energies

MDPI

Article

Uni-Messe: Unified Rule-Based Message Delivery Service for Efficient Context-Aware Service Integration †

Takuya Nakata [1], Sinan Chen [1,*], Masahide Nakamura [1,2]

[1] Graduate School of System Informatics, Kobe University, 1-1 Rokkodai-cho, Nada, Kobe 657-8501, Japan; tnakata@ws.cs.kobe-u.ac.jp (T.N.); masa-n@cs.kobe-u.ac.jp (M.N.)
[2] RIKEN Center for Advanced Intelligence Project, 1-4-1 Nihonbashi, Chuo-ku, Tokyo 103-0027, Japan
* Correspondence: chensinan@ws.cs.kobe-u.ac.jp; Tel.: +81-78-803-6295

† This paper is an extended version of the conference paper: Takuya, N.; Sinan, C.; Masahide, N., Developing Event Routing Service to Support Context-Aware Service Integration. From the proceedings of the SNPD 2021, online, 26 November 2021.

Abstract: Rule-based systems, which are the typical technology used to realize context-aware services, have been independently implemented in various smart services. The challenges of these systems are the versatility of action, looseness, and the coding that is needed to describe the conditional branches. The purpose of this study was to support the realization of service coordination and smart services using context-aware technology by converting rule-based systems into services. In the proposed method, we designed and implemented the architecture of a new service: Unified Rule-Based Message Delivery Service (Uni-messe), which is an application-neutral rule management and evaluation service for rule-based systems. The core part of the Uni-messe proposal is the combination of a Pub/Sub and a rule-based system, and the proposal of a new event–condition–route (ECR) rule-based system. We applied Uni-messe to an audio information presentation system (ALPS) and indoor location sensing technology to construct concrete smart services, and then compared and evaluated the implementation to "if this then that" (IFTTT), which is a typical service coordination technology. Moreover, we analyzed the characteristics of other rule-based systems that have been serviced in previous studies and compared them to Uni-messe. This study shows that Uni-messe can provide services that simultaneously combine versatility, ease of conditional description, looseness, context independence, and user interface (UI), which cannot be achieved using conventional rule-based system services. By using Uni-messe, advanced heterogeneous distributed service coordination using rule-based systems and the construction of context-aware services can be performed easily.

Keywords: rule-based system; context-aware service; smart service integration; value-added service; heterogeneous distributed service

Citation: Nakata, T.; Chen, S.; Nakamura, M. Uni-Messe: Unified Rule-Based Message Delivery Service for Efficient Context-Aware Service Integration. *Energies* **2022**, *15*, 1729. https://doi.org/10.3390/en15051729

Academic Editor: Virginia Pilloni

Received: 29 December 2021
Accepted: 22 February 2022
Published: 25 February 2022

1. Introduction

From the continuous development of *smart homes* [1,2] and *smart cities* [3,4], the more *efficient* diversified *integration* of *value-added* services [5,6] has become a key issue. Value-added services are commonly used to link things in the physical world with computational resources in the cloud via a network [7]. For example, Assisted Living by Personalized Speaker (ALPS) [8] is an innovative service that creates added value by connecting commonplace objects, such as a speaker and a motion sensor, in the cloud to reduce the stress of caregivers by automatically notifying care receivers. Users of smart services often make trivial requests to render the service more convenient, such as linking a microphone to ALPS so that users can take notes verbally. However, integrating and extending smart services is a costly change from the developer's side [9]. By dynamically coordinating *heterogeneous distributed* services [10] according to various situations (i.e., *contexts*) within the physical world and cyberspace [11,12], it is possible to provide the most appropriate

service (i.e., *context-aware service*, such as [13,14]) for the specific place, time, and person. The efficient integration of heterogeneous distributed services is the key step in realizing context-aware services. A *rule-based system* is used to realize context-aware services [15,16]. It is an inference-type artificial intelligence (AI) [17,18] that executes services based on *event–condition–action (ECA)* [19] rules. Generally, many rule-based systems are often implemented and utilized independently within each smart service [8,20] ; however, the development of rule-based systems is costly to implement. In recent years, research on the conversion of rule-based systems into services and the deployment of commercial services has been conducted for developing smart services efficiently. Typical services include *RuCAS* [21] and *IFTTT* [22–24]; however, there is the problem that these service-ized rule-based systems are often inadequate in terms of quality [25]. For example, RuCAS is a service that allows ECA rules to register into a Web user interface (UI) [26] but it lacks versatility and looseness of action. IFTTT is a service that links to various commercial services with an excellent UI; nonetheless, it requires coding to describe the conditional branches due to the strong connectivity between services. Hence, it is necessary to build a service-oriented rule-based system of sufficient quality to support and improve the efficiency of developing a wide range of smart services.

The goal of this study was to propose a new service, *Unified Rule-Based Message Delivery Service (Uni-messe)*, which can support the coordination os context-aware services. Uni-messe is a service that provides a *rule-based* system as an *application-neutral* service for rule management and evaluation. The rule-based system in Uni-messe operates on the newly proposed **event–condition–route (ECR)** rules instead of the conventional ECA rules. Moreover, although pure HTTP requests are often used for communication between a rule-based system and external services, we adopted *Pub/Sub* [27,28] for communication between Uni-messe and external services.

The digest version of this paper has been published in an international conference paper [29]. The most significant changes made in this version include the extension of the *event model* (Section 3.3) and the *route model* (Section 3.5), the application of practical services (Section 5), and the comparative evaluation (Section 6). We believe that these additions qualify the study to constitute a new contribution as a journal paper. In Uni-messe, we succeeded in creating a rule-based system that combines six features that cannot be simultaneously realized by conventional rule-based systems: **implementation, generality, condition simplicity, loosely coupled-ness, context independency, and the availability of the UI**. This makes it extremely easy to realize context-aware services using rule-based systems. Meanwhile, it can also enable us to build advanced smart services by easily linking heterogeneous distributed services.

Based on the proposed method, we implemented Uni-messe concretely and converted it into a service. Moreover, we applied two smart services [8,30] to Uni-messe. Furthermore, we compared and evaluated the two smart services to those implemented using IFTTT. We also made a qualitative comparison between Uni-messe and the previous studies of other rule-based systems [8,21,22,31]. The remainder of this paper is organized as follows. Section 2 describes the details of smart service technology and related research as the research background. Section 3 proposes the architecture of Uni-messe. Section 4 describes the implementation and usage of Uni-messe. The application to practical service and evaluation are presented in Sections 5 and 6, respectively, followed by the conclusions in Section 7.

2. Preliminaries

2.1. Smart Service

A smart system [32] is a system that realizes value-added services by linking information in the physical world with computational resources in cyberspace through a network. Research into and the development of the *smart services* [33] running on smart systems has been very active and our laboratory has been working on various services, such as a mind monitoring service for the elderly that uses LINE application and ALPS [8], a speaker

service to help to prevent forgetfulness. One of the common features of these smart systems is that they dynamically coordinate heterogeneous distributed services to read and be aware of the context of a room or a person using sensors or services and then provide appropriate services according to the context (i.e., *context-aware services*). Many products using context-aware technology, such as IFTTT and Nature Remo [34], have been put to practical use and are becoming popular with the public. Technologies and concepts that are commonly used to realize smart services through the coordination of heterogeneous distributed services include microservice architecture [35,36], Pub/Sub, and rule-based systems. Since there are many common aspects in the implementation of each smart service, such as the use of the above technologies, efficient implementation can be achieved by using frameworks such as FIWARE, IFTTT, and RuCAS as the basis for implementation [37].

2.2. Technologies for Realizing Smart Services

Microservice architecture is an architectural paradigm that exposes system functions in small service units and loosely couples them to realize new services. Since smart services require very the complex coordination of various sensors and services, they come with issues with development cost and feasibility [38]. Hence, in order to improve the efficiency of service development, it is necessary to link services whilst maintaining their looseness [39]. Certain technologies, such as Pub/Sub and RESTful application programming interfaces (APIs) [40], are used to enhance this looseness. The *Publish/Subscribe messaging infrastructure (Pub/Sub)* is a middleware that provides asynchronous communication between services through an intermediary (i.e., a broker). In Pub/Sub, messages are exchanged through many topics, which serve as P.O. boxes for the messages. For each topic, the service can perform two operations: publish, which passes messages to the topic, and subscribe, which receives messages from the topic. The publishers and subscribers do not need to be aware of each other, and the timing of publishing and subscribing is asynchronous. In addition, a variety of registered services can publish and subscribe to events, allowing for the complex coordination of multiple services. Due to these properties of Pub/Sub, it is possible to perform the sophisticated coordination of services whilst maintaining *looseness*. In order to realize context-aware services, it is necessary to implement concrete service coordination logic on top of service coordination infrastructure, such as microservice architecture and Pub/Sub, to realize context-sensitive service coordination. A *rule-based system* is an intuitive approach to constructing service coordination logic. A rule-based system is an inference-type AI system that performs processing based on ECA rules for its events, conditions, and actions. ECA rules operate by evaluating an event with a condition and then firing an action if the conditional expression is satisfied. The conditional expression of the condition rule must be registered in the rule-based system in advance in order to evaluate the event, and the action to be fired when the conditional expression is satisfied must also be associated in advance. A rule-based system can be used intuitively to implement context-aware services because they can mediate between an event generated by a service and another service fired by an action.

2.3. Challenges of the Existing Smart Service Implementation Infrastructure

The implementation of smart services is a combination of these technologies; however, the implementation is similar for many systems. In traditional smart services, such as ALPS, the rule-based systems have been embedded and implemented within each smart service. In recent years, many frameworks have been produced that provide a combination of the technologies required for smart services, and these frameworks can be used to improve the efficiency of smart service development. RuCAS is a Web service for creating and managing context-aware services. FIWARE Perseo [41] is a multifunctional, general purpose rule-based system that is implemented as a module of the FIWARE architecture, the context management infrastructure. IFTTT is a widely used commercial service that allows a rich variety of service coordination to be easily performed with a Web UI; however, the conventional framework has limitations, such as the fixed number of service applications

that can be linked and the lack of generality in the contexts that can be handled. Rule-based systems in context-aware applications have different semantics of rules and different appropriate service invocation methods depending on the service [19,42,43]. In this study, we proposed a rule-based system for context-aware applications. Hence, we believed that we could improve the efficiency of smart service development by providing these rule-based systems as a service that performs neutral rule management and evaluation processing for applications.

3. Proposed Method

3.1. Purpose & Key Idea

The purpose of this study was to enhance the efficiency of smart service development by making the rule-based system, which ordinarily needs to be implemented separately in many smart services, available as a neutral service. In this study, we proposed a new service: *Unified Rule-based Message Delivery Service (Uni-messe)*, which supports context-aware service coordination. Uni-messe is an application-neutral service that performs the event evaluation process commonly performed by rule-based systems that are implemented by individual services. Specifically, it receives events from various applications, evaluates them, and delegates the processing to the appropriate application based on the evaluation results. In designing Uni-messe, we considered the architecture of a rule-based system that would be suitable for context-aware applications and identified the necessary processes. Our key idea was to extract the processes that are neutral to the application and convert them into services. The approach of this study was as follows:

- The architecture of Uni-messe (Section 3.2)
- Application-neutral ECR data model (Sections 3.3–3.5)
- Event evaluation and routing with condition rules (Section 3.6)

3.2. The Architecture of Uni-Messe

Uni-messe was constructed as a rule evaluation and routing service based on the Pub/Sub message infrastructure. Depending on the events and conditions generated by each application, events are forwarded to the appropriate application. The design and scope of the services in Uni-messe and its surrounding applications are as follows. Uni-messe provides the functions of rule management, event evaluation (expression evaluation), and processing instructions (routing). Each routing rule is defined as an *event–condition–route (ECR) rule*. On the other hand, individual applications that are linked to Uni-messe define, make sense of, interpret, and perform the processing (actions) for each rule. Each application uses the rule management function of Uni-messe to create, read, update, and delete (CRUD) the rules in advance. In other words, the responsibilities of Uni-messe and the individual applications are clearly different. The responsibility of Uni-messe is to instruct the service as to which process to perform according to the results of the event evaluation (routing), not to execute the process. The execution of the process is the responsibility of the individual application. In general, ECA rules are used in rule-based systems, which use actions instead of routes; however, Uni-messe introduces the new concept of routing. This is because Uni-messe is not responsible for executing the service processing (actions), but rather for instructing the service to execute the processing (routing). Figure 1 shows the architecture of the relationship between Uni-messe, Pub/Sub, and the external applications, which include: App A, a human detection sensor; App B, a thermometer; App C, an IoT speaker; App D, a smart door. The flow of operation of the Uni-messe architecture is as follows:

- 1. *Event Source*: Each application recognizes the context of the room and people, and then passes events to Uni-messe via Pub/Sub;
- 2. *Event Routing*: Uni-messe evaluates the event using ECR rules, selects an appropriate route to hold the topic on Pub/Sub as a result of the evaluation, and passes the event on to all selected topics;

- 3. *Event Destination*: The application receives the events at arbitrary times via Pub/Sub, and then freely interprets the events and executes the arbitrary processes.

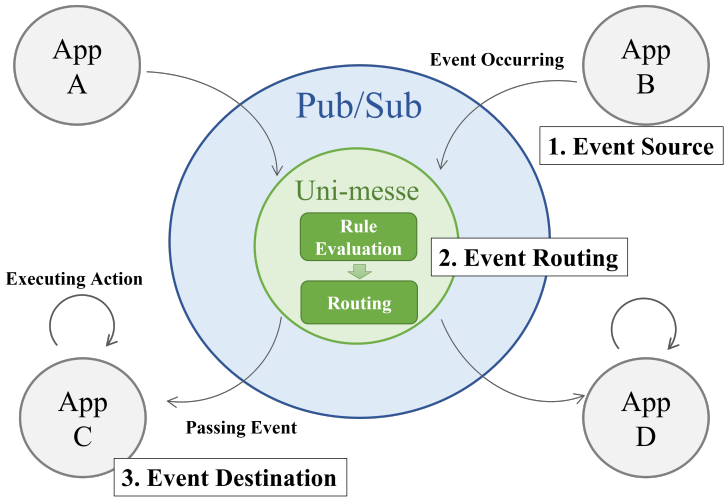

Figure 1. The Uni-messe architecture.

3.3. Event Model

An *event* is a structure that represents a context with the seven elements of *6W1H (who, whom, when, where, what, why, and how)*. When each event is detected by a service sensor, the service formats the information obtained at the time of detection into the seven 6W1H elements and sends them to Uni-messe. The events that Uni-messe evaluates, routes, and passes on to the services are also expressed according to the 6W1H format. Each element of 6W1H is described below. *Who* expresses the name or ID of the person or device that caused the event. *Whom* represents the name or ID of the person or device that caused the event or was affected by the event. *Where* expresses the location where the event occurred. *When* expresses the time that the event occurred. The string representing the time conforms to ISO8601. *What* expresses various attributes of a detailed event that cannot be expressed by other attributes in key-value format. The string that represents the key-value format conforms to JavaScript Object Notation (JSON). *Why* is a list that traces how the event occurred and serves as a stack trace. *How* expresses the means by which the event was detected, such as the name or ID of the service or sensor. Specifically, 6W1H has the ID of the event numbered by Pub/Sub and the ID of the condition rule that evaluates the event. By referring to these IDs, it is possible to trace the history of the event in detail. In this study, we further extended the event data model and introduced the notions of the `data` attribute and the `attributes` attribute. The data attribute is a string that simply describes the entire event. The attributes attribute is a collection of child attributes that describe the event in detail, and the conventional 6W1H attributes are included in the attributes attribute as child attributes. The attributes attribute can also hold child attributes that are not included in 6W1H. The data attribute and the child attributes of the attributes attribute that are not the 6W1H attributes are not evaluated by Uni-messe at all, but play a role as additional information that can be read and written arbitrarily by the services that are linked through Uni-messe.

3.4. Condition Model

A *condition* is a structure that consists of two elements: an evaluation expression and a reference to a route. An evaluation expression holds a conditional expression for each attribute that can be evaluated using the values of the 6W1H attributes of an event. A reference to a route is a set of multiple routings (routes) to be executed when all conditions of the evaluation expression are satisfied. The following is an explanation of the evaluation formula. An evaluation formula is defined for each of the 6W1H elements,

and the evaluation result of the entire condition rule uses the logical product of the true and false value, which is the evaluation result of each evaluation formula. In other words, the evaluation result of a condition rule is true only when all of the 6W1H evaluation formulae are true. In the evaluation process for how, who, whom and where, a basic string evaluation is performed for the registered keywords. The data structure of the condition is maintained in pairs of keywords and evaluation methods. The string evaluation process for the keywords of each attribute can be performed by selecting just one of the following four processes:

- Exact match judgment;
- Partial match judgment;
- Complete non-matching judgment;
- Judgment of whether the keyword is included within an element of a set.

The when element is maintained as a kind of cron expression that expresses a periodic time and only returns a true value when the when element of the event is a time included in the cron expression. The cron expression used in Uni-messe is an extension of the Spring Boot framework and, unlike the cron expressions commonly used in Unix and other systems, it can be expressed with a precision of seconds. For the what element, complex condition evaluations are performed using logical expressions. The data structure of the what element in the condition is maintained as a string that can be evaluated in Java, i.e., a logical expression in Java. The variables contained in this logical expression are replaced based on the key value of the JSON of the what element of the event, and whether or not it is true as a Java conditional expression is then evaluated. For the why element, no evaluation expression is introduced and it always returns a true value. The evaluation expression for each of these seven attributes can be set to return a true value no matter what attribute value is passed as an event by setting a blank string. For example, users can set whom, where, and why to always be true and then only evaluate the four attributes of how, who, when, and what. The references to the route rules are held by the condition as the IDs of multiple routes. When the evaluation result of all 6W1H attributes is true and the evaluation result of the entire condition rule is true, all of the routes that are associated with that condition are fired in order.

3.5. Route Model

A *route* is a structure that represents the destination of an event. The route rule in the conventional Uni-messe data model is a simple data structure that only holds one topic name in the Pub/Sub to which the event is passed. In contrast, the route rule data model extended in this study has the following three attributes:

- Multiple topic names;
- Data attribute;
- Attributes attribute.

Since the extended route rule holds multiple topic names, it can send events to all of the topics that it holds at the time of firing. The purpose of adding the data attribute and attributes attribute is to allow meaning to be assigned to the routing by using strings and attributes. The event message after the routing in the conventional Uni-messe does not contain any information other than the 6W1H elements. Hence, it is difficult to describe how the process is changed depending on which condition–route rule was used for the routing. The data attribute of the extended route rule is a string that replaces the data attribute of the event message to be sent. The attributes attribute of the extended route rule comprises multiple attributes that are added to the attributes attribute of the event message to be sent, which are separate from the 6W1H attributes. These changes to the route data model change the behavior of the route rules in Uni-messe when they are fired. In other words, in the event routing flow in the conventional method sends the event message directly to a single Pub/Sub topic. In the extended Uni-messe, the behavior is changed

to send event messages with the data attribute and the attributes attribute to multiple Pub/Sub topics.

3.6. Event Evaluation and Routing with Condition Rules

The role of the condition in ECR rules is to link a set of 6W1H evaluation expressions to a set of routes. In other words, when a set of evaluation results becomes true, all of the routes connected through the condition are fired. When a route fires, the data attribute and the attributes attribute of the route rule are added to the event received from the application. At this time, the condition that caused the route to fire is appended to the why element of the event to record the causal relationship of the event. After that, the formatted event is published to all topics that are held by the route. As a concrete example of this evaluation and routing, we describe a case where the condition and route are defined as shown in Listing 1 and 2 and the event is received as shown in Listing 3. In this case, the result of the 6W1H evaluation based on the condition rule is as follows. The how, whom, and where attributes become true as a result of the string evaluation. The who attribute always returns true because it is not specified in the condition rule. As for the when attribute, the conditional expression is from 9:00 to 17:00 on any date, so the time of 12:00 on 1 January 2021 in the event satisfies the condition. For the what attribute, the key `errorDetection` is set to false in the event and when this is applied to the variable in the conditional expression of the condition rule, it satisfies the equality sign and becomes true. The why attribute always returns true. As a result, since the evaluation results of all 6W1H are true, the route with the ID 83 that corresponds to the condition is fired. The route rule rewrites the event as shown in Listing 4. The changes to the event include rewriting the data attribute, adding the `isHomeApplianceOperation` attribute to the attributes attribute, and adding the IDs of the event and the condition to the why attribute. The changed event is then sent to the Pub/Sub topics `room.light.on` and `room.curtain.open` in the route's `topicList`.

Listing 1. An example of a condition rule in Uni-messe.

```
{
  ''conditionId'': 27,
  ''howCondition'': ''Human-Detection Sensor'',
  ''howOperator'': ''=='',
  ''whoCondition'': '''',
  ''whoOperator'': '''',
  ''whomCondition'': ''[nakata,chen,nakamura]'',
  ''whomOperator'': ''in'',
  ''whereCondition'': ''Kobe University'',
  ''whereOperator'': ''=='',
  ''whenCondition'': ''* * 9-17 * * *'',
  ''whatCondition'': ''errorDetection==false'',
  ''routeIdList'': [83]
}
```

Listing 2. An example of a route rule in Uni-messe.

```
{
  ''routeId'': 83,
  ''topicList'': [''room.light.on'', ''room.curtain.open''],
  ''data'': ''Lab members have entered the room.'',
  ''attributes'': {
    ''isHomeApplianceOperation'': true
  }
}
```

Listing 3. An example of an event in Uni-messe.

```
{
  ''messageId'': 20890,
  ''data'': ''Human detected.'',
  ''attributes'': {
    ''how'': ''Human-Detection Sensor'',
    ''who'': '''',
    ''whom'': ''nakata'',
    ''where'': ''Kobe University'',
    ''when'': ''2021-01-01T12:00:00'',
    ''what'': ''{\''errorDetection\":false}'',
    ''why'': []
  }
}
```

Listing 4. An example of a post-evaluation event in Uni-messe.

```
{
  ''messageId'': 20901,
  ''data'': ''Lab members have entered the room.'',
  ''attributes'': {
    ''how'': ''Human-Detection Sensor'',
    ''who'': '''',
    ''whom'': ''nakata'',
    ''where'': ''Kobe University'',
    ''when'': ''2021-01-01T12:00:00'',
    ''what'': ''{\"errorDetection\":false}'',
    ''why'': [''e20890'',''c27''],
    ''isHomeApplianceOperation'': true
  }
}
```

4. Implementation

4.1. Technologies for Implementation

The technologies used for the implementation of Uni-messe are summarized in Table 1. For the back-end implementation of the Uni-messe service, we used the *Kotlin* language, a Java Virtual Machine (JVM) language, and *Spring Boot*, a Web application framework. We also built a *MySQL* server as a database server to manage the ECR rules. For Pub/Sub, which is the underlying technology of Uni-messe, we introduced a separately implemented Pub/Sub service that could be used externally via an API. In addition to providing a RESTful API, we provided a front-end UI that could be operated as a Web application from a browser as a means for users to utilize Uni-messe. For this front-end UI, we used the *TypeScript* language, *React* library, and *Material-UI* UI framework. We also used *Apache Tomcat* as the Web application server.

Table 1. A list of the technologies used.

Usage	Technology
Back-end Language	Kotlin + Spring Boot
Front-end Language	TypeScript + React + Material-UI
Database Server	MySQL
Web Application Server	Apache Tomcat

4.2. The Usage of Uni-Messe

This section describes how to use Uni-messe. The first step in using Uni-messe is to design the sequence of events from the time the service generates an event to the time it

is evaluated, routed, receives the event, and executes the service. The specific steps are as follows:

- Step 1: Decide how to describe the event using the 6W1H attributes;
- Step 2: Decide how to evaluate the event;
- Step 3: Determine the destination service to which the event is passed;
- Step 4: Decide on the format of the instructions that the service receives and processes.

After designing the flow of the service, the next step is to configure the following five items:

- The Pub/Sub topics;
- The route rules;
- The condition rules;
- The services that generate events;
- The services that receive events and execute processing.

The contents of these settings are easily determined from the design of the four steps that the users just performed. The Pub/Sub topics should be registered corresponding to the destination services determined in Step 3. For registration, users can use the topic registration API provided by Pub/Sub. Next, a route rule is written with the Pub/Sub topic name as the topicList attribute, and the instructions are then determined in Step 4 as the data attribute and attributes attribute. A condition rule can also be set by making the ID of a route rule that has already been set a routeIdList attribute and further describing the event evaluation method determined in Step 2. The registration of the route and condition rules can be easily performed using the API provided by Uni-messe or the Web application shown in the figure. In the service that generates the events, the events are created in the 6W1H format according to the expression method decided in Step 1. The event is passed to the Uni-messe service by sending it to a dedicated Uni-messe topic using the Pub/Sub API. The service that receives the event and executes the processing can receive instructions via the event by subscribing to the Pub/Sub topic to which the event is passed. Since the instructions are designed in a format that is easy to interpret through the process in Step 4, the service can easily interpret the instructions and execute the process. Using these five settings, services can be linked using Uni-messe.

4.3. The Web UI of Uni-Messe

This section describes how to use the Uni-messe Web UI, which is shown in Figure 2. The Web UI is a single-page application, which means that all functions can be handled on a single page. The operations that can be performed in the Web UI are shown in Table 2. In the route rule registration, in addition to the attributes of the route rule, the owner of the rule and the name of the application that uses the rule can be set. In the route rule search, in addition to searching by route rule ID, rules can also be narrowed down by owner and application. The rules in the search results can be updated or the rules themselves can be deleted. In the registration of condition rules, it is possible to define the 6W1H conditional expressions, except for Why, and it is possible to select how to evaluate the attributes that evaluate strings. As with the route rule, the owner and application can also be set. The condition rule can be searched for by ID, owner, and application. The rules in the search results can be updated or the rules themselves can be deleted. Moreover, it is also possible to temporarily disable the evaluation of the selected rule without deleting the condition rule. The Web UI also provides some event manipulation for testing. You can send test data to Uni-messe as an event to verify whether the rule works correctly. It is also possible to obtain a list of all events received by Uni-messe within 24 h. Users can check whether the events sent by the service to Uni-messe have been received correctly.

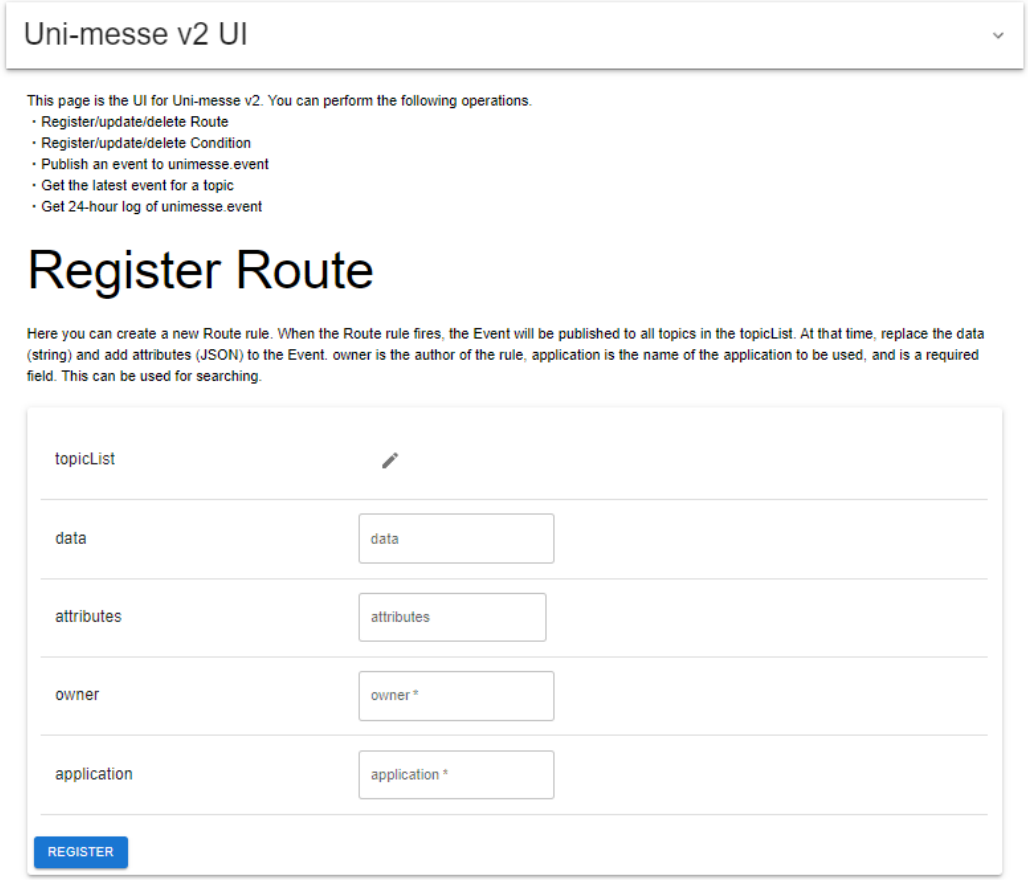

Figure 2. The Web UI of Uni-messe: route registration.

Table 2. The operations available in the Web UI.

Object	Operation
Route Rule	Registration Update Deletion Search
Condition Rule	Registration Update Deletion Activation/Deactivation Search
Event	Sending Test Data Viewing Logs

5. Application to Practical Service

We experimented with the applications of Uni-messe. We applied Uni-messe to two services: ALPS [8] and indoor location sensing [30].

5.1. Uni-Messe-Based Indoor Location Sensing

In the indoor location sensing service, video images of the home were read in real time from a fixed-point camera set high up in the room, and the coordinates of the location feature points were acquired using location sensing technology. Then, Uni-messe detected whether the location feature point coordinates were included in the named rectangular area defined by the user in real time. In this study, we used Uni-messe to build a service that operated home appliances by triggering the real-time detection of indoor location

sensing. To link indoor location sensing with Uni-messe, we needed to send an event to a Uni-messe topic on Pub/Sub when a location feature point was detected as being inside a given rectangular region. The implementation required to achieve this was the insertion of a code that executed the event-sending API of Pub/Sub into the source code of the indoor location sensing. It was also necessary to register condition rules to evaluate events and route rules to hold the destination and instructions of the events using the Uni-messe Web UI. With these implementations and configurations, it was possible to trigger the detection of indoor location sensing and convey instructions to the home appliance service via the Pub/Sub topics.

5.2. Uni-Messe-Based IoT Speaker Announcer

ALPS is a system that installs IoT speakers with motion sensors in key locations around the home and presents information based on time and location in audio form by collaborating with ECA rules through a server. The ECA rules in ALPS operate in a cycle of evaluating the location and time using condition rules and then obtaining the action, which is a sentence spoken by the speaker, as a result. The ALPS architecture shown in Figure 3 consists of an ALPS server, which handles the ECA rules, and an ALPS client, which is realized by a Raspberry Pi equipped with a motion sensor and speaker, communicating via Pub/Sub. In addition, the ALPS server has the ability to register new ALPS clients and maintains a Web UI screen that allows the easy registration and deletion of ECA rules. In this study, we implemented a new ALPS by replacing the ALPS server with Uni-messe, which restructured the entire ALPS architecture. The architecture of the new ALPS with Uni-messe is shown in Figure 4. In the conventional ALPS architecture, the ALPS server and ALPS clients communicate via Pub/Sub. Hence, the ECA rule management and evaluation functions of the ALPS server could be completely replaced by Uni-messe by simply changing the Pub/Sub topics, thereby eliminating the need to implement most of the ALPS server. As for the content of speech, it was managed in the action rules from the previous ALPS. However, since there were no action rules in the new ALPS, it was managed in the route rules. On the other hand, other modules needed to be implemented in the same way as in the conventional ALPS architecture. Specifically, an ALPS client using a Raspberry pi, an ALPS client registration system, and an ECR rule management Web UI screen were required. Since the ALPS client and the registration system are functions that do not exist in the rule-based system, they needed to be implemented as modules. The ECR rule management screen also needed to be newly implemented because it needed to have a user authentication function so that only the user of each ALPS client could write rules.

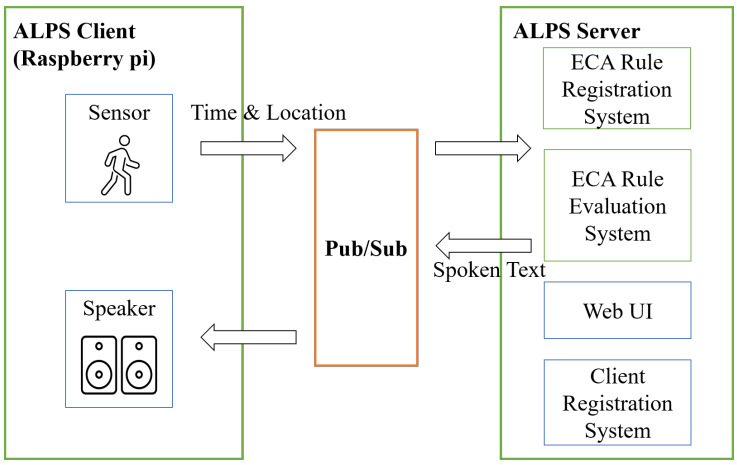

Figure 3. The previous ALPS architecture.

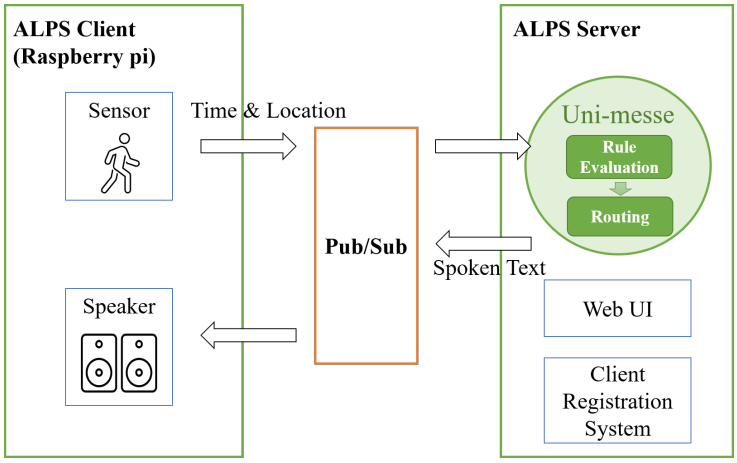

Figure 4. The new ALPS architecture with Uni-messe.

6. Evaluation

6.1. Experiment Setup

We compared and evaluated the experiments applying Uni-messe to ALPS and indoor location sensing to the conventional IFTTT method. As a conventional method, we used IFTTT, which is a service coupling platform, and compared its architecture and implementation to those of Uni-messe. Furthermore, based on the comparison, we then compared the quality of Uni-messe to various conventional rule-based systems, including IFTTT. Specifically, we focused on the following six qualities for comparison:

- Implementation;
- Generality;
- Condition simplicity;
- Loosely coupled-ness;
- Context independency;
- The availability of the UI.

The first important point was to know whether the original implementation of the rule-based system was necessary for the construction of the service. In other words, we verified whether or not each service was configured to maintain its own rule-based system. Generality is an indicator of whether or not the rule-based system maintained by a service can be used to build a wide range of smart services. This can be judged comprehensively based on the degree of freedom in the description of rules and the degree of freedom in the coupling method used with the service. Condition simplicity is an indicator of whether conditional expressions for event evaluation can be easily described without the need for advanced coding. Loosely coupled-ness indicates whether the rule-based system and external services are maintaining a low dependency on each other when building smart services. When the services are tightly coupled, it is difficult to build smart services because the implementation of the services depends on the processing contents and composition of the rule-based system. Context independency indicates whether there is a need to manage the context for using the rule-based system. Rule-based systems deal with events, which are the time differences of contexts representing the state of things, and although they do not inherently need contexts in themselves, it is possible to represent events using contexts. However, representing events with contexts has the disadvantage of increasing the complexity of the system since it requires the design of a context data model and the construction of a context database. The presence or absence of a UI indicates whether the rules can be easily managed using the UI. When a UI does not exist, the difficulty of using the rule-based system increases because specialized work, such as coding, is required to use the library or API.

6.2. IFTTT and Uni-Messe Comparison

6.2.1. Architecture Comparison

The most significant difference between the IFTTT and Uni-messe architectures was the service coupling method. Figure 5 shows service connections with IFTTT. IFTTT combines multiple trigger services with multiple action services, and can activate the action service by conditionally evaluating the events generated by the trigger service. In this case, the three elements of the trigger service, action service, and inter-service coupling were treated together as a single service. Hence, in order to combine the trigger service with another action service using IFTTT, it was necessary to register the service again to combine the three elements mentioned above. In addition, JavaScript code had to be written in order to perform conditional evaluations using IFTTT.

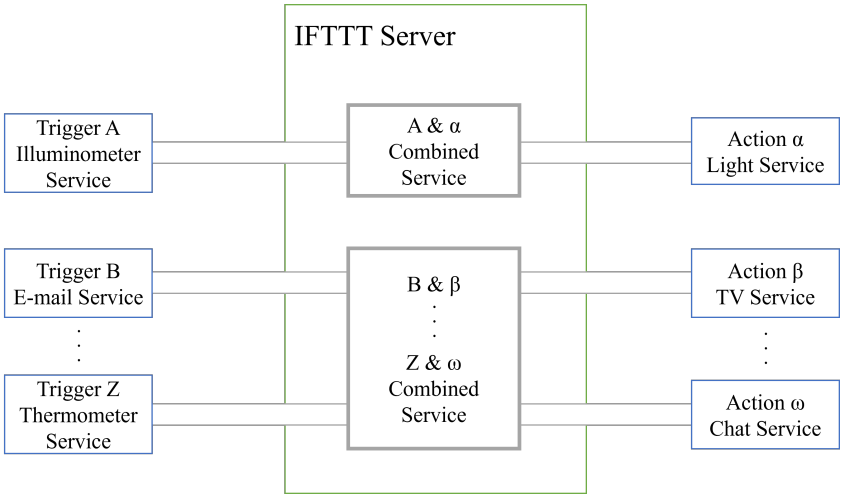

Figure 5. Service connections with IFTTT.

In contrast, the trigger service and action service in Uni-messe were very loosely coupled, as shown in Figure 6. In other words, the trigger service in Uni-messe only sent event messages and did not detect how they were received at all. The action service in Uni-messe also only received event messages and did not detect from where the messages were received. However, it was possible for the action service to obtain information about the sender via the event message. The reason the services were loosely coupled on Uni-messe was that all events received by Uni-messe are evaluated and routed in an aggregated manner. Hence, all services communicating through Uni-messe still had the possibility to be coupled with all other services depending on the configuration of the rules. Conversely, there could be services that did not receive events, i.e., did not combine with Uni-messe, depending on the evaluation results of the event, as shown by the dotted arrow combination in Figure 6.

6.2.2. C1: Indoor Location Sensing Implementation Using IFTTT

We compared and evaluated the implementation of IFTTT and Uni-messe in the development of services for both indoor location sensing and ALPS. The criteria for evaluation were the number of implementation steps and the complexity required for each of the four elements: trigger service, action service, event condition evaluation formula, and overall service connection, as well as software engineering indicators, such as reusability and the software connectivity of each element. More specifically, we examined how to use IFTTT to link indoor location sensing with home appliance operation. First, as preparation for the trigger service, in order to send data from the indoor location sensing to the IFTTT, it was necessary to obtain the URL for sending data from the IFTTT server and then to write the source code for sending the data from the indoor location sensing to this URL. Next, to prepare for the action service, it was necessary to link the IFTTT with the home

appliance operation service. At this time, when the home appliance services (Amazon Alexa, iRobot, etc. [44]) were already prepared by the IFTTT, they could be linked without any using any special settings. To link a service that is not compatible with IFTTT or a self-made service, users need to register the destination URL of the target service in the IFTTT. Finally, it was necessary to create the evaluation part to execute the action service by evaluating and conditionally branching the data sent by the trigger service. To create the evaluation part, JavaScript code needed to be written. Furthermore, one of the biggest problems with IFTTT is that all three elements (trigger service, action service, and evaluation) are managed together as a single connection, so it is necessary to manipulate common connection data to register and update each of the three settings. This means that a single common connection data needs to be manipulated to register and update each of the three settings. This not only dramatically increases the complexity of the data structure, but also makes it impossible to reuse the trigger service, action service, and evaluation, which greatly impairs scalability.

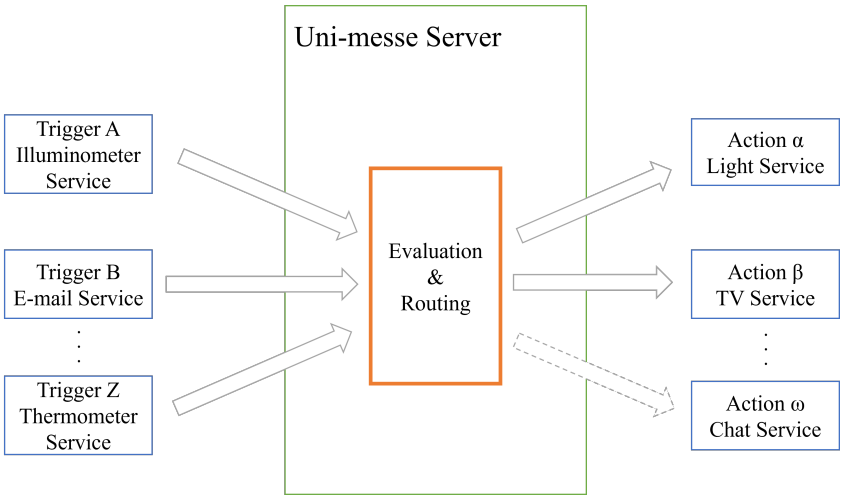

Figure 6. Service connections with Uni-messe.

6.2.3. Discussion of C1

When comparing the applied implementations of IFTTT and Uni-messe for indoor location sensing, we found that Uni-messe had the following four advantages. First, Uni-messe could use a common Pub/Sub API for the trigger service's data transmission API and there was no need to set up a new one to obtain a URL. Second, the description of the conditional branching in Uni-messe was not JavaScript code, but rather an intuitive evaluation expression description for the 6W1H conditions, which could be easily manipulated on a form input-like screen using the Web UI, thereby making the cost of conditional description very small. The third advantage was that Uni-messe could delegate the processing to the action service by simply passing event messages to Pub/Sub. IFTTT had the disadvantage of requiring the API provided by the action service to be registered with IFTTT, which increased the density of the coupling between the IFTTT and the action service. Finally, the data structure of Uni-messe was more sophisticated than that of IFTTT. Each of the event, condition, and route rules in Uni-messe corresponded to the three elements of the trigger, evaluation, and action services in IFTTT. However, unlike IFTTT, each rule was managed separately rather than in a data structure where each rule was connected as a single connection. Thus, the data structure of Uni-messe was very sophisticated, which had a great advantage in service coordination in that rules could be individually edited and reused.

6.2.4. C2: ALPS Implementation Using IFTTT

The ALPS client itself, the ALPS client registration system, and the ECR rule management Web UI screen needed to be implemented in the same way as in Uni-messe. We

describe the implementation in which the core ECA rule management and evaluation functions of ALPS were substituted by IFTTT. First, to prepare the trigger service, it was necessary to obtain a separate data destination URL for each of the ALPS clients from the IFTTT server. Next, to prepare for the action service, it was necessary to register the same number of actions in the IFTTT as the number of sentences to be spoken by the ALPS clients. At this time, it was necessary to completely describe the API for speaking sentences in all of the actions. In addition, a data destination URL needed to be registered in the IFTTT for each ALPS client. Finally, the time and location at which the ALPS client successfully sensed the person needed to be coded as a conditional expression using JavaScript. Then, all three of these elements needed to be managed simultaneously in a non-reusable format as the connection data for each ALPS client.

6.2.5. Discussion of C2

When comparing the implementation of IFTTT and Uni-messe for ALPS, we found that Uni-messe had the following four advantages. First, Uni-messe could collect data from multiple ALPS clients in an aggregated manner through Pub/Sub, eliminating the need to prepare a separate URL for each ALPS client. Second, Uni-messe allowed the direct description of instructions in the form of the data attribute and attributes attribute in the route rules, meaning that instructions to the ALPS clients could be efficiently managed as reusable and easily updated route rules. Additionally, unlike IFTTT, there was no need to describe the content of speech in a complex API format when registering it to Uni-messe, thereby making the registration process easier. The third advantage was that the branching of the data transmission from Uni-messe to multiple ALPS clients could be easily carried out by simply changing the Pub/Sub topic and there was no need to register URL information in Uni-messe. Finally, since the time and place, which are the targets of the conditional evaluation in ALPS, corresponded to the when and where attributes, which are the 6W1H attributes of Uni-messe, the conditional evaluation formula in Uni-messe could be described in an extremely simple manner.

6.3. Results

From the above comparison, it was found that IFTTT had the complexity of requiring coding for the conditional description and it also lacked the looseness between services. Uni-messe solved the complexity and looseness problems of IFTTT. In addition, Uni-messe is a service that solved the six quality issues of various conventional rule-based systems: implementation, generality, condition simplicity, loosely coupled-ness, context independency, and the availability of the UI. Table 3 compares the quality of various rule-based systems. Smart services that maintain their own rule-based system, such as ALPS, sacrifice the versatility of a rule-based system, while other qualities, such as the ease of writing conditional evaluations, are increased in a way that is optimized for services. RuCAS succeeded in converting rule-based systems into services; however, only HTTP GET requests can be used to command the services. Hence, it lacks the versatility to realize a variety of smart services and has the added disadvantage that the rules change depending on the API of the service. FIWARE Perseo is an extremely sophisticated and versatile rule-based system with a variety of conditional descriptions and support for Simple Mail Transfer Protocol (SMTP); however, it also has many drawbacks. Since Perseo is a module of the FIWARE architecture, which is the context management infrastructure, it not only requires an understanding of the entire FIWARE architecture, but it also imposes a major restriction on the data model in that every event that is handled must be re-expressed as a change in context. In addition, its cost of use is very high, as Event Processing Language (EPL) must be learned and used to describe conditions and there is no UI that can directly control Perseo. Furthermore, since direct commands are given via protocols, such as HTTP and SMTP, loose coupling between services cannot be achieved. From these comparisons, it is clear that the Uni-messe proposed in this research is a practical service that solves the six quality issues faced by conventional rule-based systems.

Table 3. The results of the comparison between Uni-messe and other existing rule-based systems.

	Implementation	Generality	Condition Simplicity	Loosely Coupled-Ness	Context Independency	Availability of UI
ALPS	x	x	o	o	o	o
RuCAS	o	x	o	x	o	o
Perseo	o	o	x	x	x	x
IFTTT	o	o	x	x	o	o
Uni-messe	o	o	o	o	o	o

6.4. Advantage and Limitation

One of the most significant benefits of using Uni-messe for smart service development is that it eliminates the need for the proprietary implementations of other rule-based systems. Another major advantage is that Uni-messe is provided as a neutral service. Uni-messe is versatile enough to be combined with a variety of smart services and loosely coupled with other services via Pub/Sub, so services that work with Uni-messe can be freely designed without depending on the specifications of the rule-based system. Since the architecture is context independent, there is no need to adopt the FIWARE architecture and there is a high degree of freedom when describing events. Furthermore, since there is no need to use source code or EPL for the conditional descriptions and there is a UI that allows for easy rule registration, it is easy to design and manage ECR rules that describe the coordination of the services. The above advantages make Uni-messe an excellent event routing infrastructure that can streamline the development of a wide range of smart services.

The first limitation of Uni-messe is that as the number of condition rules increases, the processing for evaluating one event becomes heavier. This can be solved by simultaneously running multiple Uni-messe systems to distribute the load. Moreover, although most of the events in smart services can be expressed in the 6W1H format, it is difficult to use Uni-messe as a general purpose rule-based system for industries other than smart services because events cannot be expressed in the 6W1H format. However, since the purpose of Uni-messe is to support the construction of smart services, this is not a major problem. Moreover, although condition and route rules can be easily registered in the UI, they have to be registered manually, which creates a certain degree of complexity.

7. Conclusions

In this study, the rule-based system Uni-messe that was converted into a service was proposed for application-neutral rule management and evaluation. The main outcomes include the following: (1) the proposal of a new ECR model; (2) the development of a Uni-messe architecture using the ECR model and Pub/Sub; (3) the ease with which Uni-messe integrates context-aware services. Based on the implementation and evaluation, we found that Uni-messe is more effective than systems used in previous studies for constructing a context-aware service realized by a rule-based system. However, context-aware services can also be realized using learning-based AI, such as Deep Learning, instead of inference AI, as used in rule-based systems. Since learning-type AI [45] can utilize various big data obtained from people's daily lives, the range of services that can be provided by smart services is expected to expand and thus, the future possibility of context-aware services relying on rule-based systems remains questionable. Hence, it is necessary to expand the possibilities of smart services by using Deep Learning to construct rules in rule-based systems. Moreover, the advantage of inference-based AI is that it can retrospectively evaluate the computation process, which is an important issue in smart services, and this can be used to ensure safe operation and increase the variety of services provided by evaluating the time series of the computation processes using temporal logic. As future work, we aim to consider a mechanism that can dynamically construct services by automating the registration of ECR

rules. We also aim to use Uni-messe to build smart services that are closely adapted to the lives of individuals.

Author Contributions: Writing—original draft preparation, T.N.; writing—review and editing, S.C.; supervision, M.N.; validation, S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This research was partially supported by JSPS KAKENHI, grant numbers JP19H01138, JP18H03242, JP18H03342, JP19H04154, JP19K02973, JP20K11059, JP20H04014, and JP20H05706 and the Tateishi Science and Technology Foundation (C) (No.2207004).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dong, B.; Shi, Q.; Yang, Y.; Wen, F.; Zhang, Z.; Lee, C. Technology evolution from self-powered sensors to AIoT enabled smart homes. *Nano Energy* **2021**, *79*, 105414. [CrossRef]
2. Khan, A.T.; Li, S.; Cao, X. Control framework for cooperative robots in smart home using bio-inspired neural network. *Measurement* **2021**, *167*, 108253. [CrossRef]
3. Gopikumar, S.; Raja, S.; Robinson, Y.H.; Shanmuganathan, V.; Chang, H.; Rho, S. A method of landfill leachate management using internet of things for sustainable smart city development. *Sustain. Cities Soc.* **2021**, *66*, 102521. [CrossRef]
4. Gheisari, M.; Najafabadi, H.E.; Alzubi, J.A.; Gao, J.; Wang, G.; Abbasi, A.A.; Castiglione, A. OBPP: An ontology-based framework for privacy-preserving in IoT-based smart city. *Future Gener. Comput. Syst.* **2021**, *123*, 1–13. [CrossRef]
5. Liu, W.; Yan, X.; Wei, W.; Xie, D. Pricing decisions for service platform with provider's threshold participating quantity, value-added service and matching ability. *Transport. Res. E-Log.* **2019**, *122*, 410–432. [CrossRef]
6. Dan, B.; Zhang, S.; Zhou, M. Strategies for warranty service in a dual-channel supply chain with value-added service competition. *Int. J. Prod. Res.* **2018**, *56*, 5677–5699. [CrossRef]
7. Ishaq, M.; Afzal, M.H.; Tahir, S.; Ullah, K. A Compact Study of Recent Trends of Challenges and Opportunities in Integrating Internet of Things (IoT) and Cloud Computing. In Proceedings of the 2021 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube), Quetta, Pakistan, 26–27 October 2021; pp. 1–4.
8. Akashi, T.; Nakamura, M.; Yasuda, K.; Saiki, S. Proposal for a Personalized Adaptive Speaker Service to Support the Elderly at home. In Proceedings of the 22nd IEEE-ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel Distributed Computing (SNPD2021), Taichung, Taiwan, 24–26 November 2021; pp. 2–7.
9. Rademacher, F.; Sachweh, S.; Zündorf, A. A Modeling Method for Systematic Architecture Reconstruction of Microservice-Based Software Systems. In *Enterprise, Business-Process and Information Systems Modeling*; Nurcan, S., Reinhartz-Berger, I., Soffer, P., Zdravkovic, J., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 311–326.
10. Islam, M.S.U.; Kumar, A.; Hu, Y.C. Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions. *J. Netw. Comput. Appl.* **2021**, *180*, 103008. [CrossRef]
11. Ahmad, I.; Pothuganti, K. Smart Field Monitoring using ToxTrac: A Cyber-Physical System Approach in Agriculture. In Proceedings of the 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 10–12 September 2020; pp. 723–727.
12. Leng, J.; Zhang, H.; Yan, D.; Liu, Q.; Chen, X.; Zhang, D. Digital twin-driven manufacturing cyber-physical system for parallel controlling of smart workshop. *J. Amb. Intel. Hum. Comp.* **2019**, *10*, 1155–1166. [CrossRef]
13. Li, X.; Chen, C.H.; Zheng, P.; Jiang, Z.; Wang, L. A context-aware diversity-oriented knowledge recommendation approach for smart engineering solution design. *Knowl.-Based Syst.* **2021**, *215*, 106739. [CrossRef]
14. Chavhan, S.; Gupta, D.; Nagaraju, C.; Rammohan, A.; Khanna, A.; Rodrigues, J.J. An efficient context-Aware vehicle incidents route service management for intelligent transport system. *IEEE Syst. J.* **2021** . [CrossRef]
15. Favi, C.; Garziera, R.; Campi, F. A rule-based system to promote design for manufacturing and assembly in the development of welded structure: Method and tool proposition. *Appl. Sci.* **2021**, *11*, 2326. [CrossRef]
16. Ying, H.; Lee, S. A rule-based system to automatically validate IFC second-level space boundaries for building energy analysis. *Autom. Constr.* **2021**, *127*, 103724. [CrossRef]
17. Xu, F.; Uszkoreit, H.; Du, Y.; Fan, W.; Zhao, D.; Zhu, J. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *CCF International Conference on Natural Language Processing and Chinese Computing*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 563–574.
18. Wetzstein, G.; Ozcan, A.; Gigan, S.; Fan, S.; Englund, D.; Soljačić, M.; Denz, C.; Miller, D.A.; Psaltis, D. Inference in artificial intelligence with deep optics and photonics. *Nature* **2020**, *588*, 39–47. [CrossRef] [PubMed]
19. de Ipiña, D.L.; Katsiri, E. An eca rule-matching service for simpler development of reactive applications. In *Published as a Supplement to the Proceedings of Middleware*; IEEE: New York, NY, USA, 2001.

20. Nespoli, P.; Díaz-López, D.; Mármol, F.G. Cyberprotection in IoT environments: A dynamic rule-based solution to defend smart devices. *J. Inf. Secur. Appl.* **2021**, *60*, 102878. [CrossRef]
21. Takatsuka, H.; Saiki, S.; Matsumoto, S.; Nakamura, M. RuCAS: Rule-Based Framework for Managing Context-Aware Services with Distributed Web Services. *Int. J. Softw. Innov. (IJSI)* **2015**, *3*, 57–68. [CrossRef]
22. My Applets-IFTTT. Available online: https://ifttt.com/home (accessed on 1 August 2021).
23. Ovadia, S. Automate the Internet With "If This Then That" (IFTTT). *Behav. Soc. Sci. Libr.* **2014**, *33*, 208–211. [CrossRef]
24. Mi, X.; Qian, F.; Zhang, Y.; Wang, X. An Empirical Characterization of IFTTT: Ecosystem, Usage, and Performance. In *Proceedings of the 2017 Internet Measurement Conference*; Association for Computing Machinery: New York, NY, USA, 2017; pp. 398–404.
25. Esaki, K.; Azuma, M.; Komiyama, T. Introduction of quality requirement and evaluation based on ISO/IEC square series of standard. In *International Conference on Trustworthy Computing and Services*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 94–101.
26. Hussain, J.; Ul Hassan, A.; Muhammad Bilal, H.S.; Ali, R.; Afzal, M.; Hussain, S.; Bang, J.; Banos, O.; Lee, S. Model-based adaptive user interface based on context and user experience evaluation. *J. Multimodal User Interfaces* **2018**, *12*, 1–16. [CrossRef]
27. Guo, L.; Zhang, D.; Li, G.; Tan, K.L.; Bao, Z. Location-aware pub/sub system: When continuous moving queries meet dynamic event streams. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Australia, 31 May–4 June 2015; pp. 843–857.
28. Zhao, Y.; Li, Y.; Mu, Q.; Yang, B.; Yu, Y. Secure pub-sub: Blockchain-based fair payment with reputation for reliable cyber physical systems. *IEEE Access* **2018**, *6*, 12295–12303. [CrossRef]
29. Nakata, T.; Chen, S.; Nakamura, M. Developing Event Routing Service to Support Context-Aware Service Integration. In Proceedings of the 22nd IEEE-ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel Distributed Computing (SNPD2021), Taichung, Taiwan, 24–26 November 2021; pp. 267–272.
30. Chen, S.; Saiki, S.; Nakamura, M. Using Human Pose Estimation for User-Defined Indoor Location Sensing. In Proceedings of the 2021 International Workshop on Pervasive Information Flow (PerFlow'21), Held in Conjunction with the 19th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2021), Kassel, Germany, 22–26 March 2021; pp. 495–501.
31. The Open Source Platform for Our Smart Digital Future—FIWARE. Available online: https://www.fiware.org/ (accessed on 1 August 2021).
32. Yacchirema, D.C.; Sarabia-Jácome, D.; Palau, C.E.; Esteve, M. A smart system for sleep monitoring by integrating IoT with big data analytics. *IEEE Access* **2018**, *6*, 35988–36001. [CrossRef]
33. Beverungen, D.; Müller, O.; Matzner, M.; Mendling, J.; Vom Brocke, J. Conceptualizing smart service systems. *Electron. Mark.* **2019**, *29*, 7–18. [CrossRef]
34. Nature Remo. Available online: https://nature.global/nature-remo/ (accessed on 1 August 2021).
35. Cerny, T.; Donahoo, M.J.; Trnka, M. Contextual understanding of microservice architecture: current and future directions. *ACM SIGAPP Appl. Comput. Rev.* **2018**, *17*, 29–45. [CrossRef]
36. Di Francesco, P.; Lago, P.; Malavolta, I. Migrating towards microservice architectures: an industrial survey. In Proceedings of the 2018 IEEE International Conference on Software Architecture (ICSA), Seattle, WA, USA, 30 April–4 May 2018; pp. 29–2909.
37. Jain, A.K.; Gupta, B. Rule-based framework for detection of smishing messages in mobile environment. *Procedia Comput. Sci.* **2018**, *125*, 617–623. [CrossRef]
38. Cheng, B.; Wang, M.; Zhao, S.; Zhai, Z.; Zhu, D.; Chen, J. Situation-aware dynamic service coordination in an IoT environment. *IEEE ACM Trans. Netw.* **2017**, *25*, 2082–2095. [CrossRef]
39. Sampaio, A.R.; Kadiyala, H.; Hu, B.; Steinbacher, J.; Erwin, T.; Rosa, N.; Beschastnikh, I.; Rubin, J. Supporting microservice evolution. In Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 17–22 September 2017; pp. 539–543.
40. Bogner, J.; Wagner, S.; Zimmermann, A. Collecting service-based maintainability metrics from RESTful API descriptions: static analysis and threshold derivation. In *European Conference on Software Architecture*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 215–227.
41. Introduction-Perseo Context-Aware Complex Event Processing. Available online: https://fiware-perseo-fe.readthedocs.io/en/latest/ (accessed on 29 December 2021).
42. Amin, M.M.; Maseleno, A.; Shankar, K.; Perumal, E.; Vidhyavathi, R.; Lakshmanaprabu, S. Active Database System Approach and Rule Based in the Development of Academic Information System. *Int. J. Eng. Technol.* **2018**, *7*, 95–101. [CrossRef]
43. Tamani, N.; Ahvar, S.; Santos, G.; Istasse, B.; Praca, I.; Brun, P.E.; Ghamri, Y.; Crespi, N.; Becue, A. Rule-based model for smart building supervision and management. In Proceedings of the 2018 IEEE International Conference on Services Computing (SCC), San Francisco, CA, USA, 2–7 July 2018; pp. 9–16.
44. See All Services-IFTTT. Available online: https://ifttt.com/services (accessed on 29 December 2021).
45. Chimatapu, R.; Hagras, H.; Kern, M.; Owusu, G. Hybrid deep learning type-2 fuzzy logic systems for explainable AI. In Proceedings of the 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Glasgow, UK, 19–24 July 2020; pp. 1–6.