Institute of Architecture of Application Systems

A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences

Jasmin Guth[1], Uwe Breitenbücher[1], Michael Falkenthal[1], Paul Fremantle[2], Oliver Kopp[3], Frank Leymann[1], Lukas Reinfurt[1]

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

[2]School of Computing, University of Portsmouth, UK
paul.fremantle@port.ac.uk

[3]Institute of Parallel and Distributed Systems, University of Stuttgart, Germany
kopp@ipvs.uni-stuttgart.de

BibTeX

```
@inbook{Guth2018_AnalysisIoTPlatformArchitectures,
  author    = {Guth, Jasmin and Breitenb{\"u}cher, Uwe and Falkenthal, Michael
               and Fremantle, Paul and Kopp, Oliver and Leymann, Frank and
               Reinfurt, Lukas},
  editor    = {Di Martino, Beniamino and Li, Kuan-Ching and Yang, Laurence T.
               and Esposito, Antonio},
  title     = {A Detailed Analysis of IoT Platform Architectures: Concepts,
               Similarities, and Differences},
  booktitle = {Internet of Everything: Algorithms, Methodologies, Technologies
               and Perspectives},
  year      = {2018},
  pages     = {81--101},
  doi       = {10.1007/978-981-10-5861-5_4},
  publisher = {Springer}
}
```


University of Stuttgart
Germany

A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences

Jasmin Guth, Uwe Breitenbücher, Michael Falkenthal, Paul Fremantle, Oliver Kopp, Frank Leymann, and Lukas Reinfurt

Abstract The IoT is gaining increasing attention. The overall aim is to interconnect the physical with the digital world. Therefore, the physical world is measured by sensors and translated into processible data, and data has to be translated into commands to be executed by actuators. Due to the growing interest in IoT, the number of platforms designed to support IoT has risen considerably. As a result of different approaches, standards, and use cases, there is a wide variety and heterogeneity of IoT platforms. This leads to difficulties in comprehending, selecting, and using appropriate platforms. In this work, we tackle these issues by conducting a detailed analysis of several state-of-the-art IoT platforms in order to foster the understanding of the

Jasmin Guth
Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany,
e-mail: guth@iaas.uni-stuttgart.de

Uwe Breitenbücher
Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany,
e-mail: breitenbuecher@iaas.uni-stuttgart.de

Michael Falkenthal
Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany,
e-mail: falkenthal@iaas.uni-stuttgart.de

Paul Fremantle
School of Computing, University of Portsmouth, Portsmouth, UK,
e-mail: paul.fremantle@port.ac.uk

Oliver Kopp
Institute for Parallel and Distributed Systems, University of Stuttgart, Stuttgart, Germany,
e-mail: kopp@ipvs.uni-stuttgart.de

Frank Leymann
Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany,
e-mail: leymann@iaas.uni-stuttgart.de

Lukas Reinfurt
Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany,
e-mail: reinfurt@iaas.uni-stuttgart.de

(i) underlying concepts, (ii) similarities, and (iii) differences between them. We show that the various components of the different platforms can be mapped to an abstract reference architecture, and analyze the effectiveness of this mapping.

1 Introduction

The vision of the Internet of Things (IoT) describes a future where many everyday objects are interconnected through a global network. They collect and share data of themselves and their surroundings to allow widespread monitoring, analyzation, optimization, and control [27]. Until recently this was merely a vision, but in recent years this has slowly developed into a reality. Ever decreasing prices, dimensions, and energy requirements of electronics now allow tiny devices to unobtrusively measure their surroundings. Many devices use low-energy communication technology to send those measurements to other, more powerful components, such as bluetooth gateways, mobile phones, or WiFi hotspots. Devices are increasingly incorporating long-range wireless technologies such as LoRa[1] or existing 2G and 3G networks. Local edge processors, hubs, or internet services in turn analyze and process IoT sensor data to create new knowledge, which can be used to act back on the environment through actuators. In short, the IoT can be seen as a giant cyber-physical control loop. In that context, the term "Machine-to-Machine communication" (M2M [9]) is often used to describe such a setting.

Different incarnations of IoT systems for varying use cases have been created over the years by companies and research institutions. Smart Homes are one example of such IoT systems [30]. In other areas, similar developments are underway, such as Connected Cars [20], Smart Cities [31], Demand Side Management [22], Smart Grids [11], or Smart Factory systems [24].

While local processing of the data generated by these systems is possible and a reasonable approach for use cases where low latency is required, cloud based platforms are used for processing and analyzing larger data sets [7]. As a result, over one hundred [29] such platforms have been created over the last few years. Some examples include AWS IoT[2], FIWARE[3], OpenMTC[4], and SmartThings[5].

These platforms come in various shapes and sizes. While standardization efforts are ongoing, there are no generally agreed-on standards for IoT at this time [8]. Rather, development of these platforms has often taken place in silos [38]. These different environments have influenced not only the choice of concepts and technology, but also the choice of terminology. As a result, the platform landscape has become very heterogeneous. At the same time, however, all these solution do roughly the same

[1] https://www.lora-alliance.org/

[2] https://aws.amazon.com/en/iot/

[3] https://www.fiware.org/

[4] http://www.openmtc.org/

[5] http://www.smartthings.com/

things: they allow connecting different devices, accessing and processing their data, and using the knowledge gained through this activity to create automated control.

The heterogeneity of these approaches creates an issue for someone who has to select one of these solutions. Finding the right platform for a use case becomes very time consuming when each solution uses different technologies and terminology. You have to read and understand the descriptions and documentation of each platform to make a decision. This requires not only time, but also the technical knowledge to be able to understand and compare the different concepts.

A reference architecture which maps to existing architecture descriptions and offers a unified terminology helps in this selection process. Not only does it allow for easier comparison between platforms, but it also provides a useful framework as a starting point for new developments. Therefore, we define in the next section an IoT reference architecture based on existing platforms. The IoT reference architecture is kept purposely abstract to make it applicable in a wide range of situations. It was first introduced in our former work by Guth et al. [17]. The work at hand extends that previous work by a refinement of the description, the extension of the analysis to eight IoT platforms, and a more extensive survey on related work.

The remainder of this paper is structured as follows: In Sect. 2 we present our reference architecture. We describe the different components and the possible ways they communicate. In Sect. 3 we compare our architecture against eight existing platforms to show that it is generally applicable. To deepen and clarify the differences between the considered platforms we extended our previous work by describing the mapping of our reference architecture onto more IoT solutions. We also provide a summarized comparison, illustrated within Table 1. In Sect. 4 we investigate the differences of our reference architecture to other existing approaches. Finally, we summarize, conclude, and outline possible future work in Sect. 5.

2 Reference Architecture

This section presents an IoT reference architecture (Fig. 1) which (i) offers a unified terminology and (ii) maps to existing architecture descriptions. We start by defining all components shown in Fig. 1 starting from the bottom. To clearly distinguish between the concepts presented in this work and similar or equal concepts presented in the considered platforms and related work, we accentuate the elements of the IoT reference architecture presented in this work by italics.

2.1 Sensor

A *Sensor* is a hardware component which captures information on the physical environment by "respond[ing] to a physical stimulus (as heat, light, sound, pressure, magnetism, or a particular motion)" [25]. For instance, by measuring the humidity

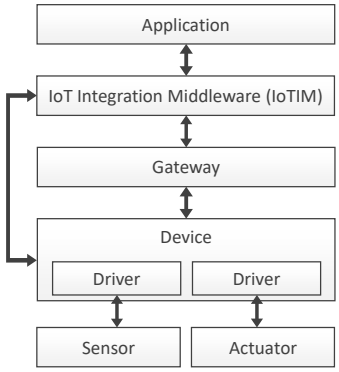

Fig. 1 IoT reference architecture based on [17]

within a room, a *Sensor* positioned within that room captures the humidity level of the room. *Sensors* transmit the captured information using electrical signals to *Devices*, to which they are connected to. This connection can be established (i) by wire or (ii) wirelessly. Wired connection includes an integration of *Sensors* into a *Device*. A *Sensor* may be configured using software, but cannot run software by itself.

2.2 Actuator

An *Actuator* is a hardware component which manipulates the physical environment. *Actuators* receive commands from their connected *Device* and translate these electrical signals into some kind of physical action. For instance, an *Actuator* turning on or off a ventilation within a room acts on the physical environment by influencing the humidity of the room. Similar to *Sensors*, the connection to the device can be established (i) by wire or (ii) wirelessly, whereby a wired connection includes the integration into a *Device*. Furthermore, an *Actuator* may be configured using software, but cannot run software by itself.

2.3 Device

A *Device* is a hardware component which (i) is connected to *Sensors* and/or *Actuators* by wire or wirelessly or (ii) even integrates these components. *Devices* have a processor and storage capacity to run software and to establish a connection to the *IoT Integration Middleware*. For instance, the outdoor module of the Netatmo weather station[6] represents a *Device* with integrated *Sensors*. Thus, *Devices* are the entry point of the physical environment to the digital world. A *Driver* is software running

[6] `https://www.netatmo.com/product/weather/weatherstation`

on the *Device* enabling uniform access to heterogeneous *Sensors* and *Actuators*. *Devices* are either (i) self-contained or (ii) connected to another, bigger system. The *IoT Integration Middleware* represents such a system.

2.4 Gateway

In case a *Device* is not capable of directly connecting to further systems, it is connected to a *Gateway*. A *Gateway* provides required technologies and mechanisms to translate between different protocols, communication technologies, and payload formats. It forwards communication between *Devices* and further systems. For instance, the indoor module of the Netatmo weather station[6] is a *Device* with integrated *Sensors*, acting as a *Gateway* for the outdoor module of the Netatmo weather station. When the *Gateway* receives a message in a proprietary binary format from the *Device*, it translates the binary format into a more common format, such as JSON, and forwards the data to the intended system over IP, for example. If necessary, the *Gateway* may likewise translate commands sent from systems to *Devices* into communication technologies, protocols, and formats supported by the respective *Device*.

2.5 IoT Integration Middleware

The *IoT Integration Middleware* (IoTIM) serves as an integration layer for different kinds of *Sensors*, *Actuators*, *Devices*, and *Applications*. It is responsible for (i) receiving data from the connected *Devices*, (ii) processing the received data, (iii) providing the received data to connected *Applications*, and (iv) controlling *Devices*. An example for processing is to evaluate condition-action rules and sending commands to *Actuators* based on this evaluation. A *Device* can communicate directly with the *IoT Integration Middleware* if it supports an appropriate communication technology, such as IP over Ethernet or WiFi, a corresponding transport protocol, such as HTTP or MQTT, and a compatible payload format, like, e.g., JSON. Otherwise, the *Device* communicates over a *Gateway* with the *IoT Integration Middleware*. The *IoT Integration Middleware* is not limited to the functionality described above. It may comprise all kinds of functionality that are required by a certain cyber-physical system, for instance, a time-series database or graphical dashboards. Additionally, the management of *Devices* and users as well as the aggregation and utilization of received data may be performed. For instance, the SmartThings[5] platform can be used with the Netatmo *Devices*. Typically, an *IoT Integration Middleware* can be accessed using APIs, for example, HTTP-based REST APIs.

2.6 Application

The *Application* component represents software which uses the *IoT Integration Middleware* (i) to gain insight into the physical environment and/or (ii) to manipulate the physical world. It does so by requesting *Sensor* data or by controlling physical actions using *Actuators*. For instance, a software system that controls the temperature of a building represents an *Application* connected to an *IoT Integration Middleware*. An *Application* can also be another *IoT Integration Middleware*.

2.7 Summary

This section presented the reference architecture consisting of six component types. When implementing the architecture, components can be omitted. This might be the case, if the platform is only used to measure changes of the physical world. For instance, a platform gathering the CO2 level within the air, may have no *Actuators* connected, if the system is only used to measure and collect the data. Another example for omitted components are platforms with connected *Devices* capable of the required technologies to communicate directly with the *IoT Integration Middleware*, so no *Gateway* is needed for an appropriate message exchange.

3 Comparison of IoT Platforms

In this section, the IoT reference architecture is mapped onto four open-source platforms and four proprietary IoT solutions. The selected open-source platforms are FIWARE[3], OpenMTC[7], SiteWhere[8], and Webinos[9]. The proprietary solutions are AWS IoT[2], IBM's Watson IoT Platform[10], Microsoft's Azure IoT Hub[11], and Samsung's SmartThings[5]. During the comparison the component's functionality and their naming are the key areas that are compared with the reference architecture. The comparison of each platform will be described in detail. Additionally, a composite comparison is presented, where the main aspects are discussed. This section extends our previous analysis presented by Guth et al. [17] by a detailed analysis of four more platforms: Webinos, IBM's Watson IoT Platform, Microsoft's Azure IoT Hub, and Samsung's SmartThings. Moreover, we refined our previous analysis of FIWARE, OpenMTC, SiteWhere, and AWS IoT [17] in terms of a more detailed reference architecture mapping. In addition, we added a detailed comparison table of the platforms in Sect. 3.9.

[7] http://www.open-mtc.org/

[8] http://www.sitewhere.org/

[9] http://www.webinos.org/

[10] http://www.ibm.com/internet-of-things/

[11] https://azure.microsoft.com/de-de/suites/iot-suite/

3.1 FIWARE

The architecture and the mapping of the open-source platform FIWARE[3] onto our IoT reference architecture is shown in Fig. 2. FIWARE is funded by the European Union and the European Commission. It provides an enhanced OpenStack-based[12] cloud, where its capabilities and Catalogue is hosted. The FIWARE Catalogue contains Generic Enablers (GEs), representing a rich library of components. Within the architecture in Fig. 2, only the GEs of the IoT part are shown. FIWARE only distinguishes between Devices and NGSI[13] Devices. Since the FIWARE documentation describes that devices may have integrated sensors and actuators, all device components are comprised by our *Device*, *Sensor*, and *Actuator* components. Devices can communicate directly with the IoT Back-End or via a Gateway, which is positioned within the IoT Edge. Both the IoT Gateway and the IoT NGSI Gateway enable and manage the communication of devices with the IoT Back-End. Consequently, the IoT Edge represents the *Gateway* of our IoT reference architecture. The IoT Back-End and the

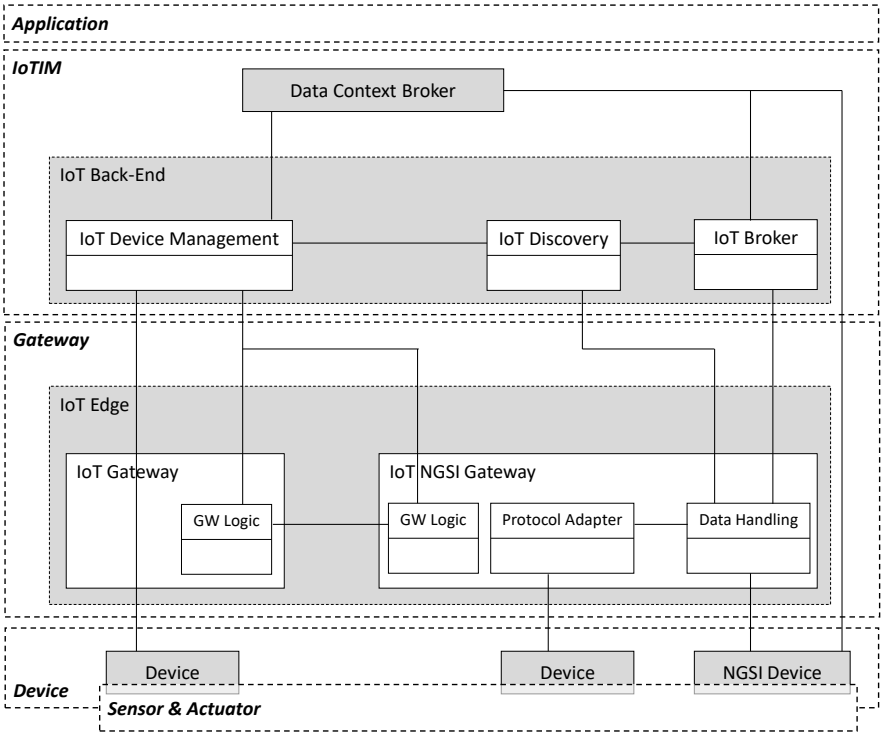

Fig. 2 FIWARE architecture based on [12]

[12] `https://www.openstack.org/`

[13] Next Generation Service Interfaces [28]

Data Context Broker provide the main functionality of FIWARE, and therefore they are encapsulated by our *IoT Integration Middleware*. The documentation of FIWARE describes how further applications can be connected through the Data Context Broker to the platform. Although the application component is not represented in the FIWARE architecture diagram, based on this description we therefore position our *Application* component on top of the Data Context Broker.

In regards to FIWARE, our IoT reference architecture covers each component of the architecture. As described above, the definition of the device component differs from ours and, hence, the *Sensor*, *Actuator*, and *Device* components partly overlap. Nevertheless, there is still an appropriate mapping to our definition.

3.2 OpenMTC

Fig. 3 shows the architecture of OpenMTC[7], which is an open-source, cloud-enabled IoT platform, and its comparison against our IoT reference architecture. OpenMTC is composed of the following components: The Front- and Back-End, the Sensors & Actuators component 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, as well as a component to connect Other M2M Platforms to the Back-End. Obviously, the Sensors & Actuators component represents our *Sensors* and *Actuators*. The documentation of OpenMTC further describes that the Sensors & Actuators components along with the lowest level of the Front-End, which enables communication, represent *Devices* equivalent to our reference architecture. The Core Features and Connectivity components of the Front-End, the connectivity between the Front- and Back-End, as well as the Connectivity component of the Back-End provide all required functionality to enable the communication between a device and the middleware, such as message translation. Consequently, those components represent our *Gateway*. Since the OpenEPC component (offering connectivity between the Front- and Back-End) provides further functionality, such as applying rules and filtering, it is encompassed by the *IoT Integration Middleware* as well. Additionally, the *IoT Integration Middleware* encapsulates the Connectivity, Core Features, and Application Enablement components of the OpenMTC Back-End. Those components provide the main functionality of the platform. The Application Enablement components provide all functionality to connect further applications to the middleware. Hence, the Applications and Other M2M Platform components are covered by the *Applications* component of our reference architecture.

Considering OpenMTC, its architecture can be mapped onto our IoT reference architecture. Nevertheless, the *Device*, *Sensor*, and *Actuator* components, as well as the *Gateway* and *IoT Integration Middleware* are partly overlapping, which is still appropriate to the definition of our reference architecture.

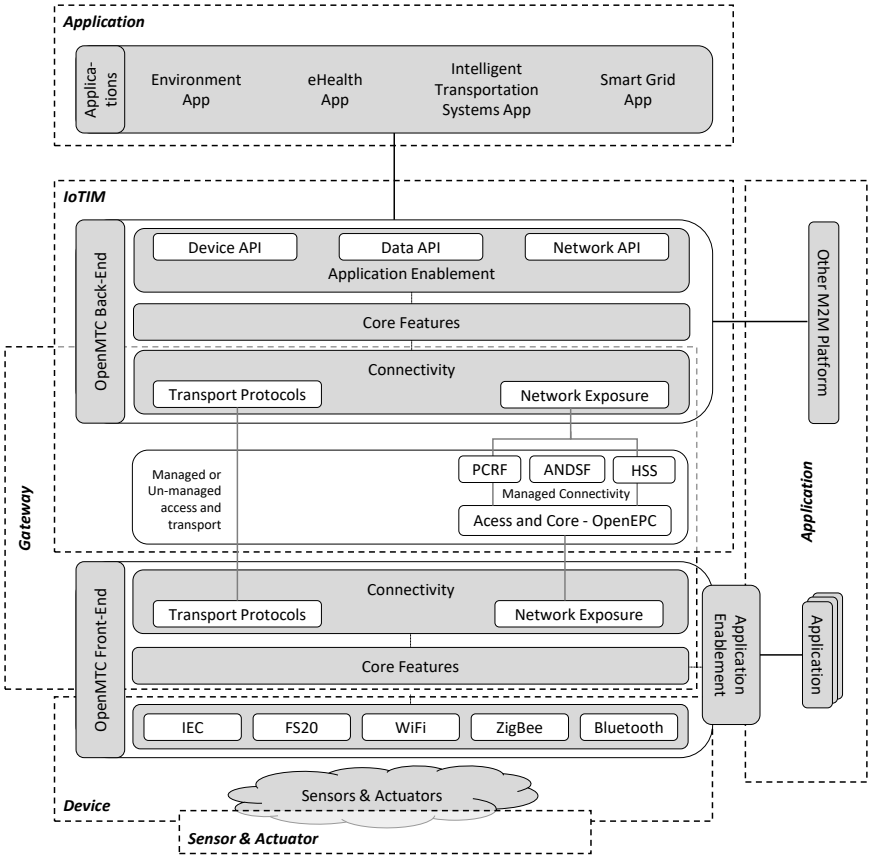

Fig. 3 OpenMTC architecture based on [13]

3.3 SiteWhere

Fig. 4 shows the architecture and the mapping of the open-source IoT platform SiteWhere[8] onto our reference architecture. The Data from Devices and Commands to Devices components are comprised by the *Device* component of our reference architecture. Furthermore, they also represent our *Sensor* and *Actuator* components, since they are not explicitly depicted within the architecture. As the devices can communicate via diverse protocols with the platform, the concept of a *Gateway* is present between the devices and the platform [32], but not pictured as a separate component. The main functionality of the platform is provided by the SiteWhereTenant Engine, including the Device Management and the Communication Engine. Consequently, those components are comprised by the *IoT Integration Middleware* of our reference architecture. The REST APIs and Integration component enables the connection of further *Applications* to the platform.

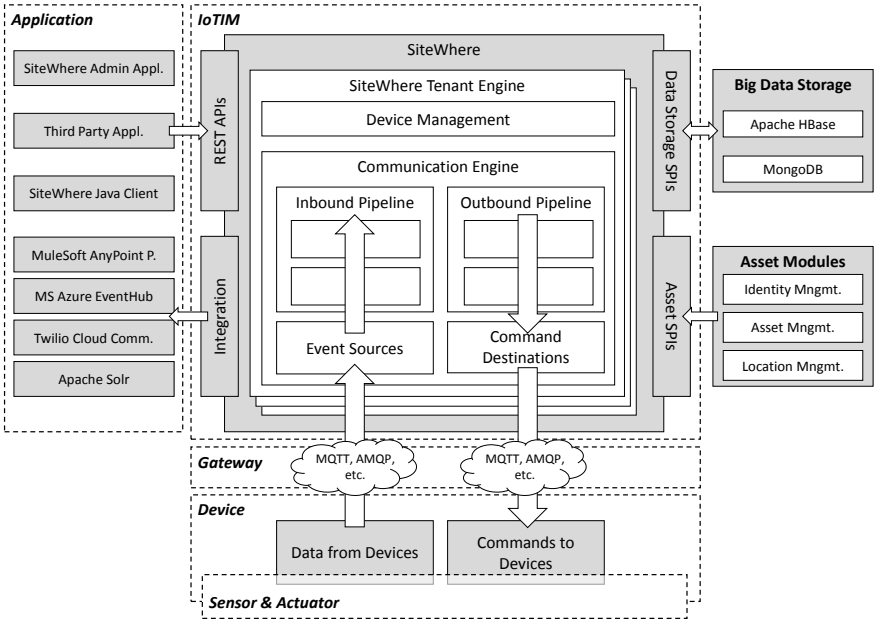

Fig. 4 SiteWhere architecture based on [32]

Considering the architecture of SiteWhere, each component of our IoT reference architecture is represented. Even though, some components are overlapping, our definition of the IoT reference architecture components still holds.

3.4 Webinos

The Webinos[9] middleware and architecture is an open-source middleware for the IoT and mobile devices, sponsored by the European Union FP7 project. The aim of Webinos is to provide a secure framework for personal devices to communicate and for individuals to publish data to third-parties and to other individuals. As such it takes a different approach to an IoT platform by being centered around the person. The mapping of the Webinos approach onto our IoT reference architecture is shown within Fig. 5. The main components of the Webinos architecture are the Personal Zone Hub (PZH), and the Personal Zone Proxy (PZP). The PZH provides the *Gateway*, where each *Device* connects to. The PZH also provides local communications between devices by acting as a messaging hub. In this regard it performs the functions of our *IoT Integration Middleware*. The PZH does not inherently support any *Applications* to run locally, but it provides the APIs that allow third-party applications to be built and to communicate with devices, which is a core function of the *IoT Integration*

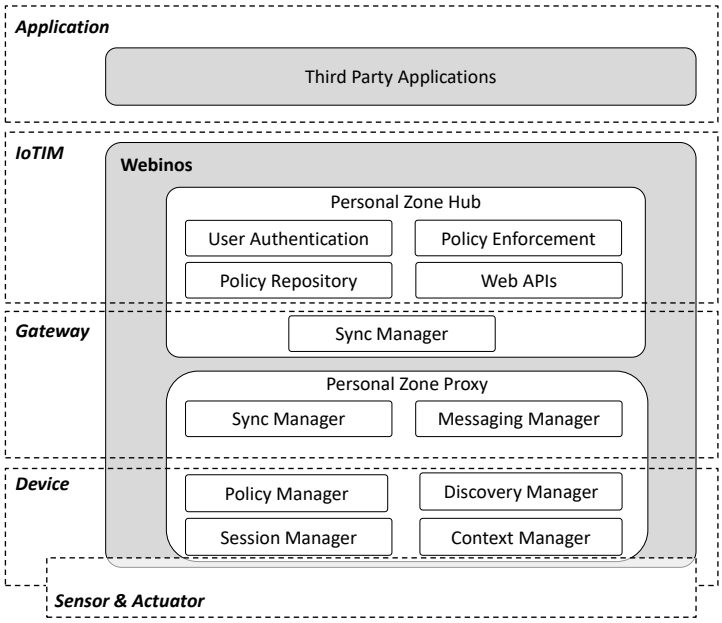

Fig. 5 Webinos Architecture based on [15, 34]

Middleware layer in our architecture. Each device runs a local component, the PZP, that aggregates sensor data and actuator commands and communicates with the PZH. One unique aspect of Webinos is that when multiple PZPs (on multiple devices) have connected to the same PZH, they can then communicate in a peer-to-peer fashion. The PZP and PZH sync to allow this to happen.

Each component of our IoT reference architecture is represented in the Webinos system. Because the PZH and PZP overlap in function, the separation into our reference architecture is more complicated. However, there is a clear mapping that certain components are performing *Gateway* and *IoT Integration Middleware* functions, as depicted in the diagram. As an example, because the PZPs can communicate in a peer-to-peer fashion without the PZH acting as an intermediary, we must assign some of the *Gateway* functionality to the PZP. Each PZP is deployed on a *Device*. Once again, the Webinos system does not explicitly call out the difference between *Devices*, *Sensors*, and *Actuators*, but there is full support for both sensors and actuators in Webinos and, therefore, it supports the reference architecture.

While the reference architecture does not explicitly deal with the person-centered approach of Webinos, we can clearly map each person's Webinos system to an individual instance of the reference architecture.

3.5 AWS IoT

Fig. 6 shows the architecture of Amazon Web Services IoT[2] (AWS IoT) and its mapping onto our reference architecture. AWS IoT is a managed cloud platform for the IoT, pursuing the concept of Things instead of devices. Since AWS uses Things synonymous to devices with integrated sensors and actuators, the Things component is covered by our *Device*, *Sensor*, and *Actuator* components. Furthermore, a *Gateway* component is not represented within the architecture, but following the documentation [2], it is located between the Things and Message Broker components. The Message Broker, Thing Shadows, Thing Registry, Rules Engine, and the Security & Identity components provide the main functionality of the platform. Hence, they represent the *IoT Integration Middleware* component of our IoT reference architecture. Our *Application* component encapsulates the already integrated data processing services, such as AWS Lambda or Amazon Kinesis, and, additionally, the IoT Applications component, which enables the connection of further applications.

Considering AWS IoT, each component of our IoT reference architecture is represented. Even though AWS follows a different concept of devices, it is still appropriate to our definition of the components.

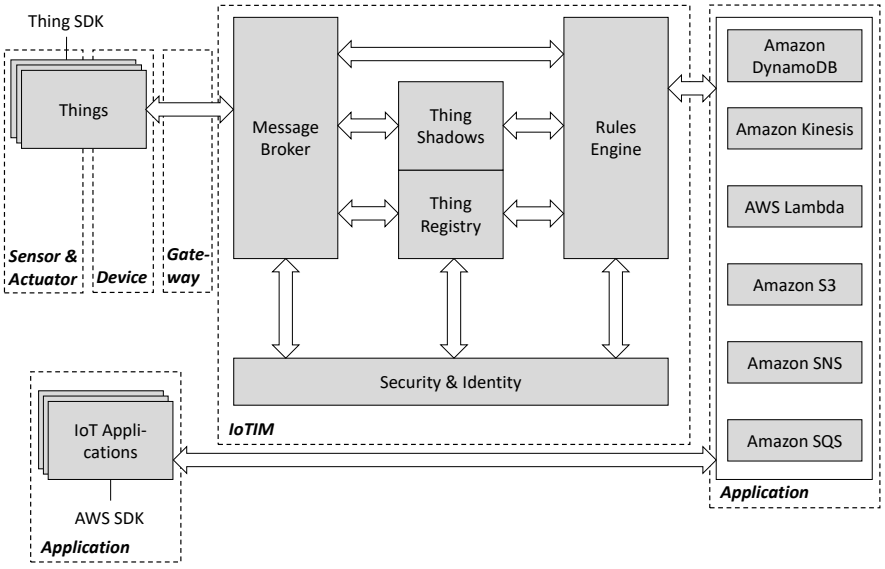

Fig. 6 AWS IoT architecture based on [2]

3.6 IBM Watson IoT Platform

Fig. 7 shows the architecture of IBM's cloud-based Watson IoT Platform[10]. Within the figure, *Devices*, *Sensors*, and *Actuators* are not represented. Since the Connect component takes care of the connection of *Devices* to the platform, the *Device*, *Sensor*, and *Actuator* components of our terminology partly cover the Connect component. Furthermore, the Connect component is responsible for a corresponding message translation, hence, it is encompassed by our *Gateway* component. The Connect component also provides further event handling functionality, thus, it is covered by our *IoT Integration Middleware* as well. In addition, the Analytics, Risk Management, and Information Management components provide the core functionality of the platform, thus, they are covered by the *IoT Integration Middleware* of our terminology. The Bluemix Open Standards Based Services component and the Flexible Deployment component build the basis of the platform. The IoT Industry Solutions and Third Party Apps components are encompassed by our *Application* component, since they enable the connection of further applications.

With consideration to the IBM Watson IoT Platform, our IoT terminology can be mapped onto it. Even though the *Device*, *Sensor*, *Actuator*, and *Gateway* components are not represented in particular, they are part of the Connect component.

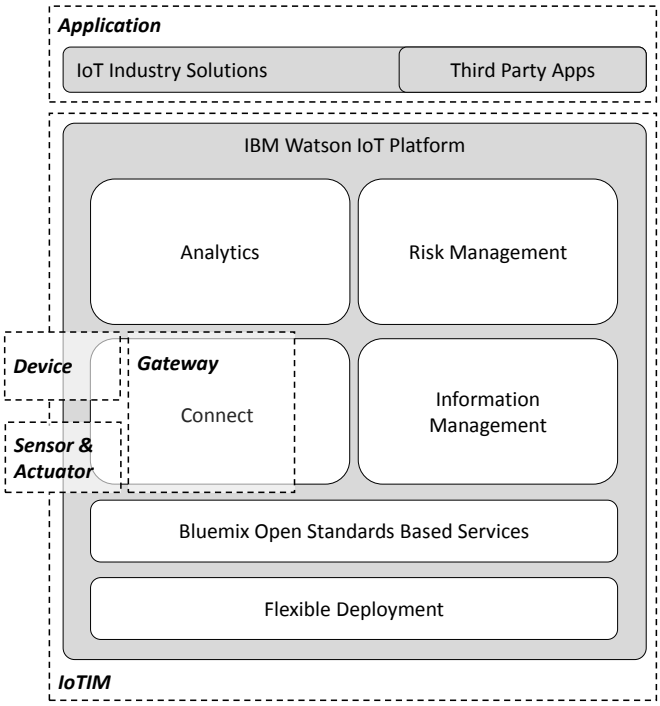

Fig. 7 IBM Watson IoT Platform architecture based on [19]

3.7 Microsoft Azure IoT Hub

The Azure IoT Hub[11] is a managed, cloud-based service, provided by Microsoft. Its architecture and the mapping onto our IoT reference architecture is shown in Fig. 8. The main component is the IoT Hub, where all remaining components are connected to. Since Microsoft only separates between IP-capable and Personal Area Network (PAN) Devices, those map to our *Device*, *Sensor*, and *Actuator* components. IP-capable devices may communicate directly or via a Cloud Protocol Gateway with the IoT Hub, whereby PAN Devices additionally need a Field Gateway to perform local management services, such as managing access and information flow. Hence, the Cloud Protocol and the Field Gateway are covered by our *Gateway* component. The core functionality of the solution is provided by the IoT Hub, the Event Processing and Insight, the Device Business Logic, Connectivity Monitoring, and the Application Device Provisioning and Management components, hence, they are covered by our *IoT Integration Middleware*. Furthermore, the Application Device Provisioning and Management component also enables the connection of further *Applications*.

With regard to the Microsoft Azure IoT Hub, each component of our IoT reference architecture is represented. Some components are overlapping and, again, it is

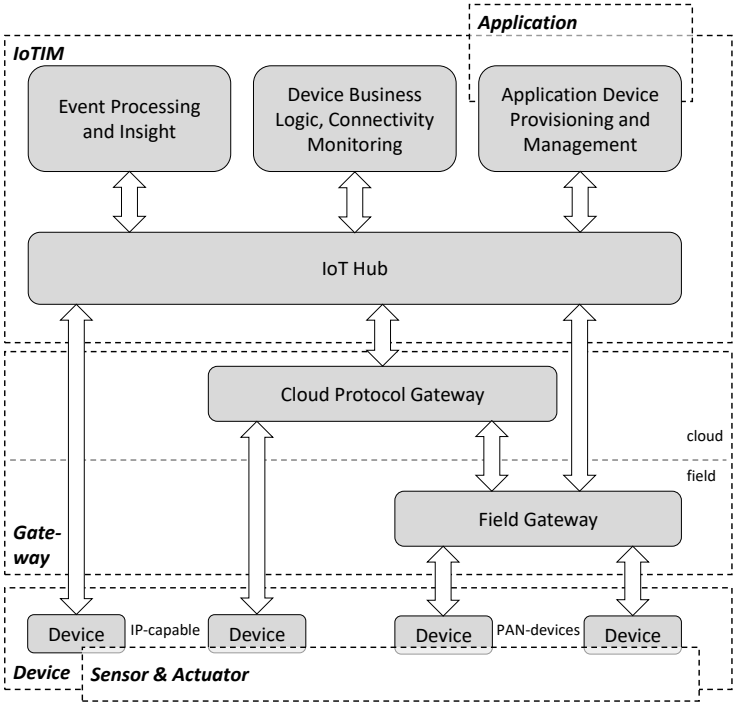

Fig. 8 Microsoft Azure IoT Hub architecture based on [6]

not further distinguished between *Devices*, *Sensors*, and *Actuators*, but this is still appropriate to the definition of our IoT reference architecture components.

3.8 SmartThings

SmartThings[5] is an IoT platform provided by Samsung for a smart home environment. The architecture and the corresponding mapping with our IoT reference architecture is shown in Fig. 9. It is composed of three core elements, namely the Device Type Handlers, the Subscription Processing, and the Application Management System including the SmartApp Management & Execution, where all further components are connected to. SmartThings combines Sensors, Actuators, Devices, Users, and Things within one component. They further distinguish between this composed component and another Clients (-Devices) component. Since Clients (-Devices) may also contain *Sensors* and *Actuators*, both components are covered by our *Device* component, with the *Sensor* and the *Actuator* components partly overlapping. Since the Device Type Handlers translate event-messages into a normalized SmartThings event, and the Hub Connectivity and the Client Connectivity enables the connection of Devices to the platform, those components represent the *Gateway*. The core functionality of the platform is provided by the Subscription Processing and the Application Management System, including the SmartApp Management & Execution components.

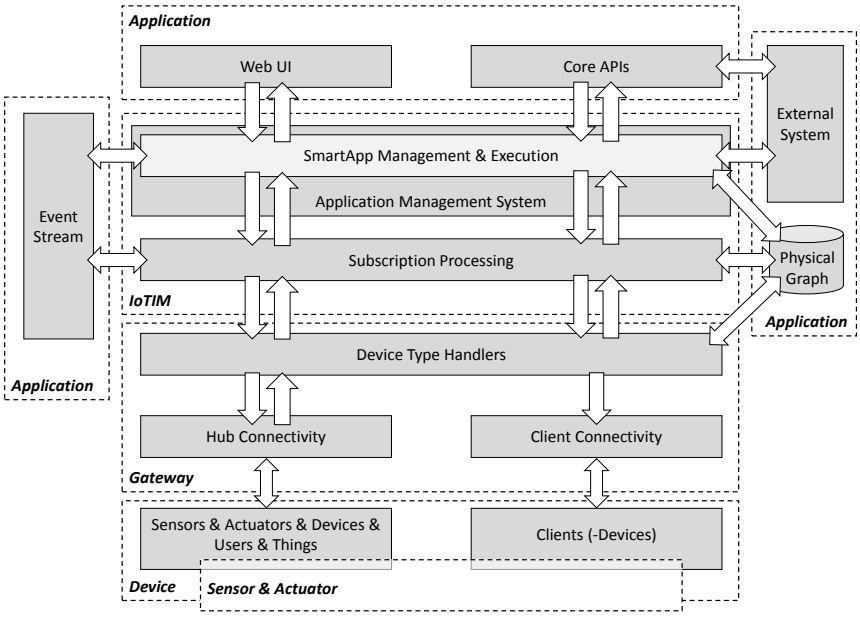

Fig. 9 SmartThings architecture based on [33]

Consequently, they cover the *IoT Integration Middleware* of our reference architecture. The Event Stream, Web UI, Core APIs, External System, and Physical Graph components represent possibly connected *Applications* to the platform.

Regarding SmartThings, our IoT reference architecture covers each component of the architecture. As described above, the *Sensor*, *Actuator*, and *Device* components are overlapping, since SmartThings uses them within composed components. Nevertheless, this is appropriate to our definition.

3.9 Summary of the Comparison

Reflecting each comparison described above, our introduced IoT reference architecture is represented within each considered platform. Table 1 shows a summarized overview of the comparison: The rows are defined by the components of our reference architecture and the columns display each considered IoT platform, whereby the table cells indicate each component of the IoT platform matching to our reference architecture components.

Only the architecture of OpenMTC and SmartThings represent a *Sensor* and *Actuator* component. All remaining platforms, besides the Microsoft Azure IoT Hub, just mention those components within their documentation. The *Device* component is not represented within the architecture of OpenMTC and the IBM Watson IoT Platform, but mentioned within the documentation. The remaining platforms represent a *Device* component within their architecture. Furthermore, the platforms FIWARE, AWS IoT, and the Microsoft Azure IoT Hub further distinguish the concept of "Intelligent" Devices, which have already some kind of logical functionality included. Following, those "Intelligent" Devices are covered by our *Device*, *Gateway*, and *IoT Integration Middleware* components, respective to the level of integrated logical functionality. Since those differences within the definition and granularity of the IoT platforms' components are present, we mapped them onto our clearly separated components, to clarify the concept and the used granularity. Besides SiteWhere and AWS IoT, all platforms represent the concept of our *Gateway* within their architectures. Nevertheless, the documentations of SiteWhere and AWS IoT also embrace the functionality of our *Gateway*. Obviously, each platform represents the core functionality, i.e., our *IoT Integration Middleware* within the architecture. The differences lie in the granularity and the number of the components comprising the functionality of the *IoT Integration Middleware*. Furthermore, each platform enables the connection of further *Applications*. FIWARE does not represent a corresponding component within its architecture, but it is mentioned within the documentation.

Platform	FIWARE	OpenMTC	SiteWhere	Webinos	AWS IoT	IBM Watson IoT Platform	MS Azure IoT Hub	Samsung SmartThings
Sensor / Actuator	-*	Sensors & Actuators	-*	-*	-*	-*	-	Sensors & Actuators & Devices & Users & Things
Device	Device / NGSI Device	-*	Data from Devices + Commands to Devices	PZP: Policy + Session + Discovery + Context Manager	Things	-*	Device	Sensors & Actuators & Devices & Users & Things + Clients (-Devices)
Gateway	IoT Edge	Front-End: Core Features + Connectivity + Back-End: Connectivity	-*	PZP: Sync + Messaging Manager + PZH: Sync Manager	-*	Connect	Cloud Protocol Gateway + Field Gateway	Hub and Client Connectivity + Device Type Handlers
IoT Integration Middleware	IoT Back-End + Data Context Broker	Back-End: Connectivity + Core Features + Application Enablement	SiteWhere Tenant Engine	PZH: User Authentication + Policy Repository + Policy Enforcement + Web APIs	Message Broker + Thing Shadows + Thing Registry + Rules Engine + Security & Identity	Analytics + Risk Management + Connect + Information Management + Bluemix Open Standards Based Services + Flexible Deployment	IoT Hub + Event Processing and Insight + Device Business Logic, Connectivity Monitoring + Application Device Provisioning and Management	Application Management System + Subscription Processing
Application	-*	Applications + Other M2M Platform	Integration- + REST-components	Third Party Applications	Amazon Services + IoT Applications	IoT Industry Solutions + Third Party Apps	Application Device Provisioning and Management	Event Stream + Web UI + Core APIs + External System + Physical Graph
	* Not represented in the figure of the architecture, but described within the documentation.							

Table 1 IoT platform comparison summary

4 Related Work

In this section, the IoT reference architecture is related to previously published IoT architectures, architecture reference models, domain models, and taxonomies.

Bauer et al. [5] describe seven functional components between a device and an application layer as part of an IoT reference architecture. The components are the Management, Service Organization, IoT Process Management, Virtual Entity, IoT Service, Security, and Communication. The Communication component can be mapped onto the *Gateway* of the presented IoT reference architecture in this work, while the remaining components build the *IoT Integration Middleware*, respectively. In contrast to our work, the *Sensor*, *Actuator*, *Device*, and *Application* components are not specifically defined.

Fremantle [14] introduces an IoT reference architecture comprising of five layers. The device layer encompasses *Devices*, *Sensors*, and *Actuators*, but does not detail the latter two in particular. The relevant transports layer abstracts the same concept as our *Gateway*. The aggregation/bus layer as well as the event processing and analytics layer correspond with our *IoT Integration Middleware*. Thus, they provide the core functionality of an IoT platform. Finally, further *Applications* as presented in this work are subsumed by Fremantle as client and external communications. Since this IoT reference architecture lacks unambiguous definitions of all components it does not pursue our goal to provide a clear terminology to understand commonalities and differences of diverse IoT platforms, it is less effective than our reference architecture.

Cisco [10] introduces a seven-layered IoT reference model. The *Devices*, *Sensors*, and *Actuators* as presented in this work are comprised in the Physical Devices and Controllers of Cisco's reference model, while the *Gateway* layer equals their Connectivity concept. The Edge (Fog) Computing, Data Accumulation, and Data Abstraction layer corresponds to the *IoT Integration Middleware* of our IoT reference architecture, whereas the Application layer corresponds roughly to the combination of the components *IoT Integration Middleware* and *Application*. Finally, the capability to connect arbitrary *Applications* to the *IoT Integration Middleware* is reflected by the concepts Collaboration and Processes by Cisco. Since the concepts introduced by Cisco are not unambiguously defined in their reference model, the concepts presented in this work can be used to exactly determine the meaning of Cisco's concepts by mapping the reference model by Cisco onto our IoT reference architecture.

The three-layer architecture by Zheng et al. [37] contains similar concepts as those outlined in our reference architecture and is also basis for the works by Wu et al. [35], Atzori et al. [4], and Aazam et al. [1]. Gathering data from and acting on the physical world is described by the abstract concept of a Perception Layer and corresponds with the combination of our *Sensors*, *Actuators*, and *Devices*. Pre-processing of gathered data and transmission to an integrating middleware is covered by the Network Layer, which corresponds to the interplay of *Device* and *Gateway* in our IoT reference architecture. The Application Layer is also a more coarse-grained concept and reflects the core functionality of the platform. Thus, it maps onto our *IoT Integration Middleware* and *Applications*. Further approaches by Atzori et al. [3, 4] and Xu et al. [36] are similar layered architectures and grasp the field of IoT from a

service-oriented architecture (SOA) perspective. While they focus on the design of IoT architectures, they lack a clear and unambiguous definition of the concepts, which they introduce and rely on. Neither of these works map their introduced concepts onto existing technologies and platforms, which is one contribution of our work.

Kim et al. [21] investigated diverse IoT applications and abstracted a generic platform model from them. They introduce the concept of Things, which are closely related to *Devices* as presented in this work. Gateways provide connections to a Platform in cases that Things cannot communicate directly with the Platform. Service Users as well as Service and Software Providers are connected to the Platform by RESTful APIs. In cases where no complex data processing is required on the Platform, a Service Use can also connect directly to devices, e.g., to gather metering data. All components of this model are covered of our reference architecture, besides the user.

The IoT Reference Model discussed by Krčo et al. [23] is based upon the IoT Domain Model by Haller et al. [18]. The concepts Augmented Entity, User, Device, Resource, and Service are introduced. A definition of these concepts is given but it is not abstract and unambiguous enough for mapping different IoT platforms upon each other to foster their understanding. For instance, on the one hand, a device is described as a hardware component, which integrates sensors and/or actuators and is, therefore, responsible for monitoring and interacting with real-world objects. On the other hand, a device is also capable of connecting to further IT systems. This example shows that the concept of a device is only roughly defined, thus, it is unclear if the device may also takes on the role of a gateway or if such an indirection is not foreseen, which implies that devices always communicate directly with the platform.

Mineraud et al. [26] review 39 existing IoT platforms according to six criteria including for instance data ownership or developer support. Concerning the architecture, they distinguish between cloud-based and local IoT platforms, but they do not provide a detailed analysis of the architectures as we do.

A high-level taxonomy for the components of an IoT platform is introduced by Gubbi et al. [16]. It contains the components hardware, middleware, and presentation. Hardware covers sensors, actuators, and embedded communication hardware, while middleware covers on-demand storage and computing tools for data analytics, and presentation provides visualization and interpretation tools. However, the taxonomy is very coarse-grained and not detailed enough to foster a clear understanding of the introduced concepts, which leads to possibly diverse interpretations.

5 Conclusion

The IoT is slowly turning from vision into reality: IoT platforms play a central role within this evolution by providing significant building blocks. A lack of standardization and development in silos has led to a heterogeneous platform landscape. We argue that, as a result of this heterogeneity, comparing and selecting one of these platforms is a difficult task. Not only do they use different concepts and technologies,

but also the terminology is not clearly defined. Many concepts and parts of these platforms are described with synonyms or homonyms, or differ in granularity.

To help with these problems, we introduced an IoT reference architecture which is based on existing platforms. We defined each component and described the communication between them. These components do not necessarily have to stay separated, but can be combined. We compared our reference architecture to eight existing platforms, four of which are open-source. We showed that the components of our architecture map to those of the existing platforms. When comparing or evaluating these different platforms, our IoT reference architecture can be a useful tool. Besides, it may be useful by providing a common basis on which to base new IoT platform designs.

Future work could present a technical definition of the reference architecture. Furthermore, this work will build the basis for a decision support approach, which provides a selection of IoT platforms based on user-given preferences. This will help a user to determine a suitable IoT solution for his case.

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

References

1. Aazam, M., Khan, I., Alsaffar, A.A., Huh, E.N.: 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)
2. Amazon Web Services: AWS IoT Documentation (2016). URL `https://aws.amazon.com/de/documentation/iot/`
3. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: A survey. Computer Networks **54**(15), 2787–2805 (2010)
4. Atzori, L., Iera, A., Morabito, G., Nitti, M.: The Social Internet of Things (SIoT) – When social networks meet the Internet of Things: Concept, architecture and network characterization. Computer Networks **56**(16), 3594–3608 (2012)
5. Bauer, M., Boussard, M., Bui, N., De Loof, J., C., M., Meissner, S., Nettsträter, A., Stefa, J., Thoma, M., Walewski, J.W.: IoT Reference Architecture. In: Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model. Springer Berlin Heidelberg (2013)
6. Betts, D.: Microsoft Azure – Übersicht über Azure IoT Hub. `https://azure.microsoft.com/de-de/documentation/articles/iot-hub-what-is-iot-hub/` (2016)
7. Bonomi, F., Milito, R., Natarajan, P., Zhu, J.: Fog Computing: A Platform for Internet of Things and Analytics. In: Big Data and Internet of Things: A Roadmap for Smart Environments, pp. 169–186. Springer (2014)
8. Borgia, E.: The Internet of Things vision: Key features, applications and open issues. Computer Communications **54**, 1–31 (2014)
9. Boswarthick, D., Elloumi, O., Hersent, O. (eds.): M2M Communications. John Wiley & Sons Inc (2012)
10. Cisco: The Internet of Things Reference Model (2014). URL `http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf`
11. Farhangi, H.: The path of the smart grid. IEEE power and energy magazine **8**(1), 18–28 (2010)
12. FIWARE: FIWARE Wiki (2016). URL `https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Main_Page`

13. Fraunhofer FOKUS: OpenMTC Platform Architecture (2016). URL http://www.open-mtc.org/index.html#MainFeatures
14. Fremantle, P.: A Reference Architecture for the Internet of Things (2015). URL http://wso2.com/wso2_resources/wso2_whitepaper_a-reference-architecture-for-the-internet-of-things.pdf
15. Fuhrhop, C., Lyle, J., Faily, S.: The webinos project. In: Proceedings of the 21st International Conference on World Wide Web, pp. 259–262. ACM (2012)
16. Gubbi, J., Buyya, R., Marusic, S., M., P.: Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems **29**(7), 1645–1660 (2013)
17. Guth, J., Breitenbücher, U., Falkenthal, M., Leymann, F., Reinfurt, L.: Comparison of IoT Platform Architectures: A Field Study based on a Reference Architecture. In: Proceedings of the International Conference on Cloudification of the Internet of Things (CIoT). IEEE (2016)
18. Haller, S., A., S., Bauer, M., Carrez, F.: 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 (2013)
19. IBM: IBM Internet of Things Architecture Overview. https://www.iot-academy.info/mod/page/view.php?id=478 (2016)
20. Kargupta, H.: Connected Cars: How Distributed Data Mining Is Changing the Next Generation of Vehicle Telematics Products. In: International Conference on Sensor Systems and Software. Springer (2012)
21. Kim, J., Lee, J., Kim, J., Yun, J.: M2M Service Platforms: Survey, Issues, and Enabling Technologies. IEEE Communications Surveys & Tutorials **16**(1), 61–76 (2014)
22. Kopp, O., Falkenthal, M., Hartmann, N., Leymann, F., Schwarz, H., Thomsen, J.: Towards a Cloud-based Platform Architecture for a Decentralized Market Agent. In: Informatik 2015, Lecture Notes in Informatics (LNI). Gesellschaft für Informatik e.V. (GI) (2015)
23. Krčo, S., Pokrić, B., Carrez, F.: Designing IoT and Architecture(s). In: Proceedings of the IEEE World Forum on Internet of Things (WF-IoT). IEEE (2014)
24. Lucke, D., Constantinescu, C., Westkämper, E.: Smart Factory – A Step towards the Next Generation of Manufacturing. In: Manufacturing systems and technologies for the new frontier, pp. 115–118. Springer (2008)
25. Merriam-Webster: Full definition of SENSOR (2016). URL http://www.merriam-webster.com/dictionary/sensor
26. Mineraud, J., Mazhelis, O., Su, X., Tarkoma, S.: A gap analysis of Internet-of-Things platforms. Computer Communications **89-90**, 5–16 (2016)
27. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of things: Vision, applications and research challenges. Ad Hoc Networks **10**(7), 1497–1516 (2012)
28. Open Mobile Alliance Ltd.: NGSI Context Management (2012). URL http://technical.openmobilealliance.org/Technical/release_program/docs/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf
29. Postscapes: IoT Cloud Platform Landscape. Vendor List. http://www.postscapes.com/internet-of-things-platforms/ (2016)
30. Ricquebourg, V., Menga, D., Durand, D., Marhic, B., Delahoche, L., Loge, C.: The Smart Home Concept : our immediate future. In: 1st IEEE international conference on e-learning in industrial electronics. IEEE (2006)
31. Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., Oliveira, A.: Smart Cities and the Future Internet: Towards Cooperation Frameworks for Open Innovation. In: The Future Internet Assembly. Springer (2011)
32. SiteWhere LLC.: SiteWhere System Architecture (2016). URL http://documentation.sitewhere.org/architecture.html
33. SmartThings, Inc.: SmartThings Documentation. http://docs.smartthings.com/en/latest/architecture/index.html (2016)
34. Webinos Project: webinos project deliverable – Phase II architecture and components. Tech. rep. (2012)

35. Wu, M., Lu, T.J., Ling, F.Y., Sun, J., Du, H.Y.: Research on the architecture of Internet of Things. In: Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE). IEEE (2010)
36. Xu, L.D., He, W., Li, S.: Internet of Things in Industries: A Survey. IEEE Transactions on Industrial Informatics **10**(4) (2014)
37. Zheng, L., Zhang, H., Han, W., Zhou, X., He, J., Zhang, Z., Gu, Y., Wang, J.: Technologies, Applications, and Governance in the Internet and of Things. In: Internet of Things – Global Technological and Societal Trends. River Publishers (2009)
38. Zorzi, M., Gluhak, A., Lange, S., Bassi, A.: From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view. IEEE Wireless Communications **17**(6), 44–51 (2010)

All links were last followed on Wednesday 30th November, 2016.