Institute of Architecture of Application Systems

Comparison of IoT Platform Architectures: A Field Study based on a Reference Architecture

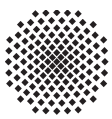
Jasmin Guth, Uwe Breitenbücher, Michael Falkenthal, Frank Leymann, and Lukas Reinfurt

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{guth, breitenbuecher, falkenthal, leymann, reinfurt}@iaas.uni-stuttgart.de

BibTeX:

```
@inproceedings{Guth2016_ComparisonIoTPlatformArchitectures,
  author    = {Guth, Jasmin and Breitenb{\"u}cher, Uwe and Falkenthal, Michael
               and Leymann, Frank and Reinfurt, Lukas},
  title     = {Comparison of IoT Platform Architectures: A Field Study based on
               a Reference Architecture},
  booktitle = {2016 Cloudification of the Internet of Things (CIoT)},
  year      = {2016},
  month     = {Nov},
  pages     = {1--6},
  doi       = {10.1109/CIOT.2016.7872918},
  publisher = {IEEE}
}
```

© 2016 IEEE Computer Society. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.



Universität Stuttgart
Germany

Comparison of IoT Platform Architectures: A Field Study based on a Reference Architecture

Jasmin Guth, Uwe Breitenbücher, Michael Falkenthal, Frank Leymann, and Lukas Reinfurt
Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany
Email: [lastname]@iaas.uni-stuttgart.de

***Abstract*—The Internet of Things (IoT) is gaining increasing attention. The overall aim is to interconnect the physical with the digital world. Therefore, the physical world needs to be measured and translated into processible data. Further, data has to be translated into commands to be executed by actuators. Due to the growing awareness of IoT, the amount of offered IoT platforms rises as well. The heterogeneity of IoT platforms is the consequence of multiple different standards and approaches. This leads to problems of comprehension, which can occur during the design up to the selection of an appropriate solution. We tackle these issues by introducing an IoT reference architecture based on several state-of-the-art IoT platforms. Furthermore, the reference architecture is compared to three open-source and one proprietary IoT platform. The comparison shows that the reference architecture provides a uniform basis to understand, compare, and evaluate different IoT solutions. The considered state-of-the-art IoT platforms are OpenMTC, FIWARE, SiteWhere, and Amazon Web Services IoT.**

I. Introduction

The Internet of Things (IoT)[1] is gaining increasing attention. The idea of IoT is to interconnect the physical world with the digital world [1]. Therefore, sensors measure parameters of the physical world as well as changes of it. Consequently, this information is translated into data processible by computers [2]. Furthermore, the aim of IoT is to act on the physical world through actuators, e.g., the temperature of a room can be measured and monitored, if a threshold is exceeded the air-conditioner is turned on. As soon as the desired temperature is reached the air-conditioner is turned off. Due to smart home applications, such as the described example, IoT has already arrived within our daily life. Along with this development, the impact of cloud computing rises as well since devices are often accessed through the cloud and along with the trend towards smart cities, a huge amount of data has to be processed.

Diverse integration approaches are provided, such as FIWARE[2] or Amazon Web Services IoT[3]. However, the heterogeneity of different integration approaches leads to multiple selection problems. The major problem is to find a suitable IoT platform for a given field of application. Although IoT platforms provide similar or even equal functionality, their implementation and the underlying technologies differ. This leads to diverse concepts and architectures, which complicates a comparison of multiple platforms. For instance, some IoT solutions use the term "things" for a component, whereby others use the term "devices". It is unclear what "things" exactly are and if "things" and "devices" are equal. Since there is no general architecture applied, users have to dive deep into the platforms' descriptions and have to understand each architecture and their components from scratch. This procedure is time-consuming and foreknowledge is required. The result of the discussion above is that an abstract reference architecture is needed to provide a basis for comparing diverse IoT platforms.

In this paper, we tackle these issues by introducing an abstract IoT reference architecture, which is based on several state-of-the-art IoT platforms. In contrast to many other reference architectures, such as the reference models introduced by Cisco [3] or Fremantle [4], our reference architecture is kept abstract on purpose to ensure a broad applicability. Therefore, our reference architecture does not present new concepts, but provides a more abstract view on the components of IoT platforms and their possible connections. Many existing reference architectures provide a detailed view on IoT platforms. The more detailed each reference architecture gets, the more heterogeneous they become as a whole. Thus, the aim of our reference architecture is to build an abstract terminology that serves as a uniform knowledge basis. Within this paper, we define each component of the reference architecture and compare three open-source platforms and one proprietary platform by mapping their architectures onto our reference architecture. Thereby, we further ease the comparison of different platforms. Our comparison shows that the reference architecture is generally applicable and demonstrates how to understand the investigated architectures based on our reference architecture.

The remainder of this paper is structured as follows: In Section II, we introduce the derived IoT reference architecture defining all components and their possible communication. In Section III, we compare our reference architecture to four state-of-the-art IoT platforms. We compare our IoT reference architecture against existing approaches in Section IV. In Section V, we conclude the paper and outline future work.

II. IoT Reference Architecture

The IoT reference architecture described in the following is derived from a comparison of several IoT platforms including open-source as well as proprietary ones. Figure 1 shows the different components and their intercommunication. For the sake of simplicity, the components are depicted without cardinalities. Furthermore, components can also be omitted.

[1]The term "Cyber-Physical-System" (CPS) can be used as a synonym since both terms are recently mentioned coincidentally.
[2]https://www.fiware.org/
[3]https://aws.amazon.com/de/iot/

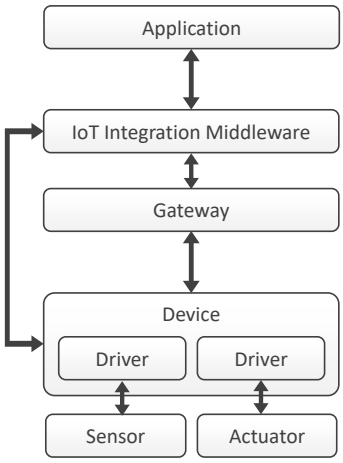


Fig. 1. IoT reference architecture

For instance, if a cyber-physical system is only used to measure the parameters of the physical environment, the system would have no actuators. In contrast to existing reference architectures, we kept ours abstract on purpose since the aim of our reference architecture is to serve as a uniform, abstract terminology, which eases the comparison of different platforms. To distinguish our terminology from the ones used by the considered platforms, the component names of our IoT reference architecture are written in italics in the following.

A. Sensor

A *Sensor* is a hardware component, which is used to measure parameters of its physical environment and to translate them into electrical signals, for example, by measuring the temperature or humidity of a room. If required, a *Sensor* may be configured using software, but cannot run software itself. Typically, *Sensors* are connected to or are integrated into a *Device* to which the gathered data is sent. Prominent examples for *Devices* are RaspberryPis, BananaPis, the Arduino boards, or BeagleBones. The connection can be established by wires or wireless, for instance, via radio.

B. Actuator

An *Actuator* is a hardware component, which can act upon, control, or manipulate the physical environment, for example, by giving an optic or acoustic signal. *Actuators* receive commands from their connected *Device*. They translate electrical signals into some kind of physical action. Just like *Sensors*, *Actuators* are typically connected to or are even integrated into a *Device*, whereby the connection can be established by wires or wirelessly. If required, *Actuators* can be configured using software but cannot run software themselves.

C. Device

A *Device* is a hardware component, which is connected to *Sensors* and/or *Actuators* via wires or wirelessly or even integrates these components. To process data from *Sensors* and to control *Actuators*, typically software in the form of *Drivers* is required. A *Driver* in our architecture enables other software on the *Device* to access *Sensors* and *Actuators*. It represents the first possibility to use software to process data produced by *Sensors* and to control *Actuators* influencing the physical environment. Thus, *Devices* are the entry point of the physical environment to the digital world. *Devices* are either (i) self-contained or (ii) connected to another system, e.g., to an *IoT Integration Middleware*. If they are self-contained, they build a black box of functionality, e.g., to control an air-conditioner by evaluating data from a connected temperature *Sensor*.

D. Gateway

Devices are often connected to a *Gateway* in cases when the *Device* is not capable of directly connecting to further systems, e.g., if the *Device* cannot communicate via a particular protocol or because of other technical limitations. To solve these problems, a *Gateway* is used to compensate such limitations by providing required technologies and functionalities to translate between different protocols and by forwarding communication between *Devices* and other systems. A *Gateway* is, therefore, responsible for supporting the required communication technologies and protocols in both directions and for translating data if necessary. For instance, a *Device* communicates with a *Gateway* via an IoT protocol, such as ZigBee or MQTT. When the *Gateway* receives a message in a proprietary binary format from the *Device*, the *Gateway* translates the information into JSON or XML and forwards the data to a system in the world wide web. Likewise, the *Gateway* may translate commands into communication technologies, protocols, and formats supported by the respective *Device*. The *Gateway* may already execute some data processing functions, such as data aggregation, depending on its processing capabilities.

E. IoT Integration Middleware

The *IoT Integration Middleware* is responsible for receiving data from the connected *Devices* to process the received data, for example, by evaluating condition-action rules, to provide the received data to connected *Applications*, and to control *Devices* in terms of sending commands to be executed by the respective *Actuators*. A *Device* can communicate directly with the *IoT Integration Middleware* if it supports an appropriate communication technology, such as WiFi, a corresponding transport protocol, such as HTTP or MQTT, and a compatible payload format, such as JSON or XML. Otherwise the *Device* communicates over a *Gateway* with the *IoT Integration Middleware*. Thus, from a functional point of view, it serves as an integration layer for different kinds of *Sensors*, *Actuators*, *Devices*, and *Applications*.

The *IoT Integration Middleware* is not limited to the functionality described above. It may comprise all kinds of functionality that is required by a certain cyber-physical system, for instance, a rules engine or graphical dashboards. Additionally, the device and user management as well as the aggregation and utilization of received data may be performed inside this component. Typically, an *IoT Integration Middleware* can be accessed using APIs, e.g., HTTP-based REST APIs.

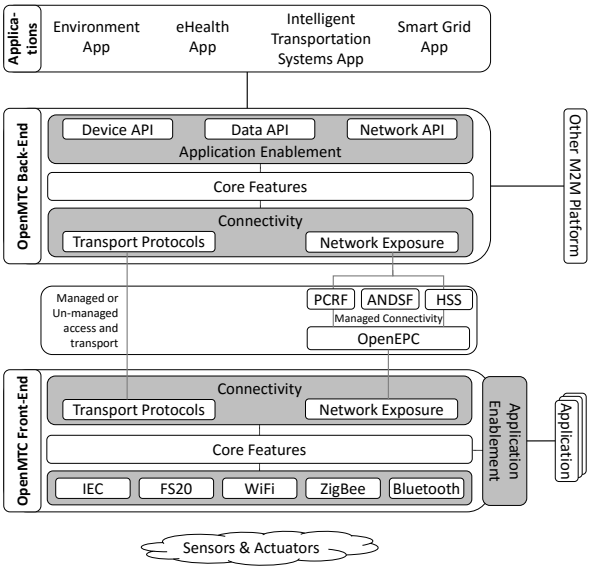


Fig. 2. OpenMTC architecture based on [6]

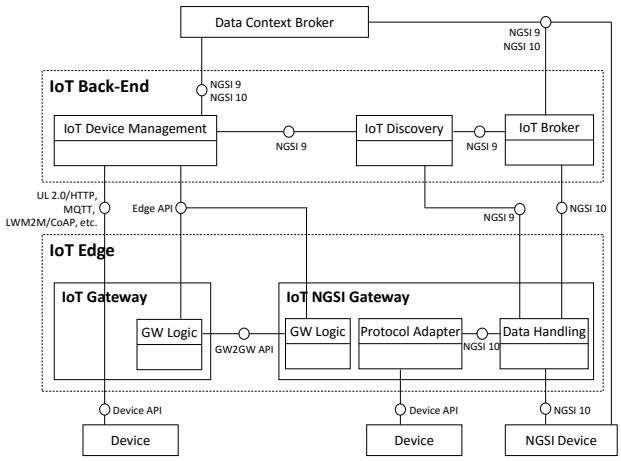


Fig. 3. FIWARE architecture based on [7]

F. Application

The *Application* component represents software that uses the *IoT Integration Middleware* to gain insight into the physical environment by requesting *Sensor* data or to control physical actions using *Actuators*. For example, a software system that controls the temperature of a building represents an *Application* connected to an *IoT Integration Middleware*. An *Application* in this reference architecture can also be another *IoT Integration Middleware*, for example, to integrate multiple systems.

III. Comparison of the IoT Platform Architectures

We compare our IoT reference architecture to three open-source platforms and one proprietary IoT platform. Throughout the mapping, the different naming of the components as well as their provided functionality have been considered. The detailed comparison of all technologies is discussed by Guth [5]. In accordance with the extent of this paper, the comparison and the major differences of the open-source platforms OpenMTC[4], FIWARE[2], and SiteWhere[5], and the proprietary solution of Amazon Web Services[3] are summarized in the following.

A. OpenMTC

OpenMTC implements an open-source, cloud-enabled IoT platform. Considering the architecture shown in Figure 2, the OpenMTC platform is divided into the following building blocks: the Front- and Back-End as well as the Sensors & Actuators beneath the Front-End, the connectivity between the Front- and Back-End, Applications positioned on top of the Back-End and on the right side of the Front-End, and a component to connect other M2M Platforms to the Back-End. Corresponding to the documentation of the OpenMTC platform, the Sensors & Actuators comprise not only *Sensors* and *Actuators* of our reference architecture, but also *Devices*. Furthermore, the *Devices* component of our IoT reference architecture includes the lowest part of the OpenMTC Front-End, which represents the communication technologies connecting the *Devices* to the platform. Thus, the components *Sensor*, *Actuator*, and *Device* of our reference architecture are partly overlapping when mapped onto the OpenMTC architecture. The remaining OpenMTC Front-End parts, namely Core Features and Connectivity, as well as the components of the gap between the Front- and Back-End build the functionality to translate the messages from the *Devices* to the middleware and vice versa. Hence, those parts are encompassed by the *Gateway* of our reference architecture. The OpenEPC component in the gap between the Front- and Back-End already provides functionality, such as filtering and applying rules. Accordingly, this component is covered by the *IoT Integration Middleware* as well. Furthermore, the OpenMTC Back-End components Connectivity, Core Features, and partly the Application Enablement are comprised by the *IoT Integration Middleware* of our reference architecture since they provide the core logic of the platform. More detailed, the Connectivity component is responsible for the Device Management, the Core Features component provides all further functionality of the platform, and the Application Enablement manages the connection to *Applications*. Both Application Enablement components, and both Application components of the OpenMTC Back- and Front-End, as well as the Other M2M Platform component are encompassed by the *Application* component of our IoT reference architecture. They represent all possibly connected further *Applications*.

Regarding the OpenMTC platform, each component of our IoT reference architecture is represented. Some of the components are partly overlapping, which is appropriate to the abstract definition of our IoT reference architecture following the explanations in Section II.

[4]http://www.open-mtc.org/
[5]http://www.sitewhere.org/

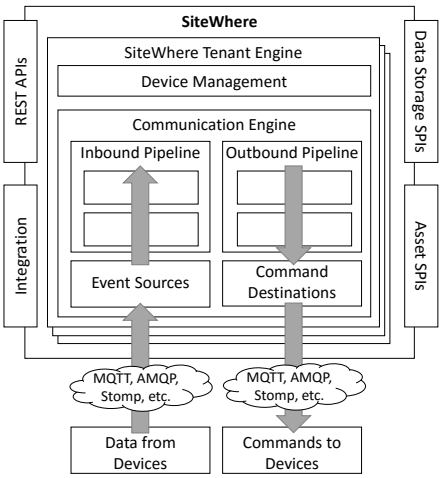


Fig. 4. SiteWhere architecture based on [8]

B. FIWARE

FIWARE is an open-source, cloud-based infrastructure for IoT platforms funded by the European Union and the European Commission. It is an enhanced OpenStack-based[6] cloud, which hosts capabilities and the FIWARE Catalogue, containing a rich library of components called Generic Enablers (GEs). The GEs of the IoT part are shown in Figure 3, spread over the IoT Edge and the IoT Back-End. Furthermore, the Devices are located below the IoT Edge and the Data Context Broker is positioned on top of the IoT Back-End. FIWARE follows the approach to represent only *Devices*, which have integrated *Sensors* and *Actuators*, and they further separate NGSI[7]-capable devices. Accordingly, the *Sensor*, *Actuator*, and *Device* components of our reference architecture are partly overlapping and comprise the Device components of the FIWARE architecture. The IoT Edge further contains the IoT Gateway and the IoT NGSI Gateway, which are both responsible for establishing and managing the communication between the devices and the IoT Back-End. Hence, the IoT Edge is encompassed by the *Gateway* of our reference architecture. The core functionality of the platform is located within the IoT Back-End and the Data Context Broker, which are consequently comprised by our *IoT Integration Middleware*. Our *Application* component is not represented within the figure of the architecture, but FIWARE also enables the connection of *Applications* through the Data Context Broker. Thus, our *Application* component is likewise covered.

Considering FIWARE, our IoT reference architecture can be mapped onto it and each component is covered. Like before, the *Sensor*, *Actuator*, and *Device* components are partly overlapping, which is appropriate to our definition.

[6]https://www.openstack.org/

[7]The Open Mobile Alliance defines the standard of Next Generation Service Interfaces (NGSI) [9]. NGSI are context management function specifications of the NGSI Enabler, which provides access to information about Context Entities through interfaces.

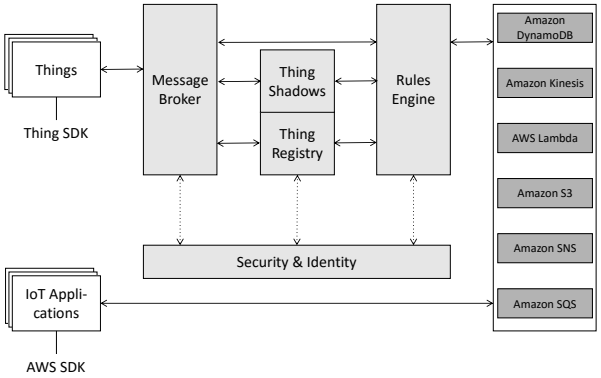


Fig. 5. AWS IoT architecture based on [10]

C. SiteWhere

SiteWhere is an open-source IoT platform. Its architecture is shown in Figure 4. It is composed of a core element, where devices and further *Applications* can be connected to. Since SiteWhere does not divide the device component more precisely, it is comprised by our *Sensor*, *Actuator*, and *Device* components. The concept of a *Gateway* is not represented within a particular component, but it is located between the Devices and the SiteWhere core element [8]. The core element consists of the Tenant Engine encapsulating the Communication Engine, which ensures the internal event handling. Consequently, it is encompassed by the *Gateway* of our reference architecture. Our *IoT Integration Middleware* covers the Tenant Engine, where the core functionality of the platform is embedded. Additionally connected to the SiteWhere core are the Integration component, REST APIs, Asset SPIs, and Data Storage SPIs, which enable the connection of further systems and *Applications* to the platform.

Regarding SiteWhere, our IoT reference architecture covers each component of the architecture. As described above, the *Sensor*, *Actuator*, and *Device* components are overlapping, since SiteWhere does not further distinguish between them. Nevertheless, this is appropriate to our definition.

D. AWS IoT

Amazon Web Services IoT (AWS IoT) is a managed cloud platform for the IoT, its architecture is shown in Figure 5. Noticeably, they do not have a *Device* component since AWS uses the idea of Things. The term Things is used as a synonym for *Devices*, which can have integrated *Sensors* and *Actuators*. Following this, the Things component of the AWS IoT architecture is comprised by the *Sensor*, *Actuator*, and *Device* components of our reference architecture. The *Gateway* component of our IoT reference architecture is not represented, but located between the Things and the Message Broker [10]. The core logic of the platform is located within the Message Broker, Thing Registry, Thing Shadows, Rules Engine, Security & Identity, and partly the Message Broker, and hence, they are encompassed by the *IoT Integration Middleware*. Since AWS is a cloud service provider, multiple data processing

services are already integrated. Likewise, the IoT Applications component enables the connection of further *Applications* to the platform.

Regarding the AWS IoT platform, each component of our IoT reference architecture is represented. Again, the definition of the *Device* component differs from the ones described above, but it is also appropriate to our definition of the components.

E. Summary of the Comparison

Our IoT reference architecture can be mapped onto each considered platform. Consequently, each component of our IoT reference architecture is represented in each investigated platform. One major difference is that each platform uses the term "device" in a different way since the granularity of the device components differs strongly. FIWARE and SiteWhere mention *Sensors* and *Actuators* only within their documentation, and AWS IoT does not separate between *Sensors*, *Actuators*, and *Devices* at all. Furthermore, OpenMTC, FIWARE, and AWS IoT use the device term even for smart devices, where they have already some kind of logic integrated and assume partly the functionality of our *Gateway*.

Noticeably, each IoT platform uses the approach of our *Gateway* slightly different: OpenMTC and FIWARE already integrate a possibility to filter the incoming data, whereby the remaining solutions comply with our definition. In accordance to that, the comparison of our *IoT Integration Middleware* to OpenMTC and FIWARE showed that it is shifted over the *Gateway*. Additionally, the *Application* components of the considered solutions demonstrate that each of them provide the possibility to connect further applications to the platform. AWS IoT already provides additional integrated *Applications* since AWS is a cloud service provider.

IV. Related Work

This section presents work related to our IoT reference architecture. Therefore, IoT architectures, architecture reference models, domain models, and taxonomies are considered.

Bauer et al. [11] introduce an IoT reference architecture describing seven functional components between a device and an application layer: the Management, Service Organisation, IoT Process Management, Virtual Entity, IoT Service, Security, and Communication. Besides the Communication component, which can be mapped onto our *Gateway*, the remaining components build our *IoT Integration Middleware*. The *Device* and *Application* components are not defined in particular. Since our approach was to provide an abstract reference architecture and a definition of all components, this approach leads to a detailed reference architecture and, thereby, focusses on the middleware.

The IoT reference architecture introduced by Fremantle [4] contains five layers. The device layer corresponds with our *Devices*, but it is not further divided into *Sensors* and *Actuators*. The relevant transports layer is equal to our *Gateway*. The aggregation/bus layer and the event processing and analytics layer provide the core functionality of an IoT platform. Hence, they correspond to our *IoT Integration Middleware*. The client/external communications provide further *Applications*. Clearly, the discussed reference architecture corresponds with ours, but it does not provide an unambiguous definition of all components. As a result, it does not pursue our goal to provide an abstract terminology and basis for the comparison of diverse IoT platforms.

Cisco introduces a seven-layered IoT Reference Model [3]. The Physical Devices and Controllers correspond with our *Devices*, *Sensors*, and *Actuators* since Cisco does not differ between those components. The Connectivity layer is equal to our *Gateway*. The Edge (Fog) Computing, Data Accumulation, and Data Abstraction layer represent our *IoT Integration Middleware*. The Application layer partly corresponds with our *IoT Integration Middleware* and our *Application* component. Furthermore, the Collaboration and Processes correspond to our *Applications*. Again, our IoT reference architecture can be mapped onto the discussed reference model. Nevertheless, Cisco's reference model does not focus on the definition of the components and is, therefore, not unambiguous, which is required to support the comparison of diverse IoT platforms.

Zheng et al. [12] introduce a three-layer architecture containing similar concepts as those outlined in our reference architecture. This work is used in diverse other works by Wu et al. [13], Atzori et al. [14], and Aazam et al. [15]. The Perception Layer represents the connection point to the physical world and is responsible, e.g., for gathering the information and for collaboration. This layer corresponds with our *Sensors*, *Actuators*, and *Devices*. The Network Layer takes care of transmitting and pre-processing the gathered data, which is covered by our *Device* and *Gateway*. The Application Layer provides the core functionality of the platform. Thus, it represents our *IoT Integration Middleware* and *Applications*. There are further approaches of layered architectures based on service-oriented architectures introduced by Atzori et al. [14] [16] and Xu et al. [17]. The review of those approaches shows that they do provide a basis for the architectural design, but they do not introduce a common definition or naming of the components. Consequently, both approaches do not pursue our goal to provide an abstract terminology as basis for the comparison of IoT platforms. In addition, one major contribution of our work is the mapping of the reference architecture to existing technologies to support the understanding of those.

Kim et al. [18] introduce a platform model derived out of diverse applications. The Things (*Devices*) are connected through a Gateway or directly to the Platform, and the Platform is connected to Service and Software Providers and to the Service User. Both connections outgoing from the platform are through a RESTful API. Furthermore, the Service User can communicate directly with a Thing. Besides the user, all components of this model are covered by our reference architecture. As above, this approach does not introduce a definition or uniform naming of the contained components.

The IoT Domain Model introduced by Haller et al. [19] builds the basis for an IoT Reference Model discussed by Krčo et al. [20]. Haller et al. introduce five concepts:

Augmented Entity, User, Device, Resource, and Service. Even though the definition of those components is given, it is not detailed enough for comparing different IoT platforms. For instance, a device is a hardware component, which is responsible for monitoring and interacting with real-world objects. Hence, sensors and/or actuators are already integrated or connected to the device. Furthermore, a device can provide the connectivity to IT systems. Since the definition is imprecise, it is unclear if the device can act as a gateway or communicates directly with the platform. Hence, this approach is not pursuing our goal of an unambiguous reference architecture.

Gubbi et al. [21] define an high-level taxonomy for the components of an IoT platform containing three components: (i) the hardware, which covers sensors, actuators, and embedded communication hardware, (ii) the middleware, which covers on-demand storage and computing tools for data analytics, and (iii) the presentation, which provides visualization and interpretation tools. Clearly, this taxonomy is applicable to our reference architecture as well, but it is not detailed enough to pursue our goal. Due to the lack of specification of the components, they can be interpreted diversely. For instance, an interpretation tool, which is categorized into the presentation component, can also be understood as a computing tool for data analytics, which is part of the middleware component.

V. Conclusions & Future Work

IoT platforms are gaining increasing attention. However, due to a missing clear definition of the components within an IoT platform, we introduced an unambiguous IoT reference architecture. In contrast to existing reference architectures, the architecture presented in this paper is more abstract to enable a uniform terminology and to ease the comparison of platforms. Within our reference architecture, each component as well as the communication between them is defined abstractly. Depending on the circumstances, several components can be combined. For instance, a Smart Phone represents a *Device* with integrated *Sensors* and *Actuators*. From a Smart Watch's perspective, a Smart Phone can also comprise its *Gateway* if the watch cannot communicate directly with the *IoT Integration Middleware*.

We compared our IoT reference architecture to three open-source and one proprietary IoT platforms. Respective to the mappings described in Section III, our IoT reference architecture can be mapped onto each considered IoT solution. The consideration of multiple platforms showed that the definition of the components of the architectures contain synonyms, homonyms, and that they differ strongly within the granularity of their components. Our unambiguous reference architecture maps to them and, therefore, cleared the understanding of the IoT platforms' components. Our IoT reference architecture can be used as a basis for the comparison and evaluation of different IoT solutions. It may ease the selection process and provides a common basis for the design of a new IoT platform.

Future works could present a more detailed and technical description of each component including, for instance, a definition of the cardinalities or communication interfaces of the reference architecture's components.

Acknowledgments

The research leading to these results has received funding from the German government through the BMWi projects NEMAR (03ET4018) and SmartOrchestra (01MD16001F).

References

[1] Khan, R. et al., "Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges," in *Proceedings of the 10th International Conference on Frontiers of Information Technology*. IEEE, Dec. 2012.

[2] Salim, F. and Haque, U., "Urban computing in the wild: A survey on large scale participation and citizen engagement with ubiquitous computing, cyber physical systems, and Internet of Things," *International Journal of Human-Computer Studies*, vol. 81, Sep. 2015.

[3] Cisco, "The Internet of Things Reference Model," 2014. [Online]. Available: http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf

[4] Fremantle, P., "A Reference Architecture for the Internet of Things," 2015. [Online]. Available: http://wso2.com/wso2_resources/wso2_whitepaper_a-reference-architecture-for-the-internet-of-things.pdf

[5] J. Guth, "Architectural Design of an Abstraction Layer for the Integration of Heterogeneous Cyber-Physical Systems," Master's thesis, University of Stuttgart, Mar. 2016. [Online]. Available: http://www.iaas.uni-stuttgart.de/institut/mitarbeiter/guth/MA_Guth_JA.pdf

[6] Fraunhofer FOKUS, "OpenMTC Platform Architecture," 2016. [Online]. Available: http://www.open-mtc.org/index.html#MainFeatures

[7] FIWARE, "FIWARE Wiki," 2016. [Online]. Available: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Main_Page

[8] SiteWhere LLC., "SiteWhere System Architecture," 2016. [Online]. Available: http://documentation.sitewhere.org/architecture.html

[9] Open Mobile Alliance Ltd., "NGSI Context Management," May 2012. [Online]. Available: http://technical.openmobilealliance.org/Technical/release_program/docs/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf

[10] Amazon Web Services, "AWS IoT Documentation," 2016. [Online]. Available: https://aws.amazon.com/de/documentation/iot/

[11] Bauer, M. et al., "IoT Reference Architecture," in *Enabling Things To Talk: Designing IoT solustions with the IoT Architectural Reference Model*. Springer Berlin Heidelberg, 2013.

[12] Zheng, L. et al., "Technologies, Applications, and Governance in the Internet and of Things," in *Internet of Things - Global Technological and Societal Trends*. River Publishers, 2009.

[13] Wu, M. et al., "Research on the architecture of Internet of things," in *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*. IEEE, 2010.

[14] Atzori, L. et al., "The Social Internet of Things (SIoT) – When social networks meet the Internet of Things: Concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, Nov. 2012.

[15] Aazam, M. et al., "Cloud of Things: Integrating Internet of Things and Cloud Computing and the Issues Involved," in *International Bhurban Conference on Applied Sciences and Technology*. IEEE, 2014.

[16] Atzori, L. et al., "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, Oct. 2010.

[17] Xu, L. et al., "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, Nov. 2014.

[18] Kim, J. et al., "M2M Service Platforms: Survey, Issues, and Enabling Technologies," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, 2014.

[19] Haller, S. et al., "A Domain Model for the Internet of Things," in *Proceedings of the IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*. IEEE, Aug. 2013.

[20] Krčo, S. et al., "Designing IoT and Architecture(s)," in *Proceedings of the IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, 2014.

[21] Gubbi, J. et al., "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, Sep. 2013.

All links last followed on October 23, 2016.