The Design Once, Provide Anywhere Concept for the Internet of Things Service Implementation

Mohd Anuaruddin Ahmadon[1*] and Shingo Yamaguchi[1]

[1]Graduate School of Sciences and Technology for Innovation,
Yamaguchi University, 2-16-1 Tokiwadai, Ube, 755-8611, JAPAN

*Corresponding Author

Abstract:
In today's IoT landscape, most platforms primarily focus on connecting devices through application-centric strategies. This approach entails using separate protocols for device intercommunication and accommodating devices with diverse capabilities for various situations. However, the current methodology of designing and implementing IoT services proves unreliable due to device heterogeneity and environmental variations. This approach incurs significant time, financial, and developmental costs for service deployment. To address these limitations, a novel platform for deploying IoT services that embrace the 'design once, provide anywhere' concept is introduced. By treating logical devices and environmental elements as separate entities while unifying them under a single adaptable platform, the proposed work aims to streamline the IoT deployment process.

Keywords:
Internet of Things · Service Design · Rapid Deployment · Elgar Platform

1. Introduction

Industry 4.0 [1], Society 5.0 [2], and Digital Transformation [3] are all related concepts, and the IoT (IoT) is crucial to their realization. The Internet of Things (IoT) is undeniably more advanced, universal, and fully mature than it was even a decade ago. Physical items, or "*things*," that may be connected to one another is not simply a novel idea; it has the potential to revolutionize the way we conduct business. Concerns have been raised about our readiness to implement IoT solutions. We need to shift our focus from "*What can I do with IoT*?" to "*What do I want IoT to do to benefit my organization*?" The IoT is not just "*things*" communicating with one another; it's also about what we can do with that information. This means that attention to IoT services is essential, alongside the current devices, protocols, and platforms. To realize these objectives, a system developer must prioritize research and development in the area of the Internet of Things (IoT). The Industry 4.0 also known as Fourth Industrial Revolution, is centered on enhancing corporate intelligence and automation, with a significant dependence on IoT. IoT refers to a network of physical things that are interconnected digitally, facilitating communication and the exchange of data. A wide range of devices, including telephones, household appliances, automobiles, and buildings, may be classified as smart gadgets. The impact of Industry 4.0 on the manufacturing process is far-reaching. This technology is used to enhance operational efficiency, enhance the accuracy of demand prediction, eradicate isolated data repositories, implement proactive maintenance practices, provide virtual training to personnel, and offer additional functionalities.

*Corresponding author: anuar@yamaguchi-u.ac.jp

Industry 4.0 includes connected vehicles, intelligent home appliances, connected surveillance systems, intelligent farming, automated manufacturing equipment, intelligent robotics, smart warehouse, and using IoT to improve manufacturing efficiency and implement monitoring and tracking mechanisms.

There have been a number of different measures launched in order to make it possible for the development and implementation of IoT technologies all over the world. Several different movements have been collaborating with one another to further the convergence of standards and communication protocols for the IoT (IoT). Examples of these include the European Telecommunications Standards Institute (ETSI)[4], the China Communications Standards Association (CCSA)[5] , the Telecommunications Industry Association (TIA)[6], the Association of Radio Industries and Businesses (ARIB)[7], the Alliance for Telecommunications Industry Solutions (ATIS)[8], the Telecommunications Standards Development Society, India (TSDSI)[9], the Telecommunications Technology Association (TTA)[10], and Japan's Telecommunication Technology Committee[11]. Through its community-driven Eclipse IoT group, the Eclipse Foundation promotes projects relating to the IoT (IoT), including gateways and ontologies. For example, through its Eclipse IoT [12] open-source community, the Eclipse Foundation[13] supports IoT development initiatives. Eclipse IoT offers a platform for creativity and cooperation in the creation of IoT technologies. With more than 170 contributors and 8.2 million lines of code, it supports over 45 projects. These initiatives support the development of gateways ("Smart Objects"), cloud backends, IoT devices, and other things. Eclipse 4diac, which concentrates on IEC 61499's[14] further development for use in distributed industrial process measurement and control systems, Eclipse Agail, a language-neutral, modular software and hardware gateway framework for the Internet of Things; and Eclipse Californium, an open-source implementation of the Constrained Application Protocol (CoAP) [15], are a few examples of Eclipse IoT projects. The collaboration between these different movements to further the convergence of standards and communication protocols for the IoT is crucial for its success. The Eclipse Foundation’s community-driven Eclipse IoT group is a great example of how creativity and cooperation can drive the development of IoT technologies.

The Internet Engineering Task Force (IETF) [16] oversees the development of IoT protocols across several IoT-related working groups and promotes IETF standards. In the United States of America, the Industrial Internet Consortium advocates the advancement of Industrial IoT (IIoT), while the IETF is responsible for promoting IETF standards. IoT-European Platforms Initiative (IoT-EPI)[17] is in charge of directing all of the research and innovation initiatives that are receiving funding from the EU-H2020 program in Europe . IoT Connectivity Alliance (ICA) [18] plays a critical role in supporting open, interconnected, and secure ecosystems for China's and the global IoT industry. In Asia, IoT Acceleration Consortium promotes the development of IoT projects to businesses and society in Japan, while IoT Acceleration Consortium promotes the development of IoT projects to businesses and society in China. A global non-profit organization such as the Organization for the Advancement of Structured Information Standards (OASIS) also had a role in the creation and implementation of open standards for IoT protocol specifications. The Internet Engineering Task Force (IETF) creates voluntary standards that are frequently adopted by network operators, Internet users, and equipment makers. As a result, the IETF contributes to the trajectory of the growth of the Internet. Constant work is being put into developing new standards as well as improving existing ones, putting them into effect, and rolling them out. The Internet Engineering Task Force's primary responsibility is to generate high-quality, up-to-date technical publications that provide an explanation of these voluntary standards. Working groups within the Internet Engineering Task Force (IETF) are the principal vehicle for the creation of IETF specifications and guidelines.

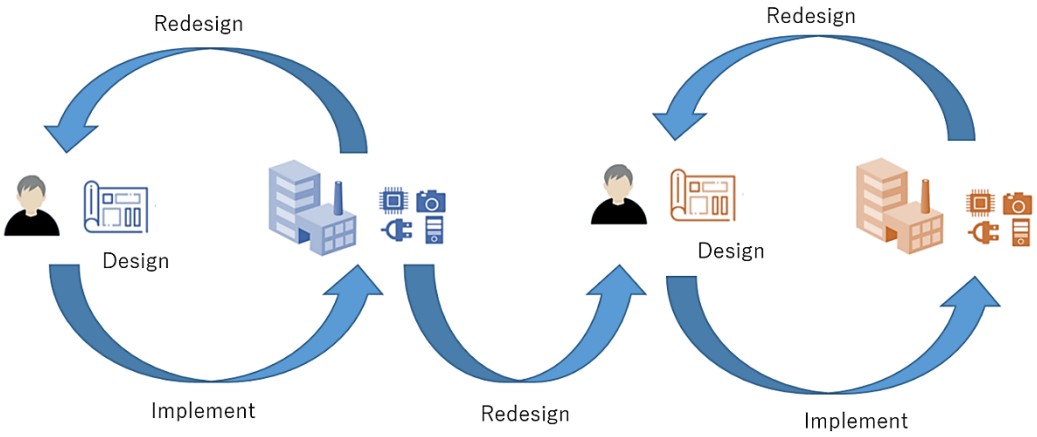


Figure 1: Repetitive problem in designing and implementing IoT services on different environment

In this chapter, a framework for the deployment of IoT services that incorporates the idea of "Design Once, Provide Anywhere is introduced." Within the context of this idea, the proposed work partitions logical devices from their surrounding environment and then connect those devices using a single and universal service architecture. In order to put the proposed notion into practice with regard to IoT services, the proposed work suggests using a platform that will be known as the Elgar Platform. Figure 1 provides an illustration of two distinct settings, both of which make use of the same IoT (IoT) service (a smart factory). In the conventional method, the designer is required to redo the design and implementation process for each environment due to the fact that different environments have varying elements and devices have varying capabilities. Our strategy eliminates the need to build and put into action the identical IoT service many times. Then, the efficacy by doing a case study on the implementation of the same IoT service design for a new setting utilizing a unique collection of devices is demonstrated.

2. Service Oriented Architecture and Design Once, Provide Anywhere Concept

An approach known as "Service-Oriented Architecture," or SOA for short, is described by the phrase "Service-Oriented Architecture." This methodology enables independent components to communicate with one another and scale in a flexible manner. SOA is a strategy that explains how to make software components reusable and interoperable by utilizing service interfaces. Services conform to defined interfaces and follow a certain architectural design in order to make it easier for them to be quickly included into new applications. This is done to expedite the process. Because this takes care of such errands for the application developer, they no longer need to know how to link or provide interoperability with pre-existing functions. This takes care of those processes for them. In the past, the developer may have either reproduced or modify previously implemented functionality.

Each individual service that constructed based on SOA is equipped with the coding and data necessary to carry out a separate and self-contained business process. The service interfaces are what offer the loose coupling, which means that they may be called with minimum or no knowledge of the underlying implementation of the service. This allows for more flexibility and adaptability. This contributes to the reduction of the dependencies that now exist between the various programs. A wide array of computer systems and programming languages are able to connect with one another because to this design. The cornerstone of SOA is loose coupling, often known as decoupling. The structure of SOA is based on decreasing the connectivity between application components, hence loose coupling is

also referred to as decoupling. The process of designing and deploying new apps and services may now take advantage of enhanced flexibility and scalability as a direct result of this development.

SOA can be understood as an architectural framework that incorporates various fundamental characteristics, such as loose coupling, reusability, seamless integration, responsiveness, and overall effectiveness. Consequently, numerous design methodologies and patterns have adopted SOA as a basic architectural concept. The methodology places emphasis on the dissection of the procedure into its fundamental components, embracing a wide range of factors including services and activities. A notion has been created for a model that integrates loose coupling by utilizing a Data Petri net architecture, in which logical devices are solely defined. In addition to the aforementioned process model, the proposed work additionally delineate the constraints associated with the characteristics of the appropriate devices. After the ontology tree has been constructed, it can be employed to determine the compatibility between the logical devices and the physical devices. Therefore, it is well recognized that SOA is pertinent in the realm of IoT system designs. The adoption of SOA can provide benefits to IoT systems. This is because SOA offers an architectural framework that enables efficient intercommunication between various components at a lower level. Moreover, SOA facilitates loose coupling, hence facilitating scalability within IoT systems. In order to fulfill the goal of "Design Once, Provide Anywhere," our methodology additionally facilitates the integration of various physical devices into the system design, without explicitly defining any particular device. The implementation of this methodology enables the effective administration of the meticulous implementation of service design through the utilization of various tools in varying circumstances. One strategy for attaining the goal of "Design Once, Provide Anywhere" in accordance with the principles of SOA entails the utilization of suitable tangible devices. Nevertheless, the primary emphasis in developing services using SOA is not on ensuring compatibility. Therefore there is a need for a new concept.

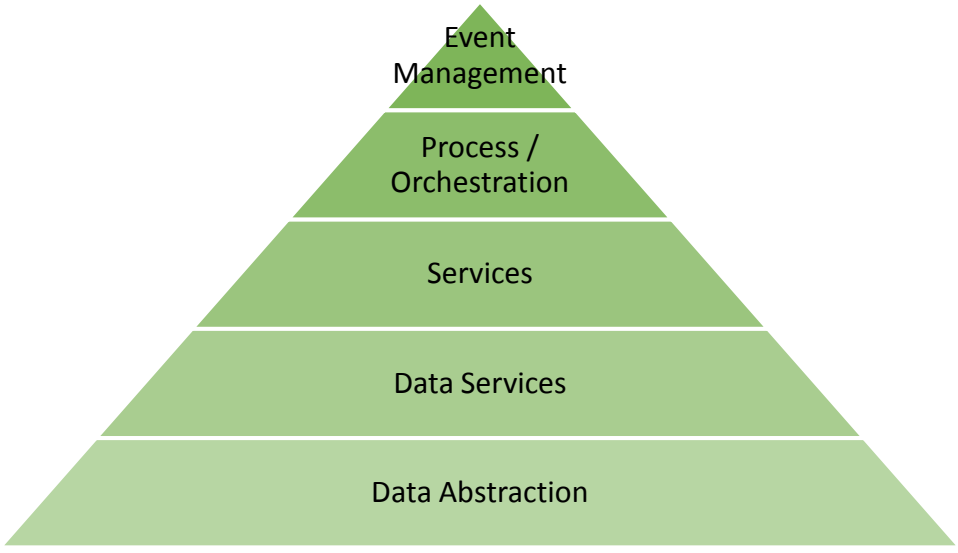


Figure 2: Service Oriented Architecture of Design Once, Provide Anywhere

The positioning of our proposal inside the several layers of the Service-Oriented Architecture (SOA) is shown in Figure 2. The process layer and the orchestration layer are the primary focal points of our investigation. The process model is utilized within the context of the data-control flow model in order to show the responsibilities and characteristics that are held by logical and physical devices. The purpose of the process model is to provide a graphical representation of the service design. It is feasible to link a logical device with any physically compatible device so long as the two devices in question adhere to

the constraint between device types that is provided by an ontology model. If this condition is met, the implementation will be successful.

3. IoT Platforms and the Real Truth

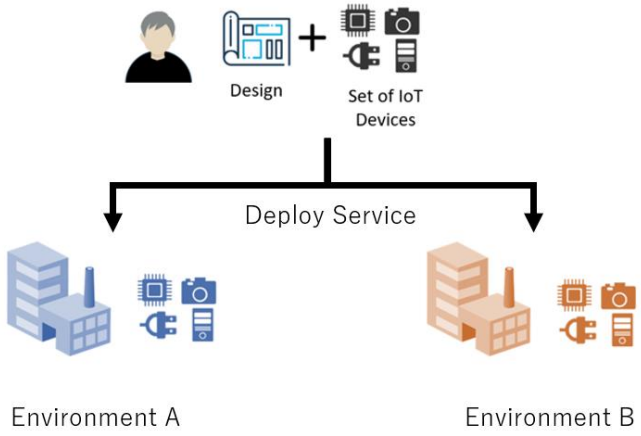


Figure 3: The implementation of the service may be achieved by the utilization of a single service design, along with the definition of the necessary equipment

IoT technology advanced by one step concurrently with the introduction of new protocols and standards. There has been a proliferation of Internet of Things platforms released as a means of hastening the development of Internet of Things infrastructures and services, as in Figure 3. Message Query Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) are two examples of well-known protocols that are used by some of the platforms, which may be cloud-based or run locally on the company's premises. Table 1 lists several IoT platforms, their companies, objectives, advantages, and disadvantages. These platforms include Curiosity IoT by Sprint, Jasper by Cisco, IoT Accelerator by Ericsson, Pelion IoT Platform by ARM, ThingWorx 8 IoT Platform by PTC Inc., Azure IoT Suite by Microsoft Corporation, IBM Watson IoT Platform by IBM Corporation, and Thingspeak IoT Platform by MathWorks Inc. Each platform has its own unique objective and offers various advantages such as complete control over device profiles and configurations, real-time visibility and control over connected devices, flexible global connectivity with multi-network access, and easy device registry with rich integration with other platforms. However, they also have their own disadvantages such as being highly dependent on specific systems or requiring dedicated administration and technical support.

Table 1: The IoT platforms that are widely used

Platform	Company	Objective	Advantages	Disadvantages
Curiosity IoT	Sprint	To enable connectivity of power efficient Wide Area Network (WAN) to its IoT-specific, virtual, distributed, and specialized network and IoT operating system.	One point of contact for full control over device setups and settings	Highly dependent on Curiosity Core and Curiosity OS. Non-transferrable.
Jasper	Cisco	Provides businesses with a cloud-based software platform to help them start, run, and generate revenues from IoT services on a global scale.	Offers real-time visibility and control over connected devices automates business processes	Requires dedicated administration and technical support. Transferring

Platform	Company	Objective	Advantages	Disadvantages
			and delivers new services faster	IoT services requires a support team.
IoT Accelerator	Ericsson	To make a specialized IoT connectivity platform that lets partners grow their services around the world.	Enhances interoperability and provides the IoT Accelerator Community with a comprehensive Internet of Things network	Dedicated to cellular IoT exclusive for large industries or mass production.
Pelion IoT Platform	ARM	To allow IoT devices safe cellular access around the world.	Offers flexible global connectivity with multi-network access secure device management and edge processing	Dedicated to cellular IoT. Data plan highly dependable on Pelion.
ThingWorx 8 IoT Platform	PTC Inc.	To facilitate simple networking for equipment used in industrial organizations.	The fundamental functions include simple connecting with sensors and RFIDs and remote connectivity after setup with built-in machine learning.	Difficult to utilize with C# custom programs and doesn't provide capabilities for scaling and clustering.
Azure IoT Suite	Microsoft Corporation	To offer a variety of services for developing IoT solutions that improve both profitability and efficiency through the use of pre-built connected solutions	Dashboards, integration with Oracle WebSphere and SAP Salesforce, and real-time data streaming are all included in this straightforward device registry solution.	Provides support only for software and cloud systems. Not much focus on hardware compatibility.
IBM Watson IoT Platform	IBM Corporation	To provide an all-encompassing platform for IoT system that simplifies the process of connecting, storing, and managing the data produced by IoT devices.	Good for scaling business. Google Cloud's per-minute pricing is less than others. Makes things simple and quick.	The platform will be deprecated and withdrawn on December 2023.
Thingspeak IoT Platform	MathWorks Inc.	To provide an open-source IoT's API that can save device data and retrieve it via the web protocol over the local network.	Provides real-time data collecting and display from internet-connected sensors and devices. Sends data from devices to	The customization requires support from the developer team since it is an open-

Platform	Company	Objective	Advantages	Disadvantages
			ThingSpeak, visualizes live data, and sends warnings.	source platform.

The IoT platforms shown in Table 1 provide numerous services and solutions for the Internet of Things. Sprint's Curiosity IoT provides low-power wide area network connectivity to its IoT-specific network and operating system. Jasper by Cisco is a cloud-based platform that helps organizations build, manage, and monetize IoT services globally. Ericsson's IoT Accelerator technology helps partners grow internationally. Pelion IoT Platform by ARM provides secure worldwide cellular connectivity for IoT devices. PTC Inc.'s ThingWorx 8 IoT Platform simplifies industrial device communication. MS Azure IoT Suite provides several services to construct IoT solutions that boost revenue and productivity with pre-built linked solutions. Watson IoT Platform by IBM Corporation provides an end-to-end platform for connecting, storing, and managing IoT data. The open-source Thingspeak IoT Platform from MathWorks Inc. stores and retrieves data from objects via HTTP over the Internet or a Local Area Network.

Table 1 outlines several IoT platforms, each of which comes with its own set of benefits and drawbacks. Curiosity IoT by Sprint provides users with full control over device profiles and settings via a centralized point of contact. Despite this, the platform is primarily reliant on Curiosity Core and Curiosity OS and cannot be transferred to another device. However, in order to provide real-time visibility and control over connected devices, automate business processes, and speed up the delivery of new services, Jasper by Cisco needs dedicated administration and technical support. Additionally, in order to transfer IoT services, a support staff is required. IoT Accelerator by Ericsson enhances interoperability and brings a solid IoT network to the IoT Accelerator Community; nevertheless, it is specialized to cellular IoT and is thus only available to major enterprises or companies that produce in huge quantities. Pelion IoT Platform by ARM provides flexible worldwide connection with multi-network access, secure device management, and edge processing; nevertheless, it is devoted to cellular IoT, and its data plan is largely reliant on Pelion. Pelion IoT Platform also enables secure device management. PTC Inc.'s ThingWorx 8 IoT Platform includes fundamental capabilities such as simple connectivity with electronic devices such as sensors and RFIDs, the ability to work remotely once setup is finished, and pre-built integration with machine learning; however, it is difficult to use with C#-specific programs and does not offer capabilities for scaling or clustering. The Azure IoT Suite offered by Microsoft Corporation has an easy-to-use device registry, comprehensive connectivity with dashboard provided by SAP's Salesforce, and Oracle's WebSphere, and easy integration to obtain real-time streaming; nevertheless, it only offers support for software and cloud-based systems and places little emphasis on the hardware compatibility of connected devices. The IBM Watson IoT Platform is being discontinued and will no longer be available after December 2023. This platform was developed by IBM Corporation and lets you expand your company by basing pricing on Google Cloud on a per-minute basis, making it more affordable than competing platforms while also putting an emphasis on making things simple and quick. Thingspeak IoT Platform by MathWorks Inc. offers real-time data collection and visualization from sensors or instruments that are connected to the internet. It also enables you to send data to ThingSpeak from your devices, create instant visualizations of live data, and send alerts. However, it is secured with the MQTT protocol, and customization requires support from the developer team because it is an open-source platform.

Based on the information provided in Table 1, it appears that most IoT platforms are dependent on providers or developers. This can make it difficult to transfer an IoT service to another environment once it has been deployed. As a result, there is a need for a consensus platform that involves providers, device manufacturers, and service designers. However, the most important focus should be on the end-user,

as they are the ultimate beneficiary of the IoT service. A platform that facilitates the deployment of IoT services across different environments and makes it easier for end-users to implement and manage these services would be beneficial in addressing these challenges.

4. Device Implementation Problem in IoT Environments

This section will define service design, logical objects, and the challenge that arises while attempting to implement them. The Internet of Things service design can be modeled using the IoT service design language. IoT logical objects is be utilized as the component that will be used for the implementation. For illustration, let us imagine we have a set of devices D that are up for consideration to implement the service specification S. As a straightforward illustration, let us take a look at a single piece of equipment like a digital air thermometer. Let us imagine we have three different devices to choose from as potential candidates for the implementation. The names of the instruments are as follows: Analog Air Thermometer t_1, NTC Thermometer from *Company X* t_2, and NTC Thermometer from *Company Y* t_3. From here we can give the question, "Are the t_1 , t_2, and t_3 devices implementable in specification S? "

In order to determine whether or not the devices can be implemented, it is necessary for us to examine the different types of devices as well as the functions they perform. It is clear from looking at the names of the devices that there is no *Temperature Sensor*. If the instruments do not fall into the category of digital air thermometers, we are unable to utilize them. Furthermore, if we specify a given device, the function must be precisely the same. However, in practice, it is difficult to locate devices that fit the design. This is the case unless the service designer and the device maker are part of the same organization or have some sort of agreement regarding the deployment of the service. With the help of the *Sprinkler* d_4, the service will attempt to regulate the daily amount of irrigation. The amount of water used for irrigation is regulated by monitoring the amount of precipitation captured by the *Rain Gauge* d_2. In addition, the temperature and humidity of the farm are being monitored at the same time with the help of the *Temperature Sensor* d_1 and the *Humidity Sensor* d_3 in the control flow. The information regarding the temperature, humidity, and rainfall is being supplied to the *Cloud Process* d_5. After that, *Cloud Process*, also known as a scheduler, will compute the amount of irrigation necessary and then transmit that information to the sprinkler. The data that is being watched are also saved in the cloud.

5. Concept of Design Once, Provide Anywhere

The concept of "Design Once, Provide Anywhere" [19] aims to simplify the deployment of IoT services by allowing service designs to be easily obtained from the internet and used with various devices. However, this strategy is fraught with difficulties because of the wide range of devices that must be employed and the requirement that they all be interoperable with one another. The Elgar Platform provides a solution to these problems by making it easier to rapidly create and deploy Internet of Things services that are flexible enough to be reused and modified. For instance, in order for smart farms to function properly in their respective surroundings, they require comparable monitoring and control services. These services connect temperature sensors, irrigation actuators, and control systems. The Elgar Platform accelerates this procedure and decreases the amount of effort necessary to assure compatibility with sensors and actuators. Traditional techniques require the service deployment process to be repeated for each farm. For instance, to establish the farm in a distinct area that has an excessively dry or excessively moist climate. This requires a unique set of sensors and actuators that can adapt to this environment. There are numerous ways to implement "Design Once, Provide Anywhere" in IoT services. Elgar Platform enables users to establish a service design by specifying control flow and logical devices, which can be implemented anywhere by adapting to varied contexts and physical devices.

Ensuring physical device compatibility and relaxing rigorous implementation to implement a group of suitable devices instead of specifying specific devices in the service design can achieve this. Another method is to utilize standardized communication protocols and data formats to connect devices from different manufacturers. Developers can spend less time ensuring device interoperability and implementing IoT services across environments.

The Elgar Platform implements "Design Once, Provide Anywhere" for IoT services. It adapts to various environments and physical devices to let users construct service designs that can be implemented everywhere. This can simplify IoT service deployment and reduce developer work to assure device compatibility. At smart farms, similar temperature, irrigation, ventilation control, and monitoring services are needed in multiple environments, but conventional approaches require repeating the service deployment process. The Elgar Platform helps developers quickly design and deploy IoT services that can be reused and adapted from independent development, eliminating the need to configure the device and environment or ensure sensor and actuator compatibility.

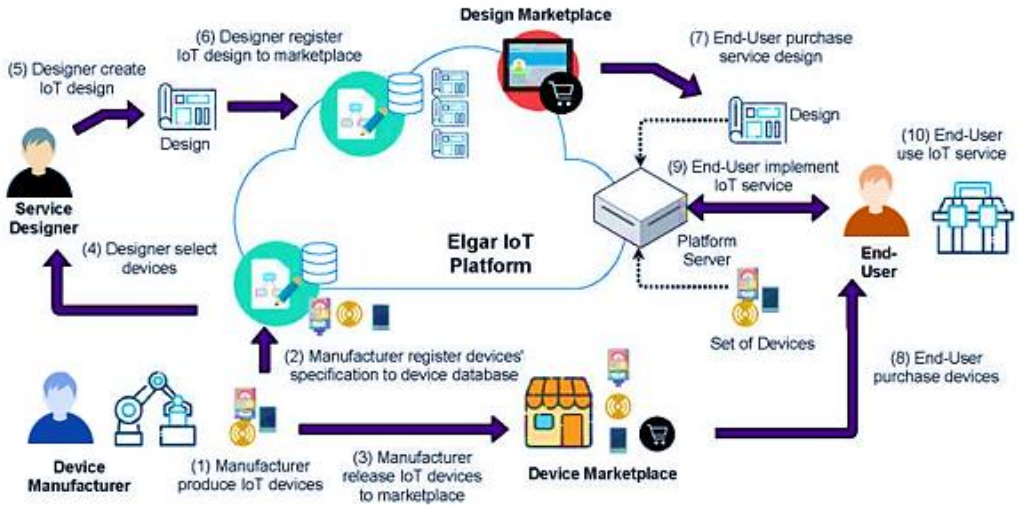


Figure 4: Our proposed concept of Elgar Platform based on Design Once, Provide Anywhere concept. The platform is used from three perspectives: Service Designer, Device Manufacturer and End-User

Basically, the concept consists of three actors; Service designer, device manufacturer and end-user. The proposed concept is illustrated in Figure 4. The Elgar Platform provides an environment for the development and deployment of IoT services that involves three main actors: device manufacturers, service designers, and end-users. Device manufacturers produce IoT devices and register their specifications in a device database before releasing them to the marketplace. Service designers select devices from the database and use them to create IoT service designs, which they then register in the marketplace. End-users can purchase these service designs and the necessary devices from the marketplace, implement the IoT service, and manage it on the platform server. This process aims to streamline the deployment of IoT services and reduce the effort required by developers to ensure compatibility between devices. The Elgar Platform provides an environment that facilitates the development and deployment of IoT services by streamlining the interactions between device manufacturers, service designers, and end-users. Device manufacturers produce IoT devices and register their specifications in a device database, making it easier for service designers to select the appropriate devices for their service designs. Service designers can then create IoT service designs using these devices and register them in the marketplace, where they can be purchased by end-users. End-users can also purchase the necessary devices from the device marketplace and implement the IoT service

using the platform server. This process aims to reduce the effort required to ensure compatibility between devices and make it easier to deploy IoT services across different environments. By providing a centralized platform for the development and deployment of IoT services, the Elgar Platform aims to facilitate the realization of the "Design Once, Provide Anywhere" concept.

In practice, the concept of "Design Once, Provide Anywhere" aims to simplify the deployment of IoT services by allowing service designs to be easily obtained from the internet and used with various devices without having to start from scratch. This can be achieved through the use of platforms like the Elgar Platform, which provides tools for device manufacturers, service designers, and end-users to facilitate the development and deployment of IoT services. Service designers can use the platform's DPN tool to create service designs using a control flow notation known as Data Petri Net (DPN) [19], and these designs can be registered in the marketplace for end-users to purchase. End-users can also purchase the necessary devices from the device marketplace and implement the IoT service using the platform server. The platform's goal is to reduce the amount of effort required by developers to ensure compatibility between devices and to make it easier to implement IoT services across a variety of environments. This will be accomplished by ensuring compatibility between hardware components and allowing for the relaxation of strict implementation, which will allow a set of compatible devices to be implemented rather than specifying specific devices in the service design.

5.1 Service Designer

In the Elgar Platform, service designers use the DPN tool to create service designs by entering environment parameters that set variables in the logical device. The DPN tool is stored using the GraphViz format and loaded into the platform. Device manufacturers provide a list of registered devices, from which service designers can choose to include in their designs. The device list is described using RDF files that include information such as the device name, resource type, quantity kind, unit, and measurement range. This information helps ensure compatibility between devices and facilitates the deployment of IoT services. Figure 5 illustrates the type of files on the platform.

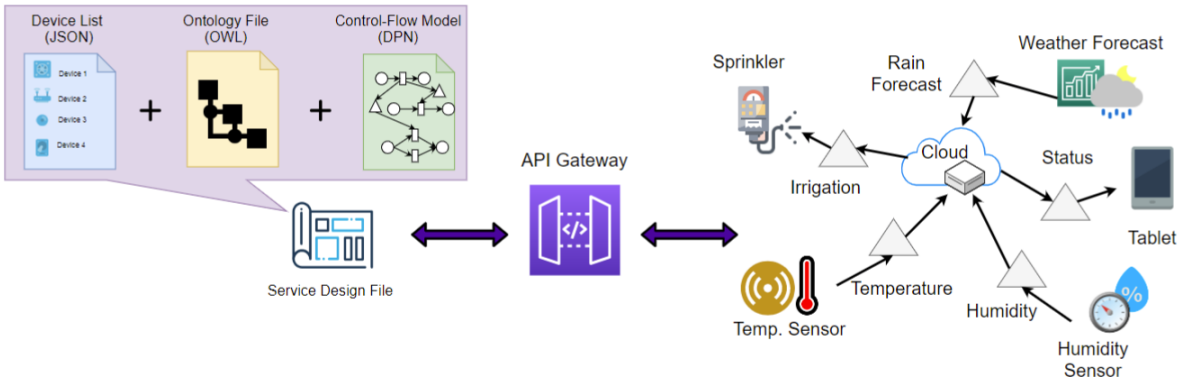
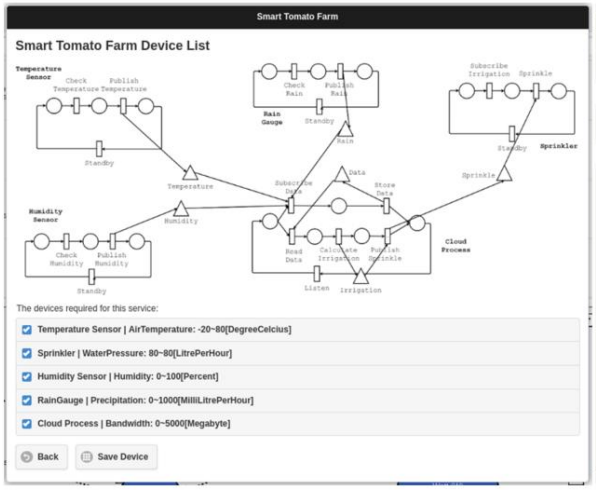


Figure 5: Overview of Elgar platform implementation

5.2 Manufacturer

The second tool provided by the Elgar Platform (Figure 6) is an ontology design tool that allows device manufacturers to create and reference ontology in an organized fashion. This ontology, known as the Elgar Ontology, is based on the W3C-registered IoT-Lite Ontology and has been expanded to include additional concepts such as *QuantityKind* and *Unit*. The ontology design tool includes a representation of the Elgar Ontology, which is organized as a tree with devices eligible for registration in particular categories. The lowest leaf of the tree consists of examples that depict devices with concrete

specifications, while the parent of the leaf classes is the superclass, which contains a description of the abstract specifications of different types of devices. When devices are selected from a subclass of a superclass, the implementation of the device is considered a strict implementation. In the case of a superclass, an increased number of devices can be accommodated within the class for implementation. The manufactured devices must be compatible with the Elgar application programming interface (API), which serves as a wrapper around protocols and separates them from the control flow. This allows manufacturers to use whichever protocols they see fit. The API includes fundamental classes and functions that can be invoked using a library, and variables such as device name, model, device type, quantity type, and unit must be defined by the manufacturer. Additionally, the manufacturer must specify the address of the broker (the central server of the platform), as well as the port number and time-to-live. The remainder of the code is used to carry out the basic operation of the device, such as invoking a function to carry out a specific instruction based on values obtained through message passing using a pre-defined protocol such as MQTT.




```
<?xml version="1.0" encoding="UTF-8"?>
<service>
    <info>
     <creator xml:lang="en
    ">M.Anuaruddin, T.Tsujimoto, N.Yamaguchi, S.Yamaguchi</creator>
     <date xml:lang="en">2018-12-11</date>
     <description xml:lang="en
    ">An IoT service for smart tomato farm.</description>
     <title xml:lang="en">Smart Tomato Farm Service</title>
     <version xml:lang="en">0.4</version>
    </info>
  <device name="Temperature Sensor">
      <type resource="TemperatureSensor" />
      <hasQuantityKind resource="AirTemperature" />
      <hasUnit resource="DegreeCelsius" />
      <hasValue min="-20" max="200"/>
  </device>
  <device name="Sprinkler">
      <type resource="IrrigationActuator" />
      <hasQuantityKind resource="WaterPressure" />
      <hasUnit resource="LitrePerHour" />
      <hasValue min="0" max="80"/>
  </device>
</service>
```

Figure 6: Overview of Elgar platform implementation

5.3 End-User

The third tool provided by the Elgar Platform is an end-user tool that allows end-users to purchase service designs and devices from the marketplace and implement IoT services using the platform. The platform provides a "boxed" solution that enables end-users to easily implement the desired IoT service by obtaining the required service design documents from the marketplace and connecting the logical devices they have acquired. End-users can customize the service by entering various environmental conditions. The end-user tool also allows users to register multiple service design documents using a platform designer display, where they can register a Resource Description File (RDF) containing an XML description of the completed design document. The registered design is then made available to the public.

6. Overcoming the Challenges of Implementing Design Once, Provide Anywhere

The interoperability of different technologies presents certain difficulties for the execution of the idea. If we are unable to find the device that was mentioned in the design, then we will not be able to satisfy the strict standards that are essential to carry out IoT services. Despite this, each of these three devices is a member of the same subclass of air devices. Therefore, we need to make it easier to implement IoT devices such that even the subclass of the devices stated in the design may be implemented for any sort of service design. Specifically, we need to manufacture it so that we can

implement it for any kind of design. We have introduced a vital property that may decide whether or not a device is implementable for the sake of simplifying the implementation process. This attribute can determine whether or not a device is implementable. In order to overcome these challenges, during the design process, the designer can evaluate the level of abstraction or strictness of the design by testing it on a set of devices and calculating the compatibility rate [19]. This rate is determined by the attributes of the devices and their level of subclass in the ontology tree. For a given physical device with a specific attribute for a task, the compatibility rate when assigned to a logical device with a constraint on that attribute can be calculated. The constraint must satisfy a true or false statement of satisfying a subset equation.

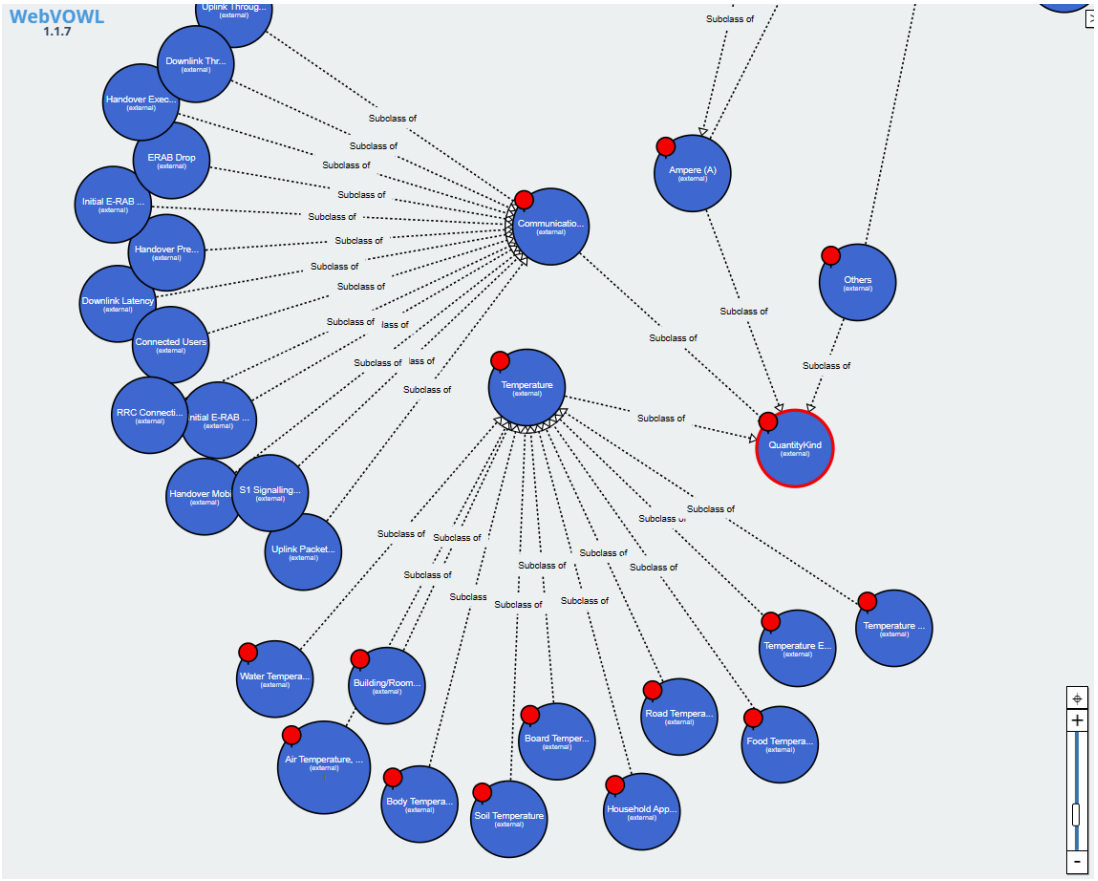


Figure 7: Visualization of Elgar ontology for the compatibility verification of devices and its specification

This allows the designer to evaluate the compatibility of the design with different sets of physical devices and adjust as needed to ensure that the design can be implemented across different environments. The compatibility rate for a given attribute of a physical device when assigned to a logical device with a constraint on that attribute can be calculated based on three types of constraints: (i) the constraint of subclass in the ontology tree, (ii) the constraint of ontology class relation, and (iii) the constraint of a value range. The first type of constraint, the constraint of subclass in the ontology tree, can be specified by setting a class for the attribute. This allows the designer to evaluate the compatibility of the design with different sets of physical devices based on their position in the ontology tree and their relationship to other classes and subclasses. The ontology is shown in Figure 7 and is available [20]. The illustration shows that one superclass of attributes may have multiple subclasses. For example, a *Temperature* class can possess subclass such as *Soil Temperature*, *Water Temperature* and *Air Temperature*. This classes are bounded to the physical device specification.

A service designer may determine whether or not a service design is compatible with a collection of candidate devices by assessing the percentage of devices that are compatible with the design. The compatibility rate is used to describe the amount of abstraction achieved by the service design. The designer has the ability to modify this level of abstraction by specifying the restrictions placed on the devices. When establishing the service design, there is a trade-off between the amount of strictness and the level of abstraction. The designer could find a balance between the two in order to develop a design that is either precise or abstract and can handle a greater number of devices. However, raising the degree of abstraction might result in a reduction in the accuracy of the device specification. This is one of the limitations that must be taken into account when deciding the most appropriate level of abstraction for a service design. The effectiveness of integrating ontology into the device application and service design process was evaluated in this study. During the assessment, the percentage of implementable devices that were accomplished via the use of traditional verification techniques will be presented. Then it will be compared with the implementation rate that was accomplished with the use of our suggested strategy. The results of the evaluation of the suggested concept are shown in Figure 8. A service design that utilized twenty-five actuators, eighty sensors, eighteen tag devices, and twenty different types of software and services was considered. First, the number of devices that can be utilized without applying to the proposed methodology will be compared with the proposed method. An ontology that consists of 120 types of actuators, 150 types of sensors, 60 tagging devices, and 145 application and service components is utilized in the evaluation process. This ontology takes into account a total of 475 different groups of technological implements. During the course of the analysis, a total of 35 controllers, 80 monitoring devices, 18 tagging devices, and 20 software programs or services were used. Therefore, the relative rate of implementation for actuators, sensors, tag devices, and software/services range anywhere from 28% to 69% when using the traditional methodologies that are defined by severe implementation protocols. The rate of implementation showed signs of progress, increasing to anywhere between 56% and 82%. between 56% to 82%.

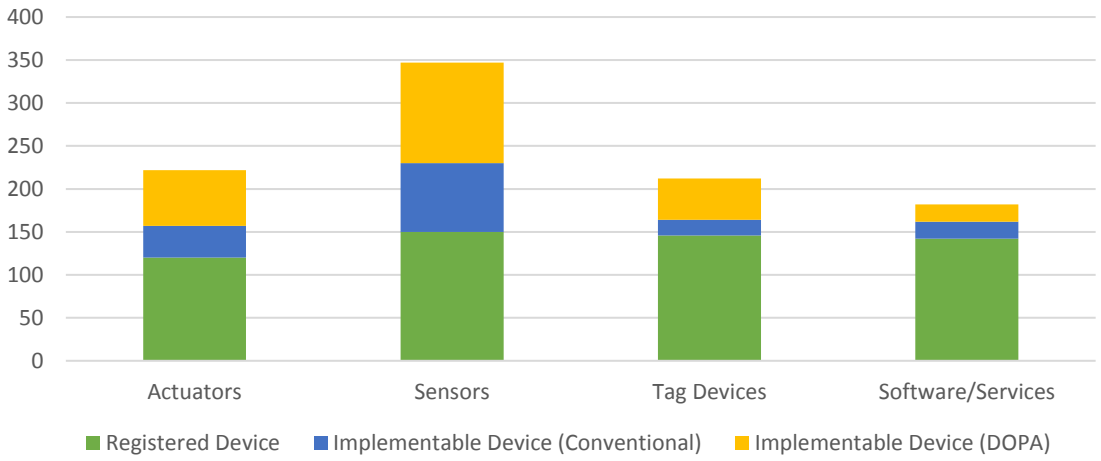


Figure 8: Comparison of implementable devices between the proposed concept and conventional concept

7. Conclusion

This chapter presents a framework for the deployment of IoT services based on the concept of "Design Once, Provide Anywhere." This approach involves separating logical devices from their surrounding environment and connecting them using a single, universal service architecture. An IoT platform embracing this concept called Elgar Platform is proposed. The platform allows for the deployment of the same IoT service design in different environments using different sets of devices,

eliminating the need to repeat the design and implementation process for each environment. This is illustrated with the example of a smart factory, where the same IoT service can be deployed in two different settings using the Elgar Platform.

The framework is based on the principles of Service-Oriented Architecture (SOA), which is a methodology that enables disparate components to connect and scale in a flexible manner. SOA describes a way to make software components reusable and interoperable through the use of service interfaces that adhere to standardized interfaces and follow specific architectural patterns. This allows for the quick incorporation of services into new applications without the need for developers to know how to connect or provide interoperability with existing functionalities. Services in an SOA include the code and data required to execute complete, discrete business operations, and are loosely coupled through their interfaces, reducing dependencies between different applications. The proposed framework applies these principles to IoT services by specifying logical devices using a Data Petri net and establishing constraints on the characteristics of compatible devices. The ontology tree is used to determine whether logical devices and physical devices are compatible, allowing for the deployment of IoT services in different environments using different sets of devices. This approach provides increased flexibility and scalability in the development and deployment of IoT services.

The Elgar Platform is an IoT service development and deployment platform that is based on the concept of "Design Once, Provide Anywhere." It enables users to create a service design by specifying the control flow and logical devices, and the service design can be implemented anywhere by adapting to different environments and physical devices. This sets it apart from other IoT platforms, which may not provide the same level of flexibility in deploying IoT services across different environments. The Elgar Platform aims to facilitate the rapid deployment of IoT services by ensuring compatibility between physical devices and allowing for the relaxation of strict implementation, where a set of compatible devices can be implemented instead of specifying specific devices in the service design. This can help reduce the effort required by developers to ensure compatibility between devices and make it easier to implement IoT services across different environments.

Acknowledgment

This research was fully supported by Interface Corporation, Japan.

References

[1] M. Ghobakhloo, M. Iranmanesh, A. Grybauskas, M. Vilkas, and M. Petraitė, "Industry 4.0, innovation, and sustainable development: A systematic review and a roadmap to sustainable innovation," Bus Strategy Environ, vol. 30, no. 8, pp. 4237–4257, Dec. 2021, doi: 10.1002/bse.2867.

[2] C. Narvaez Rojas, G. A. Alomia Peñafiel, D. F. Loaiza Buitrago, and C. A. Tavera Romero, "Society 5.0: A Japanese Concept for a Superintelligent Society," Sustainability, vol. 13, no. 12, p. 6567, Jun. 2021, doi: 10.3390/su13126567.

[3] S. Nadkarni and R. Prügl, "Digital transformation: a review, synthesis and opportunities for future research," Management Review Quarterly, vol. 71, no. 2, pp. 233–341, Apr. 2021, doi: 10.1007/s11301-020-00185-7.

[4] ETSI, "European Telecommunications Standards Institute (ETSI)." Oct. 2023.

[5] CCSA, "China Communications Standards Association (CCSA) ." Aug. 2023.

[6] TIA, "Telecommunications Industry Association (TIA)."

[7] ARIB, "Association of Radio Industries and Businesses (ARIB)."

[8] ATIS, "Alliance for Telecommunications Industry Solutions (ATIS)." Feb. 2020.

[9] TSDSI, "Telecommunications Standards Development Society, India (TSDSI)."

[10] TTA, "Telecommunications Technology Association (TTA)." Dec. 2022.

[11] TTC, "Japan's Telecommunication Technology Committee (TTC)."

[12] L. Romer, S. E. Jeroschewski, and J. Kristan, "Leveraging Eclipse IoT in the Arrowhead Framework," in NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, IEEE, Apr. 2020, pp. 1–6. doi: 10.1109/NOMS47738.2020.9110418.

[13] Eclipse IoT Community, "Eclipse IoT Community."

[14] IEC, "IEC 61499 Standard for distributed Automation."

[15] S. Suman, T. Perumal, N. Mustapha, R. Yaakob, M. A. Bin Ahmadon, and S. Yamaguchi, "CoAP-Based Lightweight Interoperability Semantic Sensor and Actuator Ontology for IoT Ecosystem," International Journal of Ambient Computing and Intelligence, vol. 12, no. 2, pp. 92–110, Apr. 2021, doi: 10.4018/IJACI.2021040106.

[16] M. Ohta, "IETF and Internet standards," IEEE Communications Magazine, vol. 36, no. 9, pp. 126–129, 1998, doi: 10.1109/35.714633.

[17] bIoTope project, "IoT European Platforms Initiative."

[18] B. Conde Gallego and J. Drexl, "IoT Connectivity Standards: How Adaptive is the Current SEP Regulatory Framework?," IIC - International Review of Intellectual Property and Competition Law, vol. 50, no. 1, pp. 135–156, Jan. 2019, doi: 10.1007/s40319-018-00774-w.

[19] M. A. Bin Ahmadon, S. Yamaguchi, A. K. Mahamad, and S. Saon, "Physical Device Compatibility Support for Implementation of IoT Services with Design Once, Provide Anywhere Concept," Information, vol. 12, no. 1, p. 30, Jan. 2021, doi: 10.3390/info12010030.

[20] Mohd Anuaruddin Bin Ahmadon, "Elgar Platform Ontology," 2017. Accessed: Oct. 23, 2023. [Online]. Available: https://github.com/anuaruddin/elgar