resources available is not enough to process all the requests. Otherwise this comparison can determine if the system is overpowered. If the *PresenceRate* is equal to 1 and the *SystemLoad* is close to 0, it means that the available resources are more than the required to process all the requests.

References

1. M. Weiser, R. Gold, and J. S. Brown, "The origins of ubiquitous computing research at parc in the late 1980s," *IBM systems journal*, vol. 38, no. 4, pp. 693–696, 1999.
2. J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.
3. M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.
4. A. Greenfield, *Everyware: The dawning age of ubiquitous computing*. New Riders, 2010.
5. Y. Rogers, "Moving on from weiser's vision of calm computing: Engaging ubicomp experiences," in *UbiComp 2006: Ubiquitous Computing*, pp. 404–421, Springer, 2006.
6. D. Tennenhouse, "Proactive computing," *Communications of the ACM*, vol. 43, no. 5, pp. 43–50, 2000.
7. R. Cáceres and A. Friday, "Ubicomp systems at 20: progress, opportunities, and challenges," *IEEE Pervasive Computing*, vol. 11, no. 1, pp. 14–21, 2012.
8. 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*, pp. 242–259, Springer, 2010.
9. L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.
10. D. Cook and S. Das, *Smart environments: technology, protocols and applications*, vol. 43. John Wiley & Sons, 2004.
11. D. Uckelmann, M. Harrison, and F. Michahelles, *Architecting the Internet of Things*. Springer Publishing Company, 1st ed., 2011.
12. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.
13. W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-oriented cloud computing architecture," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pp. 684–689, IEEE, 2010.
14. B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *Internet Computing, IEEE*, vol. 13, no. 5, pp. 14–22, 2009.
15. S. Zhang, S. Zhang, X. Chen, and X. Huo, "Cloud computing research and development trend," in *Future Networks, 2010. ICFN'10. Second International Conference on*, pp. 93–97, IEEE, 2010.
16. P. Mell and T. Grance, "The nist definition of cloud computing," 2011.
17. L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.
18. J. O'Sullivan, D. Edmond, and A. Ter Hofstede, "What's in a service?," *Distributed and Parallel Databases*, vol. 12, no. 2-3, pp. 117–133, 2002.
19. L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition," *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 311–327, 2004.

20. D. Guinard, C. Floerkemeier, and S. Sarma, "Cloud computing, REST and mashups to simplify RFID application development and deployment," in *Proceedings of the Second International Workshop on Web of Things*, p. 9, ACM, 2011.
21. S. Distefano, G. Merlino, and A. Puliafito, "Enabling the cloud of things," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pp. 858–863, IEEE, 2012.
22. J. Zhou, T. Leppanen, E. Harjula, M. Ylianttila, T. Ojala, C. Yu, and H. Jin, "Cloudthings: A common architecture for integrating the internet of things with cloud computing," in *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, pp. 651–657, IEEE, 2013.
23. F. Li, M. Vogler, M. Claeßens, and S. Dustdar, "Towards automated IoT application deployment by a cloud-based approach," in *Service-Oriented Computing and Applications (SOCA), 2013 IEEE 6th International Conference on*, pp. 61–68, IEEE, 2013.
24. U. Breitenbücher, T. Binz, K. Képes, O. Kopp, F. Leymann, and J. Wettinger, "Combining declarative and imperative cloud application provisioning based on TOSCA," *IC2E. IEEE*, 2014.
25. T. Waizenegger, M. Wieland, T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, B. Mitschang, A. Nowak, and S. Wagner, "Policy4tosca: A policy-aware cloud service provisioning approach to enable secure cloud computing," in *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, pp. 360–376, Springer, 2013.
26. U. Breitenbücher, T. Binz, C. Fehling, O. Kopp, F. Leymann, and M. Wieland, "Policy-aware provisioning and management of cloud applications," *International Journal On Advances in Security*, vol. 7, no. 1 and 2, pp. 15–36, 2014.
27. U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and M. Wieland, "Policy-aware provisioning of cloud applications," in *SECURWARE 2013, The Seventh International Conference on Emerging Security Information, Systems and Technologies*, pp. 86–95, 2013.
28. J. A. Zachman, "A framework for information systems architecture," *IBM systems journal*, vol. 26, no. 3, pp. 276–292, 1987.
29. V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level metrics to high level slas-lom2his framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," in *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, pp. 48–54, IEEE, 2010.
30. M. Armbrust, O. Fox, R. Griffith, A. D. Joseph, Y. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "M.: Above the clouds: A berkeley view of cloud computing," 2009.
31. F. Wang, S. Liu, P. Liu, and Y. Bai, "Bridging physical and virtual worlds: complex event processing for RFID data streams," in *Advances in Database Technology-EDBT 2006*, pp. 588–607, Springer, 2006.
32. C. Floerkemeier and M. Lampe, "Issues with RFID usage in ubiquitous computing applications," in *Pervasive Computing*, pp. 188–193, Springer, 2004.
33. N. Correia, "RFIDToys: A Flexible Testbed Framework for RFID Systems," Master's thesis, Instituto Superior Técnico, Portugal, 2014.