WAC 2014 1569909201

Extension of HNS-VAUI for Personal Adaptation from Human-Sensed Data

Hiroyasu Horiuchi, Kouhei Takahashi, Seiki Tokunaga, Sachio Saiki, Shinsuke Matsumoto and Masahide Nakamura
Kobe University
1-1 Rokkodai-cho, Nada-ku, Kobe, Hyogo 657-8501, Japan
Email: {horihori, koupe, tokunaga}@ws.cs.kobe-u.ac.jp, sachio@carp.kobe-u.ac.jp, {shinsuke, masa-n}@cs.kobe-u.ac.jp

***Abstract*—Research and development of home network system (called HNS.) is recently a hot topic in the area of ubiquitous computing applications. Orchestrating house-hold appliances (e.g., TVs, DVDs, speakers, air-conditioners, lights, curtains, windows, et al.) via the network, the HNS provides value-added services for home users. In our previous research, we have proposed and developed HNS-VAUI that is the interactive voice user interface using the virtual agent as a HNS user interface. The HNS-VAUI provides a useful interaction in HNS operation for users. However, in some case the proposed system usability declines, because which doesn't have feature to adopt user's personalization. Users don't only have individual home-appliance, but also they have individual favorite interaction via their HNS-VAUI. In this paper, we propose extended HNS-VAUI at the view point of personalization. Concretely speaking, HNS-VAUI can adapt to each user by learning personal preference and behavior rule. In addition to this, we conduct case study to confirm availability.**

***Keywords*—Home Network System, Virtual Agent, Personalizing, Human Agent Interface**

I. Introduction

In recent years, with the development of network technology, home network systems (HNS)[4] have been developed, which is a framework of ubiquitous application in the next generation to achieve high value-added services connecting networked home appliances and sensors. Improving the usability of the HNS user interface is an important issue in the HNS. In the HNS environment, users operate their appliances on everyday base, so the interface of HNS strongly required "easy learning" and "simple operation". Virtual agent user interface (VAUI), which is using virtual agent as an interface, is fulfill all the conditions of HNS interface. In the interaction with the agent on the screen, user can operate the system.

In our research group, we have proposed new HNS environment exploiting VAUI as a HNS user interface (HNS-VAUI)[11]. And also we have realized HNS-VAUI using MMDAgent[7] with the concept of service-oriented architecture (SOA)[10] in our HNS environment (called SO-HNS-VAUI)[3]. HNS-VAUI gives feedbacks to users with agent's motions as well as agent's voice. Owing to these feedbacks, users can enjoy to operate the HNS through more natural conversations with the agent. As a more important point of HNS interface, personal and environment adaptability are described. To generalize the application and combination of HNS has a great difficulty in practical situation because the target appliance and service of HNS lies on variety and difference. Furthermore, indivisual lifestyle habit and preference such as temperature of air conditioning or Bedfordshire are different from each other even if they are families. From these backgrounds, the adaptation and personalized interaction of HNS user interface is an inescapable problem to realize a comfortable user interface. Unfortunately, the current SO-HNS-VAUI cannot adjust a personalized interaction due to limits of the system. In order to implement personalized interaction, personal adaptation based on learning personal preference and behavior pattern is required. After this implementation, the interaction management to provide appropriate service corresponding to each user is also required, because the users of HNS-VAUI are every person in the home.

In this paper, we extend the current SO-HNS-VAUI system, in order to generate the personalized interaction and provide appropriate service corresponding to each user. By applying this extension, suitable interaction that is based on learning personal preference and behavior pattern from house log data is automatically generated and added to HNS-VAUI. Besides distinguishing the current active user with personal identification technology, HNS-VAUI can provide appropriate service corresponding to this user.

II. Preliminaries

A. Home Network System (HNS)

The home network system consists of a variety of household appliances (e.g., room light, television), and sensors (e.g., temperature, brightness). The appliances and sensors are connected via a network. Each device has control API, which allows users or external agents to control the device over the network. Application services include personal home controllers, remote monitoring/controls, appliance orchestration, energy saving, context-aware services, et al. In our research group, we have implemented an actual HNS environment, called CS27-HNS [8]. Introducing the concept of SOA, CS27-HNS integrates heterogeneous and multi-vendor appliances by standard Web services. Since every API can be executed by SOAP or REST Web service protocols, it does not depend on a specific vendor or execution platform. For example, an implementation of changing the TV channels to 6 is to access the address ''http://hns/TVService/setChannel?channel=6''.

B. Virtual Agent (VA)

Virtual agent (VA) is classified into the user interface, which is displayed on the screen and act like a real agent

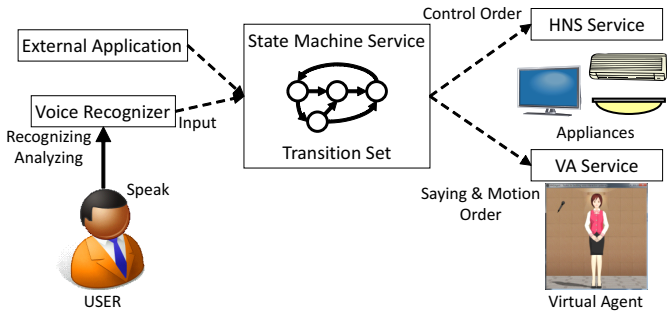

Fig. 1. SO-HNS-VAUI Overview

through the interaction with the user. The VA technology, provide realistic interaction and a sense of intimacy, which have received a lot of attention in recent years as a next generation user interface instead of conventional voice interface[9][1]. The methodology of making original VA, MMDAgent[7] is a typical software. MMDAgent is a powerful tool kit for implementing a VA spoken dialogue system and it is very popular software in Japanese subculture. We can produce various agents by changing the 3D models. The behaviour of the agent is defined by a script, which specifies motion, movement, camera work, speech synthesis, speech recognition, lip sync, application invocation, according to a state transition model with events and actions. Our MMDAgent system can be controlled through Web API by using socket communication plugin[2]. For example, in order to speech "Hello" by the agent, we just access to the URL ''http://../VAService/say?str=Hello''.

C. HNS-VAUI

HNS-VAUI is one of the HNS environments, which has the VA user interface to operate HNS services. HNS users can operate appliances via natural interaction just like a "talk with a friend" instead of unnatural one-way speaking through the use of HNS-VAUI. Additionally, HNS-VAUI avoid lengthening of feedback, because it can provide not only aurally and visually feedback using agent's voice and motion but also visualized information. By using various feedbacks, HNS-VAUI always provides user-friendly feedback even if the amount of information becomes so large.

III. Service Oriented HNS-VAUI

A. Overview

In previous research[3], we have implemented HNS-VAUI with the concept of SOA, in order to apply the HNS-VAUI to CS27-HNS, and realize for flexible cooperation between HNS-VAUI system and other application. Fig.1 shows the current HNS-VAUI system, named Service Oriented HNS-VAUI (SO-HNS-VAUI). SO-HNS-VAUI is composed of the following three layers. The first layer is the input-layer that input to SO-HNS-VAUI system, including Voice Recognizer. The second layer is the process-layer, that is, State Machine Service (SM Service). SM Service is a Web Service that receives the input and determines the content of the feedback. The last layer is the output-layer, which is composed of Web services that control HNS and agent. These Web Services in output-layer gives feedbacks to users. Each layer is connected by Web API. We used Julius[6] as a voice recognition engine.

B. State Machine Service

This section explains about SM Service that is the most important in SO-HNS-VAUI. SM Service is a Web service that manages the system and the conversation with the user. SM Service realizes the interactive interface between system and user. In order to manage the interaction, SM Service provides stateful feedback in response to external input by using a transition set. SM Service gives feedbacks by executing the Web service, such as HNS Services to control the home appliances and VA Service to control the agent behavior. Not only Voice Recognizer, but also other services can input the condition to SM Service, so SM Service can provide interaction that is triggered by external applications.

This transition set is constituted by many state transitions. These transitions are composed five elements of *transition ID*, *Current State*, *Next State*, *Condition* and *Action*. *Current State* and *Next State* are state of SM Service's transition set. *Conditions* is a condition for making the transition of the transition set. *Action* contains one or some Web service, which is executed by SM Services to give feedbacks when the transition occurs.

C. System Operation

The following is how SO-HNS-VAUI realizes the interaction. First, external applications inputs the condition to SM Service via the Web API. Then SM Service selects a pertinent transition of which condition is the same as inputted condition. Next, SM Service transitions the state of the transition set and execute registered action according to the selected transition. Finally, SM Service gives feedbacks and operates the home appliances by executing external Web Services such as VA Service and HNS Service.

D. Problem

Each user has a HNS environment of their own, so home appliance that is installed is different from each HNS environment. The configuring when operating the home appliances (e.g., Temperature of the air conditioner, Strength of the wind fan, et al.) is also different from each user. In addition to these differences, each user has favorite interaction, such as cute behaviors of their favorite agent. In order to meet these requirements that are specific to each user, HNS-VAUI should be able to individualize adapt.

To realize personal adapted HNS-VAUI, it is necessary that the HNS-VAUI edits the interaction automatically by mining for preference and behavior of the user from the usage of day-to-day. And it would be required that HNS-VAUI manages interaction for each user, in order to provide a suitable interaction to the appropriate user in an environment where there are a plurality of users, such as in-home. However, SO-HNS-VAUI is only for single user, and individual adaptation is not considered in the SO-HNS-VAUI. Therefore, SO-HNS-VAUI can't edit the interaction dynamically based on learning and provide suitable interaction for each user. This yields the following problems in SO-HNS-VAUI.

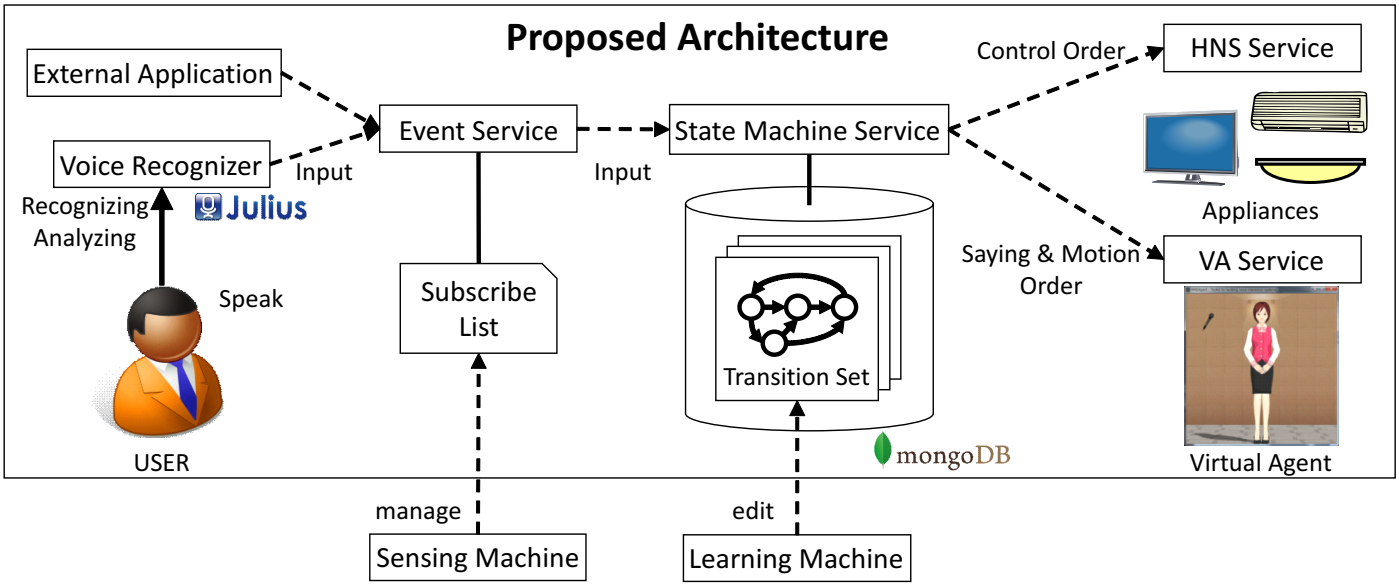

Fig. 2. Proposed System Overview

- **P1 : Dynamic manage of interaction**
  SO-HNS-VAUI manages the interaction by using the transition set. In the current system, it is not possible to add or modify transitions.
- **P2 : Switching of the transition sets**
  Since SM Service receives an input from external applications as an input to the single transition set, SO-HNS-VAUI has just only one transition set that manages interaction.

IV. Personalizable SO-HNSVAUI

In this section, we propose the Personalizable SO-HNSVAUI (PSO-HNS-VAUI) which solves P1 and P2 problems shown in the previous section. PSO-HNSVAUI is based on following two concepts :

- **R1 :** Implementing control functions as a API, in order to edit transition from external application.
- **R2 :** Enable to manage multiple transition set in order to switch interaction for each user.

In the following sections, we explain three key ideas to realize R1 and R2 concepts.

A. Dynamic editing of transition

SO-HNS-VAUI handles the own transition set. Therefore, external application can't modify the transitions in transition set. So, we managed the transitions with DB, in order to easily modify the transitions via the Web API from external application. In consequence, it will be possible to provide the interaction, which is personalized based on learning, at any time.

B. Extension of transition data

We added a new element, named *Set Name*, to the five elements that constitute the current transition. *Set Name* indicates that the transition belongs to which transition set. And SM Service controls the state of each transition set. Thereby, SM Service manages multiple transition set. Also, we extend the current API to suit this extension of SM Service. Hence, SM Service can provide appropriate interaction to each user by managing multiple transition set.

C. Determination of the transition

Multiple transition set management can be realized by the extension of SM Service shown in previously. However, an input target for transition set have to be assigned in order to use suitable transition set for each user. To implement this assignment function, we attach a new application **Event Service** on SO-HNS-VAUI. Event Service is a Web Service Application placed in between input and SM Service. When Event Service receives input data, Event Service subscribes the input for registered external applications. The external application management is one of the Event Service functions and it can be edited the list of the application via Web API.

As a result of these extensions, PSO-HNS-VAUI ready for personalizing by way of external interaction management. Through the adapting some learning method for the system, we will get the personalized interaction and appropriate service corresponding to each user.

D. Implement

Based on IV, we have extended current system. Fig.2 shows system overview. The following is system operation flow. When a user orders an operation, the Voice Recognizer analyzes the operation and generates a condition according to analyzing the result. And Voice recognizer inputs the condition to Event Service. Then Event Service determines a transition

TABLE I. EXAMPLE OF TRANSITION

Transition ID	Set Name	Condition	Current State	Next State	Action
T0000	Mike	ACTIVATE	idle	active	SAY:what_can_i_do_for_you, DO:SMILE
T0001	Mike	DISACTIVATE	active	idle	SAY:bye, DO:BYE
T0002	Emily	ACTIVATE	idle	active	SAY:may_i_help_you, DO:GREETING
T0003	Emily	DISACTIVATE	active	idle	SAY:see_you_soon, DO:BYE
T0004	Emily	TV_ON	active	active	SAY:i_turn_on_the_tv, HNS:TV_ON
T0005	Emily	TV_OFF	active	active	SAY:i_turn_off_the_tv, HNS:TV_OFF
T0006	Mike	LIGHT_ON	active	active	SAY:i_turn_on_the_light, HNS:LIGHT_ON
T0007	Mike	LIGHT_OFF	active	active	SAY:i_turn_off_the_light, HNS:LIGHT_OFF
T0008	Emily	RECOMMEND_FAN	idle	idle	SAY:it's_hot_i_turn_on_the_fan HNS:FAN_ON
T0009	Mike	RECOMMEND_FAN	idle	recommend fan	SAY:shall_i_turn_on_the_fan
T0010	Mike	YES	recommend fan	idle	SAY:i_turn_on_the_fan, HNS:FAN_ON
T0011	Mike	NO	recommend fan	idle	SAY:i_do_not_turn_on_the_fan
T0012	Emily	GOOD_MORNING	idle	idle	SAY:good_morning_emily, DO:SMILE

set to input by referring to the list, and inputs the condition to the transition set. Next, SM Service selects pertinent transition of which the condition is the same as the inputted condition from the specified transition set. And SM Service changes the state of the transition set and execute registered action according to the selected transition.

We explain how to realize these requirements in IV, in the following sections.

1) Management of the Transition: In this part, we explain how to manage transitions. We used MongoDB for managing transitions. Table I shows some transition. Transition is composed of new element *Set Name* and other five elements. For example, a transition, of which *Transition ID* is T0000, means the following :

- This belongs to a transition set for Mike.
- This is a transition from state *idle* to state *active*.
- This is caused by a condition ACTIVATE.
- This has two *action*
  - The agent speaks "What can I do for you?".
  - The agent performs "SMILE" motion.

Besides, we implemented Web APIs that add or modify transition. For example, in order to add other actions (e.g., to open curtain) to *action*, of which *Transition ID* is T0012, to access the address ''http://../SMService/addActionToTransition?transitionID= T0015&Action=HNS:CURTAIN_OPEN''. In addition, there are other APIs such as `addTransition` that adds a transition to the database, `editTransition` that changes transition's information. Thus, we can add or modify transition via Web API, and so we can manage interaction from external application.

2) Switching transition set: In this part, we explain how to switch transition set. We have added an argument, which specify the transition set, to the current input API of SM Service. For example, to access the address ''http://../SMService/input?str=TV_ON&setName=Emily'' means that *condition* "TV_ON" is inputted to the transition set for Emily.

In addition, we have implemented the Event Service. We explain the flow of accessing the above address with Event Service. First, access the following address so as to add URL which inputs to the transition set for Emily to Event Service's list. ''http://../EventService/addURL?url=http://../SMService/input?str=%keyword&setName=Emily''. As a result, the address ''http://../SMService/input?str=%keyword&setName=Emily'' is added to the list. Next, external application (mainly Voice Recognizer) access the address ''http://..

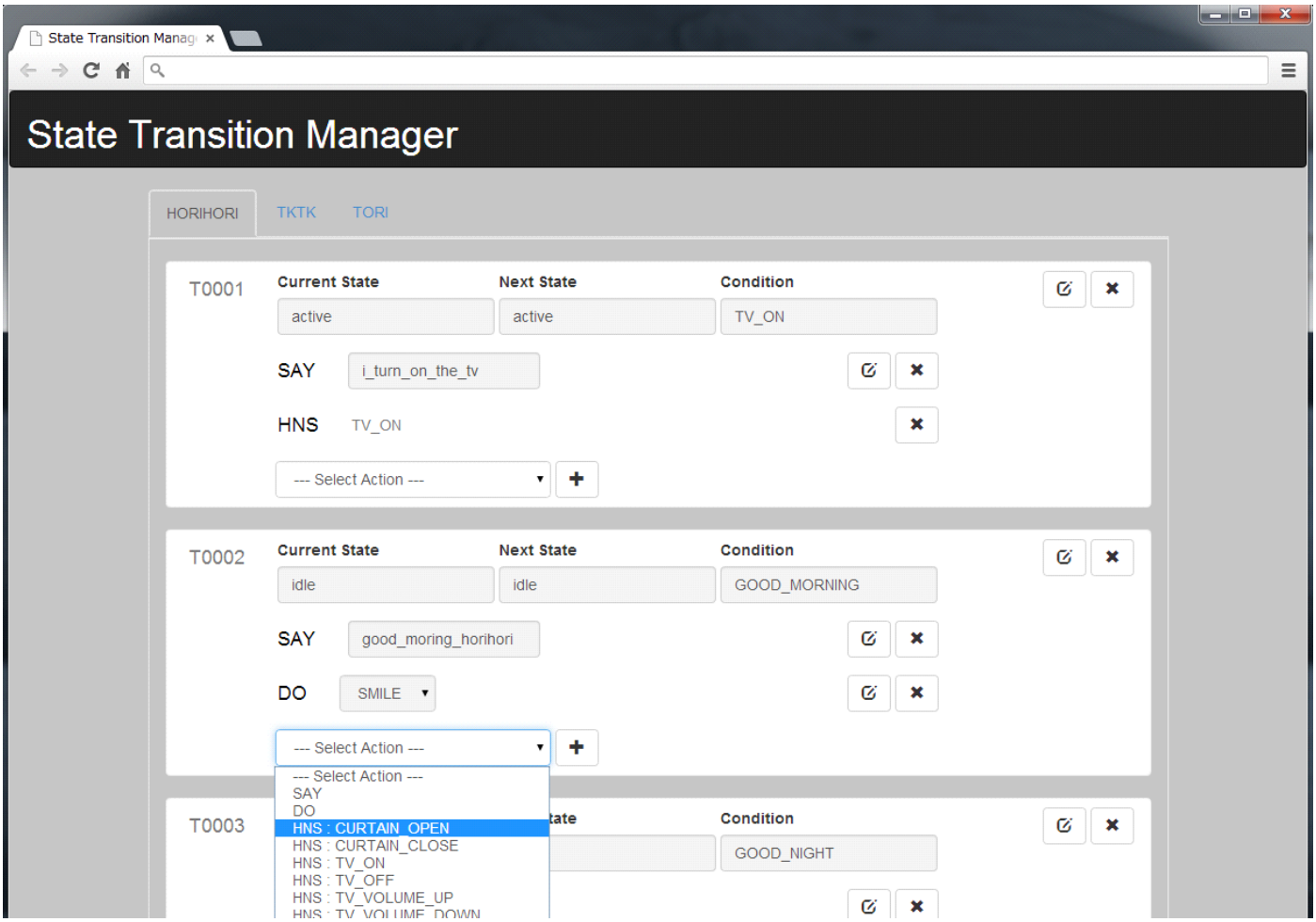

Fig. 3. Screenshot of "State Transition Manager"

/EventService/update?keyword=TV_ON''. Then, Event Service replaces the string of "%keyword" in address of all in the list with "TV_ON", and access each replaced address like this address ''http://../SMService/input?str=TV_ON&setName=Emily''. Thus, Event Service can input to the specified transition set. Therefore, Event Service inputs to the suitable transition set so as to synchronize the list of URL and the present circumstances automatically.

V. Case Study

A. Transition Management with GUI application

We explain the example of modifying transitions with external application. We have implemented a GUI application, named "State Transition Manager", in such a way as to manage transitions easy and intuitively. "State Transition Manager" is a Web application, which is implemented with HTML and JavaScript. This application uses the APIs of SM Service for managing transitions. In implementing, we also use jQuery(Ver. 2.0.3) and Twitter Bootstrap(Ver. 3.1.0) as a CSS framework. Fig.3 shows a screenshot of State Transition Manager. Nav (in upper part of the page) represents the transition set of each user. Containers under nav, express each transition. Every transition can be deleted by clicking the delete button in the upper right corner of the container. *Current State*, *Next State*, and *Condition* can be edited by clicking the edit button to the left of the delete button. Each *action* has an edit button and a delete button and can be managed individually. Also new *action* can be added by choosing an action from selectbox.

Next, we will show an example of the use of "State Transition Manager". In the case of adding action that open curtain to Transition T0002, at first, choose "HNS:CURTAIN_OPEN" from slectbox (Fig.3 shows the very moment). Then click add button to the right of selectbox. When the add button was clicked, "State Transition Manager" added selected action to SM Service via API.

B. Input determination by external application

We explain the example of determining of transition set to input by using external application. In our laboratory, we have operated our entrance/leaving management service, called "In/Out Manager". Fig.4 shows a screenshot of In/Out Manager. These buttons represent each student. Students in the laboratory are represented by the colored buttons. Every student clicks their personal button when they leave and when they come. Thus, "In/Out Manager" is aware of the entry/exit

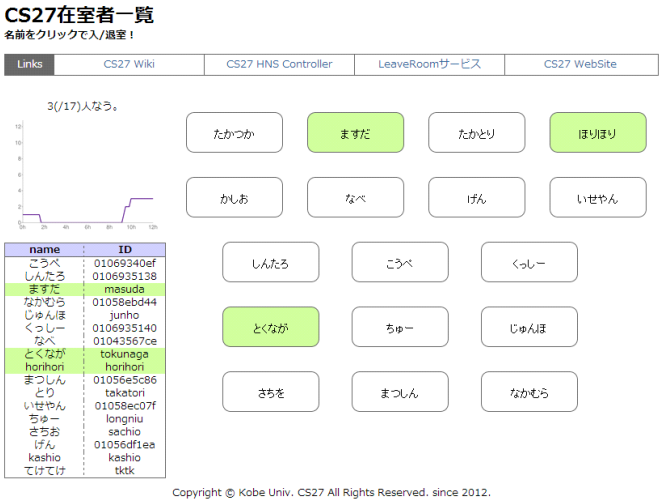

Fig. 4. Screenshot of "In/Out Manager"

of students. This application also can be cooperation with external service.

We used "In/Out Manager" for determining the transition set to input. By synchronizing the Event Service's list and the entry/exit of each user, external service can input to the appropriate transition set.

C. Discussion

Through the case study, it was shown that this system could modify the transitions from external application. This case study also represent that this system could choose an appropriate transition set based on the situation which is obtained by using the external service. However, these applications, which were used in the case study, don't work on SM Service automatically. It is necessary to develop applications that can modify transitions based on learning or switch the system transition state by sensing the context automatically. Besides, as a scale becomes big, management and editing of transitions are complicated, because transition is required for each operation and home appliance. We believe that this problem can be solved by patterning the transition. Furthermore, to use multiple transition set may cause the service conflict problem. This problem can be solved, by applying the solution in [5].

VI. Conclusion

The VAUI is appropriate for the use of HNS user interface. In order to improve the user experience of HNS, personalized interaction is absolutely imperative. In this paper, we extended the current HNS-VAUI system with the concept of personal adaptation. As a result of extension, a new HNS-VAUI named PSO-HNS-VAUI ready for personalizing by way of external interaction management. Through the adapting some learning method for the system, we will get the personalized interaction and appropriate service corresponding to each user. The case study illustrated how this method can be put into practice. From the results of case study, we confirmed the edit-ability of the interactions from external application and switchability of transitions based on the user. However, functions of automatic generating and modifying of transition set are not implemented yet. Moreover, the proposed system have to be evaluated the practical usability through by end-user experiments. This point should be investigated further.

Acknowledgment

This research was partially supported by the Japan Ministry of Education, Science, Sports, and Culture [Grant-in-Aid for Scientific Research (C) (No.24500079), (No.12877795), Scientific Research (B) (No.23300009)] and Sekisui House, Ltd.

References

[1] J. Cassell, "Embodied conversational interface agents," *Communications of the ACM*, vol. 43, no. 4, pp. 70–78, 2000.

[2] CUBE370, "Mmdagent & project-naip wiki," http://cube370.wiki.fc2.com/.

[3] H. Horiuchi, S. Saiki, S. Matsumoto, and M. Nakamura, "Implementing service framework for agent-based interactive interface in home network system," in *IEICE Technical Report*, February 2014, pp. 61–66.

[4] H. Igaki, M. Nakamura, and K. Matsumoto, "Design and evaluation of the home network systems using the service oriented architecture," in *Proc. International Conference on E-Business and Telecommunication Networks(ICETE04)*, vol. 1, August 2004, pp. 62–69.

[5] K. Ikegami, S. Matsumoto, and M. Nakamura, "New definition of environment feature interactions in home network system," in *Workshop on Dependability of Network Software Applications 2010 (DNSA 2010)*, November 2010, hiroshima, Japan.

[6] Julius Development Team, "Open-source large vocabulary CSR engine Julius," http://julius.sourceforge.jp/en_index.php?q=index-en.html.

[7] MMDAgent Project Team, "Mmdagent - toolkit for building voice interaction systems," Nagoya Institute of Technology, http://www.mmdagent.jp.

[8] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K. Matsumoto, "Constructing home network systems and integrated services using legacy home appliances and web services," *International Journal of Web Services Research*, vol. 5, no. 1, pp. 82–98, 2008.

[9] M. Ochs, C. Pelachaud, and D. Sadek, "An empathic virtual dialog agent to improve human-machine interaction," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 1*, 2008, pp. 89–96.

[10] M. P. Papazoglou and D. Georgakopoulos, "Service-oriented computing," *Communication of the ACM*, vol. 46, no. 10, pp. 25–28, 2003.

[11] S. Soda, M. Nakamura, S. Matsumoto, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "Implementing virtual agent as an interface for smart home voice control," in *In Asia-Pacific Software Engineering Conference (APSEC2012)*, December 2012, pp. 342–345.