https://doi.org/10.48009/4_iis_2018_26-34

Issues in Information Systems
Volume 19, Issue 4, pp. 26-34, 2018

BRINGING CLARITY TO THE JAVA IOT JUNGLE

Eldar Sultanow, Capgemini, eldar.sultanow@capgemini.com
Alina Chircu, Bentley University, achircu@bentley.edu

ABSTRACT

Everyone is talking about the Internet of Things (abbreviated as IoT) – by now the future of many industries is barely imaginable without it. IoT frameworks, technologies, cloud solutions, generic platforms and development tools are proliferating at a high rate. Even though experts recognize the need for coherent standards, the world of IoT resembles a jungle, with a myriad of different, non-standardized components. In this paper, we attempt to bring some clarity to the jungle by examining the existing challenges, summarizing the IoT frameworks, technologies and tools that are being developed for the popular Java software platform, and proposing a novel, integrated reference architecture for explaining how different IoT components can be organized by architecture layers and how they can work together. The reference architecture and related IoT components summaries can help software engineers and other IoT practitioners understand the building blocks for the design, development, implementation, operations and life-cycle management of end-to-end IoT solutions. They can also serve as organizing aids for researchers interested in IoT standards, methodologies, tools, and architectures or for educators interested in teaching these IoT concepts in the classroom.

Keywords: Internet of Things (IoT), IoT Applications, IoT Devices, IoT Networks, IoT Reference Architecture (RA), IoT Sensors, IoT Standards, Java platform, Software Development, Software Engineering

INTRODUCTION

The Internet of Things (IoT) is a system in which “things” in the physical world (inanimate objects or even living beings) possess unique IDs as well as sensing, processing and networking capabilities and can connect among themselves and with the Internet at large (Amazon, n.d.; Postscapes, n.d.). The early vision for IoT – and the IoT term – emerged in late 1990s, in the context of using RFID (Radio-Frequency Identification) in supply chains (Ashton, 2009). Many other names describe similar concepts – including, among others, web of things, industrial Internet, Industry 4.0, ubiquitous computing, ambient intelligence, machine to machine (M2M), wireless sensor networks, future Internet, cyber-physical systems, or smart things and processes (Postscapes, n.d.; Hofmann & Rüsch, 2017; Papert & Pflaum, 2017; Prasse, Nettstraeter, & ten Hompel, 2014). According to Gartner, the number of connected things in use worldwide will grow from 8.4 billion in 2017 to 20.4 billion in 2020 (Gartner, 2017).

IoT generates data from the physical world, which in turn can lead to valuable, actionable insights. Examples include measuring the vibrations of the rotor blades of a wind turbine and predicting when maintenance activities must be implemented before the blade becomes defective, adjusting lighting for unoccupied rooms, processing information from the environment and adjusting the speed of a vehicle in order to avoid accidents, monitoring patients in hospitals, or tracking and tracing products through the supply chain. This can lead to improvements in internal and outward-facing processes, and to new ways to create and capture value, thus potentially transforming entire industries (Iansiti & Lakhani, 2014; Purdy & Davarzan, 2015) and generating trillions of dollars for the global economy (Gartner, 2017; Purdy & Davarzan, 2015). However, the IoT concept is not yet fully operational in practice, and challenges still exist (Hofmann & Rüsch, 2017; Guerrero-Ibanez, Zeadally, & Contreras-Castillo, 2015).

Some of the most popular applications of IoT right now are in areas such as supply chain, logistics, and transportation (Guerrero-Ibanez et al., 2015; He, Yan, & Xu, 2014; Hofmann & Rüsch, 2017; Papert & Pflaum, 2017; Prasse et al., 2014). However, designing, implementing and operating commercial-grade systems for smart manufacturing, intelligent logistics, intelligent transportation, or connected vehicles requires strong cooperation among the many companies involved in these software engineering steps for IoT (Papert & Pflaum, 2017). Experts agree that an ecosystem view is needed in order to define the architecture of IoT systems, the interoperability standards, and how to seamlessly integrate different IoT components into an end-to-end solution (Chen & Helal, 2008; Guerrero-Ibanez et al., 2015; Kim et al., 2016; Papert & Pflaum, 2017). But even though global standards are essential for IoT success (Kim et al., 2016), the world of IoT still resembles a jungle, with many different, non-standardized components (Chen & Helal, 2008).

In this paper, we attempt to bring clarity to this jungle by examining the existing challenges, analyzing and categorizing the range of IoT components in a systematic way, and proposing a novel, integrated reference architecture for explaining how different IoT components can work together. To this end, our paper answers the following research questions: What are the components of an integrated end to end IoT solution? What are the options available to IoT developers for each architecture component, and how do these options compare? We answer these questions in the context of a major development language, Java, and its ecosystem. This flexible, device-independent language for building, deploying, running and updating applications, is consistently ranked among the top most popular programming languages in the world (Cass, 2017; Gewirtz, 2017; Stack Overflow, n.d.).

This paper is organized as follows: the second section examines the challenges inherent in the IoT environment, the third section presents the proposed architecture and summarizes the associated technologies, and the fourth section concludes with contributions and future research directions.

UNDERSTANDING IOT SOLUTION CHALLENGES

Successful IoT solutions need to take into account the specific characteristics of the IoT world – the complexity of the IoT architecture enabling the connection of many heterogeneous devices to the Internet and the complexity of software engineering (the development, implementation, operation and management of IoT software) in this heterogeneous environment. We discuss each one of these challenges in the following sections.

IoT Architecture Challenges

While a generic IoT architecture standard does not yet exist, many researchers and practitioners agree that end-to-end IoT solutions require three types of components (also called layers) for both hardware and software: sensing, networking, and applications (Chen, Xu, Liu, Hu, & Wang, 2014; Eclipse.org, 2016; Microsoft, n.d).

The sensing layer includes the IoT-enabled devices, or things, that can sense and interact with their environment. The devices usually contain a microcontroller (with one or more processing units for executing software, memory, general-purpose input/output interfaces, and specific interfaces such as wireless networking or USB) as well as other parts (such as a power source, sensors for light, heat, motion, sound or other environmental inputs, analog-to-digital and digital-to-analog converters, and various actuators – motors, smart materials, etc. - that can control the environment, usually by converting a source of energy into motion). Devices can be built from different chips connected to one circuit board, integrated on a single chip, or built on several chips packed closely together.

Additional processing capabilities can be added though field gateways, which can aggregate communication from multiple devices to the network and process and store data at the edge in order to minimize problems due to network latency and reliability.

Next, the networking layer consists of the networking infrastructure – the wired and wireless networks connecting some devices directly using common networking protocols, and the protocol gateways that help connect other, more constrained devices, to the existing networks.

Last, but not least, the application layer consists of back-end servers and enterprise applications. Communication between these back-end applications and the other architectural components is facilitated by an IoT software platform, or hub, running either inside an enterprise data center or in the cloud (Eclipse.org, 2016; Microsoft, n.d.).

This IoT architecture presents several challenges. First, IoT devices can be designed using many different hardware and software platforms (Microsoft, n.d.), resulting in tremendous heterogeneity. Second, IoT devices are generally constrained in terms of power consumption, processing capability, data storage, security, and networking capabilities (Eclipse.org, 2016; Microsoft, n.d.; Morin, Harrand, & Fleurey, 2017). For example, some devices have only intermittent, slow network connections and use simple networking protocols (Microsoft, n.d.; Morin et al., 2017; Rellermeyer et al., 2008). Third, the architecture needs to be easily scalable and support millions of devices and a very large volume (millions of events per second) and variety of data (Eclipse.org, 2016; Microsoft, n.d.).

IoT Software Engineering Challenges

IoT software engineering (the development, implementation, operation and management of IoT software applications) operates in a jungle of non-standardized hardware and software (Chen & Helal, 2008; Morin et al., 2017. The early days of pervasive computing involved trying to tame the sensors "wild jungle" with ad hoc system integration of sensors, actuators, appliances, and computers, but this led to solutions that were not scalable. Adopting a service-oriented approach does not necessarily solve the problem, since the task of designing the services corresponding to various devices could be allocated to either the device manufacturer or the system integrator (Chen & Helal, 2008).

Developing and deploying IoT software is a very complex task for several reasons (Morin et al., 2017). One reasons is the distributed nature of IoT – requiring that the software be distributed over many heterogeneous processing nodes (with different roles, capabilities and protocols, as described in the previous section).Another reason for complexity is the complementary nature of the heterogeneous nodes, which needs to be explicitly exploited by developers in order to efficiently use the computing power distributed across nodes. Last, but not least, another reason for complexity is the shared nature of IoT hardware (as several IoT and non-IoT applications have to run in the same environment without interactions, as in the case of an industrial machine running an internal production application but also communicating status information to an IoT application) (Morin et al., 2017).

Using a flexible, device-independent development language to build, deploy, run and update applications can help address some of the above-mentioned challenges. Java, a general-purpose programming language, provides such a solution (Oracle, 2015). Java is uniquely suited for IoT application development due to its wide acceptance, features, and related open-source ecosystems. First, according to IEEE, Java is currently among the top three most popular programming languages in the world (Cass, 2017), after being for many years number one worldwide. A 2017 survey of over 51,000 software developers from 213 countries and territories around the world indicates Java is one of the top 3 languages used by professional developers, and ranks above C#, Python, C++, and C (Stack Overflow, n.d.). Java is consistently ranked among the top five programming languages in many surveys based on metrics such as number of web searches, number of projects on SourceForge, Freecode, and GitHub, and number of job openings (Gewirtz, 2017). In addition, the most recent IoT developer survey from the Eclipse IoT Working Group indicates Java is the top programming language for IoT (Ramel, 2018). Second, experts argue that Java's features are superior to other programming languages (for example, in terms of automatic garbage collection, robust exception handling, built-in threading model, standards and third-party libraries for IoT) (Krasner, 2017). Java can run on a variety of IoT processing nodes (including those with limited resources), using a Java Virtual Machine (VM) to abstract networking and hardware differences among devices, gateways, and enterprise back-end systems and to make development across all these IoT entities easy and consistent. In addition, the Java architecture makes it easy to add applications and evolve the functionality of the end-to-end solution as needed over time (Oracle, 2015; Thumar, 2017). Historically, Java was used for embedded systems development because of its ability to work with constrained resources (such as processing power and memory). As IoT devices started to be developed, Java's ease of development and maintenance, code reuse abilities, easy integration with native systems, and availability of qualified developers became clear advantages (Mathews, 2002). Today, Java is gaining acceptance as a language for embedded IoT systems due to its lower cost of development and better design outcomes, according to a 2017 survey of over 900 developers (Krasner, 2017). Last, but not least, Java is the basis for many open source projects – such as Eclipse Foundation projects that provide tools, frameworks and runtimes for developing, deploying and managing applications in a variety of domains, including IoT. This helps create a vibrant ecosystem which is supported by many organizations (including Bosch, Huawei, IBM, Intel, Red Hat, SAP, Siemens, etc.) and has generated many recent open source IoT solutions (Eclipse IoT Working Group, n.d.).

Modular software design is another way to address some of the IoT software engineering challenges. The challenge is to build "a coherent application out of a possibly large collection of unrelated software modules" (Rellermeyer et al., 2008). To do so, industry practitioners usually rely on OSGi, a set of specifications and development model for modular Java applications. In OSGi, the many different components of one application can hide their implementations from other components and communicating through services (OSGi Alliance, n.d.). This enables dynamic assembly of components and simplifies software engineering in a distributed heterogeneous environment (Rellermeyer et al., 2008; OSGi Alliance, n.d.). Java and OSGi are already being used to develop solutions for important IoT applications, such as smart grids and energy management systems (Pichler, Veichtlbauer, & Engel, 2015). A new option for software modularity is the new Java Platform Module System (JPMS), which is included in the latest Java release, Java 9. JPMS was developed as a result of a multi-year long technical specification standard effort - Project Jigsaw – in order to address some of the shortcomings of OSGi and provide a simpler modularity solution for developing large Java applications (Deitel, n.d.; Jcp.org, n.d.).

Additional solutions to these challenges are currently under development as well. For example, ThingML (Internet of Things Modeling Language), a modeling language which supports collaboration among multiple hardware and software experts and co-development of IoT solutions, is currently being tested in a commercial e-health application, among others (Morin et al., 2017).

TOWARDS AN INTEGRATED IOT REFERENCE ARCHITECTURE

Given the IoT challenges identified in the previous section, understanding how the IoT components (devices, networks, and applications) work together is essential for developing end-to-end IoT solutions (Kim et al., 2016; Chen & Helal, 2008). In the following sections, we analyze and organize the different IoT components and propose an integrated IoT reference architecture.

Things

Things, also called smart objects, are IoT devices – stand-alone wearables and gadgets (such as Samsung Gear and FitBit) or embedded systems and boards (such as Arduino) (Singh & Kapoor, 2017). Popular embedded platforms for IoT devices include Arduino, Raspeberry Pi, Tessel, Espruino, Pinoccio, Beaglebone Black, and others. These platforms differ in terms of their computing power (processors, graphics, clock speed, memory), connectivity (input-output, network, or other interfaces), communication standards, and development environment (Singh & Kapoor, 2017).

Consumer things include wearables, smart phones, and smart consumer objects such as TVs, appliances, alarm systems, smoke detectors, climate controllers, mailboxes, monitoring devices for water quality or leaks, etc. Business things include smart offices objects, energy and building management devices, IoT-enabled assets, products, warehouse storage locations, etc. Industrial things include smart machines, smart transportation systems (for traffic management, parking monitoring), and smart city objects (monitoring devices for air quality, trash collection, water quality and availability, energy usage, street lighting, etc.).

Transfer Standards

Transfer standards facilitate network connections for devices. According to the IoT approach, devices are connected to each other within a network and with at least one server. The IoT devices can exchange data between each other before it is sent to a server. For example, in a warehouse, field gateways installed at various locations collect the location information of moving things (such as forklift trucks or persons equipped with active beacons) and then send this in batches to one or more servers in a consolidated manner. The IoT devices can also send their data directly to the servers. The transfer standards clarify the question on how data is exchanged in the course of this communication, where the focus is on the transfer itself and not on the content being transferred. This is similar to describing the communication mode between two individuals (post, email or telephone) without being concerned about the content or the language in which communication takes place. The most widely used transfer standards are Wi-Fi, ZigBee, Bluetooth, WiMAX, LTE, and NB-IoT.

Protocol Standards

If IoT devices communicate with each other or with a server then they must “understand” each other. If a device sends JSON-coded environmental data (e.g. temperatures or air pressure) to the server, but the server is expecting business documents (like invoices or master data) in EDIFACT format, communication is not possible. The corresponding “agreements” between devices and servers are called protocols (or messaging protocols for clarity), which help answer questions such as: “which type of data is transferred (semantics)?”, “how is it formatted” (syntax)?”, “how does one identify that a message has been lost?,” etc. These questions are independent of how the data is transferred, which is addressed by the transfer standards. The most widely used protocol standards include MQTT, MQTT-SN, AMQP, XMPP, CoAP, HTTP, OPC UA, and LWM2M. As messaging protocol standards with different characteristics (message size, overhead, power consumption, bandwidth, reliability, security, etc.) proliferate, protocols need to be carefully evaluated in order to determine their fit for a particular IoT application (Naik, 2017).

Protocol-compliant Communication
Now that it is clear what language IoT application components speaks - be it sensors, RFID readers, objects tagged with active beacons and of course the servers - and which protocol they use for messaging, we must define the components which implement the communication conforming to the protocol. It is these components that implement device/client-side and server-side communication using any of the protocols described above.

Most implementations of the protocols presented in the previous section (such as AMQP, MQTT, or XMPP) use a broker to facilitate IoT communication (ADLINK, 2016). A broker provides trusted message prioritization, routing, filtering and delivery between a publisher and a subscriber. An IoT device, gateway or cloud application can each act as either publisher or subscriber. Broker configurations include a centralized broker system (with all messages going through one server), a centralized multi-broker system (with several brokers on different servers, each with its own message queue), and a decentralized broker system (which has no central server and uses the IP multicast protocol coupled with local client-side functionality for persistence, security and transactions) (ADLINK, 2016). Examples of brokers include HiveMQ, Mosquitto, Moquette, and Pivotals RabbitMQ.

Other Tools, Frameworks and Platforms
IoT device control and integration frameworks include openHAB and Eclipse Edje. IoT simulation frameworks include Eclipse Ditto, which provides access to digital twins (software abstractions of real-world devices used for analyzing past performance, optimizing current operations, and predicting future repair needs for industrial machines). Tools for modelling, development and implementation of IoT applications include Eclipse 4diac, Eclipse Vorto, tools developed for other projects and integrated in Eclipses Open IoT Stack for Java, Eclipse hawkBit, Papyrus for IoT, SensIDL, Reactive Blocks, Node-RED, Virtual Developer, and Resin.io. Frameworks for IoT application development include various Eclipse projects, such as Concierge, Hono, Milo, Californium, Leshan, Kura, SCADA, OM2M, Wakaama, and Ponte, and the cloud-based platform Temboo. Finally, IoT cloudiIntegration solutions which facilitate the collection, processing and analysis of IoT data using scalable computing platforms include AWS IoT, Microsoft Azure IoT, and IBM Watson IoT Services.

Integrated IoT RA
To show how the previously-discussed IoT components work together to deliver end-to-end IoT solutions, we develop an integrated reference architecture that organizes the components by architecture layers. Figure 1 summarizes our results. The left part of the figure (titled development) presents the device design and application development architecture, while the right part of the figure (titled production) presents the operational (solution in action) architecture. The development architecture focuses on the IoT frameworks, technologies and tools that support the software development lifecycle steps (specify, develop, and build/deploy, numbered 1 to 3in light grey in the Figure 1), and highlights how different actors (device manufacturers, platform vendors, and software designers and developers for the different IoT components) can use them. The operational architecture highlights the different architectural layers for things (constrained devices) and gateways (hardware, firmware and run-time environment), networking standards (transport and protocol), and cloud applications (back-end and front-end), and provides examples of frameworks, technologies and tools available in each layer. In addition, the operational architecture provides examples of how different components support the steps necessary for an end-to-end IoT solution (produce data, collect it, queue it, process it, store it, and utilize it , numbered 1 to 6 in dark grey in Figure 1). By integrating the two perspectives – development and operational – the RA becomes a high-level map that can help organizations understand all the necessary building blocks for an end-to-end IoT solution.

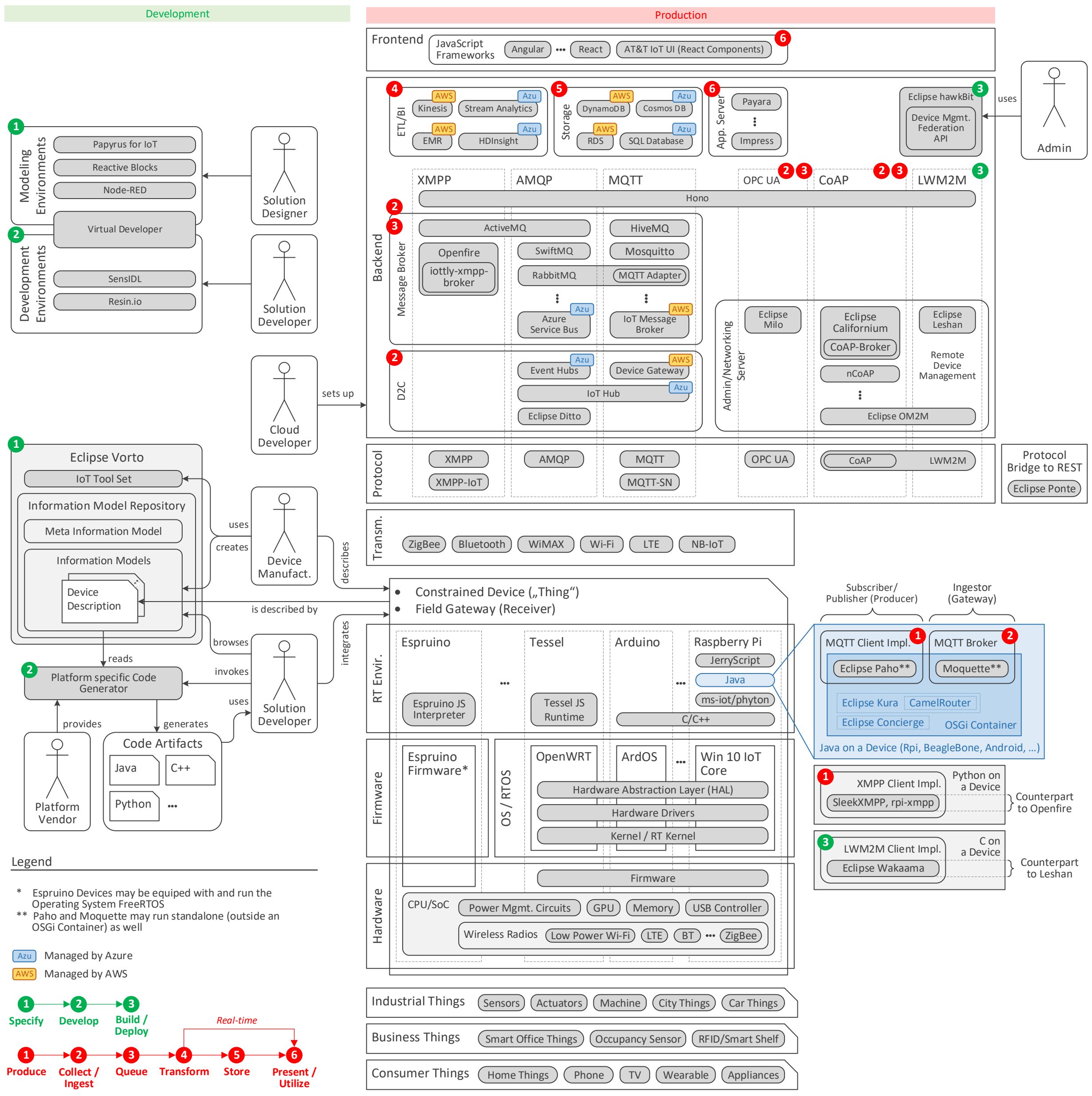

Figure 1. Reference architecture for IoT with Java

CONCLUSIONS AND FUTURE WORK

In this paper, we propose a solution for making sense of the complex IoT jungle of heterogeneous devices and standards – a reference architecture for end-to-end IoT solutions. While previous research has addressed standards within various IoT categories such as sensors (Chen & Helal, 2008) and networking (Pichler et al., 2015; Naik, 2017), no integrated end-to-end framework for the development, implementation, and management of IoT applications exists. To our knowledge, our paper is the first attempt to provide such a framework in the form of an integrated reference architecture for IoT. We build the reference architecture from the perspective of Java – a widely-used software engineering platform that is very well suited for IoT.

The reference architecture detailed in this paper provides a comprehensive overview of the required components, organized by architecture layers, for developing IoT applications with Java – such as network standards, protocols, frameworks and development tools. The analysis provides a comparison of the features of various architectural components. To develop the architecture, we followed the steps in the design science process including defining the research gap (analyzing the existing technologies for IoT, and developing and describing the artifact (the reference architecture), and showing how the artifact fills the research gap in IoT reference architecture and how real world components can be used to fill this gap (Riege, Saat, & Bucher, 2009). The proposed architecture can be helpful in choosing the best development options for achieving high scalability and reliability of IoT applications in a variety of use cases and on devices with limited resources (such as limited processing power and memory). Our analysis can help software engineers and other IoT practitioners who make decisions about the design, development, implementation, operations and life-cycle management of end-to-end IoT solutions. The framework can also help researchers interested in IoT reference architectures and in developing new IoT standards, methodologies and tools.

This paper has several limitations. First, we focus only on the Java IoT environment. Future papers should investigate similar architectures based on other popular languages, such as C++ or Python. Second, we focus on the larger picture when building and describing the end-to-end IoT architecture, since we found that this approach is missing from previous research. However, this makes it harder to have a detailed discussion of the characteristics of the architectural components and their interactions without exceeding article page limits. Third, our research provides some but not all types of artifact testing recommended for design science (Riege et al., 2009). We provide an evaluation of the reference architecture against an identified research gap and an evaluation of the real world components necessary to fill the research gap. We were not able to evaluate the artifact against the real world by allowing potential users to try it out under real world conditions and comment on its value. Given the complexity of the reference architecture, we believe this is a task better suited for a separate investigation. Last, but not least, we do not go into detail regarding security – which is an important aspect of IoT development, deployment and management. Note, however, that security solutions for the device, network and application levels of the framework already exist – such as authentication, trust establishment, federated architectures, and security awareness at the user level (Mahmoud, Yousuf, Aloul, & Zualkernan, 2015). In addition, at the network level, specific security challenges for many of the protocols discussed in this paper have been identified, and solutions have been proposed as well (Shin, Kobara, Chuang, & Huang, 2016; Singh, Rajan, Shivraj, & Balamuralidhar, 2015; Zamfir, Balan, Iliescu, & Sandu, 2016).

This paper contributes to the IoT literature by developing an IoT reference architecture and providing an analysis of its components, based on a comprehensive review of the available IoT technologies in the Java space. The reference architecture presented in this paper is a proof of concept, and we will continue to expand it as new IoT developments occur. Future research can test the framework in practice by looking at enterprise-level value, software engineering process improvements, or value from an individual software developer perspective.

REFERENCES

ADLINK (2016). *Messaging technologies for the Industrial Internet and the Internet of Things*. Retrieved from: http://www.prismtech.com/sites/default/files/documents/Messaging-Whitepaper-051217.pdf

Amazon (n.d). Was ist das Internet of Things (Internet der Dinge)?. Retrieved from: https://aws.amazon.com/de/iot/what-is-the-internet-of-things/

Ashton, K. (2009, June 22). That 'internet of things' thing. *RFID Journal*. Retrieved from: http://www.rfidjournal.com/articles/view?4986

Cass, S. (2017, July 18). The 2017 top programming languages. *IEEE Spectrum*. Retrieved from: https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

Chen, C., & Helal, S. (2008). Sifting through the jungle of sensor standards. *IEEE Pervasive Computing,* 7(4), 84-88.

Chen, S., Xu, H., Liu, D., Hu, B., & Wang, H. A Vision of IoT: Applications, challenges, and opportunities with China perspective. *IEEE Internet of Things Journal.* 1(4), 349-359.

Deitel, P. (n.d.). Understanding Java 9 modules. Retrieved from: https://www.oracle.com/corporate/features/understanding-java-9-modules.html

Eclipse.org (2016). *The three software stacks required for IoT architectures*. Retrieved from: https://iot.eclipse.org/resources/white-papers/Eclipse%20IoT%20White%20Paper%20-%20The%20Three%20Software%20Stacks%20Required%20for%20IoT%20Architectures.pdf

Eclipse IoT Working Group (n.d.). About us. Retrieved from: https://iot.eclipse.org/

Gartner (2017, February 7). Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016. Retrieved from: https://www.gartner.com/newsroom/id/3598917

Gewirtz, D. (2017, October 4). Which programming languages are most popular (and what does that even mean)?. *ZDNe*t. Retrieved from https://www.zdnet.com/article/which-programming-languages-are-most-popular-and-what-does-that-even-mean/

Guerrero-Ibanez, J.A.; Zeadally, S.; Contreras-Castillo, J. (2015). Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and Internet of Things technologies. *IEEE Wireless Communications, 22*(6), 122-128.

He, W., Yan, G., & Xu, L. D. (2014). developing vehicular data cloud services in the IoT environment. *IEEE Transactions on Industrial Informatics, 10*(2), 1587-1595.

Hofmann, E., & Rüsch, M. (2017). Industry 4.0 and the current status as well as future prospects on logistics. *Computers in Industry, 89*, 23-34.

Iansiti, M., & Lakhani, K.R. (2014). Digital ubiquity: How connections, sensors, and data are revolutionizing business. *Harvard Business Review*, *92*, 91-99.

Jcp.org. (n.d.) JSR 376: JavaTM platform module system. Retrieved from: https://www.jcp.org/en/jsr/detail?id=376

Kim, J., Yun, J., Choi, S.C., Seed, D.N., Lu, G., Bauer, M., Al-Hezmi, A., Campowsky, K., & Song, J. (2016). Standard-based IoT platforms interworking: Implementation, experiences, and lessons learned. *IEEE Communications Magazine, 54*(7), 48-54.

Krasner, J. (2017). Real-time Java and the IoT: Bridging the chasm between it and embedded connected devices. *Embedded Market Forecasters.* Retrieved from: https://www.ptc.com/-/media/Files/PDFs/Developer-Tools/JavaMay-2017Final-EMF-White-Paper-RT-Java--IoT.pdf?la=en&hash=85F2DB9D0961AA735B465CD16CA8546D89ABF723

Mahmoud, R., Yousuf, T., Aloul, F., & Zualkernan, I. (2015). Internet of Things (IoT) security: Current status, challenges and prospective measures. *Proceedings of the IEEE International Conference for Internet Technology and Secured Transactions (ICITST)*, London, U.K., 336-341.

Mathews, M. (2002, June 28). The future of Java for embedded applications. *EE Times*. Retrieved from https://www.eetimes.com/document.asp?doc_id=1200917

Microsoft (n.d). Overview of the Azure IoT Hub service. Retrieved from: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-what-is-iot-hub#iot-device-connectivity-challenges

Morin, B., Harrand, N., & Fleurey, F. (2017). Model-based software engineering to tame the IoT jungle. *IEEE Software, 34*(1), 30-36.

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. *Proceedings of the IEEE International Systems Engineering Symposium (ISSE)*, Vienna, Austria, 1-7.

Oracle (2015). Java and the Internet of Things: Automating the industrial economy. Available online: http://www.oracle.com/us/solutions/internetofthings/java-iot-industrial-automation-2430562.pdf

OSGi Alliance (n.d.). The dynamic module system for Java. Retrieved from: https://www.osgi.org/

Papert, M., & Pflaum, A. (2017). Development of an ecosystem model for the realization of Internet of Things (IoT) services in supply chain management, *Electronic Markets, 27*(2), 175–189.

Pichler, M., Veichtlbauer, A., & Engel, D. (2015). Evaluation of OSGi-based architectures for customer energy management systems. *Proceedings of the IEEE International Conference on Industrial Technology (ICIT)*, Seville, Spain, 2455-2460.

Postscapes (n.d.). Best Internet of Things definition. Retrieved from: https://www.postscapes.com/internet-of-things-definition/

Prasse, C., Nettstraeter, A., & ten Hompel, M. (2014). How IoT will change the design and operation of logistics systems. *Proceedings of the IEEE International Conference on the Internet of Things (IOT)*, Cambridge, MA, USA, 55-60.

Purdy, M., & Davarzan, L. (2015). The growth game-changer: How the Industrial Internet of Things can drive progress and prosperity. *Accenture*.

Ramel, D. (2018, April 30). IoT dev survey names AWS top cloud service, Java no. 1 language. *ADTMag*. Retrieved from: https://adtmag.com/articles/2018/04/30/iot-survey.aspx

Rellermeyer, J.S., Duller, M., Gilmer, K., Maragkos, D., Papageorgiou, D., & Alonso, G. (2008). The software fabric for the Internet of Things. In Floerkemeier C., Langheinrich M., Fleisch E., Mattern F., & Sarma S.E. (Eds.), *The Internet of Things* (pp. 87-104), Berlin, Germany: Springer.

Riege, C., Saat, J., & Bucher, T. (2009). Systematisierung von Evaluationsmethoden in der gestaltungsorientierten Wirtschaftsinformatik. In Becker, J., Krcmar, H., & Niehaves, B. (Eds.), *Wissenschaftstheorie und gestaltungsorientierte Wirtschaftsinformatik* (pp. 69-86), Heidelberg, Germany: Springer.

Shin, S., Kobara, K., Chuang, C. C., & Huang, W. (2016). A security framework for MQTT. *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, Philadelphia, PA, USA, 432-436.

Singh, K., & Kapoor, D. (2017). Create your own Internet of Things: A survey of IoT platforms. *IEEE Consumer Electronics Magazine, 6*(2), 57-68.

Singh, M., Rajan, M. A., Shivraj, V. L., & Balamuralidhar, P. (2015). Secure MQTT for Internet of Things (IoT). *Proceedings of International Conference on Communication Systems and Network Technologies (CSNT)*, Gwalior, MP, India746-751.

Stack Overflow (n.d.). Developer Survey Results 2017. Retrieved from: https://insights.stackoverflow.com/survey/2017

Thumar, C. (2017, August 8). The role of Java in IoT. *jaxenter*. Retrieved from https://jaxenter.com/role-java-iot-135590.html

Zamfir, S., Balan, T., Iliescu, I., & Sandu, F. (2016). A security analysis on standard IoT protocols. *Proceedings of the International Conference on Applied and Theoretical Electricity (ICATE)*, Craiova, Romania, 1-6