Development of Smart Mobility Application based on Microservice Architecture

Masatoshi Nagao¹, Takahiro Ando¹, Kenji Hisazumi², Akira Fukuda¹
¹Graduate School / Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan
²System LSI Research Center, Kyushu University, Fukuoka, Japan

Abstract—This study proposed the use of microservice architecture for smart mobility applications. For application development in the field of intelligent transport systems, considering a system that is simple, sustainable, and has long-term operability is necessary. Using the microservice architecture, evolutionary and flexible applications can be easily developed using the services that already exist. This study introduces the overview of microservice architecture and discuss its usefulness for the development of smart mobility applications based on the bus timetable recommendation application developed herein using this architecture.

Keywords: Smart Mobility, Intelligent Transport Systems, Microservice Architecture, Application, Mixed Reality

1. Introduction

Smart mobility society is a society to realize smooth and comfortable mobilities with intelligent transport systems (ITS). With the development of ITS, the realization of smart mobility society is expected to increase. Applications/systems related to smart mobility such as destination recommendation guide applications and driving safety support systems are being actively developed. However, much of this research is based on the perspective of civil works and city planning and research from the viewpoint of information platform is limited [1]. For the development of smart mobility applications, confirming the appropriate architecture in advance, including its effect on the life cycle of the system, is essential.

Some applications related to ITS use augmented reality (AR) technology as the user interface. Recently, a technology called mixed reality (MR), which is related to AR, is being advocated by Microsoft. MR uses the information of objects in the real world and displays life-like virtual objects. Utilizing this characteristic, MR can be used as the user interface for applications related to ITS.

This study is based on the idea that creation of a new implementation for a function can be avoided by taking advantage of other services for the same function already provided.

2. Related Technology

2.1 Microservice Architecture [2]

Microservice architecture is a system configuration that combines multiple small-scale services (microservices) and realizes a new application. Each microservice is a service that provides a different function, and the connection between each service is realized by a lightweight communication method. The difference between monolithic architecture and microservice architecture is shown in Fig. 1. Lewis and Fowler mentioned the nine demands of the microservice architecture as follows [2].

Componentization via Services
Major functions are not performed as libraries, but are decomposed as services and can be individually managed.

Organized around Business Capabilities
Make it a developer team with a wide range of skills to individually build and operate services.

Products not Projects
Increase value as a business by developing and operating continuously.

Smart Endpoints and Dumb Pipes
Communicate between services using a simple and lightweight protocol.

Decentralized Governance
Design using technology that is appropriate for each service. No need to standardize on a single
3. Development of Smart Mobility Application Using Microservice Architecture

Various kinds of data are used in the development of smart mobility applications. For example, the destination recommendation application requires map data, congestion information, weather information, transportation fee data, etc. Moreover, because such data is frequently updated the application must be flexibly adaptable to different situations for providing accurate data. Furthermore, developers need to consider application usability at all times. In summary, a smart mobility application will need to satisfy the following items.

- Easily deal with many types of data
- Improve usability
- Adapt to the frequently updated data

In this study, we consider that using microservice architecture as a development method is most suitable to satisfy these items.

To satisfy the first requirement, each data can be divided as a service and managed independently in microservice architecture. By assigning work to the service side, developers do not need to manage the data when dealing with many types of data.

To satisfy the second requirement, the concept of Design for Failure (Section 2.1) of the microservice architecture can be applied. Applications can be developed such that even when a certain service fails, they can be available except for certain functions. In other words, the scenario wherein the entire application cannot be used owing to failure of a certain service can be avoided. Therefore, using microservice architecture improves usability.

Finally, since each type of data is divided into services, it can be easily updated. The service manager only has to manage the data to ensure continuous operation of the service. In summary, microservice architecture is suitable for smart mobility applications. Furthermore, using services that already exist, constructing a new service is easy.

4. Overview of the Bus Timetable Recommendation Application

In this study, we developed a smart mobility application using MR by applying microservice architecture. In this section, we describe the outline of the bus timetable recommendation application developed herein and explain the element technology used.

4.1 Application Behavior

Figure 3 shows the outline of the bus timetable recommendation application. This application uses two independent timetable services and performs the following operation.

---

2.2 Mixed Reality

MR is a technology wherein virtual information and objects are projected to the real world through a head-mounted display, and both the information in the real and virtual worlds interacts with each other. The concept of MR is shown Fig. 2. There are two types of MR devices: holographic and immersive. Using the holographic-type MR device, the real world can be seen through a transmission lens and an AR-like experience can be achieved by displaying objects on the lens. The immersive-type MR device recognizes the real world through a camera and a VR-like experience can be achieved by projecting an environment composed of virtual objects on the field of view. In brief, MR includes the concepts of both AR and VR.

Examples of a typical immersive device include the Acer Windows Mixed Reality Headset AH101 [3] and Samsung Odyssey [4] and that of a typical Holographic device include the Microsoft HoloLens [5] and Meta2 [6]. In this study, we use Microsoft HoloLens, which can operate independently (without connecting with other personal computers).

![Fig. 2: The concept of Mixed Reality](image-url)
1) When HoloLens recognizes the prespecified AR marker, it sends a data request to the corresponding timetable services.

2) Based on the request, the timetable services converts the data into the JavaScript Object Notation (JSON) format and return it.

3) HoloLens displays the recommended time on the MR space from the received data in the JSON format.

4.2 Element Technology

Vuforia [7]

Vuforia is a software development kit for mobile devices that enables the development of AR applications. It can register any image as an AR marker. The behavior when the marker is recognized is described on Unity (Section 6.1).

REST API

Representational state transfer (REST) application programming interface (API) is a type of web service design model. The specifications are as follows.

- Individual resources can be referred to by uniform resource locator (URL) and communicate using the hypertext transfer protocol (HTTP) method.
- The four operations of the REST API are shown in Table 1.
- Notify the result by the HTTP status code. The meaning of the code number is briefly given below.

1) (100 number) Information
   - The request is accepted and processing is continued.
2) (200 number) Success
   - The request was accepted and processing is completed.
3) (300 number) Redirection
   - Additional processing is required to complete the request.

<table>
<thead>
<tr>
<th>Name</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>Create data</td>
</tr>
<tr>
<td>GET</td>
<td>Get data</td>
</tr>
<tr>
<td>PUT</td>
<td>Update data</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete data</td>
</tr>
</tbody>
</table>

Table 1: Operation of REST API

4) (400 number) Client Error
   - Processing could not be completed because there was an error in the request from the client.

5) (500 number) Server Error
   - The server could not complete processing the request.

JSON

JSON is a type of data interchange format and is described as a name-value pair in the `{key : value}` format. This format is characterized by high readability, lightweight data. In the application developed in this study, we launch simple timetable services that can return data in the JSON format. This service is realized using a tool called “JSON Server,” which can easily launch RESTful mock server.

5. System of the Bus Timetable Recommendation Application

This section describes the design and specifications of the application outlined in the previous chapter.

5.1 Overall View of the System

Figure 4 outlines the entire application including the main modules that make up the system. It expresses the name/attribute/function for each module and represents the relationships by lines. Each module is explained below.

Display panel

The “Display panel” and “Service communication part of the bus A(B)” are connected. This module
The state machine diagram representing the state transition of the application is shown in Fig. 5. It shows the operation when the application is executed on HoloLens and the operation of the two independent timetable services. Each state is as follows.

**Behavior on HoloLens**

**Marker recognition standby**

This state shows the operation from activation of the application to recognition of the marker on the view of HoloLens. When HoloLens recognizes the marker, the application shifts to the state of “Communicating.”

**Communicating**

This state shows the operation until the timetables are acquired from the timetable services. A request is sent to each service and the application will be in the “Waiting for reception” state until a response is received from the service. If timetable data acquisition from the service is successful, the recommended time is decided by comparing the timetable data with the current time; the application then exits the state of “Communicating.” If data acquisition fails, a warning indicating that communicating with the service has failed is displayed and the application exits this state. In this state, the communication with the two services is performed asynchronously. After each communication ends and the recommended data (or error) display is completed, the application shifts to the state of “Displaying.”

**Displaying**

This state displays the recommended time. When the data of the recommended time is described over a page, it is displayed over multiple pages. In addition, the window displayed on HoloLens has the function to follow the line of sight, and in the “Displaying” state, the eye tracking process is always performed. This state has two events to shift to other states. One is an event wherein the erase button is pressed and the process of deleting the display window is executed. Subsequently, the application shifts to the “Marker recognition standby” state. Another is an event wherein the update button is pressed and the application is shifted to the “Communicating” state to update the recommendation time.

**5.2 Application Procedure**

The state machine diagram representing the state transition of the application is shown in Fig. 5. It shows the operation when the application is executed on HoloLens and the operation of the two independent timetable services. Each state is as follows.

**Bus A(B) timetable service**

This is an independent small-scale service that provides bus timetables. It analyzes the received request and transmits the specified timetable data in the JSON format to the client if the request is correct. If there are faults in the request, it notifies the client of request errors via the HTTP status code.

**Vuforia**

In the “External site,” it registers the markers that are recognized with Unity and downloads the dataset for UnityEditor. In the “Library,” it selects the marker included in the dataset and plays the role of making the “Display panel” appear when the HoloLens recognizes the marker.
5.3 User Interface

The timetable displayed on HoloLens is shown in Fig. 6. Figure 7 shows a screenshot of when the timetable in Fig. 6 is actually displayed on HoloLens. Functions shown in this figure are explained below.

1) **Current time**
   Displays the current time.

2) **Destination**
   Displays the bus company name and destination.

3) **Update button**
   Communicates again with the bus timetable server and updates the information.

4) **Erase button**
   Erases the information on the display panel.

5) **Bus status**
   Displays the current status of the bus such as “Awaiting” or “Passed.”

6) **Bus recommended time**
   Recommends and displays the time of the bus that has not passed based on the current time.

7) **Page manipulation**
   Displays the next(previous) page.

8) **Page number**
   Displays the number indicating the page currently displayed.

9) **Gaze tracking**
   The timetable board moves following the line of sight on the HoloLens.

6. Implementation and Evaluation

This section describes the development environment and procedures involved in developing the bus timetable recommendation application using microservice architecture. Moreover, it evaluates whether applying microservice architecture is suitable for the development of smart mobility applications.

6.1 Implementation Environment

When developing the bus timetable recommendation application, we create objects and describe behaviors in Unity. The development procedure of the application is as follows.

1) Create objects and describe behavior.
2) Convert to code for universal windows platform (UWP) and debug.
3) Deploy the application to Microsoft HoloLens.

**Unity [8]**

Unity is used to create and operate the object projected by Microsoft HoloLens. It is a game engine corresponding to multiple platforms and is developed by Unity Technologies. It is often used when creating games that handle 2D/3D models.

**Visual Studio [9]**

After creating the application and converting to code for UWP in Unity, we used Visual Studio to deploy the application to Microsoft HoloLens. Visual Studio is an integrated development environment provided by Microsoft for developing software.

6.2 Method and Result of the Evaluation

We qualitatively evaluated how well the nine features of the microservice architecture described in Section 2.1 are satisfied in the application developed in this study. Based on the above, we discuss how well the microservice architecture is suitable for smart mobility applications. The evaluation results are shown in Table 2 and we explain the reason for the evaluation result of each item below.

**Componentization via Services**

For implementing the bus timetable recommendation application, we use two independent services that provide bus timetables. This application acquires bus timetables from each service and displays the recommended time for each timetable by comparing it with the current time. Therefore, the main function of the application can be implemented without using a library. Therefore, we evaluate this parameter as “being satisfied (✓).”
Table 2: Evaluation result of the developed application

<table>
<thead>
<tr>
<th>Componentization via Services</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organized around Business Capabilities</td>
<td>–</td>
</tr>
<tr>
<td>Products not Projects</td>
<td>–</td>
</tr>
<tr>
<td>Smart Endpoints and Dumb Pipes</td>
<td>✓</td>
</tr>
<tr>
<td>Decentralized Governance</td>
<td>–</td>
</tr>
<tr>
<td>Decentralized Data Management</td>
<td>✓</td>
</tr>
<tr>
<td>Infrastructure Automation</td>
<td>–</td>
</tr>
<tr>
<td>Design for Failure</td>
<td>✓</td>
</tr>
<tr>
<td>Evolutionary Design</td>
<td>✓</td>
</tr>
</tbody>
</table>

Organized around Business Capabilities

In this study, we developed a prototype of the application. However, because it was developed individually, we did not organize a team to construct and operate the service. Therefore, we evaluate this parameter as “unconfirmed (–).” Note that because functions of the application are decomposed into services, it is realistic to organize teams with various abilities for each service.

Products not Projects

For the same reason in the previous item, we could not verify this item. Therefore, we evaluate it as “unconfirmed.” However, we think that it is sufficiently possible to continuously circulate the life cycle for each service.

Smart Endpoints and Dumb Pipes

The communication between the services of the application is implemented using the REST API via the HTTP method, and only select from the four operations shown in Table 1 with URL specifying. Because extra functions are reduces as much as possible and only the necessary functions exist, we evaluate this parameter as “being satisfied.”

Decentralized Governance

This application has two microservices that provide timetables. Because the application is a prototype of smart mobility applications using microservice architecture, both services use the JSON server, which is a tool for launching a simple server. Therefore, the application is not implemented using multiple platforms. However, regardless of the internal implementation of the service, data communication can be done without problems as long as the interface is decided. Thus, although we think that it is sufficiently possible to satisfy this item, since we do not actually implement it, we evaluate it as “unconfirmed.”

Decentralized Data Management

Because both timetable services have their own timetable data for each service, the suitable database for each service can be used for data management. Therefore, we evaluate this parameter as “being satisfied.”

Infrastructure Automation

Because automation test of the developed application is not implemented, we evaluate this parameter as “unconfirmed.” For implementation, we think that a simple data server should be launched for testing and repeatedly communicating via various requests.

Design for Failure

This application has two microservices that provide timetables. Herein, the application is constructed such that even when one microservice is unavailable, the other service can be used. Figure 8 shows the execution of the application on HoloLens. Even if one service cannot be used, the application can display the information of the other service. In other words, a system that considers the influence of service failure has been constructed. For these reasons, we evaluate this parameter as “being satisfied.”

Evolutionary Design

The functions used in the application are divided by service units. Because services are connected only by simple communication using REST API, each service can evolve independently. For this reason, we evaluate this parameter as “being satisfied.”

General Review

Based on the above evaluation results, no item is concluded as “not suitable.” Therefore, we conclude that the microservice architecture is fully suitable to develop smart mobility applications. Furthermore, it is possible to completely satisfy the “unconfirmed” parameters depending on the implementation.

7. Related Work

Considerable research has been conducted on microservice architecture. Reference [10] mentions how to divide the system into microservices and refers to the communication patterns between services. Our application adopts the asynchronicity recommended in this research as the communication method of the internet of things application.
Reference [11] describes the experience when a monolithic software configuration is moved to a microservice architecture. Using this architecture can enable a system to expand flexibly; however, moving to this architecture requires considerable labor. In conclusion, it should not be determined easily to shift microservice architecture.

In reference [12], a smart mobility application that supports driving by projecting virtual objects in the real world is developed. Examining the comfort of driving when using AR (or MR) shows overall good evaluation results and indicates that AR (or MR) is useful in the field of traffic engineering.

8. Conclusions

In this study, we proposed the use of microservice architecture for smart mobility applications. Using this architecture, the functions of the application can be divided as a services, so that each function can be easily managed and updated according to the life cycle of each service. We developed the bus timetable recommendation application as a smart mobility application based on microservice architecture. Results of examining whether the features of microservice architecture are satisfied through this application show that it is fully suitable to develop smart mobility application.

As future work, we plan to develop a large-scale project. The application developed in this research was a small-scale prototype, and we could not examine its influence for long-term operation. Therefore, a large-scale development is important to confirm its long-term operability. In addition, we will have to consider the communication of large capacity data. In the current system, because communication between services is performed by a simple method, only communication with lightweight data is assumed. However, large capacity data such as map data will be required when developing smart mobility applications. We will continue to discuss how this is the most suitable method for the development of smart mobility applications.

Acknowledgment. This work is partially supported by JSPS KAKENHI Grant Number JP15H05708.

References