Proposing Tap-Track Platform for Facilitating Implementation of Button-Driven Smart Services

Ryota Murate
Faculty of Engineering,
Kobe University,
1-1 Rokkodai-cho, Nada,
Kobe, Japan
rmurate@es4.eedept.kobe-u.ac.jp

Tomorou Nakahashi
Graduate School of Engineering,
Kobe University,
1-1 Rokkodai-cho, Nada,
Kobe, Japan
tomorrow@ws.cs.kobe-u.ac.jp

Sinan Chen
Graduate School of Engineering,
Kobe University,
1-1 Rokkodai-cho, Nada,
Kobe, Japan
chensinan@gold.kobe-u.ac.jp

Sachio Saiki
School of Data and Innovation,
Kochi University of Technology,
185 Miyanokuchi,
Kochi, Japan
saiki.sachio@kochi-tech.ac.jp

Masahide Nakamura [1,2]
[1]*Kobe University*,
[2]*RIKEN Center for Advanced Intelligence Project*,
Tokyo, 103-0027, Japan
masa-n@cs.kobe-u.ac.jp

Kiyoshi Yasuda
Graduate School of Engineering,
Kobe University,
1-1 Rokkodai-cho, Nada,
Kobe, Japan

Yuta Tuyuzaki
Oyumino Central Hospital,
6-49-9 Oyumino Minami,
Chiba, Japan

***Abstract*—In recent years, the movement toward a super smart society as a future society is being promoted as Society 5.0. In this movement, smart services [1] have become very diverse, and the burden on users has increased. Therefore, in this paper, we focus on button-driven smart services, and propose a platform "Tap Track" that makes it easy to implement button-driven smart services. In Tap Track, users can create services that run when a button [2] is pressed with no code, and users can assign their desired services to the button. The processing that can be executed when the button is pressed is playing media files, sending commands to external services, and so on. The usage history of the button can be looked back later as a life log. In the case study, we created a weather forecast confirmation service and a life log service using Tap Track and verified its operation.**

***Index Terms*—button-driven smart service, life log, no-code development**

I. Introduction

A super-smart society is defined as a society that "provides the necessary goods and services to the necessary people, at the necessary time, and in the necessary amount, responds in detail to society's various needs, allows everyone to receive high-quality services, and enables people to live vibrantly and comfortably, overcoming various differences such as age, gender, region, and language." A series of initiatives toward achieving this goal are being promoted under the name "Society 5.0." [3] Such a modern society, a wide variety of services have been born, and it is common to use services on smartphone and PC apps. However, this has led to the problem of interface diversification, increasing the burden on users.

The goal of this study is to propose a button-driven smart service, which refers to services that the user can call by simply pressing a button. As related technologies, there are Amazon Dash Button and AWS IoT Button, but the implementation has constraints and the difficulty is high, which is a problem. Therefore, in this study, we propose a button-driven smart service development and execution platform "Tap Track" with the aim of facilitating the implementation of button-driven smart services. Therefore, the requirements to be met are defined as follows. R1: Being able to perform daily, repetitive actions with the push of a button, R2: Being able to easily and easily create your own favorite services,

To meet the requirements, the overall architecture of Tap Track is composed of the following four elements. A1: Button function generation, A2: Button function assignment, A3: Drive by button, A4: Look Back,

Users can create button functions, assign them to buttons, execute services by pressing buttons, and look back execution history. In the case study, we create a "weather forecast confirmation service" and a "life log service" as specific examples of the implementation and execution of button-driven smart services using Tap Track. This is expected to be effective in supporting the implementation of various button-driven smart services.

II. Preliminaries

A. Realization of a "Super Smart Society" and "Society 5.0"

According to the 5th Science and Technology Basic Plan of the Smart Cabinet Office [4], a super-smart-society will

"provide the necessary goods and services to the necessary people, at the necessary time, and in the necessary amount," and "respond in detail to society's various needs." It is defined as a society in which all people can receive high-quality services, overcome various differences such as age, region, and language and live vibrantly and comfortably. In this society, it is expected that people and robots/AI will coexist to improve the quality of life, provide customized services that respond to the diverse needs of users in detail, provide services that anticipate potential needs and support human activities, eliminate service disparities due to region and age, and create an environment where anyone can become a service provider. By integrating cyberspace and physical space, the movement toward a super-smart society is shared as the future society, and a series of efforts to realize it are further evolved and promoted as "Society 5.0".In the current era, which can be called the so-called Fourth Industrial Revolution, many services have been born to support people's lives. [5]The challenge is how easily you can access the various services.

B. Current Status of Service Usage

Current services are commonly used on smartphone and PC apps. There are two issues to be solved here. The first is that the interface is diversified. For example, when using a service to operate home appliances and a service to play music, you need to install, start, and operate each app, and you need to go back and forth between the two apps. This has increased the burden on users to access services. The second is that there are cases where interactive operations are not necessary for using the service. When considering operating home appliances, operations such as turning on, turning off, starting, and stopping do not require interactive operations that require the user to enter data.

C. Expectations for Button-Driven Services

Here, "button-driven smart services" refers to services that can be called up with the simple operation of pressing a button. Button-driven smart services allow you to quickly use services with simple actions and are convenient for simplifying complex operations. This makes it easier for new users to use the service and allows them to use the service without any hassle. Furthermore, button-driven smart services are an effective control method, especially when combined with IoT devices. [6]From these characteristics, it can be said that the demand for button-driven smart services is increasing.

D. Related Technologies

The Amazon Dash Button [7] is a button-driven smart service provided by Amazon. By pressing this button, you can easily order specific products. It is mainly used for ordering consumables, making it a convenient tool for quickly replenishing everyday items, but it is not intended for executing services that you want to use constantly.

The AWS IoT Button [8] is a button-driven service provided by Amazon that allows you to execute AWS Lambda functions with the press of a button. This feature enables a variety of actions to be triggered, but since it requires specialized knowledge for configuration and use, it is primarily a tool for professionals.

IFTTT (If This Then That) [9]is a tool that allows you to customize and combine existing apps and services to meet your needs. Users can combine existing services offered by IFTTT, but cannot create new applications themselves. It is primarily intended for automating processes by combining existing services and apps.

STREAM DECK [10] is a device that allows you to assign frequently used functions and actions on your PC to buttons. Users can use this device to quickly execute many operations, but it is also for utilizing existing functions rather than creating new applications.

A common issue with these tools is that their available functions are limited to existing provided features. Additionally, the distribution of these platforms presents another challenge.

E. Research Focus in This Study

In this study, we focus on the problem that the implementation of button-driven smart services is difficult. Button-driven smart services are a simple mechanism in which a service is executed when a button is pressed, but for developers, the implementation cost is about the same as for normal services. When a developer creates a new service, they need to implement functions such as detecting that a button has been pressed and processing that is executed when a button is pressed by securing computing resources, building servers, developing front-end UIs, developing server-side code, designing databases, and deploying services. Furthermore, coding knowledge is essential for this.

III. Proposed method

A. Purpose and Requirements

As a study on platforms that facilitate the implementation of button-driven smart services, we propose a platform that allows you to register, implement, and use button-driven services without specialized knowledge. The requirements to be met are as follows. R1: Being able to perform daily, repetitive actions with the push of a button, R2: Being able to easily and easily create your own favorite services,

B. Approach

To meet this requirement, we propose a button-driven smart service implementation platform "Tap Track". Tap Track is a key idea to provide a GUI service that allows you to create your favorite button-driven service with no code, and to create a button press detection function and processing that is executed when a button is pressed with no code.

C. Overall architecture

Figure 1 shows the overall architecture with Tap Track introduced. Tap Track consists of the following elements. A1: Button function generation, A2: Button function assignment, A3: Drive by button, A4: Look Back,

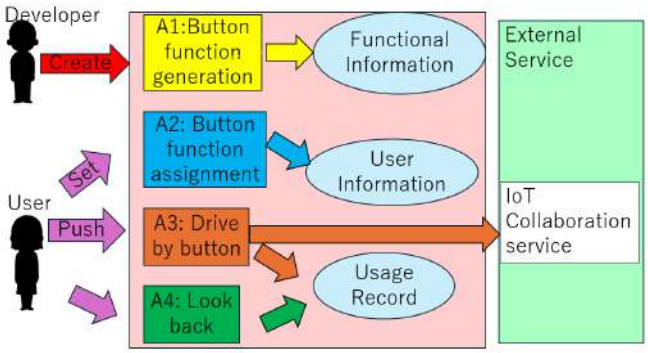

Fig. 1. System architecture.

Fig. 2. Setting button behavior form.

Service developers refer to people who implement services on Tap Track. By A1: Button function generation, service developers create the button function they want to assign to the button and register it in the button function information database. Users refer to people who use services on Tap Track. Users select the required button function from the button function information database and save it in the user information database by A2: Button function assignment. Then, by A3: Drive by button, the button function saved in the user information database is called from the button function information database and executed when the button is pressed. When the button function is executed, “who”, “when”, and “which button function” are saved, and by A4: Look Back, users and developers can check the execution history of the service.

1) A1: Button function generation: Here, users create button functions that support daily life and want to assign to buttons. The following settings can be done with no code. S1: Definition of button function behavior, S2: Reaction to pressing the button, S3: Sending an HTTP request, S4: Button function publication range,

S1: Regarding the definition of the behavior of the button function, simply saying that it is operable with just one button does not mean that there is only one behavior required. Specifically, in the case of a button function that displays the weather forecast when the button is pressed, you only need to perform the same behavior each time you press the “Get weather image” button, but in the case of a button function that turns on the electricity, you need two behaviors: “Turn on the power” and “Turn off the power”. Such behavior is set in the following three types. S2: Regarding the reaction to pressing the button, if there is no feedback to the user even if the button is pressed, the user does not know whether it was pressed or not, and there is a possibility that it will be pressed multiple times. Such a design should be avoided. Therefore, the reaction can be set to play media files. In the proposed method Tap Track, you can select the setting method from the following four. S2-1: Upload media files, S2-2: Reference URL, S2-3: Read Text, S2-4: None,

S3: Regarding sending an HTTP request, this setting is for setting up cooperation with external services. S4: Regarding the publication range of the button function, this setting is to set the range in which the button function is published. The button function created by the user can be shared. With this setting, anyone can become a developer. The setting can be selected from the following three. S4-1: Private, S4-2: Public, S4-3: Limited. Each button function is assigned an ID, and in the case of limited release, only those who know the ID can use it.And, when the publication setting is set to ”public,” the button functionality is made available to the entire world, allowing it to meet the needs of an unspecified number of users.

2) A2: Button function assignment: Assign the button function created in A1 to each button of the button-driven service. Users can call up their favorite services from the button. You can assign your own button function and button function that others have set the publication range to “public” to the button. Limited release button functions can be assigned to buttons by searching using the ID.

3) A3: Drive by button: When you press the button, the button function assigned by A2 is executed. The button has a “state”, and when the behavior set in A1 is “button”, the state does not change even if the button is pressed. When the user presses the button, the button function assigned to the service is executed. The screen becomes the user’s personal button.

4) A4: Look Back: Users can check which button they pressed when and how many times. It is displayed in the following three types of graph formats. The bar graph shows when and how many times the button was pressed. The pie chart shows the time and proportion of the button state that continued within the specified period. The timeline shows when and which state button was pressed on the timeline.

IV. IMPLEMENTATION

A. Used Technologies

In this paper, we implemented the proposed button-driven smart service “Tap Track”. The backend was implemented using the Java [11] language and Spring Boot [12]. MySQL

Fig. 3. Setting media file form.

was used as the database server, Apache Tomcat was used as the web server, and the front end was implemented using HTML, SCC, JavaScript, and google charts.

B. Usage and Sample Description

Figure 2, Figure 3, and Figure 4are the button function creation screens. In Figure 2, you can select how to set the behavior of the button function and the media file to be played as a reaction. In Figure 3, you enter the button function name, the media file to be played as a reaction, and the description of the button function. In Figure 4, you set the HTTP request sent by the button function.

Figure 5 is the button function selection screen. This screen displays the button functions you created and the button functions that others have set the publication range to "public". By entering the ID assigned to each button function in the "Search by button function ID" field, you can assign button functions that other users have added as "limited release" to the button.

Figure 6 is the button screen that actually drives the service.Buttons $0 \sim 10$ are displayed, and you can drive the service by pressing the button.

V. Performance Evaluation

To evaluate how much easier Tap Track makes the creation of button-driven smart services, we compared the time required to create a service with and without using Tap Track.

The service in question is an attendance management system where pressing a button once records clocking in, and pressing it again records clocking out.

First, when creating the service without Tap Track using the conventional method, seven steps are required: deciding on database design, setting up the environment including writing a Dockerfile, creating a backend for API handling, creating a user addition form, displaying an attendance record graph,

Fig. 4. Setting HTTP request form.

deploying for external access, and configuring the STREAM DECK to be used as the button. When the author actually carried out these steps, it took 32 minutes for decision-making and environment setup, 2 hours and 36 minutes for creating the backend, 15 minutes for creating the user addition form, 1 hour and 20 minutes for displaying the graph, 16 minutes for deployment, and 6 minutes for configuring the STREAM DECK, totaling 5 hours and 5 minutes.

On the other hand, when using Tap Track, it was completed in just 1 minute by simply following the steps and entering the necessary information using the GUI.

VI. Case Study

A. Application Overview

In the case study, we use Tap Track implemented in 4. to create a button-driven smart service with the purpose of creating a button-driven smart service.

B. Weather Forecast Confirmation Service

1) A1: Create button function: First, move to the button function creation screen. Since the button function is the same

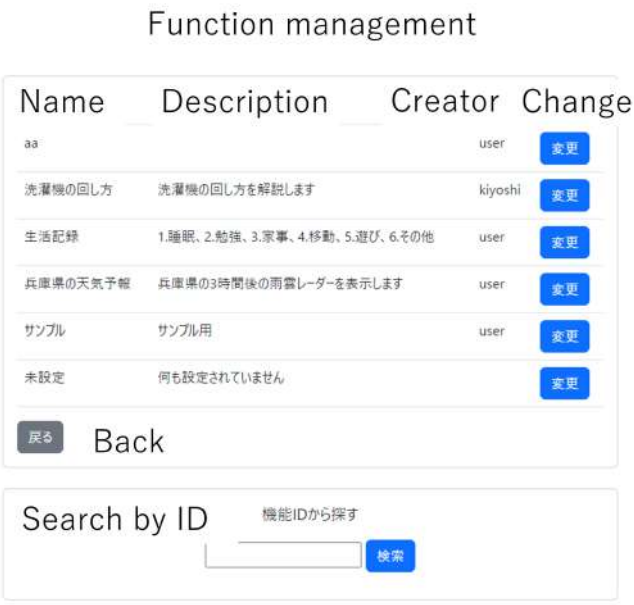

Fig. 5. Button function selection form.

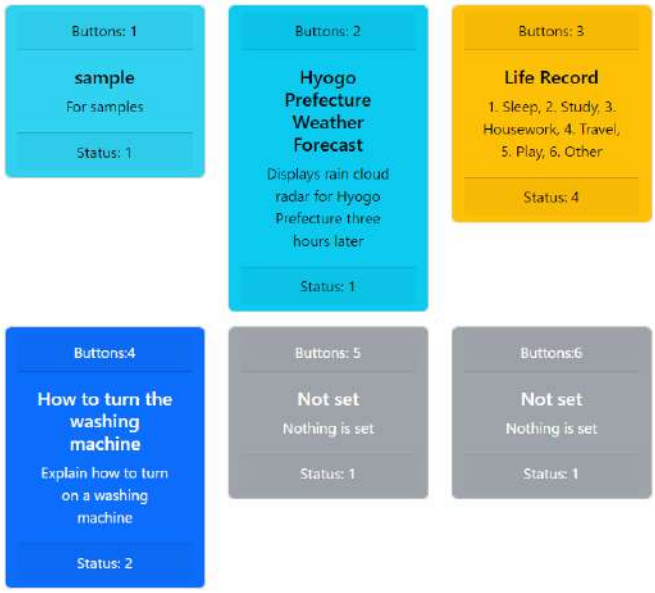

Fig. 6. Button screen.

operation of “Get weather image” every time, the behavior is set to “Trigger”. When you want to set the media file to be played as a reaction as the weather forecast image, the setting method of the media file to be played as a reaction is to use an external service, so select “Reference URL”. Next, enter the button function name, the media file to be played as a reaction, and the description of the button function. The button function name is “Hyogo Prefecture Weather Forecast”. This time, the weather forecast image uses the image of the rain cloud radar 3 hours later provided by tenki.jp [13].

Fig. 7. Weather forecast image.

2) A2: Assign button function: Next, assign the button function to the button. This time, we will assign the button function to button 2. Select button 2 from the button function setting screen and select “Hyogo Prefecture Weather Forecast” created earlier from the list of button functions displayed.

3) A3: Drive by button: When you press button 2, the weather forecast image is displayed as shown in Figure 7.

C. Life Log Service

Create a button function that allows you to keep a life record by pressing a button. In particular, when you finish studying, turn off the power of the desk lamp.

1) A1: Create button function: First, move to the button function creation screen. Set the recording items to “sleep”, “study”, “housework”, “play”, and “other” for a total of 6 items, so set the behavior to “Loop 6”. Set the setting method of the media file to be played as a reaction to “Upload”. Next, enter the button function name, the media file to be played as a reaction, and the description of the button function. The button function name is “Life Record”. Upload a simple sound effect mp3 file as the media file. In the description field, describe how to record which state in a way that is understandable. (1. Sleep, 2. Study, 3. Housework, 4. Movement, 5. Play, 6. Other) Finally, use Switch Bot [14] [15] to turn off the power of the desk lamp when you finish studying. In the screen of Figure 4 HTTP request sending settings, set “Sending an HTTP request”. Get the device ID and token from the Switch Bot app and set it as follows (TABLE I).

2) A2: Assign Button Function: Next, assign the button function to the button. This time, we will assign the button function to button 3.

3) A3: Drive by Button: When you press button 3, the number of states changes. The state changes each time you press the button, and when it is 6, it returns to 1 when you press it next. By pressing the button to match the number of states with the behavior you want to record, you can keep a

TABLE I

HTTP REQUEST SETTINGS.

URL	https://api.switch-bot.com/v1.0/devices/{deviceID}/commands/
Method	POST
Header	Authorization: {token}
Body	{ " commandType " : " command " , " command " : " turnOff " , " parameter " : " default " }

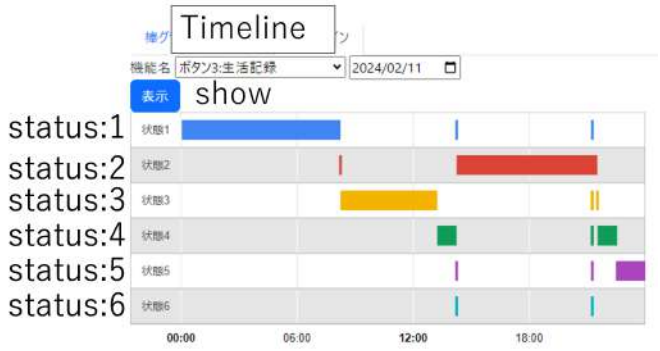

Fig. 8. Shown timeline.

life record. For example, when you want to record sleep, press the button until the state of button 3 becomes 1.

4) A4: Look Back: The life record you made can be checked later as shown in Figure 8.The state corresponds to the following. (1. Sleep, 2. Study, 3. Housework, 4. Movement, 5. Play, 6. Other)

D. Discussion

By creating a button-driven smart service using the proposed method Tap Track, the operation of the weather forecast confirmation service and the life record service was confirmed. By using Tap Track, it is expected that the difficulty of implementing button-driven smart services can be solved because button functions can be created without code. And, by using Tap Track, it is possible to implement and use various services on one platform, and it is expected that the diversification of interfaces can be solved. However, it was found that Tap Track is limited to button input being limited to operations on a web browser. It is necessary to click on the screen to operate the button, which is not sufficient. For example, it is considered that it will be possible to respond to various smart services by enabling various methods such as remote operation from a smartphone and operation by a motion sensor, but this is a future issue.

VII. LIMITATION

This study is preliminary, focusing on the implementation and proposal of a platform to facilitate the development of button-driven smart services, as well as a preliminary evaluation of development performance. A limitation is that we could not thoroughly assess whether Tap Track meets users' needs adequately. Future research should involve longer-term data collection and detailed evaluation. Specifically, it is necessary to assess user needs more accurately through usability tests and user satisfaction surveys. This is expected to provide deeper insights into the effectiveness of Tap Track.

VIII. CONCLUSION

In this paper, we proposed Tap Track as a platform that makes it easy to implement button-driven smart services, with the aim of proposing a platform that allows you to register, implement, and use button-driven services without specialized knowledge. Users can implement services on this platform without code. Tap Track is a platform that provides a GUI as a foundation for executing button-driven smart services and makes it easy to implement button-driven smart services. In addition, users can select the necessary services according to their needs, assign them to buttons themselves, execute services by pressing buttons, and check the execution history of services. In this study, we created a weather forecast confirmation service and a life record service as a case study and confirmed the operation to demonstrate the usefulness of the proposed method Tap Track. Future issues include diversification of input at the time-of-service driving and evaluation based on actual use.

ACKNOWLEDGMENT

This research was partially supported by JSPS KAKENHI Grant Numbers JP20H05706, JP22H03699, JP22K19653, JP23H03401, JP23H03694, JP23K17006.

REFERENCES

[1] M. M. Daniel Beverungen, Oliver Müller, "Conceptualizing smart service systems," in *Electronic Markets*, vol. 29, no. 16, 2019, pp. 7–18.
[2] R. Plotnick, "At the interface the case of the electric push button, 1880–1923 ," in *PROJECT MUSE*, vol. 53, no. 4, 2012, pp. 815–845.
[3] M. M. Nair, A. K. Tyagi, and N. Sreenath, "The future with industry 4.0 at the core of society 5.0: Open issues, future opportunities and challenges," in *2021 International Conference on Computer Communication and Informatics (ICCCI)*, 2021, pp. 1–7.
[4] C. Office, "The 5th science and technology basic plan," https://www8.cao.go.jp/cstp/kihonkeikaku/5honbun.pdf, 2016.
[5] S.-Y. CHOU, A. DEWABHARATA, and T. H.-K. YU, "From industry 4.0 to the fourth industrial revolution," *Journal of information and management*, vol. 38, no. 1, pp. 14–25, 2018.
[6] T. Nakahashi, S. Chen, and M. Nakamura, "Study of service to assist platform deployment of heterogeneous iot," in *2022 23rd ACIS International Summer Virtual Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Summer)*, 2022, pp. 48–55.
[7] Amazon, "Amazon dash terms of use," https://www.amazon.co.jp/gp/help/customer/display.html?nodeId=201730770, 2023, (Accessed on 6 February 2024).
[8] AWS, "Aws iot 1-click (one-click creation of aws lambda triggers) — aws," https://aws.amazon.com/jp/iot-1-click/, 2023, (Accessed on 6 February 2024).
[9] IFTTT, "Ifttt," https://ifttt.com/, 2023, (Accessed on 23 July 2024).
[10] Elgato, "Stream deck," https://www.elgato.com/en/stream-deck, 2023, (Accessed on 23 July 2024).
[11] O. Corporation, https://www.java.com/, java.
[12] VMware, "Spring boot," https://spring.io/projects/spring-boot, 2024.
[13] J. M. Agency, "tenki.jp," https://tenki.jp/, 2024, (Accessed on 6 February 2024).
[14] SwitchBot, "Switchbot (official site)," https://www.switchbot.jp/, 2024, (Accessed on 6 February 2024).
[15] ——, "Switchbot api," https://github.com/OpenWonderLabs/SwitchBotAPI, 2024, (Accessed on 23 July 2024).