Comparing IoT Platforms under Middleware Requirements in an IoT Perspective

An fulfillment analysis of IoT middleware requirements in IoT Platforms that uses REST to communicate with external applications

Artur Oliveira, Daniel Melo, Geiziany Silva, Thiago Gregório
CESAR – Centro de Estudos e Sistemas Avançados do Recife
Recife, Brazil
e-mail: artur@arturluiz.com, danielfarias.ti@gmail.com, geiziany.mendes@gmail.com, thiago.gregorio@gmail.com

***Abstract*—The increasing number of heterogeneous objects in IoT and the different kinds of software that consume data generated by those objects, have created an opportunity for middleware software to arise acting like an adapter, simplifying communication among them. As Representational State Transfer (REST) is broadly used as communication protocol by Web applications, this paper aims to analyze the fulfillment of Internet of Things (IoT) middleware requirements in IoT Platforms that use REST to communicate with external applications.**

Keywords-IoT; REST; Middleware Requirements; IoT Platforms.

I. INTRODUCTION

Internet of Things (IoT) means everything connected to the Internet, such as sensors, smart objects, clothes, toys, anything at all. The condition is that those things can produce something meaningful. The good news is that, with all people's creative imagination, almost any data may have a meaning under certain circumstances and can become useful information. To show how the IoT is going to be a world changer, it is predicted that the quantity of things in IoT would reach an astonishing number of 212 billion things generating constant data about practically everything in real world [3].

But not all those things (the 'T' from IoT) talk the same language, as they are made by different companies or groups. It is normal that they do not have a common interface to transmit their data directly to services and would be unfeasible that the services have an internal component to understand a wide variety of different components. Having this in mind, middleware software receives this responsibility. In addition to providing a large understanding how to talk to things, it also translates the data to a known language (protocol) by external applications.

Representational State Transfer (REST) is an architectural style based on Hypertext Transfer Protocol (HTTP) and an approach to communications, which has been widely used in integrations between different applications.

This paper's main purpose is to analyze the fulfillment of IoT middleware requirements in IoT Platforms that use REST to communicate with external applications. We start in Section II by presenting concepts related to the Internet of things. Section III continues by presenting an overview of the REST architectural style. Then, Section IV described the concepts of middleware and its different types that have emerged over time. In Section V, the functional and non functional requirements of middleware for IoT are listed. In Section VI, we analyze the fulfillment of IoT middleware requirements in IoT Platforms that use REST. Finally, Section VII concludes the paper.

II. INTERNET OF THINGS

The future of communication and Information technology (IT) is represented by the technological revolution of the Internet of Things [2]. The IoT was leveraged by technological advances in wireless communications, Radio-frequency identification (RFID), and the strong growth of the World Wide Web (www). The main objective of the IoT is to allow anything existing in the world can be identified, addressed, controlled and monitored by the Internet anytime and anywhere, interconnecting the physical and the virtual world through communication between two new dimensions: human-to-thing (H2T) and thing-to-thing (T2T), both promoted by IoT [1].

Through the interconnection of virtual and physical worlds, sensors play a vital role in achieving the connecting bridge between these worlds. The sensors are designed to collect data from their environment, with the intention to generate information about the context, allowing monitoring and controlling anything that can be connected to the environment [2].

Smart objects of IoT are expected to reach 212 billion entities deployed worldwide by the end of 2020. The expectation is that the Machine to Machine (M2M) traffic flow constitutes 45% of all Internet traffic [3]. So, the implantation of the IoT paradigm directly impacts the daily lives of people that will be motivated by the use of new technologies based on the interaction of physical devices [1].

III. REPRESENTATIONAL STATE TRANFER (REST)

The architectural style REST emerged in the mid-2000s, proposed by Roy Thomas Fielding through the doctoral dissertation: "Architectural Styles and the Design of Network-based Software Architectures" [4][5]. This style made people feel encouraged to use protocols and Web features to map requests in various representations, providing

 ISBN: 978-1-61208-505-0

the resource management and processing by means of a uniform interface operation [6]. The REST style is based on HTTP protocol, Uniform Resource Identifier (URI), Extensible Markup Language (XML) and HyperText Markup Language (HTML) [7]. The main features around the REST architecture are described below.

A. The Uniform Resource Identifier (URI)

The creation of an identifier for Web resources is necessary in order for anything can be considered as a Web feature, an audio, a video, or an encapsulated process, for instance. The URI is used to identify and address each Web resource, directly navigating to the specific resource. The relationship between resources and URI is that a resource may correspond to multiple URIs, but a URI corresponds only to a single resource that characterizes a relationship of one-to-many [6].

B. Resource Representation Transformation

The representation of a resource defines its current status, including the data itself and metadata. A resource can be represented through the transformation to some known formats: Extensible Hypertext Markup Language (XHTML), Atom, XML, JavaScript Object Notation (JSON), Plain Text, Comma-Separated Values (CSV MPEG-4 Part 14 (MP4) or JPEG [6].

C. Operation Methods GET, POST, PUT or DELETE

In the Web context operations, such as GET, PUT, POST, and DELETE are known as methods standard-based operations on the HTTP protocol. REST emphasizes the semantic use of such methods to perform the desired transactions [6].

D. Stateless Communication

The stateless communication on REST defines that each client request is treated as an independent transaction, and contains sufficient information to be totally understood. Using REST, any request is unrelated to any previous request. So, the communication consists of independent pairs of request and response [6].

E. Why use REST for IOT

The REST structure is designed to accommodate large data transfers efficiently equivalent to hypermedia data [7]. So, through the IOT model that allows anything that exists in the world to be connected and that its control and monitoring is possible [1]. It is easy to see the sensors as a resource on the Web and using RESTful Web service that is nothing more than a simple Web service that uses the HTTP protocol and REST principles, it is a way to combine HTTP and Web Service easily and clearly [8].

IV. MIDDLEWARE

Several definitions have emerged in the literature about distributed systems. One of them can summarize enough: A distributed system is a collection of independent computers that appears to its users as a single coherent system [9]. This means that, transparent to the user, multiple components (computers on the definition) require integration and collaboration between them. In principle, distributed systems must also have high scalability and availability. Users and applications should not notice that parts are being replaced or adjusted, or even that new parts are being added [9].

With the emergence of the need for more robust systems that could operate in a distributed way, the software engineering faced challenges during the development of distributed systems. New problems have arisen which did not exist in the development of centralized systems, such as network saturation or connection problems between the components involved [9]. An infrastructure that supports the development and execution of distributed applications was necessary.

The term middleware first appeared in the late 1980s [10]. There are several definitions for middleware in the literature. The common point between them is that middleware can be defined as a software layer located above the operating system and network software and below applications as shown in Fig. 1. The middleware allows interaction and communication between different applications via Application Programming Interface (APIs) and protocols supported between distributed components [11][12][13][14] proposed the following requirements for middleware: network communication, coordination, reliability, scalability and heterogeneity.

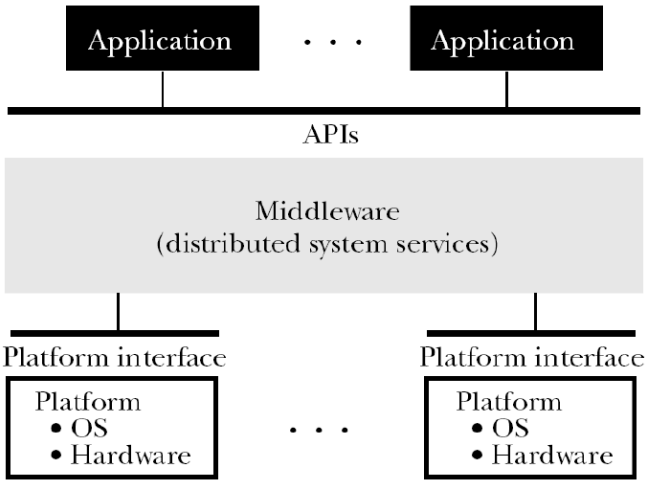

Figure 1. The set of middleware in distributed system, taken from [3..7], p. 89

Middleware gained more importance in recent years for its role in simplifying the development of new services and integration between old and new technologies [12]. In the literature, there are different types of middleware that have emerged over time and according to technological evolution. The most known are: transactional middleware, procedural middleware, object-oriented middleware and message-oriented middleware messages.

A. Transactional Middleware

It is a kind of middleware that is already considered old, which supports transactions between components that are

distributed on different servers. Transactional middleware was designed to support synchronous and distributed transactions. Its main function is to coordinate requests between clients and servers that can process these requests [9]. A transaction must support the Atomic, Consistent, Isolated and Durable (ACID) property. Atomic means that transaction either completes or it does not. The "all or nothing" strategy. Consistency should hold the system in a consistent state, independent of the status of the transaction. Isolation is the ability of one transaction to work independently from other transactions. Durability means the ability of the transaction to survive system failures (expected or unexpected) [15].

B. *Procedural Middleware*

Remote Procedure Calls (RPC) was developed by Sun Microsystems, the same responsible for the Java language, in early 80s. Their use enables an application to call functions from other applications running on remote machines. By being a synchronous communication mechanism, the client application waits while processing of the remote function is completed. Another problem caused by the PRC is the high traffic on network, requiring at that time, an evolution for the use of networks with better performance [16]. Examples: Tuxedo from BEA and Customer Information Control System (CISC) from IBM.

C. *Object-oriented Middleware (OOM)*

It is an evolution of procedural middleware. Communication is still synchronous between distributed objects. The interfaces of the services are described by specific languages and the marshalling and unmarshalling (data representation transformation in a format compatible for storage and transmission) are made automatically, unlike the procedural middleware that required this transformation was implemented [17]. Examples: Common Object Request Broker Architecture (CORBA) from Object Management Group (OMG), Distributed Component Object Model (DCOM) from Microsoft and Remote Method Invocation (RMI) from Java.

D. *Message-oriented Middleware (MOM)*

The information (messages) that travels between distributed components can be processed in two ways: message queuing and message passing. In the message queuing way, the communication is indirect, asynchronous and the messages are sent to queues. In the message passing way, the communication is direct, synchronous and operates according to the publish-subscribe model [17].

V. MIDDLEWARE REQUIREMENTS FOR IOT

Due to the nature of IoT middleware, where it is necessary to connect a large number of heterogeneous components (sensors and devices), it is recommended to archive a few minimal requirements to execute this task effectively. In [18], IoT middleware requirements are grouped into functional, those focus are functionalities themselves, and non-functional, those focus is on Quality of Service (QoS) and performance.

Fig. 2 illustrates the set of requirements explored in this paper grouped into functional and non-functional requirements. The following requirements were selected for their relevance level presented in [18]. We list their descriptions below.

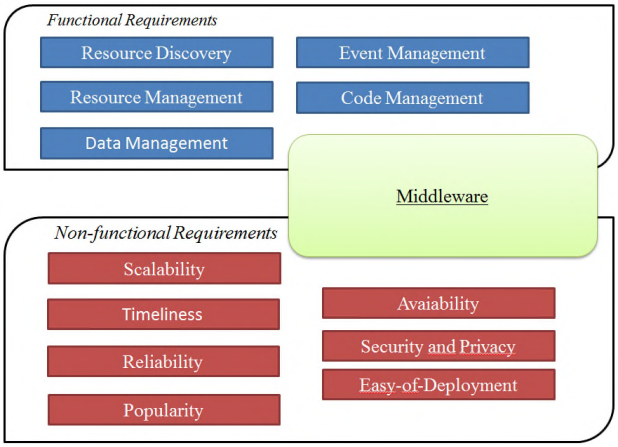

Figure 2 – The set of middleware requirements in an IoT perspective explored in this paper.

Resource Discovery allows middleware service to detect the exact moment when components connect and disconnect, making possible a reliable list of all connected components and to free up unused resources. It is also necessary that such functionalities are automated, due to infeasibility of human intervention in every situation.

Resource Management covers the efficient use of available resources, monitoring of such resources and the release procedure when they are no longer needed. This requirement is necessary due to limited resource components commonly encountered in an IoT environment.

Data Management provides access to data generated by devices and sensors, and data that may have a useful meaning to applications that consume these data from middleware like data related to network health.

Event Management allows external services connected to the middleware to be notified when an event happens in the device network that that middleware controls. Those events may be related to components availability, the interaction between them, network health etc.

Code Management allows users or external applications use a group of devices to solve a defined task, injecting necessary code in those devices. It also includes functionalities that migrate the injected code among other devices.

Scalability is the capacity to support a growing number of connected "things" generating data constantly, and external applications consuming such data with little or no loss of QoS.

Timeliness is the capacity that ensures that data which are dependent on the time they were generated are provided with a time delay which tends to zero. In the IoT context there are many components that fit into this category. This requirement turns out to be essential for a middleware that intends to act in the areas of health, transport, security etc.

 ISBN: 978-1-61208-505-0

Reliability ensures that the middleware is always operating during the execution of a task, even when there are failures. This requirement is totally dependent on devices and sensors that are connected to the middleware, since it is they who generate the data. A failure at the device level can cause the transfer, through the middleware, to consumer systems.

Availability is the capability that the middleware is or appear to be available when necessary. This requires that the frequency of errors and the recovery time is small enough as to become imperceptible to external services connected to the middleware.

Security and Privacy ensures data in the middleware is protected from being read or modified by anyone other than those who have rightful permission and prohibit malicious applications to have access to connected things and connected services.

Ease of Deployment is not an essential requirement. It concerns the low complexity in the installation and upgrade process middleware, causing an advanced technical knowledge not being required for the deployment and maintenance. The ideal scenario is an automated update process that does not interfere with Availability and Reliability requirements.

Popularity is a differential in the IoT middleware platform choosing process. Popularity is directly related to the size of the community that uses the middleware and the amount of contributions that community provides.

VI. IoT PLATFORMS ANALYSIS

IoT has undergone rapid transformation since the variety and the number of devices connected to the Internet have increased exponentially in recent years [19]. IoT has become a mainstream technology with a significant potential for advancing the lifestyle of modern societies. For this reason, there are several companies that heavily invest in the creation of solutions that covers IoT and necessary infrastructure, providing a complete platform for easy development and deployment of applications that consume IoT data.

The list of IoT platforms present in this paper was purely based on the criteria of companies that provide software solutions that enable information processing devices and sensors using REST as integration option.

The selected platforms are: Appcelerator, Amazon Web Service (AWS) IoT Platform, Bosch IoT Suite, Ericsson Device Connection Platform (DCP), EVRYTHNG, IBM Watson IoT Platform, Cisco ParStream and Xively.

Appcelerator is a platform for mobile development, including REST integration and real-time analytics through Titanium [20].

Bosch IoT Suite is composed by different services, they are Analytics, Hub, Integrations, Permissions, Remote Manager, Rollouts and Things, these services are described in [21] [22].

AWS IoT Platform provides integrations from devices to AWS Services and other devices, it also has a security layer for data and interaction [23] [24] [25] [26].

Xively's main focus is to make their clients profit from IoT. For that, they offer from IoT services (management of devices and data) to professional services (specialized consulting for clients) [27].

Ericsson DCP is a M2M platform that handles connectivity and subscription management, supporting a high number of devices and applications [28].

According to [29] "EVRYTHNG collects, manages and applies real-time data from smart products and smart packaging to drive IoT applications".

IBM Watson IoT Platform is a cloud-hosted service that simplifies access to connected devices data providing real-time connectivity throught MQ Telemetry Transport (MQTT) protocol [30] [31].

Cisco ParStream is mainly focused in analysis and time to market. [32] presents a list of its capabilities.

Table 1 shows the analysis of the functional and non functional requirements for IoT middleware depending upon the selected IoT platforms.

Through this analysis we realized that almost all IoT platforms have a requirement "not mentioned" in the documentation available and researched by the authors. This may show that the documentation is not complete enough or that the requirement "not mentioned" cannot really be found on this platform.

Another aspect realized by the analysis is that IBM Watson IoT Platform achieves all requirements presented in this paper, maybe proving to be the best choice among the listed platforms. The non-functional requirements Scalability and Reliability were found in all the chosen IoT platforms, proving to be a main requirement for any IoT platform. A possible cause for timeliness not being mentioned in a few platforms may be that they were not designed for a specific area, but for general use instead.

VII. CONCLUSION

This paper proposed to analyze the fulfillment of IoT middleware requirements in IoT Platforms that use the architectural style REST. After a brief explanation of all the concepts that surround IoT platforms, this paper has analyzed the functional and non functional requirements of middleware for IoT of the current state-of-the-art IoT software platforms.

It also can be seen that all IoT platforms, except the "Appcelerator", satisfy the majority of functional and non functional requirements of middleware for IoT. The analysis focused on requirements, such as resource discovery, resource management, data management, event management, code management, scalability, timeliness, reliability, availability, security and privacy and easy-of deployment.

As future work, we intend to analyze the cost benefit in relation to the total resources needed in the implementation of IoT platforms. Sorting the IoT platforms is more feasible for the cost reserved for this deployment.

 ISBN: 978-1-61208-505-0

TABLE 1 – ANALYSIS RESULTS OF THE FULFILLMENT OF IOT MIDDLEWARE REQUIREMENTS IN IOT PLATFORMS.

	Appcelerator	AWS IoT platform	Bosch IoT Suite	Ericsson DCP	EVRYTHNG	IBM Watson IoT	Cisco ParStream	Xively
Resource Discovery	X	√	√	√	√	√	X	√
Resource Management	X	Not mentioned	√	√	√	√	√	√
Data Management	X	√	√	√	√	√	√	√
Event Management	X	√	Not mentioned	√	√	√	Not mentioned	√
Code Management	X	√	√	√	Not mentioned	√	Not mentioned	√
Scalability	√	√	√	√	√	√	√	√
Timeliness	X	√	Not mentioned	√	√	√	√	Not mentioned
Reliability	√	√	√	√	√	√	√	√
Avaiability	√	√	√	Not mentioned	√	√	√	√
Security and Privacy	√	√	√	√	√	√	Not mentioned	√
Easy-of Deployment	X	√	√	√	√	√	X	√

REFERENCES

[1] P. F. Pires et al. (2014), "A Platform for Integrating Physical Devices in the Internet of Things". Proceedings - 2014 International Conference on Embedded and Ubiquitous Computing, EUC 2014, x'234. http://doi.org/10.1109/EUC.2014.42

[2] L. Tan, "Future internet: The Internet of Things. 2010 3rd International Conference on Advanced Computer Theory and Engineering" (ICACTE), V5–376–V5–377. http://doi.org/10.1109/ICACTE.2010.5579543

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications". IEEE Communications Surveys and Tutorials, 2015, 17(4), 2348 http://doi.org/10.1109/COMST.2015.2444095

[4] B. Costa, P. F. Pires, F. C. Delicato, and P. Merson, "Evaluating a Representational State Transfer (REST) architecture: What is the impact of REST in my architecture?". Proceedings - Working IEEE/IFIP Conference on Software Architecture 2014, WICSA 2014, 105. http://doi.org/10.1109/WICSA.2014.29

[5] T. R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", 2000, Building, 54, 162. http://doi.org/10.1.1.91.2433

[6] Z. Han, Y. Kong, and X. Wang, "Geographic stereo video web service based on the REST architecture". Proceedings - 2011 19th International Conference on Geoinformatics, Geoinformatics 2011, pp2 (40771166). http://doi.org/10.1109/GeoInformatics.2011.5980900

[7] L. Xiao-Hong, "Research and development of web of things system based on rest architecture". Proceedings - 2014 5th International Conference on Intelligent Systems Design and Engineering Applications, ISDEA 2014, 745. http://doi.org/10.1109/ISDEA.2014.169

[8] X. Zhang, Z. Wen, Y. Wu, and J. Zou, "The implementation and application of the internet of things platform based on the REST architecture". BMEI 2011 - Proceedings 2011 International Conference on Business Management and Electronic Information, 2, 43. http://doi.org/10.1109/ICBMEI.2011.5917838

[9] A. Tanenbaum, M. Van Steen, "Distributed Systems: Principles and Pradigms", 2nd ed. Pearson, pp 2-4, 2007.

[10] Middleware white paper [Online]. http://web.cefriel.it/~alfonso/WebBook/Documents/isgmidware.pdf [retrieved: July, 2016].

[11] P.A. Bernstein. "Middleware: a model for distributed system services". Communications of the ACM, pages 86-98, 1996.

[12] R. P. Bob Hulsebosch, Wouter Teeuw. "Middleware tintel state-of-the-art deliverable", 1999.

[13] J. M. Myerson, "The Complete Book of Middleware. Auerbach Publications", 2002.

[14] W. Emmerich, "Software engineering and middleware: A roadmap". Communications of the ACM, pages 117-129, 2000.

[15] D. S. Linthicum, "Application servers an eai", eAI Journal, July/August 2000.

[16] D. S. Linthicum, "Enterprise Application Integration", 1st edition, Addison-Weslley Professional, 1999.

[17] H. Pinus, "Middleware: Past and present a comparison", pp. 1–5, 2004.

[18] M. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for Internet of Things: a Survey", 4-5. 2016.

[19] M. Dayarathna, "Comparing 11 IoT Development Platforms" [Online]. Available from: https://dzone.com/articles/iot-software-platform-comparison [retrieved: August, 2016].

[20] Appcelerator Open Source. [Online]. Available from: https://www.appcelerator.com/mobile-app-development-products [retrieved: July, 2016].

[21] Bosch IoT Suite Benefits. [Online]. Available from: https://www.bosch-si.com/products/bosch-iot-suite/iot-platform/benefits.html [retrieved: August, 2016].

[22] Bosch IoT Suite white paper brochure. [Online]. Available from: https://www.bosch-si.com/products/bosch-iot-suite/downloads/white-paper-brochure.html [retrieved: August, 2016].

 ISBN: 978-1-61208-505-0

[23] AWS Amazon, How it works. [Online]. Available from: https://aws.amazon.com/pt/iot/how-it-works/ [retrieved: August, 2016].

[24] AWS Amazon, IoT Rules. [Online]. Available from: http://docs.aws.amazon.com/iot/latest/developerguide/iot-rules.html [retrieved: August, 2016].

[25] AWS Amazon, IoT Security Identity. [Online]. Available from: http://docs.aws.amazon.com/iot/latest/developerguide/iot-security-identity.html [retrieved: August, 2016].

[26] AWS Amazon, IoT thing shadows. [Online]. Available from: http://docs.aws.amazon.com/iot/latest/developerguide/iot-thing-shadows.html [retrieved: August, 2016].

[27] N. Sinha, K. E. Pujitha, J. S. R. Alex, "Xively Based Sensing and Monitoring System for IoT",1-4.

[28] Ericsson DCP [Online] . Available from: https://www.ericsson.com/ourportfolio/products/device-connection-platform [retrieved: August, 2016].

[29] Evrythng IoT Platform [Online]. Available from: https://evrythng.com/platform/ [retrieved: August, 2016].

[30] M. Kim et al., “Building scalable, secure, multi-tenant could services on IBM Bluemix”, 1-3. 2016.

[31] IBM Watson IoT documentation [Online]. Available from: https://docs.internetofthings.ibmcloud.com/ [retrieved: August, 2016].

[32] Cisco ParStram Analytics Automation [Online]. Available from: http://www.cisco.com/c/en/us/products/analytics-automation-software/parstream/index.html [retrieved: August, 2016].

 ISBN: 978-1-61208-505-0