Web of Things Simplifies Industry Applications

Sebastian Käbisch, Christian Glomb, Charif Mahmoudi, Daniel Peintner
Siemens AG,Corporate Technology,Munich,Germany
{sebastian.kaebisch, christian.glomb, charif.mahmoudi, daniel.peintner.ext}@siemens.com

Keywords: Industrial Internet of Things (IIoT), Web of Things, Industrial Domain Knowledge

Abstract Application development in industry scenarios is usually highly challenging due to the composition of different technologies such as protocols, exchange formats, and data models. The W3C Web of Things (WoT) paradigm offers an excellent opportunity to simplify interaction with heterogeneous systems and increase interoperability in IoT applications such as those for industry. As a contribution to the 2nd W3C Web of Things Workshop, this position paper gives a brief insight into the opportunities of WoT, e.g. for use cases from industry, and discusses some white gaps that can be addressed in the future as a new work item for the next period of the W3C Web of Things working group charter.

1 INTRODUCTION

The main mission of the W3C Web of Things (WoT) [17] activities is to counter the fragmentation of the Internet of Things (IoT) by developing a web-based abstraction layer capable of interconnecting existing IoT platforms, devices, and cloud services and complementing available (domain) standards such as OPC UA, OCF, and OneM2M. Over the last three years, the WoT Working Group has developed key technology building blocks to define an architectural design and a common format for describing (physical or virtual) Things and services with the help of a so called W3C WoT Thing Description (TD) [16].

It is essential to be aware that WoT is not yet another IoT ecosystem standard. From the beginning, the WoT Group has invested in collaborations and liaisons such as with OPC Foundation [12], IETF [13], and OneM2M [14] to specify technology building blocks that can be used to complement the existing established IoT domain standards and increase interoperability as well as enable cross-domain applications.

This position paper provides a brief insight into the opportunities and potential of these building blocks in the context, e.g., of industry scenarios. In addition, it identifies considerations and possible directions to be considered in the next charter period of the Web of Things standardization group.

Overall our expectations in this workshop are

- Exchange about and discovery of a variety of new WoT use cases
- Identification of white gaps of the current working assumptions in WoT
- Increase of involvement of companies and existing or new IoT initiatives (e.g. OneM2M)

2 WOT'S OPPERTUNITIES

This section provides an insight into the great potential of the Web of Thing approach developed by the W3C Web of Things group so far.

2.1 Ease of Development

In modern IoT applications, ease of development constitutes a crucial criterion that drives the choice of the framework among the large scope of existing solutions [1]. The ease of development can be measured based on three main aspects. First is the efficiency of the programmatic Application Programming Interface (API) offered by the framework. Indeed, by offering a high-level programmatic API, an IoT framework can speed up

the development process by substituting general purpose language with a domain-specific language dedicated to the interaction model offered by the framework. This design choice offers to developers a development model where they can reason within the concepts of the framework without having to explicitly map those concepts to the underlying programming language. Second is the expressivity of the web API provided by the framework. The IoT solution offers dedicated schemes that describe the devices using their web API. Those descriptions are used for metadata of the services exposed by the devices and enable applications to communicate with those devices. Third, is the interaction model offered by the framework. An the interaction model is a design model that binds an application together in a way that supports the conceptual models behind the application. It is a crucial component of the design of an IoT application as it defines the way the devices, cloud, and users interact together to enable an overall IoT experience.

The WoT provides an elegant answer to those three aspects which makes it a first chose for the majority of IoT applications. Indeed, in term of efficiency of the programmatic API, WoT offers a standard definition of a scripting API that grants an abstracted access for the developers to manipulate the properties of the devices. The WoT scripting API hides the complexity related to the specificities of the underlying protocols. Furthermore, the WoT scripting API is an enabler for portability of the business logic as a servient developed under the scripting API is easily remappable from a protocol to another to certain extend.

One of the major contributions of the WoT is capability to semantically describe things. By introducing the notion of Thing Description (TD), WoT enables a discoverable (using Thing directory) and semantically annotated description for the components of an IoT system. TDs play a significant role to ease the mapping between concepts instead of the classical approach that used the labels as principal source of information to implement a matching mechanism between concepts and their concretizations.

The simple but efficient interaction model based on Event, Action, and Property trio defines how WoT based applications communicate. This model constitutes a great fit with the protocol bindings and ease the development of so called servients (server-client) as it makes the interaction with an instance independent from all platform specific constraints.

2.2 Industry Demonstrator

Figure 1 shows an example which can be found in a similar manner in numerous other industry scenarios. Heterogeneous components using OPC UA, Modbus/TCP, and BACnet must be composed, e.g. for monitoring purposes and/or for cloud integration.

W3C Thing Descriptions can be used to describe the capacities of a manufacturing system itself and/or individual Things used in a system, regardless of the underlying technologies such as protocols or serialization formats. Based on the application abstraction of WoT with properties, actions and events, data and functionality can be used in a common way and facilitate cross-domain and cross-technology application development.

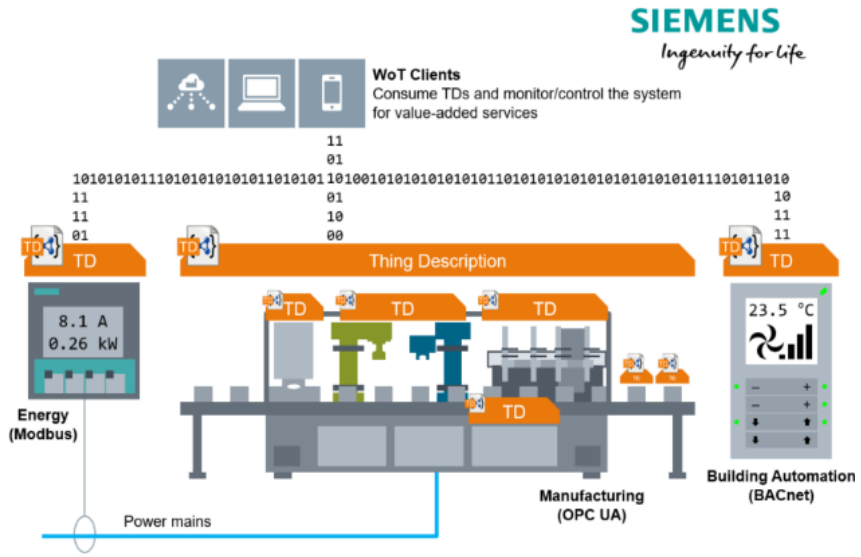

Figure 1 Web of Things for industry use cases

Siemens will present various industry-specific showcases during the W3C WoT workshop and the preceded Open Day.

2.3. Discovery, Thing Directory

Beyond the ease of development at node level introduced in the previous section, WoT pushes the notion of TD to the service discovery level. Service discovery is a notion widely and successfully used by the Service-Oriented Architecture (SOA) in the enterprise world to enable the process of finding suitable service for a specific task over the Internet using a registry. Universal Description, Discovery, and Integration (UDDI) is an ensample of registry used for webservices discovery where a provider can register a service and customers can lookup services in this registry [2].

WoT introduces a Thing Directory that extends this principle on two distinct axes. First is the extension to the IoT world. In other terms, it enables the discovery of physical things instead of the classical notion of discovery associated to services. Second is the semantic aspect of the discovery. Indeed, previous efforts such as Web Services Inspection Language (WSIL) were introduced to enhance the discovery mechanism for webservices. WoT Thing Directory innovates the way discovery is done by incorporating the interaction model provided by the TD into the discovery mechanism. The Thing Directory offers an API to lookup things based on metadata, properties, actions, or events to tailor the matchup to specificities of IoT devices. The current implementation [3] offers a CRUD API to allow users to interact with the directory. Moreover, Thing Directory introduces a notion for semantic lookup using an in-memory triple-store. This way, the Thing Directory can leverage the semantic description of the TDs and incorporate it within the lookup mechanism.

2.4. WoT Technology Landscape

Last year WoT found its way into the Open Source community when the Thingweb project [4] was accepted to become part of Eclipse IoT. With the main contributions 'node-wot' implementing the WoT Scripting API and serving as a toolbox for rapid WoT prototyping, and 'Thingweb directory' offering a database for TDs searchable via SPARQL queries and hence being the 1st choice for a central TD repository, Eclipse Thingweb and its affiliated web page [5] became the starting point for WoT development activities.

Another excellent tool provided by TU München takes care about the correct setup of TDs and gives valuable hints about errors and possible enhancements of a concrete TD instance [6].

However, since WoT aims to be the glue between IoT protocols, platforms, and domains, one of our future activities will be to spread the word about WoT in the Eclipse IoT community and get in touch with other projects and developers. One of the most concrete and interesting activities will be to develop a TD importer plugin for Eclipse Vorto [7] provided by Bosch. With that, developers will be enabled to generate source code for embedded systems based on the connectivity and semantics information hosted in a TD.

Other projects from Eclipse IoT where WoT can help to enhance the implementation are still to be identified. Therefore, we are active in the EU funded project 'Brain-IoT' [8] which is supported by the Eclipse foundation and makes heavily use of Eclipse software.

3 WOT'S CHALLENGES

Before concluding the position paper, we would like to give some suggestions for topics for the future of Web of Things that may be considered in the next charter work.

3.1 Scripting API as REC

The WoT Scripting API is a programming interface to ease the implementation of applications in a WoT runtime. It is comparable to a Web browser API and is meant to provide a standardized contract (e.g., for JavaScript applications).

The current development includes the following features: discover things, consume things, and expose things.

At the moment, the Scripting API is an optional part of the WoT Building Blocks. Hence it is possible to implement a WoT Thing without using the Scripting API. Hereafter, reasons are given why it should become nevertheless a mandatory part of WoT.

The afore mentioned interface definitions simplify application development and enable portability across vendors and network components. This means application developers may ground its work on a well-defined API. Moreover, it allows to run the *same* application on different hardware. Like

Web apps nowadays, these apps can be deployed to a device of vendor A as well as vendor B.

Web developers can focus on the actual problem they want to solve. The challenging implementation of bindings to given protocols (e.g., HTTP, CoAP, and WebSocket) can be implemented once in a stable and efficient way in the scripting API runtime.

Moreover, only the Scripting API allows a simple way to realize mash-ups of different things. Imagine a single-entry point for a service that aggregates under the hood various protocols and low level APIs with only some lines of code (e.g., JavaScript). Running code and examples can be found on the Eclipse project *thingweb* [18] which accompanies the standardization.

Let's recall the time websites have been labelled with "optimized for". Everyone can agree that this fragmentation should not happen again. Web apps are so successful because one can run them on different hardware and different operating systems. Only with well-defined and standardized interfaces WoT applications will become a first-class citizen in the Web. Hence, we propose that the Scripting API becomes a mandatory building block and a W3C recommendation in a future version. However, stakeholders from various domains might be necessary to establish the necessary common ground. In summary one can say that a well-defined Scripting API is very beneficial for the WoT landscape.

3.2 Shared Capabilities

One of the strengths of the W3C WoT Thing Description is the ability to integrate context knowledge known from certain types of applications or domain use cases. This is possible by using JSON-LD 1.1, which enables the integration of external domain knowledge (e.g. eCl@ss, SAREF, iot.schema.org) and domain-independent knowledge such as units (e.g. OM) or geographically based definitions (e.g. GeoLocation) via the @context mechanism.

Based on this mechanism, some basic Thing capability schemes can be defined and used to enrich Thing Descriptions to increase interoperability in IoT applications, regardless of domain usage. An example of such schema capabilities would be on/off, level value (=0..100%) or operational state (e.g. on, off, error). Such basic functional schemes can be centrally managed and linked by a centralized instance similar to schema.org. A perfect place to provide such capacities would be the new iot.schema.org activity, which should focus such intentions on the next step.

3.3. Eventing

The WoT interaction model defines three interaction affordance classes, namely Property, Action and Event.

The first two are generally understood in the same way. The latter, namely Event, is generally understood differently by different people active in different areas. In general, we can say that events describe push interactions initiated by the Thing. Examples of events are alarms or samples of time series that are pushed regularly.

According to the TD specification an event may contain three data fields. One for the event data itself. The other two may contain data for subscription and cancellation. The latter two depend on the protocol binding used and/or the message format (e.g., Webhooks).

We believe that the fields for subscription and cancellation need further attention and might be revisited in a future version. Further clarification is required to create a common understanding.

3.4. Protocols

So far, the W3C Thing Description offers HTTP protocol binding as standard and thus covers already a large number of use cases for IoT applications. However, the TD is designed to be open to the use of alternative established IoT protocols such as CoAP and MQTT as well as domain-specific protocols such as OPC UA, Modbus and BACnet. It would be desirable to have established and standardized RDF representations (similar to HTTP [15]) of the desired protocols in order to share the potential of the protocols and make them interoperable. A good platform to manage such protocol-specific vocabulary would be the W3C or iot.schema.org.

3.5 Alignment with other Standards

The present TD model provides a representation for (typed) web links exposed by a thing [9]. The web linking definition of the TD reflects a very common subset of the terms defined in web linking [10]. The defined terms can be used, e.g., to describe the relation to another thing such as a lamp thing is controlled by a switch thing. A similar goal is currently pursued in IETF's Thing-to-Thing research group (T2TRG) [13]. Therefore, an alignment of phrases and concepts in both W3C WoT and IETF T2TRG makes sense and is currently started by aligning corresponding parts of the TD with an IETF draft [11].

REFERENCES

[1] H. Hejazi, H. Rajab, T. Cinkler and L. Lengyel, "Survey of platforms for massive IoT," in 2018 IEEE International Conference on Future IoT Technologies (Future IoT), 2018 .

[2] Y. Zhong, Y. Fan, W. Tan and J. Zhang, "Web service recommendation with reconstructed profile from mashup descriptions," IEEE Transactions on Automation Science and Engineering, vol. 15, no. 2, pp. 468-478, 2018.

[3] Thingweb Team, "W3C WoT Thing Directory implementation," 4 10 2018. [Online]. Available: https://github.com/thingweb/thingweb-directory. [Accessed 1 4 2019].

[4] "Eclipse Thingweb," 2019. [Online]. Available: https://projects.eclipse.org/projects/iot.thingweb.

[5] "Thingweb," 2019. [Online]. Available: http://www.thingweb.io/.

[6] "Validation of Thing Descriptions," 2019. [Online]. Available: http://plugfest.thingweb.io/playground/.

[7] "Eclipse Vorto," 2019. [Online]. Available: https://www.eclipse.org/vorto/.

[8] "Brain-IoT," 2019. [Online]. Available: http://www.brain-iot.eu/.

[9] "TD Web Linking Vocabulary Definition," 2019. [Online]. Available: https://w3c.github.io/wot-thing-description/#link.

[10] Web Linking, 2017. [Online]. Available: https://tools.ietf.org/html/rfc8288.

[11] "The Constrained RESTful Application Language (CoRAL)," 2019. [Online]. Available: https://tools.ietf.org/html/draft-hartke-t2trg-coral-08.

[12] W3C and OPCF to integrate OPC-UA into the Web of Things, 2016, https://opcfoundation.org/news/opc-foundation-news/w3c-and-opcf-to-integrate-opc-ua-into-the-web-of-things/

[13] Thing-to-Thing Research Group (T2TRG), https://datatracker.ietf.org/rg/t2trg/about/

[14] oneM2M and W3C Web of Things Interworking, ftp://ftp.onem2m.org/Work%20Programme/WI-0071/WI-0071-WoT_Interworking-V0_0_1.DOC

[15] HTTP Vocabulary in RDF 1.0. Johannes Koch; Carlos A. Velasco; Philip Ackermann. W3C. 2 February 2017. W3C Note. URL: https://www.w3.org/TR/HTTP-in-RDF10/

[16] Web of Things (WoT) Thing Description, W3C Editor's Draft 17 April 2019, https://w3c.github.io/wot-thing-description/

[17] Web of Things Working Group, https://www.w3.org/WoT/WG/

[18] Eclipse Thingweb, Tools and reference implementation of the W3C Web of Things, https://github.com/eclipse/thingweb.node-wot