Interworking Models and Automated Conformance Testing for Internet of Things

JaeYoung Hwang

August 2020

Department of Computer and Information Security

The Graduate School

Sejong University

Interworking Models and Automated Conformance Testing for Internet of Things

JaeYoung Hwang

A dissertation submitted to the Faculty of the Sejong University in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer and Information Security

August 2020

Approved by Professor JaeSeung Song

Interworking Models and Automated Conformance Testing for Internet of Things

by

JaeYoung Hwang

Approved ———————————————————————

JaeHo Kim, Chair of the committee

Approved ———————————————————————

JeongWook Seo, Member of dissertation committee

Approved ———————————————————————

HyoKeun Choi, Member of dissertation committee

Approved ———————————————————————

Ki-Woong Park, Member of dissertation committee

Approved ———————————————————————

JaeSeung Song, Advisor

Abstract

At present, the Internet of Things (IoT) technology is considered to be a major technology that will solve various social problems with Artificial Intelligence (AI) and 5G, and lead the fourth industrial revolution. In addition, various objects such as sensors, home appliances, actuators, and self-driving cars are connected that can interact with each other through the Internet and are used in various fields such as smart grid, home automation, and medical fields. Therefore, the IoT is regarded as one of the promising technologies with great growth potential, and many countries including the European Union (EU), are intensively investing in Information and Communication Technology (ICT) research.

However, there are many difficulties in providing a scalable and stable IoT service that must be considered in providing the above services. Most IoT platforms and systems developed so far provide IoT services using the manufacturer's own data models and Application Programming Interfaces (APIs), so interoperability with other platforms is not possible. If interoperability is not guaranteed, IoT devices or platforms cannot interoperate or exchange data with systems developed according to different standards. Therefore, changing a system that has already been deployed is costly and also can lead to economic and technical problems such as delays in introducing large-scale IoT technologies and increasing operating costs. With regard to the IoT standard testing, for assuring the interoperability of IoT services, conformance and interoperability testing are highly important to ensure that independent implementation based on the same or different standard are interoperable. In addition, IoT devices or platforms based on various standards and protocols are

currently deployed and operated in homes, factories, and cities. While these devices and platforms help in terms of human convenience and productivity, service errors can directly affect human safety. Therefore, in order to support interoperable and stable IoT services, devices or platforms must be tested through conformance and interoperability testing before they are actually deployed. However, the method of manually testing the IoT application standard has problems because some testing procedures must be directly performed by a human. To conclude, it leads to an increase in errors occurring in the testing process as well as total testing time.

Focusing on the above problems, this dissertation describes how to solve the problems arising from IoT interoperability and conformance testing based on the oneM2M IoT standard. First, in order to solve the interoperability problem, three Interworking Models (IWMs) are presented, and a guide to select the appropriate IoT platform linking method according to the conditions is provided by describing the characteristics and advantages and disadvantages of each model. In addition, for practical examples, the main technologies of the IWMs are described in combination with the smart city field, where IoT technology is currently being actively researched. In addition, research on automated IoT standard testing methods and procedures based on triggering messages is conducted. Through the developed methods and procedures, human intervention can be minimized, which means it reduces errors that can occur in the testing process as well as total testing time.

In conclusion, by conducting the above research, it is not only possible to effectively provide interworking between IoT systems by solving interoperability problems due to the diversity of current IoT standards but also realizable to

contribute to the rapid propagation of IoT standards by reducing the time required of the testing and certification process for the testing certification body.

Keywords: IoT, Interoperability, IoT Conformance Testing, oneM2M

Contents

List of Figures

List of Tables

List of publications

This dissertation is composed of overview of the following publications and current ongoing research.

1. Hwang, J., Aziz, A., Sung, N., Ahmad, A., Le Gall, F., & Song, J. (2020). AUTOCON-IoT: Automated and Scalable Online Conformance Testing for IoT Applications. *IEEE Access*, 8, 43111-43121.

2. Hwang, J., An, J., Aziz, A., Kim, J., Jeong, S., & Song, J. (2019). Interworking Models of Smart City with Heterogeneous Internet of Things Standards. *IEEE Communications Magazine*, 57 (6), 74-79.

3. An, J., Le Gall, F., Kim, J., Yun, J., Hwang, J., Bauer, M., ... & Song, J. (2019). Toward global IoT-enabled smart cities interworking using adaptive semantic adapter. *IEEE Internet of Things Journal*, 6 (3), 5753-5765.

4. Kim, H., Ahmad, A., Hwang, J., Baqa, H., Le Gall, F., Ortega, M. A. R., & Song, J. (2018). IoT-TaaS: Towards a prospective IoT testing framework. *IEEE Access*, 6, 15480-15493.

5. Hwang, J., An J., Joo H., Lee C., Song, J. (2017), “Development and Application of Interoperability Techniques with Semantics for Global Internet of Things (GIoTs), *The Journal of Korean Institute of Communications and Information Sciences*, 42 (11), 2208-2216.

1 Introduction

In recent years, the Internet of Things (IoT) has been recognized as a major technology that can solve various social problems, and lead the fourth industrial revolution by collaborating with Artificial Intelligence (AI), 5G, and many other things [1]. In addition, varied IoT devices such as home appliances, sensors, actuators, and unmanned vehicles can be connected and interacted with each other through the Internet, and are being used in various fields such as home automation, smart grid, traffic management, and medical aid [2]. According to the McKinsey's "The internet of things mapping the value beyond the hype report" [3], it is expected that the IoT has a total potential economic impact of $3.9 trillion to $11.1 trillion per year in 2025. However, to efficiently use the IoT technologies in the real fields such as smart homes, smart factories, and smart cities, problems that IoT has must be analyzed and addressed.

1.1 Towards the Internet of Things standards

With the development of the IoT industry, various standards for the IoT have been being developed and announced. There are various IoT standards such as oneM2M, Next Generation Standard Interface (NGSI), Open Connectivity Foundation (OCF), Global Standard 1 (GS1). However, IoT services are not using only one standard and these are composed of various standards. In conclusion, in order to support interoperability for platforms and devices developed using various IoT standards, an analysis of the relevant standards is required.

oneM2M is a partnership project of eight national SDOs including Europe, India, Japan, South Korea, and North America to develop global IoT / M2M (Machine-to-machine) common platform standards. oneM2M has been developing a standard technology by targeting a common platform that can be used in various service areas without being dependent on a specific IoT domain. In this context, oneM2M is developing the reference architecture to assist a large number of IoT services such as smart factories, smart cities, smart cars. In terms of the deployment, the oneM2M platform supports useful approaches more than data exchanges for applications. oneM2M platforms provide the interworking features that can allow other IoT platforms or devices to connect to the oneM2M platform. That is, backend systems or external IoT systems can interwork by using an oneM2M standard interworking approach. For example, currently, oneM2M is working with other promising IoT standards such as Open Connectivity Foundation (OCF) and Leightweight Machine to Machine (LwM2M). In addition, beyond integrating with IoT standards, the 3rd Generation Partnership Project (3GPP) is being researched for supporting the interworking feature.

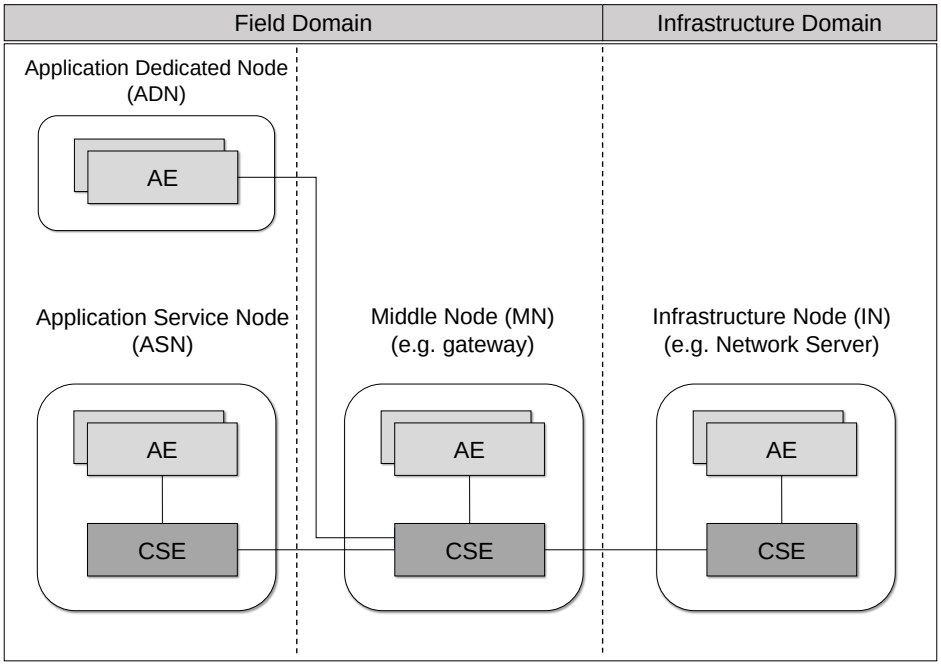

Fig. 1: oneM2M reference architecture

To implement the oneM2M IoT systems, four functional entities called nodes are defined and these are described in Fig. 1 [4], [5]. Infrastructure Node (IN) represents the IoT server in the reference architecture. IoT gateway can be illustrated as a Middle Node (MN). IoT sensors and actuators are formed in oneM2M environment as an Application Dedicated Node (ADN), Application Service Node (ASN). In addition, oneM2M adopts the Resource Oriented Architecture (ROA) model. that is all IoT devices based on oneM2M standard can be handled as resources based on hierarchical structure [6], and each resource in the structure has an identifier to classify the resources. In this regard, several resources have been being defined, but, in this dissertation, a fundamental resource to make the IoT services are considered as described in Fig. 2. [7]. Common Service Entity (CSE) supports common service functions including

registration, discovery, data management. Application Entity (AE) presents the various service logic type. <Container> play a role as data instances. <contentInstance> is storing the real sensored value. <CSE> is a root of all oneM2M child resource, and functional nodes must be comprised of at least one <CSE> or <AE>. The <subscription> resource is created under the resource that <subscription> resource wants to check the status of. The <subscription> has policies for notification and it includes the information of which, when, and how notifications are sent [8].

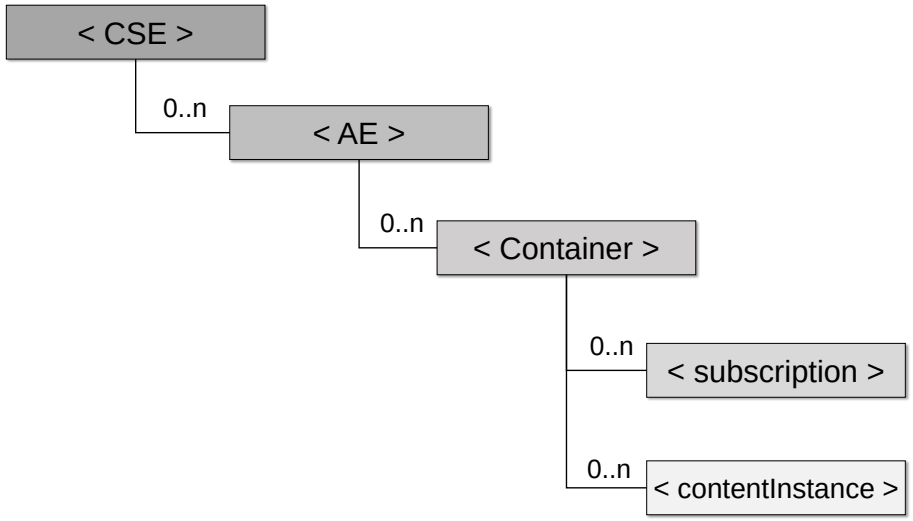

Fig. 2: oneM2M resource structure

FIWARE is an open cloud-based infrastructure for Future Internet applications and services, consist of different building blocks is called General Enablers (GE) are essentially, reusable software system [9]. Therefore, by integrating several GEs service providers can provide users with FIWARE IoT services such as smart city and smart factory and so on. Currently, based on the FIWARE, Santander smart city is being operated located in Spain. In this smart

city, it consists of a variety of actuator and sensors can be around 3000 Zigbee devices, 200 devices (mainly mobile) with GPRS communication capabilities, 2000 joint RFID tag/QR code labels deployed both at static locations (streetlights, facades, bus stops) and on-board of mobile vehicles (buses, taxis).

Basically, FIWARE is using Next Generation Service Interface (NGSI) standard. In 2009/2010 Open Mobile Alliance (OMA) specified a set of interfaces called NGSI and defined two interfaces are NGSI-9 (Context Entity Discovery Interface) and NGSI-10 (Context Information Interface) [10], [11]. Figure. 3 is showing the example of the data model of NGSI standard. The core concept of NGSI context interfaces is `Entity`. A lot of physical objects like buildings, rooms, persons can be presented as `Entity` and these are identified by using Entity identifier and a type. The `Attribute` is composed of key-value and represents the real value of the `Entity`. `Metadata` is additional information of the entity and users can attach their own `Metadata` to entity's `Attribute`.

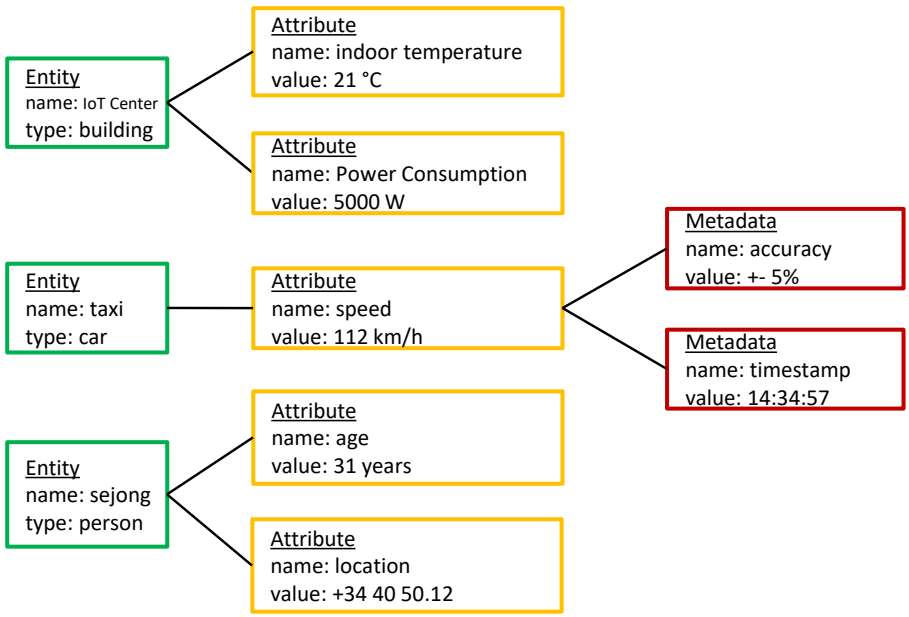

Fig. 3: Data model of NGSI

Figure. 4, 5 presents the operations supported by the NGSI-9 and NGSI-10 interface [12]. Context Management Component (CMC) is defined as an intermediary that manages the context information. In NGSI-10 API operation, Context producers produce context information and context consumers consume context information. Update operation in NGSI-10 allows the actor to update the context value in the CMC. An actor1 can subscribe to be notified from actor2 whenever a certain notification condition occurs.

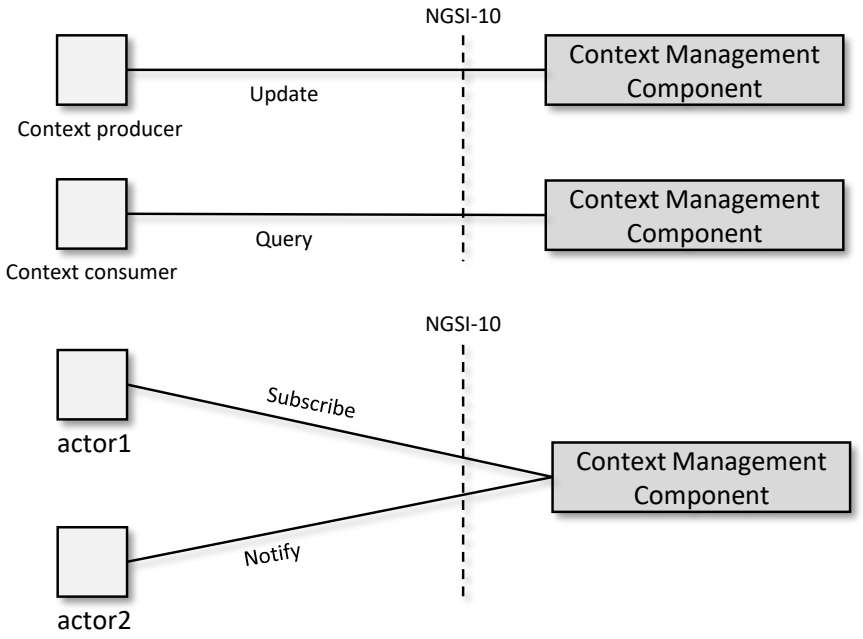

Fig. 4: NGSI-10 API operations

In the NGSI-9 interface, Actors can register themselves to the CMC, specifying what context information they can give information for, but not include the actual information. Later, the CMC can use this registration information for finding the relevant producers of context information when receiving the requests. In addition, there is a discovery operation that returns currently registered producers, and subscription operations for notifying the availability of the context producers that are eligible for the request.

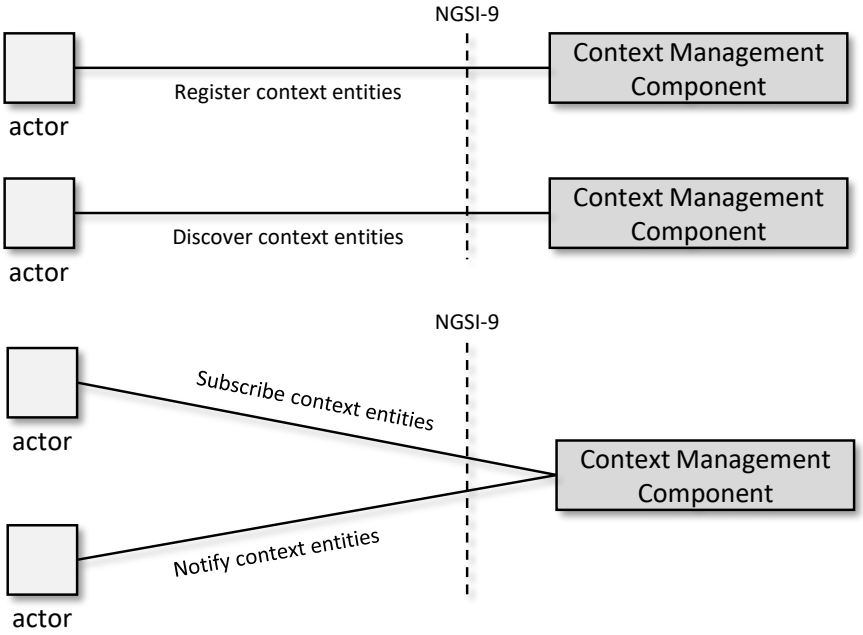

Fig. 5: NGSI-9 API operations

The FIWARE context broker called Orion is a C++ implementation of the NGSI REST API binding. Orion Context Broker enables users to manage the entire lifecycle of the context information by providing updates, queries, registrations, and subscription operations. Orion Context Broker allows users to create and manage context elements through updates and queries. In addition, users can also subscribe to context information to be notified when context information changes. It is worth nothing that FIWARE is an opensource detail information can be found here (https://github.com/telefonicaid/fiware-orion).

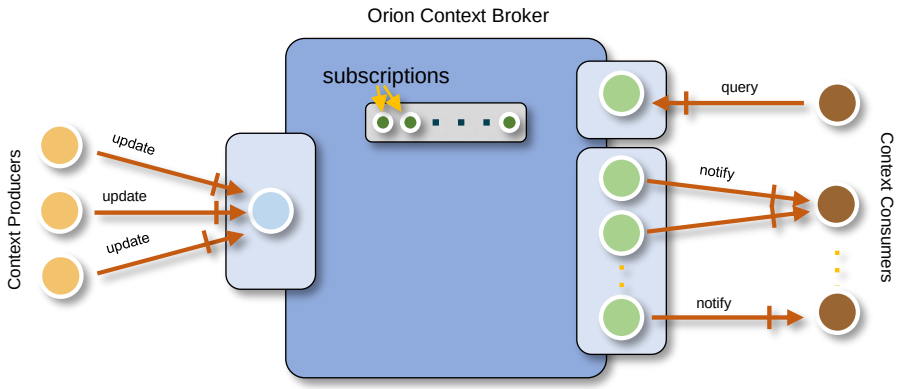

Fig. 6: FIWARE context broker

In standard activity perspective of FIWARE, at present, the Context Information Management (CIM) group which is one of the Industry Specification Group (ISG) under the European Telecommunications Standardization Organization (ETSI) is developing and maintaining the related specifications, and CIM selects the smart cities as the main application domain and it aims at defining the context-based data management API and data platforms. Technically, CIM is developing a context-based API based on FIWARE, which is being developed as a future Internet project in Europe. However, there are differences in the support perspective that CIM supports ontology-based information modeling for semantic integration and JavaScript Object Notation for Linked Data (JSON-LD). The preliminary API specification was released on April 18, and if the CIM standardization result is released continuously, it is expected that the specification for interworking with IoT platform standards such as oneM2M will be established in the future.

OCF is a specification and open source project for delivering the interconnectivity of developers, manufacturers and end-users [13]. There are four

founder members including Samsung, Intel, Cisco, and MediaTek, and at present Qualcomm, Microsoft joined OCF as a new member. OCF is providing the open-source project called IoTivity Project and it is being developed based on the OCF specification. In addition, OCF is running on varied software platforms such as Tizen, Android, iOS, Linux.

GS1 is an international organization that develops identification and distribution logistics standards using bar code and radio frequency identification (RFID) technology for global supply and demand and logistics chains. Auto-ID Labs, an international joint research institute of the GS1, developed the Open Language for the Internet of Things (Oliot) as an open-source project to develop the GS1 standard.

In addition to the standards described above, ISO/IEC JTC1 WG11, established in October 2015, is developing standards for the smart city ICT reference framework and high-level ontologies for smart cities. In addition, ITU-T formed the ITU-T Focus Group on Smart Sustainable Cities (FG-SSC) and contributed 21 relevant standard documents for activities related to smart city standardization and based on previous research, they have established a new smart city standardization organization called Study Group 20 (SG20). However, such de facto or international standardization organizations have different goals and directions for standardization, and the lack of agreement between standardization organizations raises the issue of IoT interoperability.

1.2 Problem statements

This dissertation has conducted preliminary literature research for analyzing what kind of features are essential for successfully constructing the IoT environments. As a result, six candidate features from existing IoT relevant research were gathered. More detailed features' information is described in Table 1 based on literature and these features are as follows: Interoperability, Availability, Scalability, Reliability, Performance, Security.

	Interoperability	Availability	Scailability	Reliability	Performance	Security
Ahlgren, B. et al. [1]	O	O	O	–	–	–
Gluhak, Alexander, et al.[14]	O	–	O	–	–	–
Yun et al. [15]	O	O	O	–	–	–
Sarkar, Chayan, et al. [16]	O	O	–	–	–	–
R Roman et al. [17]	O	–	–	O	–	O
H Arasteh et al. [18]	O	–	O	O	–	O
Al-Fuqaha, Ala, et al. [19]	O	O	O	O	O	O
Mehmood, Yasir, et al. [20]	O	O	O	–	–	O
Wenge, Rong, et al. [21]	O	–	–	–	–	O

Table 1: Main features for the IoT environments

- **Interoperaiblity:** End-to-End interoperability is a challenge for the smart cities interworking due to the need to handle a large number of heterogeneous things that belong to different platforms.
- **Availability:** Availability of the smart city services must be realized to provide anywhere and anytime services for citizens even users are moving different smart cities.
- **Scalability:** The scalability of the smart cities refers to the ability to add

new IoT enabled smart cities, services and functions for customers without negatively affecting the quality of existing services.

- **Reliability:** Reliability ensures the proper working of the system in terms of its specification. Reliability's aim is to increase the success probability of IoT service delivery.

- **Performance:** The IoT devices need to be evaluated to provide the best possible performance. Smart cities must not make performance degradation of smart city services.

- **Security** Since the data in a smart city includes individual, enterprise, and state economic data, providing the right level of security for sensitive information between smart cities is an important feature.

As shown in Table 1, the **`interoperability`** problem is pointed out as a problem to be solved first. More specifically, since most IoT platforms developed so far provide IoT services using the platform manufacturers' own standards and data models, there is a problem that compatibility with other platforms is not guaranteed [22], [23]. If interoperability is not guaranteed, IoT platforms cannot interoperate with or exchange data with things developed according to different standards, and cause technical and economic problems such as delays in the introduction of large-scale IoT technology and increased operating costs. In addition, McKinsey's " 2015 THE INTERNET OF THINGS MAPPING THE VALUE BEYOND THE HYPE " report [3] showed that the IoT platform should be supported for interoperability in order to achieve additional economic effects of more than 4 trillion dollars annually. Since most IoT platforms currently being deployed in various industry fields are not built using a

single unified IoT standard, so they cannot interoperate with each other. Therefore, the cost of replacing an already deployed system can be substantial to support other IoT standards. For example, in such an environment, in order to deliver IoT services developed in Europe, new services must be developed in South Korea.

With regard to the IoT standard testing, for assuring the interoperability of IoT services, conformance and interoperability testing are highly important to ensure that independent implementation based on the same or different standard are interoperable [24]. Furthermore, IoT services are pervasive in people's lives and impact on every industry area. Therefore, providing a sustainable and stable quality of IoT services is essential and to avoid fatal risks caused by the dysfunctional devices or not working products with others [25]. Therefore, testing a huge amount of IoT devices and platforms in real-time is becoming one of the most important parts of IoT technologies [26], [27]. To conclude, it is important for the IoT services to deploy interoperable and reliable IoT platforms or IoT devices. Therefore, adequate IoT testing approaches are needed to support interoperable and reliable IoT environments. However, as challenges for IoT application testing, most of the studies regarding IoT testing do not discuss standard-based IoT testing. In addition, regarding the testing intervention by a human, in the situation of the traditional IoT testing, the problem is that the human manually tests the IoT applications. It is impossible to test a lot of applications manually and human intervention can make side effects on the testing results. Therefore, the sure-fireway is that it makes the testing procedures automated.

1.3 Contributions

In this dissertation, mainly two research areas are studied for IoT environments. First, to solve the IoT standard interoperability issues, three interworking models are proposed. To interconnect the existing IoT-based services without modification or changes existing systems, by referencing the interworking models proposed in this dissertation, IoT service developers could get insight regarding the IoT standard interworking. In addition, because an evaluation matrix of the pros and cons of the three interworking models is included, according to the requirements of the IoT services, developers also can make a good selection for the desirable IWMs. For proving the above contributions, as a practical example, IWMs are explained based on the smart city scenario.

Regarding the IoT conformance testing approaches, the fully-automated testing approach is studied and described in this dissertation. In detail, standard-based triggering messages and automated testing procedures are newly developed. As a result, this approach fully automates testing procedures for IoT applications; therefore, many IoT applications can be easily tested without human intervention. In addition, together with an experimental evaluation, triggering-based IoT testing mechanisms show the advantage in terms of performance and testing time. Finally, the proposed triggering mechanism is reflected in one of the normative specifications as a standard-based testing mechanism in oneM2M IoT standard. Therefore, the triggering IoT testing feature based on the proposed approach in oneM2M has already been included in designated oneM2M testing tools. In this context, the automated testing approach contributes to the rapid propagation of IoT standards by reducing the time

required of the testing and certification process for the testing certification body. Additionally, by empirical analyzing existing IoT testing-related projects, a conceptual architecture of the IoT testing framework that can increase the reliability of the three interworking models presented above is proposed as a future work.

1.4 Dissertation outline

This dissertation is mainly organized as follows. The section 2 describes why interworking among IoT systems is needed by exemplifying the scenario based on smart cities and three interworking models are included as a solution for this issue. The section 3 illustrates an automated IoT testing mechanism for enhancing the traditional testing approach when conducting IoT application testing. Finally, the last section concludes this dissertation and proposes the future work.

2 Interworking models for IoT standards

Interoperability means systems or products can work with other systems or products without restriction and special effort and it is the most fundamental value and a cornerstone of the open Internet [28]. Lack of interoperability makes technical and business challenges such as increasing costs, slowing the introduction of new technologies and impossibility to connect with each other. In addition, technology cluster and data silos which cannot be communicated with other domains have been created. Therefore, it is important to make the existing IoT-based services interwork with other IoT services without changing existing systems. Furthermore, the new IoT platforms to be developed should also take interoperability issues into consideration.

In this regard, this section first explains interworking issues by using the two smart cities as an example. Based on that scenario, three interworking models with the aim of providing guidelines for interworking is introduced. For comparing the pros and cons of each model, an interworking model evaluation is also included.

2.1 Towards the IoT standards interoperability

As so far, this dissertation talks about interoperability issues but, there is no specific definition of what smart cities interoperability is. Therefore, the smart city interoperability definition is described. In addition, for more practical description, a smart city interworking scenario that is telling that why interworking between different smart cities is important is illustrated.

At present, most recent IoT-based smart cities are designed to support

interoperability between IoT devices and services within the city. Therefore, data generated by an IoT device can be accessed by multiple IoT services using the same standards in the city. This is an intra-city interoperability, which is applicable within a smart city (see Fig. 7 (a)). However, as more people are moving between cities, and cities are connected each other, inter-city interoperability, which is supporting interoperability between connected smart cities, is becoming an important issue to be solved by IoT technologies.

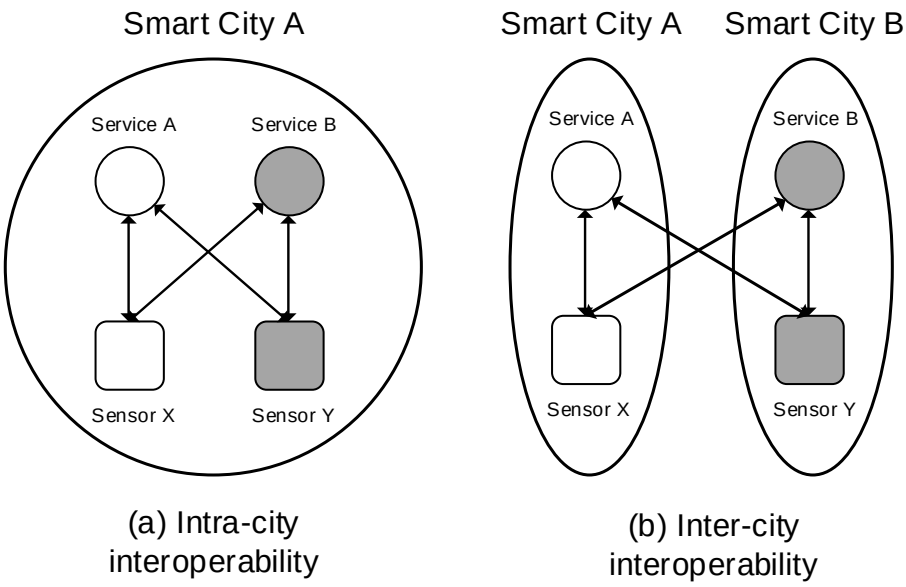

Fig. 7: Interoperability of smart cities

As far as we know, most smart cities are not supporting this inter-city interoperability (see Fig. 7 (b)). For example, let's consider that a smart city and

its IoT platform are deployed based on NGSI like Santander. If a natural disaster is just happened in that city, a disaster management service will manage various sensors and actuators to control the traffic and alarm the citizens to be in safe places using its standards-based IoT platform (Intra-city interoperability). However, the city cannot alarm the traffic coming from other cities in which used IoT standards for platforms and devices is based on oneM2M (Inter-city interoperability). In this scenario, both smart cities need to communicate with each other to exchange information and make decisions in real time to save citizens from facing the natural disaster. Depending on the technologies used for smart city IoT service platforms and data models, inter-city interoperability can be achieved differently. If two smart cities to be connected are using proprietary IoT technologies, it is very difficult to interconnect them as their data models and APIs are all different. Either all the data stored in a city IoT platform have to be translated into another IoT platform or all the deployed IoT services have to implement APIs for both IoT platforms, which require huge efforts. Even in both cases, the solutions are not scalable. On the other hand, if two smart cities are developed based on the same IoT technologies, although there needs some additioanl work to be defined, inter-city interoperability can be achieved easily. In other words, for realizing inter-city interoperability between smart cities in different conditions and environment, appropriate interworking approaches have to be considered.

In the following subsection, we introduce three interworking models for achieving intra-city interoperability and inter-city interoperability. By using these models, authorities or service providers are able to interconnect their smart cities to other cities so that the coverage of smart cities can be expanded.

2.1.1 Interworking scenario of smart cities

The majority of smart cities based on IoT are tailored to make the smart cities interoperable between IoT devices and services in the cities. However, since more people are traveling between cities, supporting the interoperability among cities are becoming more important and recognized as a paramount factor. In the present, there are not many IoT based smart cities that are supporting smart city interoperability [29].

Depending on the technologies used for smart city IoT service platforms and data models, Inter-city interoperability can be achieved in many ways. If two smart cities to be connected are using proprietary IoT technologies, it is very difficult to interconnect them as their data models and Application Programming Interfaces (APIs) are all different. Either all the data stored in a city IoT platform have to be translated into another IoT platform, or all the deployed IoT services have to implement APIs for both IoT platforms, which require huge efforts. On the other hand, if two smart cities are developed based on the same IoT technologies, although there needs some additional work to be defined, Inter-city interoperability can be achieved easily. In other words, for realizing Inter-city interoperability between smart cities in different conditions and environments, appropriate interworking approaches have to be considered, so three interworking models (IWMs) to solve the above issues were proposed in Fig. 8 and these are as follows.

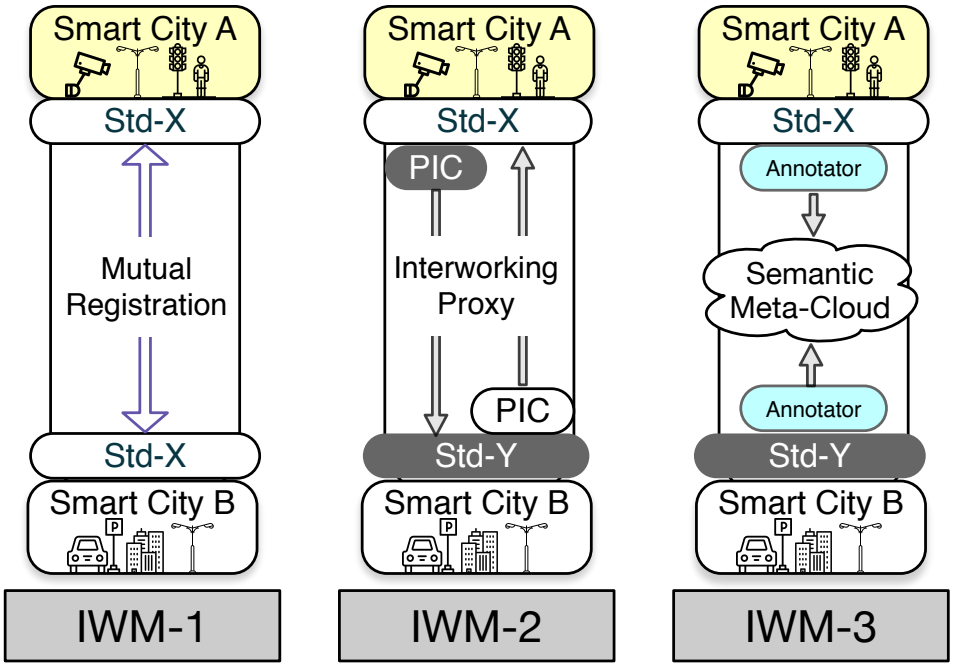

Fig. 8: Three interworking models for smart city interoperability

- ***Interworking Model-1:*** assumes that smart cities are using the same IoT standard. Standardizes interfaces are needed to have access to the other IoT platforms and exchange the data. Through those standardized interfaces, smart cities can operate their services but it has limitations in the case of citizens moving to other cities that are using different IoT standards.

- ***Interworking Model-2:*** Unlikely IWM-1, this model is under consideration based on smart cities using different IoT standards. Data standard conversion component called Interworking Proxy Entity (IPE) is used to convert the data presentation of the source platform to the target platform. In this model, in addition to changing the data presentation, the meaning of the data is also translated.

- ***Interworking Model-3:*** This model mainly focuses on creating a federation layer on top of the smart cities' IoT platform. Any data generated by IoT services of cities can be stored into the meta-cloud database through standard conversion components based on well-defined APIs. In this article, the semantic concept is used for the federation layer.

The following subsections show how the three interworking models developed to achieve interoperability between cities are used. In addition, by using these models, authorities or service providers are able to interconnect their smart cities to other cities and they can be likely to expand the coverage of smart cities.

2.1.2 The ways to realize the interoperability

As mentioned earlier, it is relatively easy to solve the problem if smart cities that are intended to interoperate are using the same standard. This is because, in general, when developing a standard, resource type, structure and APIs are developed to support the communication among IoT platforms or devices using the same IoT standard. For example, oneM2M has a reference point called `Mcc` and a resource called `<remoteCSE>`. The `Mcc` reference point is the communication flows among other CSEs. These flows allow CSE to use the services being supported by other CSEs. `<remoteCSE>` resource represents a registree CSE which is registred to the Registrar CSE. `<remoteCSE>` resource have to be located directly under the `<CSEBase>`. For exmaple, when CSE1 (Registree) registeres to CSE2 (Registrar), there will be two `<remoteCSE>` resource created. One for CSE1 (`<CSEBase1>`/`<remoteCSE2>`) and one in

CSE2 (<CSEBase2>/<remoteCSE1>). As a result, since CSEs are registered both IoT platforms, users can access IoT services operated in CSE2 at <CSEBase1> by using the <CSEBase1>/<remoteCSE2>. To conclude, with the above functions, IoT platforms or devices using oneM2M can interoperate among them, and there is no need to develop additional components or develop resources.

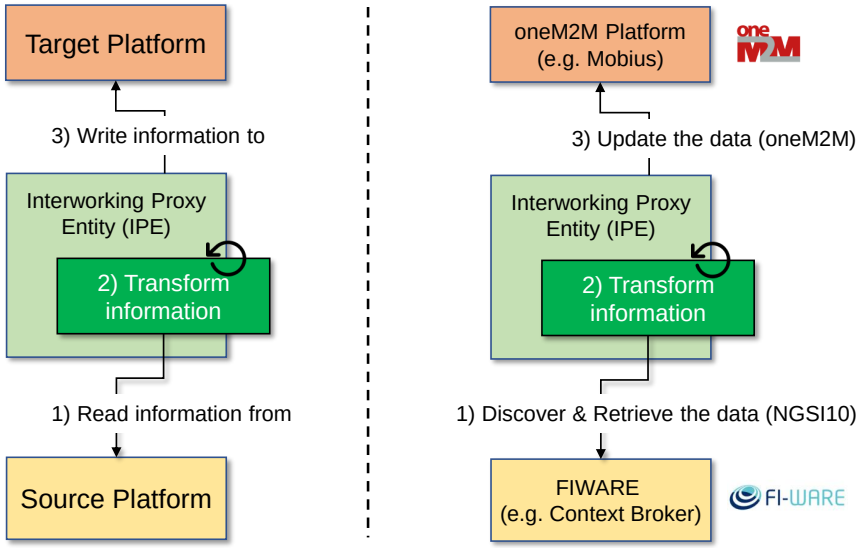

Fig. 9: Interoperability procedures with FIWARE-oneM2M example

In contrast, additional work is required when connecting systems that use different standards. Research and development are underway for architecture to secure interoperability between various IoT platforms and devices. However, since it is inefficient to completely uniform the data and structure model for each standard, a standard conversion component based on Interworking Proxy Entity (IPE), which is a component that enables the dynamic connection between each standard, has been developed. Through these IPE components, it is expected that

newly introduced platforms as well as existing IoT platforms can support interoperability. The IPE is a logical component developed to support interoperability for device-to-platform or platform-to-platform because communication standards and network interfaces are different depending on the manufacturers. However, as the demand for interworking increases, the number of interworking proxies also increases proportionally. This prevents the scalability of IoT standards and makes it difficult to efficiently manage resources, causing serious problems in providing IoT services. Therefore, to solve the above-mentioned problems, research on a method of managing various interworking proxy components through a common interface is also being conducted.

Figure. 9 shows the basic IPE structure and procedures, as well as the procedures of IPE interoperating with oneM2M and FIWARE platforms. The standard transformation component that operates in IPE reads information from the source platform and converts it to the data representation model of the target platform and stores it (Fig. 9 (a)). For example, the FIWARE-oneM2M (FO) IPE component utilizes the FIWARE platform as the source platform and the oneM2M standard platform as the target platform. When source and target platform information is provided, FO-IPE collects entity-based data stored in the standards-based FIWARE platform, converts it into the oneM2M standard-based data model, and stores it in the target platform (Fig. 9 (b)). This allows oneM2M applications to utilize data stored through the FIWARE platform. Examples of IPE components include Zwave-oneM2M (ZO), OCF-oneM2M (OO), LoRa-oneM2M (LO), GS1-oneM2M (GO) as well as the FO-IPE .

2.2 Interworking Model-1 (IWM-1)

In this subsection, IWM-1 is described as one of the ways to interconnect the smart cities. IWM-1 assumes that smart cities are using the same standard. In addition, oneM2M is selected as an example of the IoT standard. To conclude, by using oneM2M standard, this subsection shows how two smart cities can be connected.

2.2.1 oneM2M for smart cities

Besides the existing smart home and industry, the oneM2M is continuously developing and maintaining the standards in smart cities. oneM2M is currently contributing a smart city technology report (TR-0036), including domestic and overseas oneM2M smart city construction cases. Based on the existing cases, it will define various smart city models that can be configured using the oneM2M standard and derive the advantages of using standard technology.

Recently, smart cities are in the process of integrating IoT technology with existing urban infrastructure, and oneM2M can be regarded as a standard technology suitable for smart cities in this respect. Due to the nature of standard technologies, interoperability is basically guaranteed between devices, platforms, and applications that follow the standard. In addition, the Open Mobile Alliance (OMA), Lightweight M2M (LWM2M), Open Connectivity Foundation (OCF), AllJoyn, and oneM2M systems are interoperable by applying the interworking technical specifications developed in the oneM2M release 2. In addition, IoT devices equipped with network protocols such as ZigBee can also be linked to oneM2M, and by using semantic functions, large-scale smart city data collected

from various services in the future can be analyzed and used between services as described in [30]. To realize the interoperability of the cities, two types of interworking approaches as explained previously are being used.

2.2.2 IWM-1 interworking procedures

In this model, there is a assume that the IoT standard is defining the functions that allow smart cities to register and access the data with each other, and smart city interoperability can be realized if the IoT platform supports that IoT standards. The basic concept of IWM-1 is to connect smart cities using the same IoT standards. The IoT standard defines the functions that other smart cities can register or access data, and the smart city IoT platform can support each other by supporting the standard. When the IoT platform supports the above functions, the most important concept of IWM-1 is mutual registration. mutual registration is in charge of registering the resource of origin platform into the resource of the remote platform and vice versa. In addition, Additional procedures which is subscription/notification are needed to monitor the resource changes and apply the updated information to the existing resource. By using mutual registration, the information that is being used by users in smart city A can be delivered and stored in smart city B without changing any contents so that users who use a specific service in smart city A can get the same services in smart city B even though they move to smart city B.

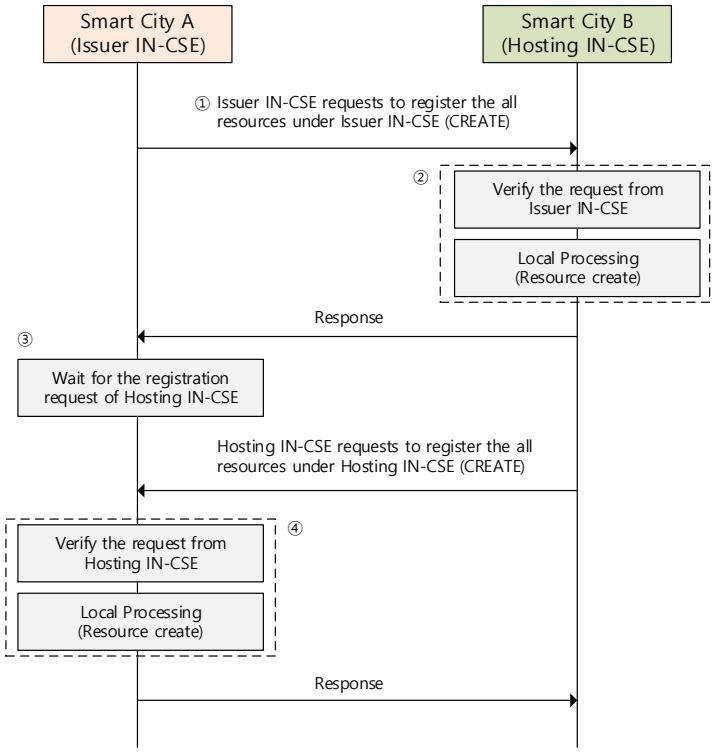

Fig. 10: oneM2M mutual registration procedures

The mutual registration procedure between IN-CSEs of the smart city is composed of the following four steps (see Fig. 10. In this figure, we assume that smart city A wants to interwork with smart city B and both cities are developed based on oneM2M standard. According to the oneM2M standard, all relevant data of cities are managed under the `<CSEroot>` resource as sub-resources. First, Issuer IN-CSE of the smart city A sends a request for registration to the hosting IN-CSE of smart city B by requesting the creation of a new `<CSEroot>` resource representing the issuer IN-CSE and all its sub-resources. The Issuer IN-CSE uses its identifier as the name of this new resource. In addition, It delivers the values that are needed to configure the new resource, for example,

the URI of the issuer. As the second step, Hosting IN-CSE verifies that the delivered identifier of Issuer IN-CSE does not exist in its managing resources already. After checking the validity of the request, a new resource is created by using the requested identifier and given attribute values. In addition, all sub-resources of the Issuer resource are created to keep information of things in smart city A. If all these procedures are successful, a success response message is going to be returned to the Issuer. As a result, all the resources of the smart city A can be used in Hosting IN-CSE. After receiving a successful response from the Hosting IN-CSE, the Issuer is waiting for the mutual registration request from the Hosing IN-CSE as the third step. In this regard, Hosting IN-CSE sends a request for registering the own resources to Issuer IN-CSE as Issuer IN-CSE did. The Issuer checks the given identifier and creates all the required resources in case the request is properly validated. As the last step, the Issuer sends a success message to Hosting IN-CSE to complete the mutual registration properly.

In this way, as we mentioned before, users can use the same services even though the users move to other cities [31]–[33]. Currently, this interworking model is being used by smart city projects such as Busan and Goyang for bridging the smart cities based on oneM2M.

2.3 Interworking Model-2 (IWM-2)

There exist many smart city IoT platforms which have different features, target domains, data models, and standards. For example, Busan and Santander cities are developed based on different standards, oneM2M and FIWARE, respectively. In this case, the data model and the way of managing IoT devices of a city are

totally different. oneM2M city platforms consider devices as a resource, while FIWARE platforms are focusing on data and events from IoT devices. If a citizen from Busan visits Santander, any IoT services cannot be accessed by the citizen as there is no interoperability on the data model and IoT services. In this case, similar to other IoT solutions that used interworking proxy to support interworking between different IoT platforms or IoT devices, a proxy interworking function can be considered as a possible way to support interoperability between different smart cities.

2.3.1 Interworking Proxy Entities (IPEs)

By using the proxy-based model illustrated in Fig. 11, data, and services are interpreted and mapped to the ones of a target smart city. This interworking function is provided by a Proxy Interworking Component (PIC). PIC has a resource mapping rule between two city platforms using different IoT standards. This resource mapping rule contains information about how a resource (i.e., representation of IoT data) is structured, configured and stored in the source and target city platforms. In this way, when there is a need for a citizen to have access to a resource in a remote city, the resource can be converted into understandable data.

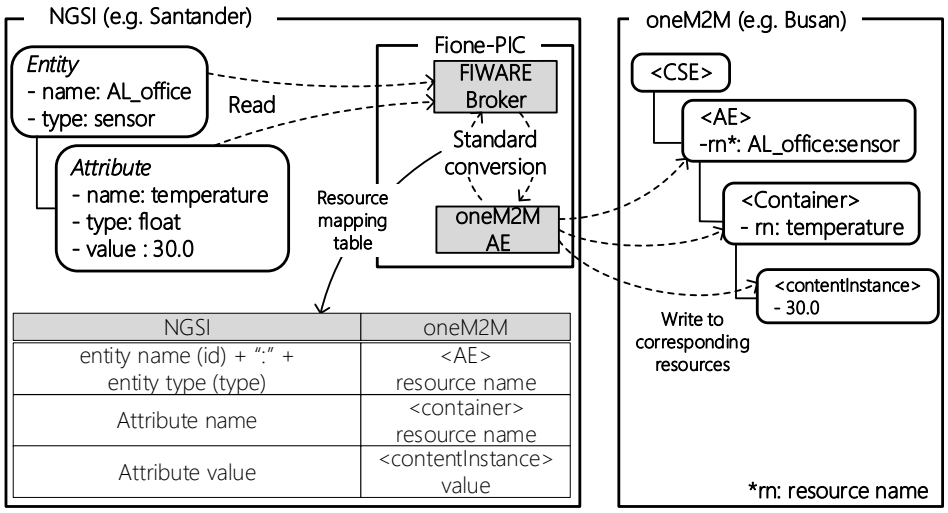

Fig. 11: Resource mapping between IoT platforms

Let us consider a case where Bob, a citizen in Busan, wants to know a temperature value of Alice's office in Santander. Busan is a smart city based on oneM2M, while Santander uses FIWARE. The oneM2M application that Bob is using does not support the NGSI interface used in Santander so that the application cannot discover a temperature device nor read the value of the sensor. To address this issue, FIWARE-to-oneM2M PIC (FIone-PIC) can be used and then Bob can get the NGSIbased data from Santander smart city. In other words, if users want to use IoT devices that use the NGSI context data model of FIWARE on the oneM2M platform, PIC is needed to convert the NGSI context data into the oneM2M resource structure.

2.3.2 IWM-2 interworking procedures

Fig. 12 describes how the FIWARE-to-oneM2M PIC converts the NGSI context data model into the oneM2M resource structure to use IoT devices without standard barriers. NGIS based IoT device registers to FIWARE ContextBroker that is the server based on NGSI standard (Step 1). PIC requests device information from the FIWARE ContextBroker based on the external input data received from the web portal. In this procedure, we assume that the web portal allows the users to select the devices registered in the FIWARE ConetxtBroker and FIWARE ContextBroker delivers device information to the PIC (Step 2, 3). PIC converts the standard from the FIWARE NGSI standard to oneM2M in order to register FIWARE device information into the oneM2M server (Step 4). PIC performs the subscription operation for handling the device data update case (Step 5). As soon as FIWARE device is updated, PIC update existing oneM2M information by using notification and transformation mechanism is already described before (Steps 6, 7, 8).

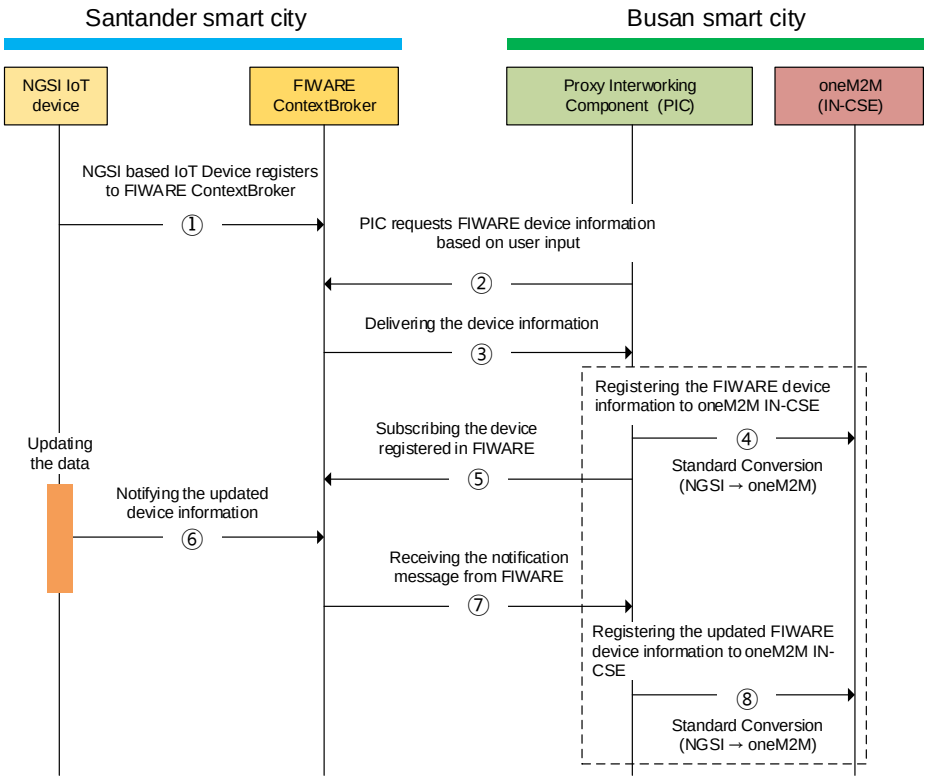

Fig. 12: Inetworking procedures between two smart cities

As a practical example using this model, there is a project called Wise-IoT (http://wise-iot.eu/en/about/), and it leads the project under the theme of smart city interworking through semantics technology. In this project, IWM-2 is adopted to support interworking between oneM2M and non-oneM2M standard devices based on NGSI, Open Connectivity Foundation (OCF), Lora, and Z-wave.

2.4 Interworking Model-3 (IWM-3)

Unlike the previous two IWMs, IWM-3 is based on the centralized cloud system. More specifically, the semantic-based central cloud is used to federate the other

smart cities by using semantic technology. The data generated by smart cities are translated according to the semantic rule and this is delivered to the semantic-meta cloud. In this subsection, first, the basic concept of semantic is illustrated and subsequently, IWM-3 describes with the architecture.

2.4.1 Semantics

The Semantic web was first proposed by Tim Berners Lee in 1998 [34], and it establishes relationships among numerous information and resources available on the Internet. As a result, machines or computers understand each other without human intervention. The purpose of the Semantic web is to develop standards and technologies to help computers easily understand information on the Web, supporting semantic search, data integration, navigation, and automation of tasks. In order to support these functions in the Semantic web, it is necessary to insert a concept for knowledge expression in a web document, establish relationships between knowledge, and include inference rules to enable intelligent information processing of computers. Through this, it is possible to accurately convey the information desired by the users [35]. As described in Fig. 13, the HTML link basically links the document node to a link called `href` tag, and links two resources, and is a simple link that does not express its own meaning. On the other hand, by providing type information to each node and link, a more meaningful network can be constructed [36].

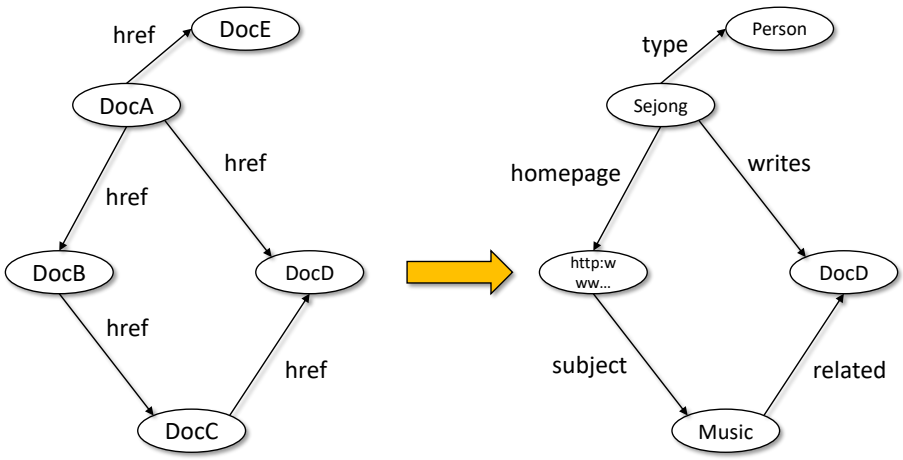

Fig. 13: Comparison between HTML and semantics

The core technology that enables the implementation of the semantic web is an ontology that formally describes the concepts in the domain and the relationships. It uses languages such as Resource Description Framework (RDF) and Web Ontology Language (OWL). By implementing ontology, computers can process data based on knowledge. Based on the application of semantic technology through the development of an ontology of specific domains, simply beyond the use of raw data generated by sensors or devices, many studies have been conducted to support IoT services by combining the current IoT technology and semantic technology.

Thanks to the advantages of this technology, research is being conducted to apply it to the IoT industry. For instance, in the oneM2M case [37], the current approach taken by oneM2M treats data as black boxes. This means that the content is opaque and applications need to know in advance how the data interpret. Therefore, semantics are essential if the M2M system is expected to

interact with real entities (“things”) because the main role of semantics is to provide a description of the relationship between things/data/information.

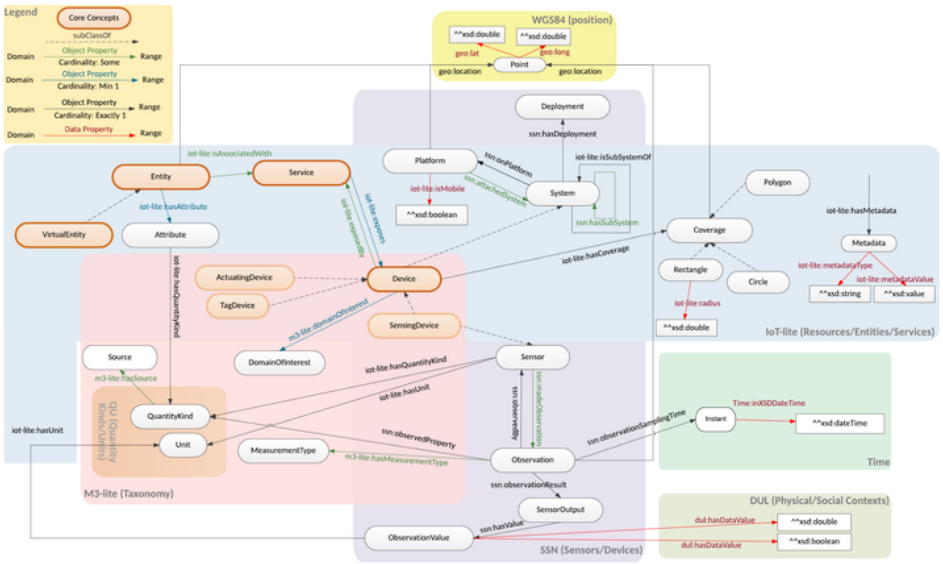

Fig. 14: FIESTA-IoT common ontology

With such advantages, the FIESTA-IoT (http://fiesta-iot.eu/) project is commenced to provide semantic interoperability between different smart city IoT testbeds and expose the services as an Experiment-as-a-Service (EaaS) model, which defines a single entry point to all FIESTA-IoT services. However, existing ontologies are not well connected and are domain-specific. That is, the ontology for the IoT service is an ontology according to specific domains, and the existing developed ontology may not be suitable for other services, so the ontology needs to be additionally extended. However, extending the ontology according to each requirement increases the complexity and is not suitable. In this context, to achieve semantic interoperability, the FIESTA-IoT project defined a common ontology called FIESTA Ontology. For constructing the FIESTA common ontology, it leverages various concepts of Semantic Sensor Network (SSN)

ontology, IoT-lite ontology, Machine-to-Machine Measurement (M3) Lite (M3-lite) Ontology, and by using these concepts, it covers most of the concepts for achieving the goal of the semantic interoperability and the federation [38].

To translate from testbeds' intrinsic formats to the one that is understood and interpreted by the FIESTA ontology, semantic annotation is needed. First, let us check what annotation is. Annotation means adding additional information to a sentence or document. It is often considered as 'comment', and comments written as a description of source code in a programming language are a kind of annotation. In computer technology, annotation means mapping the analysis result of a specific text to the text. The purpose of annotations is to provide additional information about a document, sentence, text, or specific concept, and to make the information available in a variety of places. Based on this concept, semantic annotation is used to annotate the metadata to the web resources. Metadata is summarized as "Data about the data". In the semantic web, information about web resources, etc. described in a form that a computer can read and understand is called metadata. Meanwhile, the procedure of describing metadata in a form that can be processed by a computer using a metadata expression language such as RDF, OWL, etc., and giving metadata to web content is called a semantic annotation [39], [40].

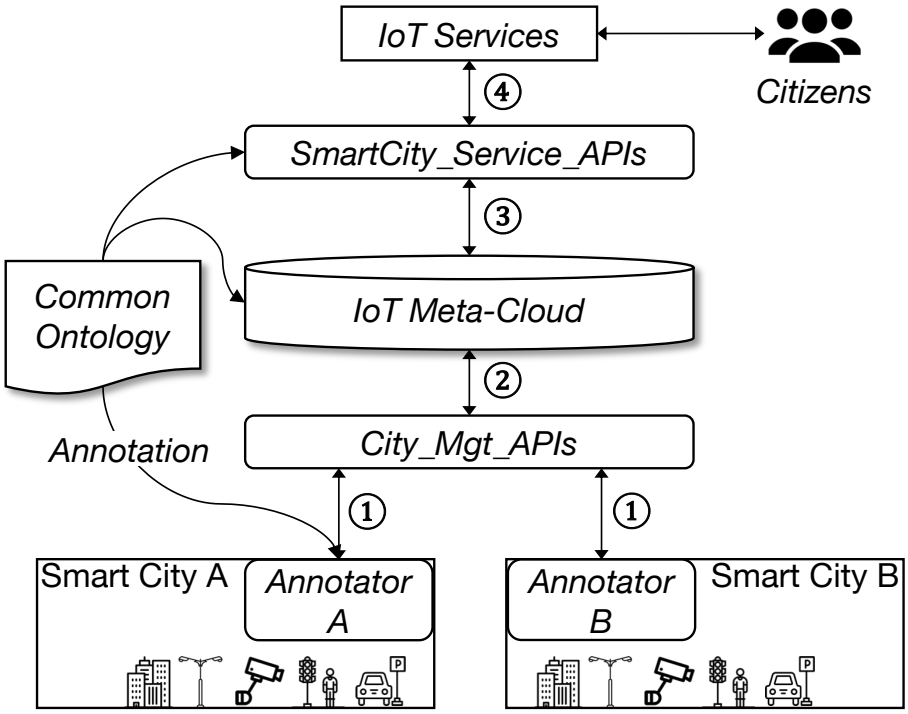

Fig. 15: Semantic interoperability architecture

2.4.2 IWM-3 interworking procedures

Semantic interoperability enables different agents, services, and applications to exchange information, data, and knowledge that arrive at equivalent meanings [41], [42]. IWM-3 is an IWM based on semantic interoperability assets, and by using this model, cities can interwork and exchange information, as this information is understandable. In addition, IWM-3 defines a centralized server platform that federates various smart cities using semantic interworking and provides a common set of semantic APIs. In addition, underlined smart cities could be developed either based on the same standards or based on different IoT platforms using different IoT standards. Figure. 15 shows the high-level

architecture of IWM-3 to make smart cities semantically interoperable. The IWM-3 comprises the four main components: 1) Unified Ontology to semantically annotate the data, 2) Meta-Cloud to store semantically annotated data pushed by the registered resources of underlined smart cities, 3) CITY-MGT-APIs for smart cities to register and manage themselves and their resources and to push observations in Meta-Cloud, and 4) SmartCity-Service-APIs which enable the end users and developers to obtain data from any smart cities registered to this meta-cloud [43].

Services	Description
getLastObservations	This service provides the latest values of a specific Sensor list in an FIESTA-IoT
getObservations	This service provides the values of a specific Sensor list for a specific time-period in an FIESTA-IoT
pushLastObservations	This service initiates a stream at the Testbed side which pushes continuously the latest values of a specific Sensor
pushSingleObservation	This service pushes continuously the latest value of a specific Sensor
stopPushOfObservations	This service stops the push of observations initiated by the "pushLastObservations" and "pushSingleObservation"

Table 2: Testbed Provider Services (TPS)

To fill the gap of understanding between different cities and their data formats, IWM-3 defines Meta-Cloud on top of all smart cities to store and manage standardized semantic information. The IoT data on each smart city platform is converted into semantic data using a standardized ontology. The ontology can encode the meanings of data in a form that does not change. Underlined smart cities consume CITYMGT-APIs to register their resources in the Meta-Cloud and push data to Meta-Cloud after converting to the semantic

data using the common ontology (Steps 1 and 2). For this data conversion, each underlined smart city has to develop and deploy its own semantic annotator by means of the transformation of the information it stores locally to data described by using the common semantic ontology. Then, IoT services use the SmartCity-Service-APIs to allow users to access data distributed around the underlined smart cities (Steps 3 and 4). In the FIESTA-IoT project, there is a set of RESTful web services called Testbed Provider Interface (TSI) for integrating the testbeds into the FIESTA-IoT platform. For example, as core services of the TSI, Testbed Provider services (TPS) are defined to provide the "pluggability" of the testbed to the FIESTA-IoT platform, and a list of services are can be considered as CITY-MGT-APIs. TPS is listed in Table 2 [44].

2.5 Interworking model evaluation

This subsection compares and analyzes the three IWMs against the five criteria, which are selected to see an impact on the stakeholders of smart city development such as platform companies and city owners.

Scalability: Depending on the number of smart cities to be interconnected, each IWM shows different scalability. IWM-1 is highly scalable, as this model mandates a single standard for all smart cities. This model can extend to as many interconnected smart cities as necessary if the city governments all agree to use the same standards. However, this is not realistic, as city governments decide their standards based on many different factors, such as culture and international relationships. IWM-2 and 3 can also provide scalability under the assumption that PIC and Annotator are easily developed and deployed. However, as the

	IWM-1	IWM-2	IWM-3
Scalability	• Pros: This is highly scalable if a single standard is mandated. • Cons: It is difficult to mandate that all cities follow the same standard.	• Pros: Interconnected smart cities can easily be extended using PIC. • Cons: It is not easy to develop PIC for all standards.	• Pros: The use of semantics ensures scalability. • Cons: It is difficult to develop annotators for all standards.
Complexity	• Pros: A standard body only needs to define standardized mutual registration procedures.	• Cons: As interworking cities grow, the complexity grows steeply.	• Pros: Each city only needs to provide an annotator for the common ontology. • Cons: The common ontology can become complicated.
Data redundancy	• Cons: All data on a platform should be replicated on a remote platform.	• Pros: A platform only needs to have data that it interworks with.	• Pros: The meta-cloud only needs to store the semantic description. • Cons: All data should be annotated and stored in the meta-cloud.
Development effort & cost	• Platform developers take on a small burden. Application developers only need to use the standardized APIs.	• Platform developers have to develop PICs while App developers just need to know their city APIs.	• Each city platform owner has to develop annotators, while App developers only need to use given APIs.
Privacy & security	• Pros: High probability that two cities are using the same security mechanism.	• Cons: May need security mechanisms for secure interworking.	• Cons: Additional common security mechanisms need to be defined.

Table 3: Summary of IWM analysis

number of interworking cities grows, it becomes more difficult to support PICs and Annotators for all cities.

Complexity: In the case of IWM-1, standard development organizations (SDOs) only need to define mutual registration procedures. Then, all the smart cities can be interconnected by implementing the procedures, which is easy. IWM-3 shows average complexity, as each smart city only needs to develop one annotator to convert its data to semantically annotated data with the common ontology. To interwork with N cities, the same number of annotators is required. On the other hand, IWM-2 shows the highest complexity. In this IWM,

each smart city has to support PIC for all the interworking cities. This means we need (N-1)*N PICs to interconnect N cities.

Data redundancy: The amount of data a smart city has to manage is very important in terms of city performance. It is not possible to support the city interworking without data redundancy. All IWMs cause data redundancy. IWM-1 shows the worst data redundancy rate among the three models as it forces both interworking cities to store their remote data on their platform. IWM-2 requires the same data redundancy as IWM-1. However, with a small enhancement feature (i.e., introducing a dynamic morphing mediation interworking proxy that is able to load only necessary interworking functions), the rate of data duplication can be decreased dramatically. Compared with IWM-1 and 2, the semantic IWM has a lower data redundancy rate, as IWM-3 only needs to store a semantic description of the data, not the data itself.

Development effort and cost: Typically, platform developers implement an IT infrastructure to operate smart cities, while application developers work on smart city services using given APIs from smart cities. In the case of IWM-1, a small amount of effort is required from both developers for interworking, as all platforms are standardized. On the other hand, IWM-2 and 3 force platform developers to exert significant effort on developing PICs and annotators. The application developers, in this case, just need to know about their own city APIs. Currently, many smart cities are being led by the government with public funding, which may come from the government, federal, or state, so that the development and deployment of the PIC and annotator are usually conducted as open-source projects which are available to the general public for use. In the future, after finishing the government funding phase, any smart city that wants to

be connected (IWM-3) to the meta-cloud and interwork (IWM-2) with other smart cities should develop and operate such software with their own funding. However, through using various open-source interworking modules developed in government funded smart city projects, the cost of developing such software can be reduced significantly.

Privacy & security: For consistently operating the smart cities, privacy and security are critical challenges in supporting data confidentiality and authentication, access control, privacy, and trust when citizens and entities move between cities. In particular, when a citizen moves to other cities that have different security and privacy regulations, important private data can be easily leaked. Even the same IoT standard is used in IWM-1, if different privacy regulations are used in the inter-connected cities, a framework managing privacy issues within a heterogeneous smart city environment is needed. IWM-2 and IWM-3 generally need a mechanism supporting secure interworking. However, such a mechanism is not always required as there exist cases that ban on data transfer across cities or nations by policymakers because of privacy and cybersecurity, or economic mercantilism. Therefore regardless of which IWM is selected, proper mechanisms for guaranteeing security and privacy across smart cities have to be considered carefully. To conclude, Table 3 shows a summary of our IWM analysis. No single IWM is better than the others are. All the models have their own pros and cons. City owners or designers should consider their environment (e.g., budget, relationship with other cities, plan for interworking, regulations) in selecting their IWM.

2.6 Summary

In this article, three interworking models that are needed to solve the IoT standard interoperability issue are proposed with smart city scenario, and also evaluated the three interworking models. By using one of these models, not only developers can easily support the variety of IoT standards but also existing IoT platforms can also support other IoT standards without any changes in the existing system.

3 Automated conformance testing for IoT applications

In the upcoming decades, people living in the cities are going to be surrounded by billions of IoT devices that will interoperate and collaborate with each other to deliver personalized and autonomic services. As a result, citizens could be the first to benefit from the technologies in the cities. However, relying on these machines to make decisions could bring profound problems if cities cannot ensure the reliability of those IoT services [45]. For instance, currently, many IoT devices are being deployed and operated for supporting various IoT services such as smart vehicles, surveillance cameras, smart buildings.

However, if IoT devices do not assure their reliability, connecting all the systems to the Internet can cause severe problems in the cities [46]. Therefore, to avoid the problems caused by dysfunctional products or not working IoT services, it is necessary to test all the IoT platforms and devices before releasing them in smart cities. Furthermore, smart cities are operating IoT platforms and devices using various standards and protocols, which means that smart cities have to consider these environments as well. Based on the previous discussion, to realize and to assure the safety of smart city environments in the highest level, this section starts by introducing the adaptable IoT testing framework to support the IoT testing. In addition, to mitigate the testing error and to reduce the testing time, an automated IoT testing architecture and procedures are proposed.

3.1 Analysis of the traditional IoT testing

For the last two decades, the interoperability is the main hurdle of IoT proliferation and adoption [17], [47], [48], and to successfully adopt the IoT into the industry fields, it is imperative to overcome its fragmentation issue between IoT services [49], [50]. Therefore, assuring the interoperability of IoT implements based on differing standards and protocols requires well known conformance and interoperability testing process in the software industry [51]–[55]. In this regard, IoT conformance testing is becoming one of the most important parts of IoT technologies [26], [27]. Conformance testing is considered to be an essential element of the IoT testing to provide a reasonable degree of gurantee and it is used to check systems or devices that are well developed against the standards developed by telecommunication organizations [56]–[58]. In addition, in the paper [24], International Telecommunication Union (ITU) has been recognizing that interoperability testing is highly important to ensure that independent implementation based on the same standard are interoperable. With the importance of IoT testing as described, in the subsequent subsection, this dissertation goes deeply to describe the conformance and interoperability testing.

3.1.1 Conformance and Interoperability Testing

Conformance testing is one of the testing approaches where a system is tested against its standard specification [59]–[61]. The aim of conformance testing is to improve the confidence of the implemented system's probability that it follows the same standards. It is possible to find the expected or unexpected (invalid)

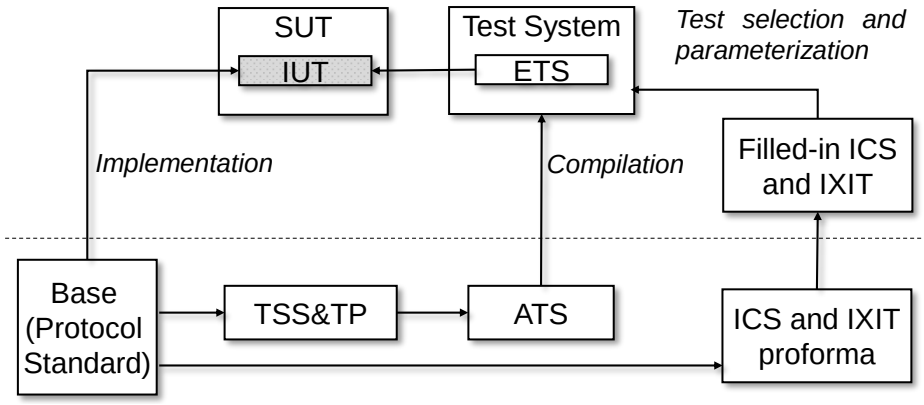

Fig. 16: Procedures of the ETSI Conformance Testing

behavior of the specific protocols but conformance testing does not cover all aspects because it only tests the systems against specific scopes or requirements specified in testing specifications. In addition, it does not actually test how well the system works with other systems. Conformance testing specifications for Global System for Mobile Communications (GSM), Universal Mobile Telecommunication System (UMTS) and Voice over Internet Protocol (VoIP) was developed by European Telecommunications Standards Institute (ETSI) and test specifications were developed according to the well-proven ISO/IEC 9646 conformance testing methodology [62]. The conformance testing specifications are also referred to the major elements of ISO/IEC, and the widely used procedures are described in Fig. 16 [62]. In addition, the main components that make up the procedures are as follows.

- **The Test Suite Structure and Test Purpose (TSS) & Test Purpose (TP):** These are derived from the relevant base standard. They provide an easy-to-read and informal description of each test. In addition, these concentrate on the meaning of the test rather than detailing how it may be achieved.

- **The Abstract Test Suite (ATS):** It is the entire collection of test cases. Each test case specifies the detailed coding of the test purposes written usually in a test specification language such as the standardized TTCN.

- **The Implementation Conformance Statement (ICS) & The Implementation eXtra Information for Testing (IXIT):** These contain additional information (e.g., specific addresses, timer values etc.) necessary for testing.

- **The Executable Test Suite (ETS):** It can be quickly and easily implemented from the ATS using the TTCN compilers available on most modem test tool platforms (C++, Java etc.).

- **System Under Test (SUT) & Implementation Under Test (IUT):** SUT is the real open system [63], and inside there is IUT which is the implementation of applications, services or protocols.

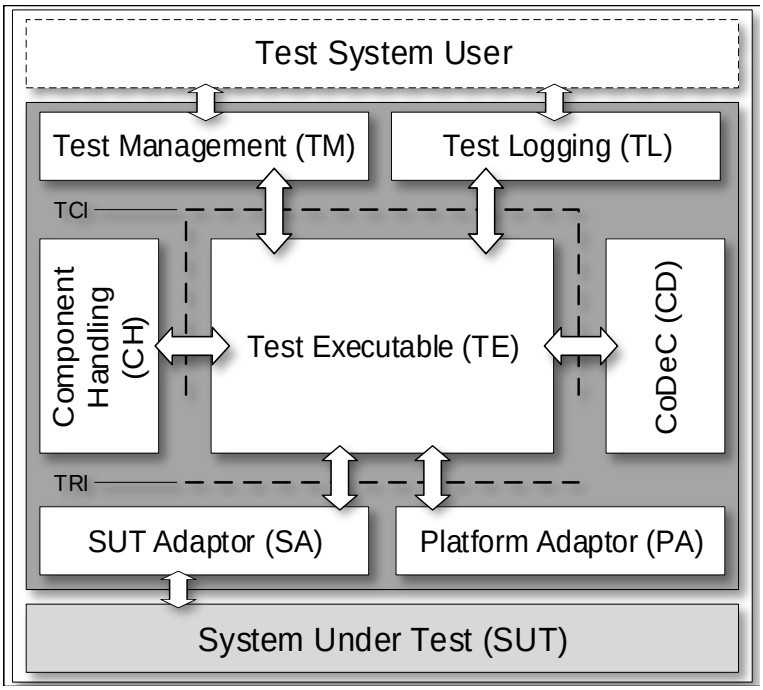

Fig. 17: TTCN-3 conceptual testing framework

Regarding the test cases described in ATS, TTCN-3 dedicated language for the protocol testing was developed and are being maintained by ETSI and also it is redesigned language based on tree and tabular combined notation (TTCN) standard (ITU-T Rec. X.292). TTCN-3 is an applicable language to the specification in terms of testing the reactive system and it is widely used in following areas such as protocol testing, service testing, module testing and Application Programming Interfaces (APIs) for the platform [64].

TTCN-3 defines a independent standardized conceptual model against the SUT, processing platform, implement the language. The well-defined sets of operations of each entity and interfaces for providing communication, management, external data, and logging can be executed by implementing and

interpreting to intermediate languages. The general concept consists of the TTCN-3 based Testing System (TS) is shown in Fig. 17. The TTCN-3 test system performs conformance testing through interactions between entities that include specific functions and each independent entity has functionalities such as communication with the System Under Test (SUT), timer management, type handling, and external function call. In addition, there are interfaces called TTCN-3 Control Interface (TCI) and TTCN-3 Runtime Interface (TRI). Through the interactions between Test Executable (TE) which is responsible for the execution of the TTCN-3 module and two interfaces including TCI, TRI, actual TTCN-3 modules are executed. Following five entities make up TTCN-3 conceptual testing system and detailed entities functionalities are as follows [65]. First, entities that communicate with TE by using TCI interface are as follows.

- **Test Management (TM):** TM entity acts as a managerial layer of a test system. On this entity, the test developer defines the arrangement of the test suite execution list. Through this layer, TTCN-3 provides support test campaigns creation, log format and handling customization.

- **Component Handling (CH):** Since TE can work in either centralized way or decentralized way, the aim of CH is for synchronizing entities which could be placed on different nodes and for managing the interaction between them.

- **CoDec (CD):** Codec is designed for encoding/decoding outgoing and incoming message during bidirectional transmissions toward SUT. In order to communicate with SUT, TTCN-3 abstract data representation format has to be converted actual format required by standard-based IUT tested.

- **Test Logging (TL):** All log events generated by the testing system are handled. The testing system must provide a log system for debugging purposes.

The following types of entities communicate with the TRI interface:

- **SUT Adapter (SA):** This is responsible for the actual communication with the SUT. That is, messages are transmitted to the SUT according to a specific procedure. In addition, messages are transmitted by the SUT with the testing results. The main activity happened inside SA is a bidirectional transmission of messages and procedure toward SUT. SA usually has concurrent and synchronous characteristics as the result of its aim to be retriever and sender of messages and procedures. Therefore, there are individual thread performances to deal with incoming and outgoing data.
- **Platform Adapter (PA):** It is mainly characterized by the support of timer function and the external functions including communication functionalities, and the timer function is used to control the flow of TTCN-3 code by using time literally. In particular, the external function makes the TTCN-3 language even more powerful because it uses a library implemented in the native language such as C++ or Java, which makes it possible to use additional functions not supported by TTCN-3.

Unlike the conformance testing that is usually testing the single systems or devices to check whether they are well following the standard, interoperability testing is based on the interaction of systems or devices. At this time, there is much literature regarding interoperability, but one book published by ETSI is

describing the interoperability testing comprehensively, so the following descriptions are based on [66]. The trend of a global interconnection demonstrated by the huge growth in the IoT makes no longer a single standard body is leading entire technologies. Usually, complex products and system are based on multiple standards from multiple standard bodies such as ETSI, IEEE, ITU-T and those have different aspects and features to support the specific services. To conclude, it is important to reduce the ambiguities, errors, unclear requirements that could lead to non-interoperability. At present, there is no single and specific definition of interoperability testing but, according to the ETSI's interoperability definition, "Interoperability can be considered to be the ability of two or more systems or components to exchange data and use information". Once a set of requirements of systems or devices using specific standard has been identified and defined, it is important to check if they provide interoperable solutions. Many issues can be identified and resolved through technical specifications, but only through interoperability events or actual tests, interoperability can be ensured. For instance, ETSI complements support for other testing activities by integrating Plugtests^{TM} into the standardization process. In this regard, the results of the Plugtests^{TM} event provide valuable feedback to other international organizations and forums.

Interoperability Test Description			
Identifier	TD_M2M_NH_01		
Objective	AE retrieves the CSEBase resource		
Configuration	M2M_CFG_01		
References	oneM2M TS-0001, clause 10.2.3.2 oneM2M TS-0004, clause 7.3.2		
Pre-test conditions	• CSEBase resource has been automatically created in CSE		
Test Sequence			
Step	RP	Type	Description
1	-	Stimulus	AE is requested to send a retrieve Request to CSE with name {CSEBaseName}
2	Mca	PRO Check Primitive	• Operation (op) = 2 (Retrieve) • To (to) = Resource-ID of requested <CSEBase> resource, assumed CSE-relative here • From (from) = AE-ID of request originator • Request Identifier (rqi) = (token-string)
3	Mca	PRO Check Primitive	• Response Status Code (rsc) = 2000 (OK) • Request Identifier (rqi) = same string as received in request message • Content (pc) = Serialized Representation of <CSEBase> resource
4	-	IOP Check	AE indicates successful operation

Fig. 18: oneM2M interoperability testing descriptor

Interoperability testing often involves activating and observing user functions, so it makes to specify test cases as a series of steps that human test drivers will perform. In some cases, it is not necessarily all, but it is useful to be able to specify some preconditions for the test. This often takes the form of instructions for configuring the network and equipment so that the testing objectives are fully met. The test steps themselves should be specified in a clear and unambiguous way, even though there are no unreasonable restrictions on how each step is performed. Clarity and accuracy are important to ensure that steps are performed accurately. The “pass” determination always means that the connected device works correctly for certain tests, but it does not imply that the

"failure" determination does not. Interconnected network equipment plays an essential role in almost all interoperability testing, but is not usually included in the equipment under test. The "failure" judgment can be caused by a network defect or unexpected behavior. Therefore, each "failure" judgment must go through a thorough investigation. If possible, monitoring equipment should be used to determine the root cause before re-inspection (if the root cause is within the tested device) before verifying the judgment result as a real failure. Test steps and judgments should be specified at a level appropriate for the function to be tested. Based on these requirements, interoperabilty test description is being developed as described in Fig. 18 [67].

3.1.2 Related work of traditional IoT testing

In [68], it briefly explained the concept of a testing framework for IoT by integrating the concept of simulation, unit integration and end-to-end integration testing. In addition, [69] explored compliance of IoT devices with privacy policy agreements and established models regarding the privacy criteria to measure the degree of compliance. However, those approaches have the disadvantages that both testing systems and test experts must be physically located at the same place to test their products.

Accordingly, the cloud-based testing system has been a great alternative for dealing with the previous issue. The characteristics of cloud computing can enhance service delivery [70], production cost and time reduction [71] and responsiveness towards requirement changes. In this regard, cloud computing can be used as an IoT testing platform remotely supporting IoT testing, logs and

results views. The F-Interop [72], a cloud-based interoperability testing framework, provides a testing expert remote testing environment supporting a variety of IoT standards and protocols and standards. However, it heavily concentrates on the interoperability aspect. [73] developed TTCN-3-based testing suites called IoT-Testware to support the widely used IoT protocols from the conformance testing perspective, but its coverage is limited to message queueing telemetry transport (MQTT) and constrained application protocol (CoAP). The IoT compatibility testing tool (ICAT) [74] was designed to support remote IoT testing, but it only examines compatibility issues of IoT devices on the level of firmware.

In addition, there are several research works for improving the problems of manual testing. Dobles et al. [75] compared the effectiveness of automated and manual tests in terms of total test time and the number and severity of defects found. The results showed that automated testing is more effective than manual testing at finding defects. To make the test script automatically, a graphic user interface (GUI)-based testing component was proposed to define and implement the test scripts automatically in a large industry system [76]. With the considerable attention on artificial intelligence (AI) technologies, a natural language-based automated testing approach was studied [77]. The proposed mechanism shows that a manual testing script written in English can be almost automatically converted to mechanical interpretation. In conclusion, much previous research has attempted to support IoT testing and automated testing. However, as far as we know, standardized conformance testing for a large number of IoT applications in constraint devices is not well considered.

3.2 Limitation of traditional software testing

In general, the conformance and interoperability testing are widely used testing approaches to validate standards, and these testing methods are being standardized and adopted by many SDOs. However, the traditional software testing by using conformance and interoperability testing is not suitable to conduct the IoT testing, and there are several issues to test IoT services.

The central issue of traditional IoT conformance testing is the human-managed operation and conventional testing procedures. For testing a larger amounts of IoT devices, vendors need to consider reducing the human interventions during the testing. In addition, variabilities of IoT services are exposing the problems of testing heterogeneity. In order to test a wide ranges of IoT platforms and devices, a flexible way to test the IoT system is needed. Traditionally, interoperability testing requires manufacturer for testing their products in the same physical location. In addition, interoperability testing can be conducted with several products having different testing configurations. However, gathering and placing them in the same physical location is not an easy task and time-consuming. Problems are classified according to each testing method, but a problem occurring in one testing method may be a problem of another testing method. For instance, testing location dependency issue is the problem that has to be solved of both conformance testing and interoperability testing. To conclude, traditional software testing problems using conformance and interoperability testing are as follows.

- ***Testing environment coordination:*** Testing IoT platforms and devices entail larger and more heterogeneous stakeholders than traditional software

testing. In addition, IoT services are naturally based on the heterogeneous environments coming from different protocols and standards [25], [78]. Therefore, creating and setting own IoT testing systems to support a heterogeneous environment is not easy and expensive tasks.

- ***Testing cost:*** To test the companies products before releasing its products and get certification from certification bodies, testing systems and testing experts have to be positioned at the same place physically in the traditional software testing: This is called Face-to-Face (F2F) testing. It is very cost-inefficient and labor-intensive [24].

- ***Testing intervention:*** In the situation of the traditional IoT device testing, the problem is that the testers manually test the IoT platforms and devices. It is impossible to test a lot of them manually and human intervention can make side effects on the testing results. Therefore, the sure-fireway is that it makes the testing procedures automated [79].

As solutions to these challenges, in this dissertation, Testing intervention is mainly handled as a challenge has to be solved; However, Testing environment coordination and Testing cost are illustrated as a ongoing work for IoT testing. In the following subsection, motivation and the solutions of automated IoT testing are explained.

3.3 Research of the automated IoT application conformance testing

This subsection elaborates on triggering messages and related procedures for testing IoT applications automatically. In addition, because the method is

performed based on an automated approach without human intervention, it is expected that there will be significant advantages in terms of testing cost or accuracy than the tester performs conformance testing manually. As a result, IoT application testing can be performed more accurately and quickly. To conclude, this section illustrates an architecture for an automated and scalable conformance testing mechanism for IoT applications. In addition, it shows that the experimental results to prove the proof-of-concept of the automated IoT testing approach.

3.3.1 Motivation of automated IoT conformance testing

In the situation of the traditional IoT application testing, the problem is that testing experts are manually testing the IoT applications. With the emergence of IoT applications, each IoT application has a list of distinct features according to the application domains such as wearable devices, home automation, automobiles etc. As a essential stage, before releasing IoT products, these have to be certified by testing experts to avoid fatal risks coming from dysfunctional devices and not working products. However, existing IoT conformance testing would not appeal to testers since it has several limitations.

In general, conformance testing consists of a test system, a SUT and a set of test cases, where the testing system executes the test cases against a SUT to assess its degree of standard compliance. To test IoT applications, many manufacturers use their own software agent having UIs for stimulating the SUT to initiate a specific behaviour or waiting the specific requests from the IoT applications. As a result, such testing tools require human developers to be involved in conformance

testing procedures. For example, to test a registration function, which requires a SUT to send a registration request to a testing system, a developer should instruct the SUT via a software testing agent that sends the corresponding stimulus to the SUT. Testing hundreds of IoT applications using such conformance testing tools is a time-consuming job, which is not efficient for testing IoT applications. In addition, there are many aspects to test the IoT application before releasing the products to the market, and all of features has to be tested. However, it is impossible to test a lot of applications manually and human intervention can make side effects on the testing results. Therefore, the sure-fireway is that it makes the testing procedures automate.

At present, many IoT standard organisations are defining and maintaining the standards for the IoT while IoT specifications regarding the interoperaiblity and conformance testing are being developed and maintained. However, most previous studied regarding the conformance testing are focusing on specific domains such as security, battery lifetime and so on, which means that these are not discussed in the context of IoT standard based testing [79]. Additionally, at the time of conducting this research, there was not an automated approach to testing IoT application conformance.

IoT conformance testing has the several steps to test the IoT applications, and the Fig. 19 shows the steps of the complete testing procedures in the testing lab with the assumption that the Testing System (TS) has all the executable test cases defined in the specification. In general, conformance testing consists of a TS and a SUT, and the TS executes the test cases against a SUT to assess its degree of standard compliance. First, according to the IoT device features, a tester needs to select the Product profiles which provide guidance on what features have

to be implemented in the IoT devices, and based on these profiles, test cases is being developed by SDOs. In the next step, the tester establishes a connection between the TS and the SUT by configuring the IP addresses and the ports. Once configured, the tester runs the TS and the SUT for execution of the test cases. Next, the testing loop starts from tester by selecting one of the test cases from a product profile and run it in the TS.

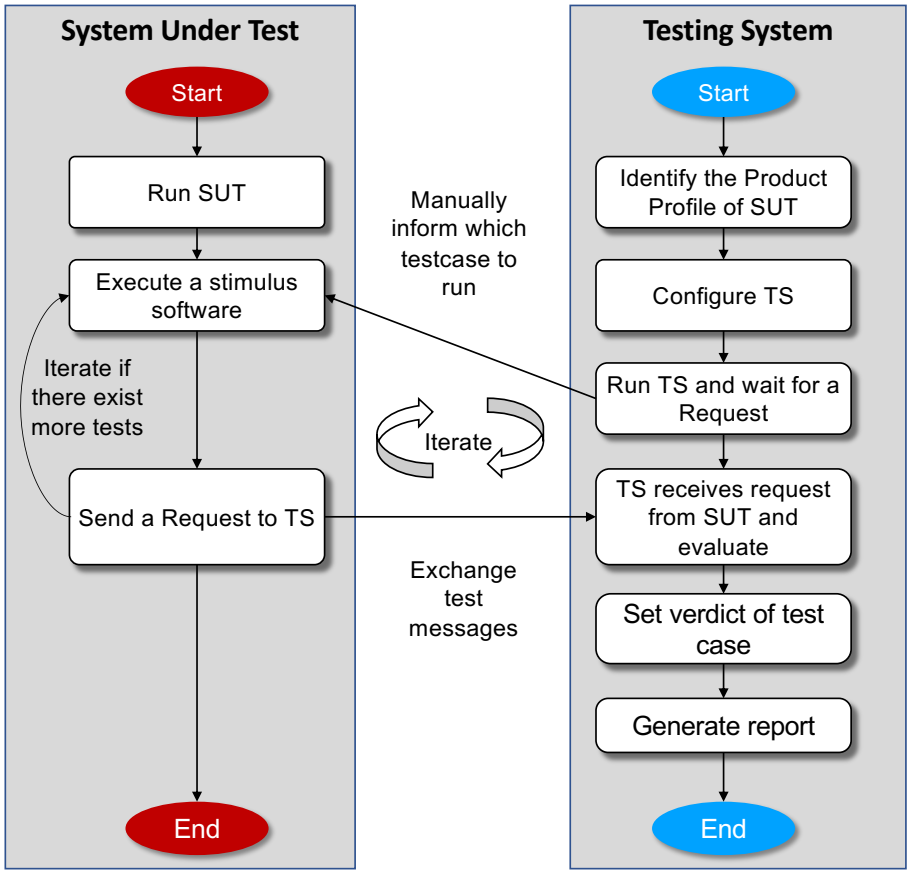

Fig. 19: Flow chart of IoT conformance testing procedures between TS and constraint SUT that needs external stimulus software.

In parallel, while the test case is running in the TS and waiting for a specific

request from SUT, one of operations such as create, read, update, or delete is constructed with headers and body format, and sent to the TS. As soon as the TS receives the request from the SUT, the TS completes the execution of the test case. Lastly, the TS sets the verdict of the test case as a result, and there are four types of verdicts: Pass-SUT behaves correctly according to the test purpose, Fail-SUT violates its test purpose, Inconclusive-neither a pass nor a fail can be assigned, Error-errors in the testing system [80]. After finishing all testing procedures, if the SUT has more features to test, the tester selects and runs each test case in the testing loop again. When the execution of all test cases is completed, the TS generates an overall report. Based on the report, if successful, this tested IoT device gets a certificate as a correct implementation of the standard.

As indicated in the flow chart of Fig. 19, some steps are executed in iterative manner. These steps in the testing procedures require human intervention for manual execution. The tester needs to select and run the test case in the TS and generate a specific request from the SUT according to the test cases. Performing these steps in a loop manually is a hectic and time consuming job, and it leads to inevitable mistakes and errors in the test report [81]. In this regard, assuming a situation where a tester wants to test an IoT device against four test cases which are named Test Case-1 (TC-1), TC-2, TC-3, and TC-4. The tester run the TC-1 in the TS, and the TS waits for the request from the SUT. Then, the tester initiates a request from the SUT according to the TC-1. The TS receives the request and completes the execution of the TC-1 and sets the verdict. As assumed, the tester needs to run all four test cases to successfully perform the conformance testing. As a next step, the tester runs the TC-2 in the TS, but mistakenly initiates the

wrong request from the SUT. The request from the SUT is not in accordance with the TC-2 expectation, and it triggers a false request. The overall report is then erroneous and affects the outcome of the certification process. This typical conformance testing method of IoT applications is not well capable for the IoT applications. There are many SMEs and individual developers that produce large amount of IoT applications. Therefore, it is legitimate to consider and examine the automatic conformance testing in the field of IoT. The key to the automatic conformance testing procedure for the IoT applications is to initiate and handle the trigger messages between the TS and the SUT to exchange information. In the next section, we thoroughly explain the triggering methodology along with the additional testing components and show how to turn exciting conformance testing approach into automated conformance testing.

3.3.2 Automated IoT testing architecture and procedures

In general, for the IoT application testing, after turning on the testing system, a testing expert may directly execute a test case and manually pass a specific requested message to the testing system to determine whether it is suitable. However, the problem is that testing experts have to manually run around dozens or hundreds test cases on IoT application functions, and during the testing process, testing experts might get unexpected testing errors, for example due to sending unintended information to testing system. Therefore, automated IoT testing architecture including Triggering message and Upper Tester (UT) was developed to support the automated conformance testing without human intervention. Figure. 20 shows the architecture for automatic conformance testing of the IoT applications in a testing lab.

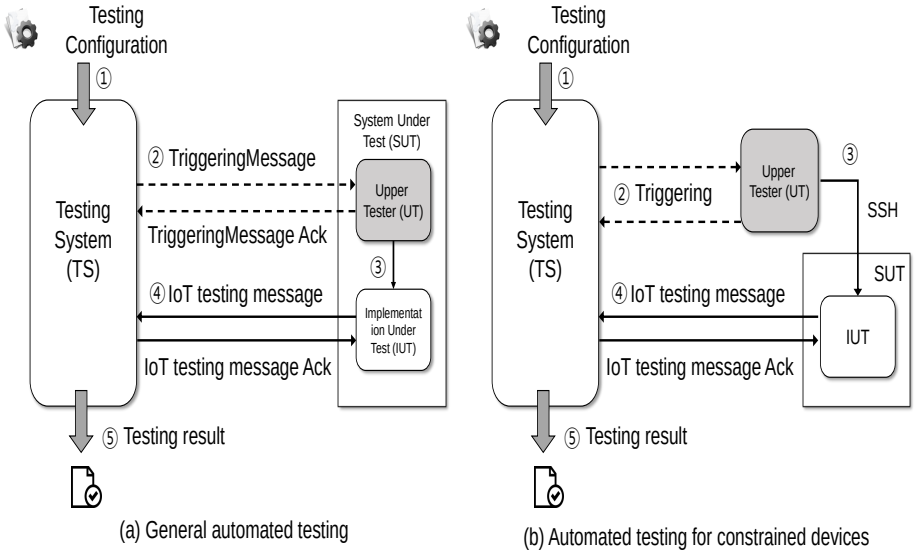

Fig. 20: The architecture of automatic conformance testing

- Testing Configuration: is the testing configuration files that contain the testing parameters written by the testing experts about what testing to perform. The testing parameters to perform specific testing cases can be IP address of testing system, test case name, IoT protocols, serialization formats and so on.

- Triggering message: is standardised message between the TS and the Upper Tester (UT). In order for TS to perform particular tests, it has to deliver the information based on the testing configuration. That is, the TS parses the configuration file and makes the triggering message including testing case name, protocol, serialization information coming from configuration file. As a next step, the TS delivers triggering message to the UT to start the conformance testing.

- Upper Tester (UT): is the key testing component that performs the triggering mechanism and reduces human intervention by automating the conformance testing process of the IoT applications. The UT receives the triggering message from the TS, and commands to the Implementation Under Test (IUT) that is a protocol implementation considered as an actual object performing communications. Triggering messages are standardized and to be sent between the TS and the UT. However, the way of performing the specific functions of the IUT can be implemented according to the IoT application manufacturers' development policies.

The actual procedures of the automatic conformance testing is similar to the procedures shown in the flow chart in Fig. 19, but in this subsection, the more detail procedures of automated conformance testing are provided. By using the two functionalities defined above, automated conformance testing for IoT applications can be performed and the general automated testing procedures are as described Fig. 20. (a).

1) The testing expert writes about what testing to perform on Testing Configuration and executes a testing case for a IoT device to be tested.

2) The testing system then delivers a triggering message containing the testing information to the UT. The triggering message contains the test case name, serialization, IoT protocol type, the address and port information of the TS. Also, if the testing system executes the test cases regarding the POST or PUT method, information about what device data should be sent to the TS is included.

3) The UT receiving the triggering message analyzes the message and checks

what testing should be performed. The UT then performs a specific function of the IUT, and it delivers the IoT testing message to the TS.

4-5) Finally, when the TS receive the IoT testing message, it analyse and verify the message. After that, testing experts can check whether the device is well implemented in accordance with the IoT standard.

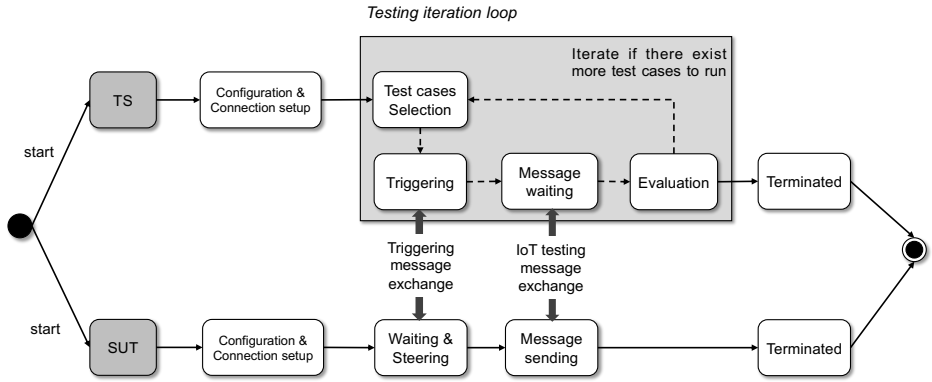

Fig. 21: Finite state machine showing automatic conformance testing for IoT Application

However, these previous procedures are only for the IoT applications having rich computation resources since UT is based on the HTTP server. There are only a few IoT devices can keep running the UT, and resource-constrained IoT devices are not suitable to deal with the UT as an HTTP server. In these resource constrained IoT devices, the UT cannot reside inside the SUT. To address this particular issue of testing, the UT that runs in resource-constrained IoT devices can dwell outside the SUT, which is depicted in Fig. 20 (b). There are various approaches to set up communication between the UT and the IUT, for example, Secure Shell (SSH) or Telnet. Accordingly, by using one of these options, the UT can communicate to the IUT in a similar way and also it can invoke a specific

operation to the IUT. For more detailed procedures of automated conformance testing, a finite state machine showing automatic conformance testing for IoT Application is presented as described in Fig. 21.

Another key to conduct the automatic conformance testing is to define the standard-based triggering message format between the TS and the UT. Triggering message is developed to deliver the control commands between Testing System and Upper Tester application. This command has to encompass testing parameters that are required for testing a specific test case. In addition, the Triggering message considers that most IoT standards are being developed based on the Resource-Oriented Architecture (ROA) which is one of the software architectures using the REpresentational State Transfer (REST) interfaces [82]. Take into account these requirements, the general triggering message is defined as follows.

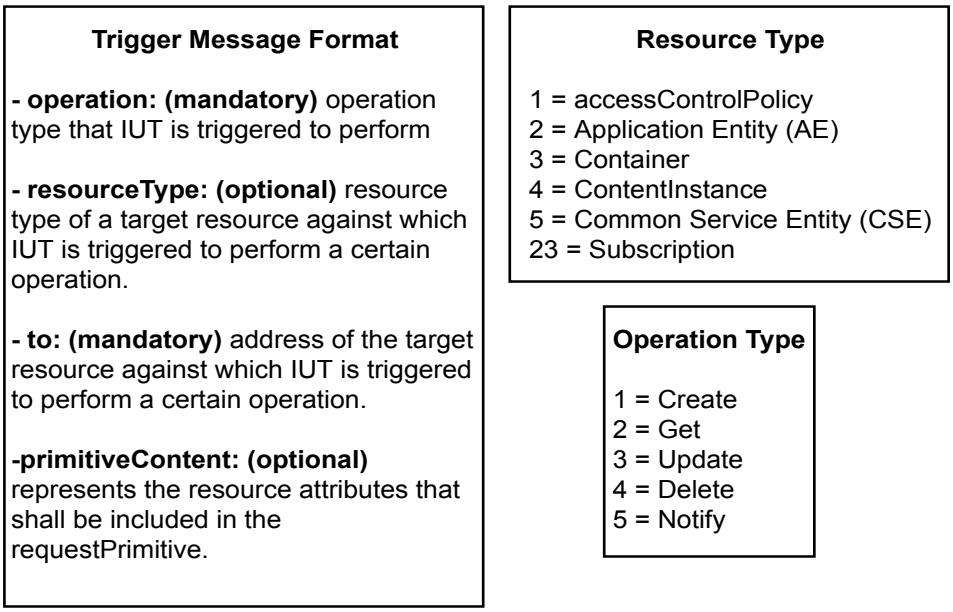

Fig. 22: The Triggering Message Format

Listing 1: Example of TriggerMessage format

```
{
  "m2m:rqp" :{
    "op": 1, //indicate CREATE operation
    "ty": 2, //indicate AE resource type
    "to": {TEST_SYSTEM_ADDRESS},
    "pc": {"m2m:ae": {
                          "lbl":"UNINITIALIZED"
    //indicate that attribute labels needs to be included
                          }
          }
      }
}
```

The body of the request has at most four key/value pairs, such as the operation, “op” and target testing system, “to” as mandatory parameters while the resourceType, “ty” and the primitiveContent, “pc” are optional parameters. First, “to” is the IP address of the TS, and it is used when IUT sends the testing request to the TS. The operation “op” parameter is numerically enumerated on numerous operations that are 1=Create, 2=Get, 3=Update, 4=Delete, and 5=Notify. If the test case has the POST or UPDATE procedure, body information has to be included in the triggering message. resourceType is the virtual representation of all kinds of logical components in the IoT systems such as data instance for saving measured value. primitiveContent represents the resource attributes that shall be included in the specific resource type. Any available information about resource type can be expressed in the form of attributes of the resource type. The response format of the Trigger message is composed with a response code. If the trigger message is correctly formatted by the IUT, then TS sends back the response message that is OK_REQUEST (2000). Otherwise, it is BAD_REQUEST (4000). The actual payload for the triggering message is shown in Listing. 1. This payload contains the parameter op: 1, which indicates a

CREATE operation, ty : 2 indicates the IoT application resource type, the to parameter points to the target resource address in our TS, and pc is an object that contains primitive contents with respect to the oneM2M standard specification. The receiving IoT application uses this pc object while sending the post request to the TS.

By using the approaches, IoT application conformance testing procedures are radically changed. Even though automated testing could seem very trivial, this approach will bring many advantages in terms of reducing testing completion time and errors coming from testers. In this regard, the conformance testing results between manual testing and automated conformance testing to prove the advantages of this approach are explained in the next section.

3.4 IoT conformance testing performance evaluation

This subsection shows the experimental results of an automatic conformance testing process for the IoT applications. In addition, oneM2M that is the global IoT standard is used to provide more practical example.

3.4.1 Evaluation configuration

As one of the oneM2M activities, oneM2M is developing the testing specification including conformance testing and interoperability testing. According to these specifications, oneM2M test cases based on TTCN-3 is being developed and provided (https://git.onem2m.org/TST/ATS) while oneM2M is defining the oneM2M Product Profiles specification including many feature sets

Profile	Description
ADN Profile 1	IoT application sensing data in a constrained IoT device such as a temperature sensor
ADN Profile 2	IoT application actuating things in a constrained device such as a dimmed light
ADN Profile 3	IoT application in a normal device such as a smartphone
ADN Profile 4	IoT application in a small originator device
IN Profile 1	Server device supporting IoT services
ASN Profile 1	IoT application in a normal actuator device
MN Profile 1	Gateway devices that support multiple different area network technologies and connect devices

Table 4: oneM2M Product Profiles

that have to be certified against the testing labs [49]. In this regard, Table 4 is presented as oneM2M product profiles example.

The oneM2M ADN Profile of Table 4 to examine the proposed technique was selected. In addition, the evaluation results of conformance testing are generated by using an IoT conformance testing system called oneM2MTester (https://github.com/IoTKETI/oneM2MTester) which is developed based on open source Eclipse Titan for oneM2M platform developers or testing engineers to test and debug their oneM2M implementations during or after the implementation development. In addition, nCUBE-thyme as SUT based on ADN profile was used. The nCUBE-Thyme is an open source for IoT devices, and it is based on the oneM2M standard (https://github.com/IoTKETI/nCube-Thyme-Nodejs). Meanwhile, oneM2M Testing working group is developing and maintaining the oneM2M test cases base on the TTCN-3, and in this code, the automated testing approach proposed in this dissertation has been reflected. Accordingly, the two types of conformance testing by setting the UT active/deactivate parameter in the configuration file can be selected. As shown in Fig. 23, a performance evaluation

to show the feasibility and practicality of the proposed conformance testing mechanism was conducted. Conformance testing for AE was performed on a computer with UBUNTU 16.04 LTS OS, an Intel Core i7-7800X CPU @ 3.50 GHz X2 processor, and 8.0 GB of memory.

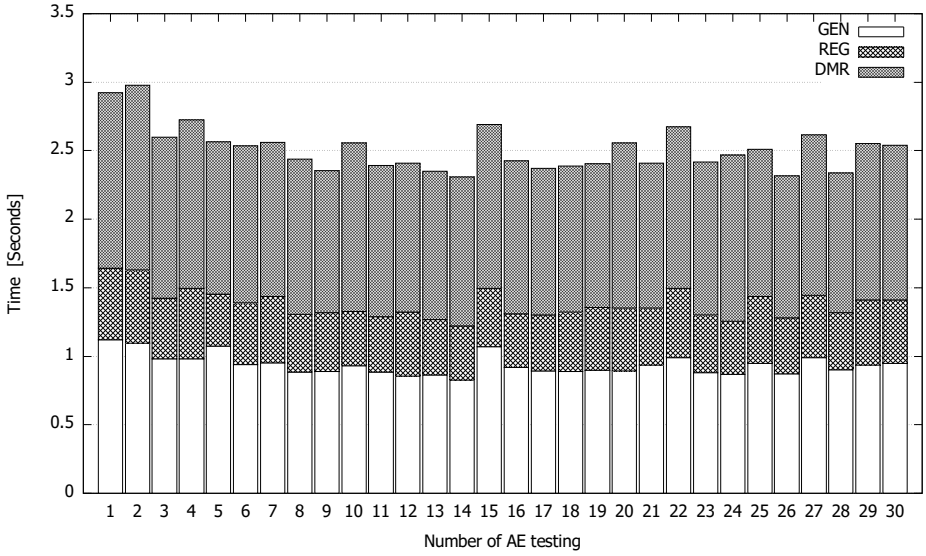

Fig. 23: The oneM2M IoT device testing performance evaluation

3.4.2 Testing performance comparison

For the testing, a total of sixty test cases for general IoT service layer capability (20 test cases), registration of IoT devices (9 test cases), data management-related features (27 test cases), and management of repository functions (4 test cases) were prepared. Such test cases were developed by the oneM2M Test Working Group as sections of testing specifications. An author of this dissertation participated and contributed to the development of the test cases together with other testing experts from industry. For each test case, AE

conformance testing was performed 30 times. It took 2.5 seconds to complete one test run for all sixty test cases, and the results showed a standard deviation of 0.165. Figure. 24 shows the GUI of the oneM2MTester after executing a set of test cases.

Testing Profile	Operation	Items	Automated testing	Hybrid testing	Manual testing
GEN	CREATE	6	0.359	11.3	30.4
	RETRIEVE	6	0.306	12.7	27.8
	UPDATE	6	0.195	9.4	28.7
	DELETE	6	0.261	10.5	29.6
REG	CREATE	8	0.466	14.4	31.4
	DELETE	1	0.205	3.5	30.2
DMR	CREATE	13	0.782	19.7	54.2
	RETRIEVE	7	0.267	14.2	27.8
	UPDATE	4	0.117	8.8	21.5
	DELETE	3	0.113	9.4	19.6
Total	-	60	3.071 (s)	113.9 (s)	274.2 (s)

Table 5: oneM2M IoT application conformance testing results with three approaches

In order to prove the performance advantage of automated conformance testing, conformance testing for an IoT application with three different testing configurations was conducted, i.e., manual testing, hybrid testing with dedicated stimulus, and automated testing. When the IoT application conformance testing was performed based on the automated testing approach as shown in Table. 5, it took 3.071 seconds to complete one test set, while stimulus tool-based hybrid testing and manual testing took approximately 113.9 and 274.2 seconds, respectively, to complete one test set.

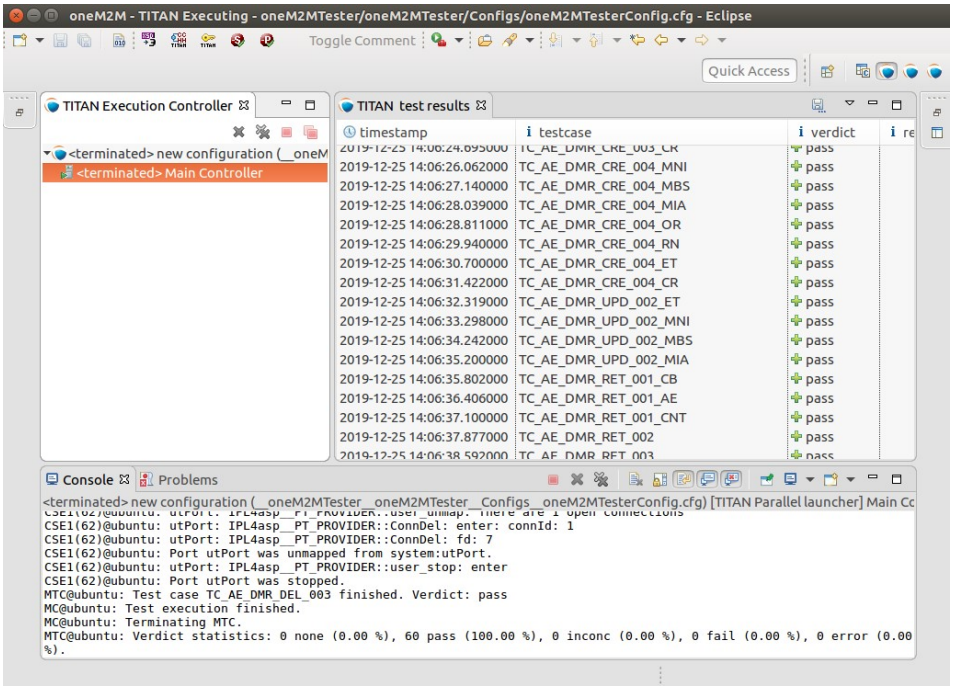

Fig. 24: A screen capture of automated conformance testing with oneM2MTester

The reason why automated testing performed better than manual and hybrid testing is that all procedures were automated, such as setting the testing parameters and the order of sending and receiving messages between the TS and the SUT so that all test cases could be verified in one execution. In other words, the method using automated testing minimized the tester's intervention, thereby reducing errors in performing testing. In the case of hybrid testing that uses dedicated software tools with predefined stimulus testers do not need to develop the tools from scratch, and they can reduce the redundant time by using tools' various support functions such as drop down testing menu, crafting test messages. However, radical problems still remain. To synchronize testing sessions between a testing system and target IoT application, at least two testers have to be involved in. In addition, if a way to test the IoT applications is

modified by an organization managing test specifications, these changes have to be reflected as soon as possible; however, it is not an easy task. As described in Table. 5, by using such tools, testers could get slightly better results than the one from manual testing. However, unless human involvement is fully replaced with standardized triggering message, it is not feasible to test large number of IoT applications.

3.5 Advanced techniques for IoT standard testing

There is a high demand for IoT standard testing framework to satisfy cost, coordination, heterogeneity, and scalability requirements. Therefore, to design and propose a novel IoT standard testing framework for resolving the issues, three IoT standard testing related ongoing research is analyzed as a preliminary step. First, F-Interop that uses the remote and distributed approach for conducting the conformance and interoperability testing is illustrated (https://www.f-interop.eu/). As subsequent research, oneM2MTester that aims to develop and distribute an open source oneM2M conformance testing tool is explained to analyze the extensible protocol support and an automated conformance testing approach (https://github.com/IoTKETI/oneM2MTester). Finally, FIESTA-IoT semantic testing which checks the contents of IoT messages to verify that these are properly annotated according to the semantic information based on referenced standards as described (http://fiesta-iot.eu/).

By using the advanced IoT standard testing techniques introduced in these three research, IoT Testing-as-a-Service (IoT-TaaS) architecture is defined. As a result, the newly proposed IoT-TaaS can complement the testing features used in

traditional testing, and also it will be helpful in terms of IoT-specific testing issues

Automated conformance testing: with regard to the conformance testing, it is being challenged by the feature of the variability of communication protocols in use. In this regard, IoT protocol bindings and automated testing can reduce the complexity of handling of such variabilities. As one of the standard activities regarding this, oneM2M that is standard for Machine-to-machine (M2M) and the IoT actively standardizes not only the basic specification including conformance and interoperability testing but also advanced IoT testing approaches such as protocol binding for testing and automated approach for IoT devices [83]. Herein, oneM2M is used to exemplify the IoT device conformance testing.

With the emerging of various IoT services, a range of communication protocols are being used by IoT devices. For instance, HyperText Transfer Protocol (HTTP) is used for general-purpose message transmission while the Message Queuing Telemetry Transport (MQTT) and the Constrained Application Protocol (CoAP) are defined for light-weight message exchange. The IoT testing framework has to consider supporting various communication protocols and standards [84]. However, it is not easy tasks for SMEs and developers to construct a testing environment for covering all these different protocols and testing features such as monitoring, reporting, and logging. To test these varied standards or protocols, a conformance testing tool that supporting them is required and oneM2MTester is being developed based on this motivation.As described in Fig. 17, TTCN-3 has the testing system and oneM2MTester is based on this architecture. In this architecture, there is an entity called SA to communicate with the SUT. In general, the TTCN-3 only supports standardized encoding schemes such as Basic Encoding Rules (BERs), Abstract Syntax

Notation One (ASN.1) and Packed Encoding Rules (PERs). To support the other protocols such as HTTP, MQTT and CoAP, testers need additional procedures to test IoT platforms and devices. In this regard, the oneM2MTester is developing the oneM2M dedicated SA including communication procedures and protocol binding procedures. In this way, any protocol can be added with the extension of the system adapter functionalities.

Compared to the traditional conformance testing, oneM2M conformance testing had the same issues in terms of testing costs. As traditional testing did, developers and vendors have to be physically the same place such as testing labs to conduct conformance testing. This typical conformance testing is not efficient and suitable for individual developers and small to medium-sized enterprises (SMEs) that are producing a lot of IoT devices in terms of testing costs and testing environment configuration since they have to bring their IoT devices to the lab. In addition, if the testing lab is abroad, the business trip fee will account for a very high percentage. Furthermore, oneM2M has defined hundreds of test cases for conformance testing. However, manually conducting lots of test cases is not feasible and time-consuming and highly labor-intensive. Therefore, it is natural to research and bring in the low-cost IoT testing mechanism.

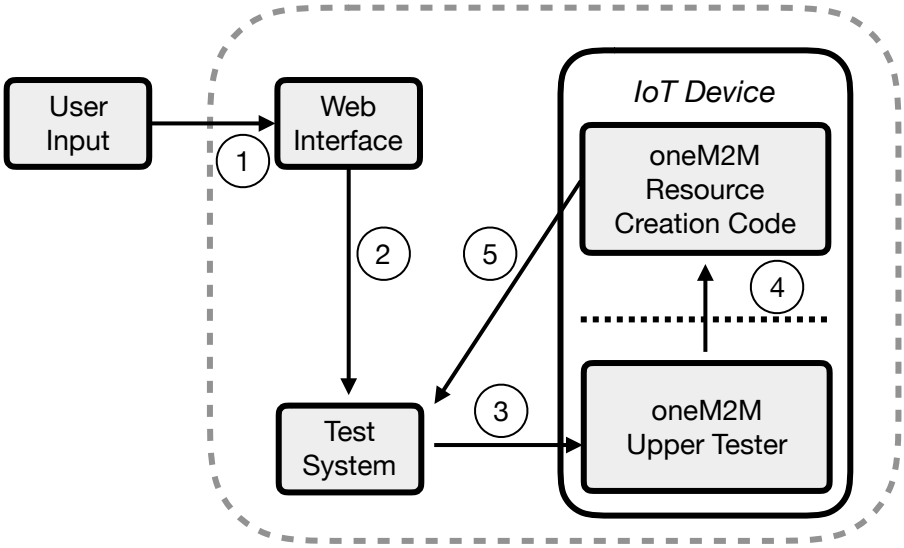

Fig. 25: Cloud-based automated IoT testing

Thus, a remote automated conformance testing framework based on web technologies can be considered as a promising solution for resolving the above problems. The core concept of web-based IoT testing is to place core conformance testing components to the cloud and provide the common APIs to the testers for setting the various configuration options based on their testing scenarios. In addition, by allowing other third parties to adopt their own protocol system adapter into the conformance testing system in the cloud, a web-based IoT testing framework can easily adopt new IoT protocols. Regarding the automated conformance testing, the key concept is to trigger the IoT devices to start the communication of a test case. This Upper Tester (UT) is playing the role to realize this. The UT is a logical software component and it can be implemented within an IUT or outside of the IUT according to the capability of the IUT. The upper tester in the SUT is a lightweight server that receives a

triggering message from the TS and coordinates the testing behaviour of the SUT. It parses a received triggering message to retrieve the data from the test case in the message body and initiates the predetermined operation. For more practical example of oneM2M automated conformance testing, brief procedures are explained in Fig. 25.

1-2) The testing expert writes about what testing to perform and executes a testing case for a specific IoT device through the web GUI. 3) The testing system then delivers a triggering message containing the testing information to the UT. The triggering message contains the test case name, serialization, protocol type, the address and port information of the testing system. Also, if the testing system executes the test cases regarding the POST or PUT method, Information about what device data should be sent in the body format is included. 4) The UT receiving the triggering message analyzes the message and checks what testing should be performed. The UT then performs a specific function of the IUT and delivers the message to the testing system. 5) Finally, when the device sends responses for the function to be verified in the testing system, verdicts such as pass and fail can be judged, and it can be checked whether the device is well implemented for the standard.

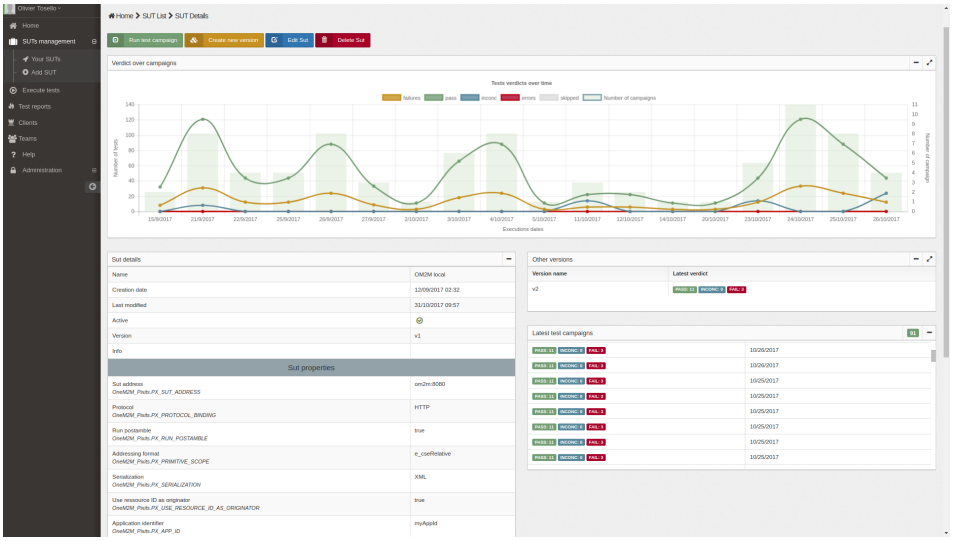

Fig. 26: oneM2MTester conformance testing result screen shot capture

Based on this concept, oneM2MTester that is web-based oneM2M conformance testing framework is developed. By using this tool, thanks to the remote IoT device testing feature of oneM2MTester, testers can test their IoT devices without visiting IoT testing labs. In addition, with the automated testing approach developers can debug their products during or after the implementation development automatically. Figure. 26 shows the captured screenshot of web-based oneM2MTester, and it shows an overview of the conformance testing results in history against an IUT based on oneM2M.

Remote Interoperability testing: A testing system for an IoT devices should provide web-based remote testing features. The main object of such a testing system is to support a platform for remote testing for accelerating the development speed in terms of standards-based IoT products [85]. A testing system called *F-Interop* has the feature that enables the IoT devices testing located in remote places [86].

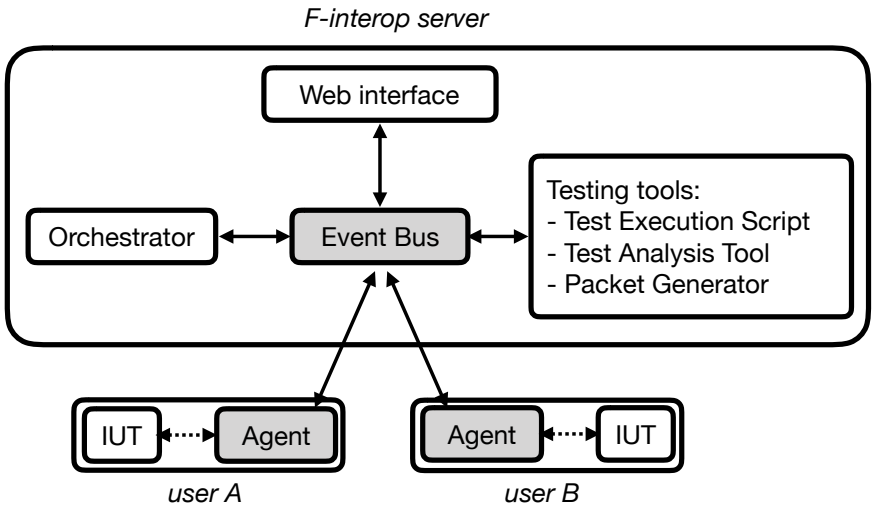

Fig. 27: High-level architecture of F-Interop

The high-level architecture of F-Interop is depicted in Fig. 27 [87]. Compared to traditional software testing, the novelty of F-Interop is using the remote testing approach by having a centralized testing area. For realizing this idea, the testing system provides an agent that is secured communication wrapper for the IUT. After the IUT connection is established through the agent, the testing system creates an isolated area for testing. When the configuration setup is finished, the *test execution script* is executed. This execution consists of three steps. `STIMULI` to trigger the IUT to initiate a specific action, `CHECK` to verify communication (e.g. capture the contents of the packet), and `VERIFY` to reflect the desired result. *Test Analysis Tools* verify output data and behavior after communication. The tool then generates verdicts for the tests in `PASS`, `FAIL` and `INCONCLUSIVE`. The testing system has own packet generator for input generation and supports a web interface to enable testers to check status checking, manually input decision and commands when needed.

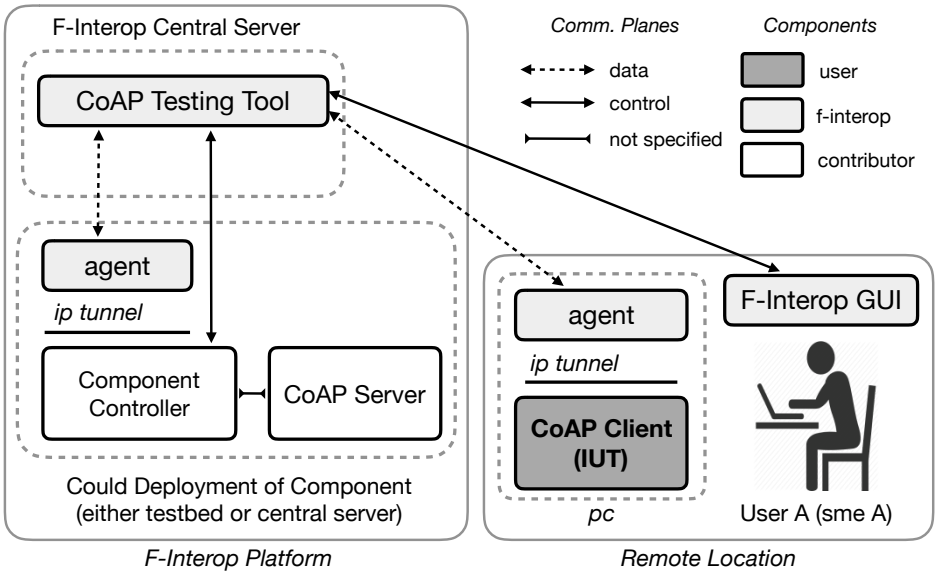

Fig. 28: Architecture of CoAP protocol testing at F-Interop

F-Interop's architecture is designed to handle the entire process of IoT interoperability testing with a focus on test management and related tasks such as tracking management and reporting verdicts. All of these tasks are controlled and coordinated by *Orchestrator* that manages the collaboration of various test components such as test sessions, message brokers and access control. Communications are encrypted and managed by the Event Bus. These include control messages, data and log packets being sent to the central event bus implemented in RabbitMQ (https://www.rabbitmq.com/). This centralized communication architecture ensures the independence of each component.

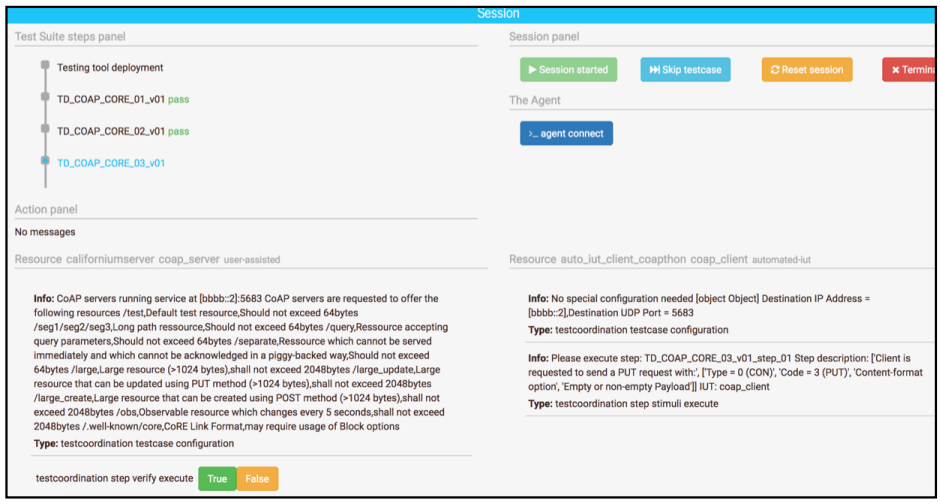

Fig. 29: GUI screen image of CoAP protocol testing as F-Interop

To exemplify the remote interoperability testing, CoAP protocol testing is illustrated. As shown in Fig. 28, F-Interop's CoAP interoperability testing consists of analyzing data exchanged between CoAP IUTs through a CoAP testing tool located on the F-Interop central server. This test tool can support three interoperability scenarios to test CoAP IUTs in remote locations. (1) the IUT is connected to the F-Interop test server. (2) IUT is connected to another CoAP device running in the testbed. (3) The IUT interacts with other vendors' IUTs through the F-Interop testing tool. Using these various test scenarios, the User A can rigorously test the interoperability of the IUT. In addition, the actual F-Interop web GUI is showed in Fig. 29.

Semantics testing for IoT devices: In the previous section, protocol-level testing mainly analyzed by researching conformance and interoperability testing. However, IoT platforms and devices are being developed based on different levels. Therefore, it is valuable to test different levels such as semantics.

Semantic testing aims to test the accuracy of the semantic description of an IoT data stream against the standard. Studies conducted on these challenges relating to Semantic Web have concluded that one of the key aspects of achieving semantic interoperability is the ontology and data annotation using these ontologies. Standardization organizations have already defined reference ontology such as ETSI-SAREF ontology (ETSI Smart Appliance Reference ontology), W3C-SSN ontology (Semantic Sensor Network Ontology) and oneM2M basic ontology. Once the reference ontology is defined, one important step for ensuring semantic interoperability is to test its conformance against the reference ontologies. However, Semantic testing of the IoT environment is especially challenging because of the large diversity of semantic modeling methods and a large number of concepts and relations between these concepts. For instance, IoT-Lite optimized performance for handling large and various semantic models of IoT by developing a light-weight semantic model.

Several research projects are working to address these challenges by proposing and applying these semantic assessment methods and tools. The H2020 *Fiesta-IoT* project is one of the projects that aims to build a unique cloud platform of semantically federated IoT testbeds for new experiments based on semantic technology. The platform provides a unique access point for rich semantic data coming from various types of testbeds (e.g. crowdsensing applications, smart cities, smart campuses and smart buildings). For an appropriate semantic database that provides accurate data about the agreed-upon ontology to be used by experimenters relying on standard semantic technology, semantic verification operations are applied to the incoming data stream. If the data does not meet all the requirements related to syntactic and semantic

accuracy, it is rejected to keep the database clean and accurate. Therefore, it provides a full report with a description of all the pitfalls that could lead to modeling errors. It assists ontology developers during the correction process. Semantic testing is essential to achieve semantic interoperability. The case study presented to demonstrate that semantic testing is needed in the IoT industry to verify that IoT systems are working as expected. As we claim, semantic testing should include multiple levels of semantic interpretation, such as vocabulary, syntax, semantics and cardinal accuracy, which are important for semantic interoperability.

One of the main reasons to have a semantic validator is to check for semantic annotations (data annotated for ontology). For example, there is an IoT platform that has introduced a meta-test bed IoT/cloud infrastructure to unite various IoT testbeds distributed around the world. These platforms aim to overcome the barriers between technology and data silos by applying semantic technology. Any user who wants to access data, regardless of location and connected IoT platform, can send data annotated against a common ontology to the semantic repository. From this perspective, the user should check the consistency of the annotation. Otherwise, the platform will reject the data. In this way, the Semantic Validation Tool is a key element of the IoT platform that helps users generate correct annotations. In fact, the tool provides a detailed validation report that lists various errors.

```
<owl:NamedIndividualrdf:about="&ns0;Bob">
   <rdf:typerdf:resource="&ns0;young"/>
   <rdf:typerdf:resource="&ns0;adult"/>
</owl:NamedIndividual>
```

Fig. 30: Example of invalid semantic resource

For instance, *Semantic Verification* can be exemplified. Validation tools can determine ontology consistency, identify subsumption relationships among classes, etc. To perform semantic validation, the developers upload the ontology to the tool, then checks the syntactic of ontology format and URI validation and return the reports. The tool also demonstrates the semantic correctness of predicates. For example, consider the list of annotation list in Fig. 30, which contains two classes (adult and young person) defined as common classes. This list includes inconsistencies when defining instances (Bobs) which is the person with young and adult classes. When a developer enters these annotated resources into the validator, the mismatch in annotations results in semantic or logical errors.

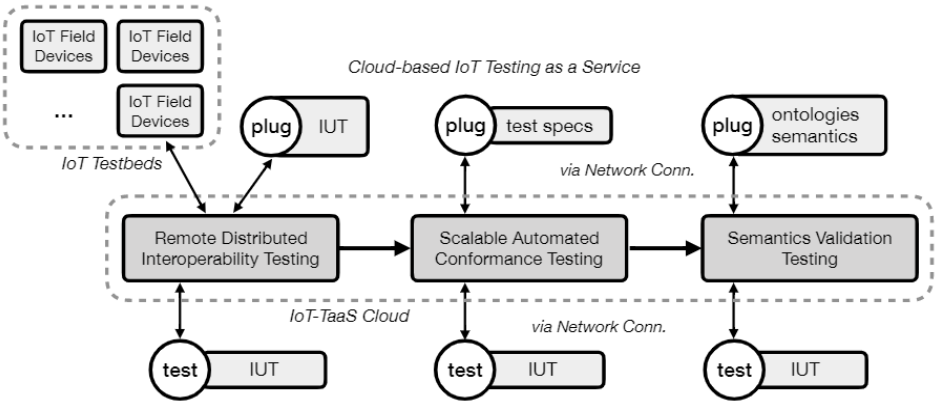

Fig. 31: Architecture of cloud-based IoT Testing-as-a-Service

This subsection introduced a series of enhancements to the traditional conformance and interoperability testing concepts, along with semantic validation to test various IoT platforms and devices. As these concepts are now implemented and adopted as tools for IoT certifications, it is expected that these concepts will play a key role in the modern IoT testing framework. These three main IoT test concepts can be developed into IoT-TaaS's "plug and test" concept as described in Fig. 31. This plug-and-test concept makes it easy to extend the functionality of the test system without modifying the test system itself. For example, conformance testing can plug test specifications (including test cases and profiles) and perform tests on the IUT. The semantic validation test also enables various ontology depending on the ontology used in the target document and can verify the correctness of the semantics used in the document. Finally, interoperability testing provided by IoT-TaaS allows other field IoT devices to verify the target IoT device. If there is a demand for different types of testing to IoT-TaaS, the new test set cases, ontology information, and new messages can be easily added to the system. These design concepts can provide a practical architectural concept for practitioners designing and implementing IoT-TaaS for the IoT ecosystem. IoT-TaaS offers significant advantages to accelerate the development of standards-based IoT products.

3.6 Summary

At present, the IoT combined with technologies such as artificial intelligence and 5G is applied across many industries as well as people's lives. However, it would be predicted that the operation without proper verification of the IoT applications might cause several problems such as the device or platform not working well,

and finally, it makes serious problems to all the industries and people's lives. Therefore, proper verification must be performed before operating IoT applications. In this regard, the traditional IoT conformance testing method requires testing experts to manually test the IoT applications, and it is not easy to set the testing environment and execute all the test cases manually. Therefore, to solve the problems above, the automated IoT conformance testing approach including architecture and implementation are developed and showed how to conduct automated IoT conformance testing with oneM2M IoT standard. This mechanism has been reflected in the oneM2M IoT standard and being used IoT testing labs such as TTA to test the IoT applications. To conclude, by using the automated testing approach, SMEs will benefit from time and cost, and SDOs can speed up the propagation of the adoption of IoT standards.

4 Conclusion

These days various industry field is considering the introduction of the IoT, but they might not operate efficiently without solving various problems that IoT have. Therefore, in this dissertation, research was conducted to solve the problems related to interoperability, availability, and reliability as problems that can occur when operating the IoT services.

First, by proposing three interworking models to solve the interoperability problem among IoT standard with the smart city scenario, the approaches to efficiently connect IoT platforms that are using the same or different standards were explained. Therefore, as these models have their own pros and cons, it would predicted that developers should carefully consider their own environment before deciding on any IWMs. In addition, an automated IoT testing method is newly defined and it proved that not only errors during the testing are reduced by minimizing human intervention but also the testing time can be shortened. As a result, it contributes to the rapid propagation of IoT standards by reducing the time required of the testing and certification process for the testing certification body. Therefore, it is expected that the IoT environments can be operated more efficiently and stably through understanding and solving various problems occurring in the current IoT services by performing the above research. In addition, the technologies developed in this dissertation are expected to be globally applicable to all industries.

Currently, AI is changing all industry aspects with its strong potential. The AI can provide automation of system management and can extract meaningful data in the huge data pool. Therefore, the AI can be used for predicting the resource usage

of IoT devices and platforms, and by using this, IoT platforms can manage all nodes automatically and elasticity. In this context, as the next promising research for the IoT, research for integrating the AI into the IoT platform is going to be conducted.

References

[1] B. Ahlgren, M. Hidell, and E. C.-H. Ngai, "Internet of things for smart cities: Interoperability and open data," *IEEE Internet Computing*, vol. 20, no. 6, pp. 52–56, 2016.

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[3] J Manyika, M Chui, P Bisson, J Woetzel, R Dobbs, J Bughin, and D Aharon, *The internet of things: Mapping the value beyond the hype. mckinsey glob inst 144*, 2015.

[4] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common m2m service layer platform: Introduction to onem2m," *IEEE Wirel. Commun.*, vol. 21, no. 3, pp. 20–26, 2014.

[5] S. Husain, A. Kunz, J. Song, and T. Koshimizu, "Interworking architecture between onem2m service layer and underlying networks," in *2014 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2014, pp. 636–642.

[6] R. Zhao, L. Wang, X. Zhang, Y. Zhang, L. Wang, and H. Peng, "A onem2m-compliant stacked middleware promoting iot research and development," *IEEE Access*, vol. 6, pp. 63 546–63 559, 2018.

[7] J. Hwang, J. An, A. Aziz, J. Kim, S. Jeong, and J. Song, "Interworking models of smart city with heterogeneous internet of things standards," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 74–79, 2019.

[8] *Onem2m ts-0001: Functional architecture*, oneM2M, 2020. [Online]. Available: `http://www.onem2m.org/technical/published-drafts/release-4`.

[9] P. Fernández, J. Santana, S. Ortega, A. Trujillo, J. Suárez, C. Domínguez, J. Santana, and A. Sánchez, "Smartport: A platform for sensor data monitoring in a seaport based on fiware," *Sensors*, vol. 16, no. 3, p. 417, 2016.

[10] M. Bauer, J. Song, S. Jeong, C. Lopéz, R. Druilhe, H. Minh, F.-J. Wu, and J. Hwang, *D2.4 - semantic interoperability components r1-1.0.2*, Wise-IoT, 2017.

[11] E. Kovacs, M. Bauer, J. Kim, J. Yun, F. Le Gall, and M. Zhao, "Standards-based worldwide semantic interoperability for iot," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 40–46, 2016.

[12] C. L. Martin Bauer SeungMyeong Jeong, H. M. Rémi Druilhe, and J. Hwang, *D1.3 - wise iot updated high level architecture and reference technologies and standards*, Wise-IoT, 2017.

[13] S. Park, "Ocf: A new open iot consortium," in *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, IEEE, 2017, pp. 356–359.

[14] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo, "A survey on facilities for experimental internet of things research," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 58–67, 2011.

[15] J. Yun, R. C. Teja, N. Chen, N.-M. Sung, and J. Kim, "Interworking of onem2m-based iot systems and legacy systems for consumer products," in

2016 International Conference on Information and Communication Technology Convergence (ICTC), IEEE, 2016, pp. 423–428.

[16] C. Sarkar, S. A. U. Nambi, R. V. Prasad, and A. Rahim, "A scalable distributed architecture towards unifying iot applications," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, IEEE, 2014, pp. 508–513.

[17] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, 2013.

[18] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-khah, and P. Siano, "Iot-based smart cities: A survey," in *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, IEEE, 2016, pp. 1–6.

[19] 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 & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[20] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, and S. Guizani, "Internet-of-things-based smart cities: Recent advances and challenges," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 16–24, 2017.

[21] R. Wenge, X. Zhang, C. Dave, L. Chao, and S. Hao, "Smart city architecture: A technology guide for implementation and design challenges," *China Communications*, vol. 11, no. 3, pp. 56–69, 2014.

[22] *The internet of things-enabled smart city framework, or ies-city framework*, Feb 11, 2019; https://pages.nist.gov/smartcitiesarchitecture/, National Institute of Standards and Technology (NIST).

[23] AFT and V. Foundation, *Stakeholders and market analysis report final v1.1*, Feb 11, 2019, http://www.inter-iot-project.eu/deliverables, Inter-IoT.

[24] E. E. Kim and S. Ziegler, "Towards an open framework of online interoperability and performance tests for the internet of things," in *2017 Global Internet of Things Summit (GIoTS)*, IEEE, 2017, pp. 1–6.

[25] S. Brady, A. Hava, P. Perry, J. Murphy, D. Magoni, and A. O. Portillo-Dominguez, "Towards an emulated iot test environment for anomaly detection using nemu," in *2017 Global Internet of Things Summit (GIoTS)*, IEEE, 2017, pp. 1–6.

[26] B. S. Ahmed, M. Bures, K. Frajtak, and T. Cerny, "Aspects of quality in internet of things (iot) solutions: A systematic mapping study," *IEEE Access*, vol. 7, pp. 13 758–13 780, 2019.

[27] B. Sand, "Iot testing-the big challenge why, what and how," in *International Internet of Things Summit*, Springer, 2015, pp. 70–76.

[28] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview," *The Internet Society (ISOC)*, vol. 80, pp. 1–50, 2015.

[29] R. Petrolo, V. Loscrì, and N. Mitton, "Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms," *ETT*, vol. 28, no. 1, e2931, 2017.

[30] *Smart cities done smarter*, Jul, 2018; http://www.onem2m.org/images/files/oneM2M_WhitePaper_SmartCitiesDoneSmarter.pdf, oneM2M.

[31] J. Kim, J. Yun, S.-C. Choi, D. N. Seed, G. Lu, M. Bauer, A. Al-Hezmi, K. Campowsky, and J. Song, "Standard-based iot platforms interworking:

Implementation, experiences, and lessons learned," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 48–54, 2016.

[32] *Onem2m-ts-0001: Functional architecture*, oneM2M, 2016. [Online]. Available: `http://www.onem2m.org/technical/published-documents`.

[33] *Onem2m-tr-0025: Application developer guide*, oneM2M, 2016. [Online]. Available: `http://www.onem2m.org/technical/published-documents`.

[34] T. Berners-Lee, J. Hendler, O. Lassila, *et al.*, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[35] *Ontology*, June, 2020; http://www.frotoma.com/sub2_1.do, Frotoma.

[36] 황석형/양해술, *시맨틱 웹을 위한 RDF/OWL 입문*. 홍릉 과학 출판사, 2008.

[37] *Tr-0007: Study on abstraction and semantics enablement*, oneM2M, 2016. [Online]. Available: `http://www.onem2m.org/technical/published-documents`.

[38] R. Agarwal, D. G. Fernandez, T. Elsaleh, A. Gyrard, J. Lanza, L. Sanchez, N. Georgantas, and V. Issarny, "Unified iot ontology to enable interoperability and federation of testbeds," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, IEEE, 2016, pp. 70–75.

[39] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff, "Semantic annotation, indexing, and retrieval," *Journal of Web Semantics*, vol. 2, no. 1, pp. 49–79, 2004.

[40] J. Euzenat, “Eight questions about semantic web annotations,” *IEEE Intelligent systems*, vol. 17, no. 2, pp. 55–62, 2002.

[41] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, “Semantics for the internet of things: Early progress and back to the future,” *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 8, no. 1, pp. 1–21, 2012.

[42] J. Strassner and W. W. Diab, “A semantic interoperability architecture for internet of things data sharing and computing,” in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, IEEE, 2016, pp. 609–614.

[43] A. Gyrard and M. Serrano, “A unified semantic engine for internet of things and smart cities: From sensor data to end-users applications,” in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, IEEE, 2015, pp. 718–725.

[44] *D3.4 - specification and implementation of common testbed interfaces v2*, FIESTA-IoT, 2017.

[45] M. L. Loper and B. Swenson, “Machine to machine trust in smart cities,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2017, pp. 1887–1889.

[46] A. Kaiser and S. Hackel, “Standards-based iot testing with open-source test equipment,” in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, 2019, pp. 435–441.

[47] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, “Future internet: The internet of things architecture, possible applications and key challenges,” in *2012*

10th international conference on frontiers of information technology, IEEE, 2012, pp. 257–260.

[48] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[49] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," *Wireless personal communications*, vol. 58, no. 1, pp. 49–69, 2011.

[50] J. Song, A. Kunz, M. Schmidt, and P. Szczytowski, "Connecting and managing m2m devices in the future internet," *Mobile Networks and Applications*, vol. 19, no. 1, pp. 4–17, 2014.

[51] A. Vallejo, J. Ruiz, J. Abella, A. Zaballos, and J. M. Selga, "State of the art of ipv6 conformance and interoperability testing," *IEEE Communications Magazine*, vol. 45, no. 10, pp. 140–146, 2007.

[52] S. Seol, M. Kim, S. Kang, and J. Ryu, "Fully automated interoperability test suite derivation for communication protocols," *Computer Networks*, vol. 43, no. 6, pp. 735–759, 2003.

[53] S. Maag and C. Grepet, "Interoperability testing of a manet routing protocol using a node self-similarity approach," in *Proceedings of the 2008 ACM symposium on Applied computing*, 2008, pp. 1908–1912.

[54] R. Hao, D. Lee, R. K. Sinha, and N. Griffeth, "Integrated system interoperability testing with applications to voip," *IEEE/ACM transactions on networking*, vol. 12, no. 5, pp. 823–836, 2004.

[55] M. Schmidt, A. Wilde, A. Schulke, and H. Costa, "Ims interoperability and conformance aspects [ip multimedia systems (ims) infrastructure and services]," *IEEE Communications Magazine*, vol. 45, no. 3, pp. 138–142, 2007.

[56] O. Koné and R. Castanet, "Test generation for interworking systems," *Computer Communications*, vol. 23, no. 7, pp. 642–652, 2000.

[57] Y. Zhang and Z. Li, "Ipv6 conformance testing: Theory and practice," in *2004 International Conferce on Test*, IEEE, 2004, pp. 719–727.

[58] W. E. Howden, "Functional program testing," *IEEE Transactions on Software Engineering*, no. 2, pp. 162–169, 1980.

[59] D. Rayner, "Osi conformance testing," *Computer Networks and ISDN Systems*, vol. 14, no. 1, pp. 79–98, 1987.

[60] M. Krichen and S. Tripakis, "Conformance testing for real-time systems," *Formal Methods in System Design*, vol. 34, no. 3, pp. 238–304, 2009.

[61] S. T. Moez Krichen, "Black-box conformance testing for real-time systems," in *International SPIN Workshop on Model Checking of Software*, Springer, 2004, pp. 109–126.

[62] S. Moseley, S. Randall, and A. Wiles, "Experience within etsi of the combined roles of conformance testing and interoperability testing," in *Standardization and Innovation in Information Technology, 2003. The 3rd Conference on*, IEEE, 2003, pp. 177–189.

[63] *Methods for testing and specification (mts); deployment of model-based automated testing infrastructure in a cloud*, European Telecommunications Standards Institute (ETSI), 2016.

[64] J. Grabowski, D. Hogrefe, G. Réthy, I. Schieferdecker, A. Wiles, and C. Willcock, "An introduction to the testing and test control notation (ttcn-3)," *Computer Networks*, vol. 42, no. 3, pp. 375–403, 2003.

[65] C. Willcock, T. Dei, S. Tobies, S. Keil, F. Engler, and S. Schulz, *An introduction to TTCN-3*. Wiley Online Library, 2005, vol. 2.

[66] *Interoperability best practices*, ETSI. [Online]. Available: `https://www.etsi.org/images/files/Events/interoperability_best_practices_handbook.pdf`.

[67] *Onem2m-ts-0013: Interoperability testing*, oneM2M, 2018. [Online]. Available: `http://onem2m.org/technical/published-drafts/release-3`.

[68] M. Bures, "Framework for integration testing of iot solutions," 2017.

[69] A. Subahi and G. Theodorakopoulos, "Ensuring compliance of iot devices with their privacy policy agreement," 2018.

[70] L. Leah Riungu-Kalliosaari, O. Taipale, and K. Smolander, "Testing in the cloud: Exploring the practice," *IEEE Software*, vol. 29, no. 2, pp. 46–51, 2012.

[71] J. Gao, X. Bai, W.-T. Tsai, and T. Uehara, "Testing as a service (taas) on clouds," 2013.

[72] R. M. Palattella, F. Sismondi, T. Chang, L. Baron, M. Vučinić, P. Modernell, and T. W. Vilajosana, "F-interop platform and tools: Validating iot implementations faster," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, vol. 6, IEEE, 2018, pp. 15 480–15 493.

[73] I. Schieferdecker, S. Kretzschmann, A. Rennoch, and M. Wagner, "Iot-testware - an eclipse project," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, 2017, pp. 1–8.

[74] W.-K. Chen, C.-H. Liu, W. W.-Y. Liang, and M.-Y. Tsai, "Icat: An iot device compatibility testing tool," 2018.

[75] I. Dobles, A. Martínez, and C. Quesada-López, "Comparing the effort and effectiveness of automated and manual tests," in *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*, IEEE, 2019, pp. 1–6.

[76] C. Klammer and R. Ramler, "A journey from manual testing to automated test generation in an industry project," pp. 591–592, 2017.

[77] S. Thummalapenta, S. Sinha, N. Singhania, and S. Chandra, "Automating test automation," pp. 881–891, 2012.

[78] E. S. Reetz, D. Kuemper, K. Moessner, and R. Tönjes, "How to test iot-based services before deploying them into real world," in *European Wireless 2013; 19th European Wireless Conference*, VDE, 2013, pp. 1–6.

[79] H. Kim, A. Ahmad, J. Hwang, H. Baqa, F. Le Gall, M. A. R. Ortega, and J. Song, "Iot-taas: Towards a prospective iot testing framework," *IEEE Access*, vol. 6, pp. 15 480–15 493, 2018.

[80] F. Muhammad, *An Introduction to Umts Technology: Testing, Specifications and Standard Bodies for Engineers and Managers*. Universal-Publishers, 2008.

[81] W. F. Gonçalves, C. B. de Almeida, L. L. de Araújo, M. S. Ferraz, R. B. Xandú, and I. de Farias, "The influence of human factors on the software

testing process: The impact of these factors on the software testing process," in *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, IEEE, 2017, pp. 1–6.

[82] Y. Hong, "A resource-oriented middleware framework for heterogeneous internet of things," in *2012 International Conference on Cloud and Service Computing*, IEEE, 2012, pp. 12–16.

[83] *Ts-0015: Testing framework*, oneM2M, 2016. [Online]. Available: `http://www.onem2m.org/technical/published-documents`.

[84] I. Schieferdecker, S. Kretzschmann, A. Rennoch, and M. Wagner, "Iot-testware-an eclipse project," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, 2017, pp. 1–8.

[85] O. Vermesan, P. Friess, *et al.*, *Internet of things-from research and innovation to market deployment*. River publishers Aalborg, 2014, vol. 29.

[86] S. Ziegler, S. Fdida, C. Viho, and T. Watteyne, "F-interop–online platform of interoperability and performance tests for the internet of things," in *Interoperability, Safety and Security in IoT*, Springer, 2016, pp. 49–55.

[87] R. Leone, F. Sismondi, T. Watteyne, and C. Viho, "Technical overview of f-interop," in *Interoperability, Safety and Security in IoT*, Springer, 2016, pp. 11–17.

국문초록

사물인터넷을 위한 인터워킹 모델 및 자동화 적합성 테스팅

세종대학교 대학원

컴퓨터공학과 정보보호

황재영

오늘날 사물인터넷 (Internet of Things) 기술은 인공지능, 5G와 함께 다양한 사회적 문제점을 해결하고, 4차 산업혁명을 이끌 주요 기술로 간주되고 있다. 또한, 센서, 가전제품, 액추에이터, 자율주행차와 같이 다양한 사물들이 인터넷을 통해 서로 연결되고 상호작용할 수 있게 함으로써 스마트 그리드, 교통관리, 홈오토메이션 및 의료 분야 등 다양한 영역에서 활용되고 있다. 따라서, 사물인터넷은 커다란 성장 잠재력을 가진 유망한 기술 중 하나로 간주되고 있으며, 유럽연합을 비롯하여 많은 국가들이 정보통신기술 (ICT, Information and Communication Technology) 연구에 집중적으로 투자하고 있다.

그러나 위와 같은 서비스를 제공하는데 필수적으로 고려되어야 할 확장가능하고 안정적인 사물인터넷 서비스를 제공하는 것에는 많은 어려움이 따른다. 지금까지 개발된 대부분의 사물인터넷 플랫폼 및 시스템들은 해당 사물인터넷 플랫폼 제조업체의 독자적인 표준에 따른 데이터 모델 및 애플리케이션 프로그래밍 인터페이스 (APIs) 등을 활용하여 사물인터넷 서비스를 제공하기 때문에

다른 플랫폼과는 상호운용성이 보장되지 않는 문제를 발생시켰다. 상호운용성이 보장되지 않을 경우 사물인터넷 디바이스 또는 플랫폼은 서로 다른 표준에 따라 개발된 시스템과의 연동 또는 데이터에 대한 교환이 불가능하다. 따라서, 다른 표준을 지원하기 위해 이미 구축된 시스템을 바꾸는 것은 상당한 비용을 수반하며, 대규모 사물인터넷 기술 도입 지연, 운영 비용 증가 등의 경제적 및 기술적인 문제를 발생시킬 수 있다. 사물인터넷 표준 테스팅 측면에서, 동일한 또는 다른 표준에 기반한 독립적인 사물인터넷 시스템들이 상호운용 될 수 있도록 보장하기 위하여 적합성 및 상호운용성 테스트가 중요하다. 또한, 다양한 사물인터넷 표준 기반의 사물인터넷 디바이스 및 플랫폼들이 가정, 공장 및 도시에 배치되어 운영되고 있으며, 이러한 디바이스 및 플랫폼들은 사람의 편의성 및 생산성 측면에서 도움을 줌과 동시에 서비스의 오류는 직접적으로 사람의 안전에 영향을 줄 수 있다. 따라서, 상호운용성 및 안정적인 서비스를 지원하기 위해 디바이스나 플랫폼들은 실제 배치되기 전에 사물인터넷 표준에 대한 적합성 및 상호운용성 테스트 등을 통하여 상호운용성 및 안정성을 확보해야 한다. 그러나, 수동으로 이루어지는 사물인터넷 애플리케이션 표준 적합성 테스팅 방법은 사람의 개입으로 인해 테스팅 과정에서 발생하는 오류의 증가 및 일부 테스팅 절차를 직접 수행해 주어야 하므로 테스팅 시간이 오래 걸리는 문제점이 있다.

위의 내용을 중심으로 본 논문에서는 사물인터넷 상호운용성 및 oneM2M 사물인터넷 표준 적합성 테스팅 수행에서 발생한 문제점을 해결하는 방안에 대해 기술하였다. 첫째, 상호운용성 문제를 해결하기 위하여 세 가지 상호운용성 모델들을 (Interworking Models, IWMs) 제시하였으며, 각 모델마다의 특징뿐만 아니라 장단점을 기술하여 조건에 맞게 사물인터넷 플랫폼 연계 방안을 선택할 수 있는 가이드를 제시하였다. 그리고 실질적인 예를 위하여, 현재 사물인터넷 기술이 활발하게 연구 중인 스마트 시티 분야와 결합하여 해당 모델들의 주요 기술을 설명하였다. 사물인터넷 적합성 테스팅 관련하여 트리거링 (Triggering)

메시지 기반의 자동화된 사물인터넷 표준 테스팅 방법 및 절차에 대한 연구를 하였으며, 개발된 방법 및 절차를 통해 사람의 개입을 최대한으로 줄여 테스팅 과정에서 발생할 수 있는 오류를 줄이고 시간을 단축할 수 있음을 제시하였다.

따라서 위와 같은 연구를 수행하여 현재 사물인터넷 표준의 다양성으로 인한 상호운용성 문제를 해결하여 사물인터넷 시스템들 간의 연동을 효과적으로 제공할 수 있을 뿐만 아니라 자동화된 테스팅 절차를 통해 사물인터넷 표준화 기관의 테스팅 및 인증 프로세스에 걸리는 시간을 단축시켜 사물인터넷 표준의 빠른 보급에 기여할 것으로 생각된다.

주요어: 사물인터넷, 상호운용성, 사물인터넷 적합성 테스팅, oneM2M

감사의 글

학위를 마무리하는 지금, 그동안 연구하였던 결과물 그리고 학회 및 회의에 참석하여 작성하였던 보고서 및 사진들을 정리하다 보니 많은 분들을 만나고 도움을 받았다는 것을 알 수 있었습니다. 먼저 모든 분들께 감사의 인사를 드리고 싶습니다.

연구실에서의 시간도 벌써 5년의 시간이 흘렀습니다. 결코 짧은 시간이 아니라고 생각됩니다. 긴 시간 동안 연구를 어떻게 진행해야 하는지 논문을 어떻게 써야 하는지 하나도 모르는 학생들 때문에 많은 고생을 하신 송재승 지도교수님께 감사의 말씀을 드립니다. 연구실 초기 학생으로서 선배가 없는 구조상 송재승 교수님은 저의 지도교수님이기도 하시지만 진로나 학업 관련해서도 도움을 주시는 선배이시기도 하였습니다. 진심으로 감사드립니다. 그리고 KETI 에서의 연구를 하였던 시간은 연구자로써 가장 많이 성장할 수 있던 시간이었다고 생각이 됩니다. 김재호 센터장님, 안일엽 팀장님을 비롯한 선배 연구원분들의 밤낮없이 열정적으로 연구하는 모습은 저에게 많은 가르침을 주셨습니다. 바쁘신 와중에도 학위 심사를 위하여 심사위원으로 참여해 주신 김재호 센터장님, 서정욱 교수님, 최효근 프로님, 박기웅 교수님께도 감사의 말씀을 드립니다. 또한, 저와 같이 학위를 진행하였던 세종대학교 SESLab 동료들에게도 감사의 말씀을 드리며, 모두 다 잘 되었으면 좋겠습니다. 마지막으로 항상 제가 잘 되기를 걱정해 주시는 할머니, 부모님, 동생 나의 가족에게 감사의 말씀을 드립니다. 가족들의 응원이 없었더라면 학위를 마칠 수 없었을 것 같습니다.

이제 학생의 신분을 벗어나 좀 더 넓은 세상으로 한 발짝 나아가려 합니다. 바로 내일, 그리고 먼 미래에 어떠한 일을 하고 있을지 알 수 없으나 저를 이끌어 주신 분들 그리고 동료들의 감사함을 마음에 새기고 나아가겠습니다. 감사합니다.