Virtual Agent as a User Interface for Home Network System

Hiroyasu Horiuchi, Graduate School of System Informatics, Kobe University, Hyogo, Japan

Sachio Saiki, Graduate School of System Informatics, Kobe University, Hyogo, Japan

Shinsuke Matsumoto, Graduate School of System Informatics, Kobe University, Hyogo, Japan

Masahide Namamura, Graduate School of System Informatics, Kobe University, Hyogo, Japan

ABSTRACT

In order to achieve intuitive and easy operations for home network system (HNS), the authors have previously proposed user interface with virtual agent (called HNS virtual agent user interface, HNS-VAUI). The HNS-VAUI was implemented with MMDAgent toolkit. A user can operate appliances and services interactively through dialog with a virtual agent in a screen. However, the previous prototype heavily depends on MMDAgent, which causes a tight coupling between HNS operations and agent behaviors, and poor capability of using external information.To cope with the problem, this paper proposes a service-oriented framework that allows the HNS-VAUI to provide richer interaction. Specifically, the authors decompose the tightly-coupled system into two separate services: MMC Service and MSM service. The MMC service concentrates on controlling detailed behaviors of a virtual agent, whereas the MSM service defines logic of HNS operations and dialog with the agent with richer state machines. The two services are loosely coupled to enable more flexible and sophisticated dialog in the HNS-VAUI. The proposed framework is implemented in a real HNS environment. The authors also conduct a case study with practical service scenarios, to demonstrate effectiveness of the proposed framework.

Keywords: *Finite State Machine, Home Network System, Interactive Voice Interface, Service-Oriented Architecture, Virtual Agent*

INTRODUCTION

Research and development of home network system (HNS, for short) is recently a hot topic in the area of ubiquitous computing (Igaki, Nakamura, & Matsumoto, 2004; Li, & Zhang, 2004). Orchestrating house-hold appliances (e.g., TVs, DVDs, speakers, air-conditioners, lights, curtains, windows, etc.) and sensors (e.g., temperature, humidity, brightness, human-presence, etc.) via the network, the HNS provides value-added services for home users.

In the HNS, improving the usability is an essential requirement. Since home users operate appliances and services every day, the user interface must be intuitive and easy to learn.

DOI: 10.4018/ijsi.2015040103

Virtual agent user interface (VAUI, for short), which uses virtual agent as an interface, is a promising approach to fulfill the requirement (Ochs, Pelachaud, & Sadek, 2008; Cassell, 2000). Using voice recognition and synthesis technologies, the VAUI allows users to talk to an agent in a screen, to make intuitive operations for the system.

In our research group, we have been studying VAUI for HNS (say, HNS-VAUI) within an actual HNS environment (called CS27-HNS) (Nakamura, Tanaka, Igaki, Tamada, & Matsumoto, 2008). In our previous study (Soda et al., 2012), we have implemented a prototype system of HNS-VAUI, using MMDAgent Toolkit (Mmdagent.jp, 2009). Extending the finite state machine of MMDAgent so as to invoke external Web services, we successfully allowed the virtual agent to operate appliances and services of the HNS. The study showed that home users enjoyed operating the HNS through natural conversations with the agent.

However, the previous prototype system was able to provide quite limited interactions, since the extension was just made locally to the state machine. Thus, the prototype strongly depended on the design of MMDAgent Toolkit, which caused the lack of flexibility and expressivity. More specifically, operations of the HNS and detailed behaviors of the virtual agent were both described in the same state machine. The tight coupling of the operations and the behaviors made the system difficult to manage and extend. Also, the original state machine was so primitive that it could not use information from external Web services. Moreover, it was impossible to update and modify the state machine during run time.

To cope with the problem, we propose a new framework that allows HNS-VAUI to provide richer interactions. Exploiting the concept of service-oriented architecture (SOA) (Papazoglou and Georgakopoulos, 2003), we decompose the tightly-coupled system into two separate services: MMC Service and MSM service. The MMC service concentrates on controlling detailed behaviors of a virtual agent. Wrapping MMDAgent, the MMC service publishes a stateless Web service that simply commands a virtual agent to speak or move. On the other hand, the MSM service manages logic of HNS operations and dialog with the agent using richer state machines. The MSM service publishes a stateful Web service that invokes external Web services (including HNS and MMC services), based on a given input and the current state. The new state machine involved in the MSM service enables the usage of external information as well as the dynamic update of the state machine. Thus, the MMC and MSM services are loosely coupled to enable more flexible and sophisticated dialog in the HNS-VAUI.

In this paper, we implement the proposed framework within the CS27-HNS. The MMC and MSM services are both implemented by the Java Web service (Apache Axis2 on Tomcat) (Axis.apache.org, 2004) (Tomcat.apache.org, 1999). Also, Julius is used for the voice recognition engine (Julius.sourceforge.jp, 2002). Using the developed system, we also conduct a case study to demonstrate effectiveness of the proposed framework. In the case study, we implement three service scenarios: (a) turning on a TV, (b) asking today's weather, and (c) recommendation of using fan. It is shown that the proposed framework allows HNS-VAUI to provide more advanced and flexible interactions.

PRELIMINARIES

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 (Tokuda, Matsumoto, & Nakamura, 2012), remote monitoring/controls, appliance orchestration (Nakamura, Igaki, Tamada, & Matsumoto, 2004), energy saving, context-

aware services (Takatsuka, Saiki, Matsumoto, & Nakamura, 2013).

In our research group, we have implemented an actual HNS environment, called CS27-HNS. Introducing the concept of service-oriented architecture (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, to change channel of a TV to 6 in CS27-HNS, a client just accesses URL http://hns/TVService/setChannel?channel=6.

Virtual Agent (VA)

Virtual agent (VA) is an animated, human-like graphical chat bot program. Displayed on a screen, VA often serves as intuitive user interface of the system through voice interactions in natural language. Due to the natural interactions and a sense of intimacy, VA has received a lot of attention in recent years as a next generation user interface. In this paper, the user interface implemented with a VA is called VAUI.

MMDAgent is known as a powerful toolkit to implement VAs. Using a three-dimensional movie model of MMD together with built-in speech recognition and synthesis technologies, MMDAgent allows a user to easily develop a custom VA with a spoken dialogue system. Figure 1 shows virtual agent "Mei" bundled with the toolkit.

In MMDAgent, behaviors of a VA are defined by a dedicated finite state machine (which we call MMDA-FSM). Events and actions in the MMDA-FSM include motion of agent, speech synthesis, speech recognition, lip sync, application invocation, camera work, loading an MMD model, etc. Individual developers crate own VAs by customizing the MMDA-FSM.

Prototyping VAUI for HNS

We have been studying intuitive user interface for our CS27-HNS. We consider that VAUI fits well for operating various appliances and services in the HNS. A VAUI specifically designed for HNS is called HNS-VAUI in this paper. With HNS-VAUI, a user can operate HNS via natural interaction just like a "talk with a friend", instead of the conventional dreary one-way controls. Moreover, HNS-VAUI can efficiently use visual information to avoid long speech feedback.

In our previous work, we have developed a prototype HNS-VAUI for CS27-HNS using MMDAgent. We have revised the original MMDAgent in order to adapt it to the CS27-HNS. Major revisions are summarized as follows.

A. Command input from external applications

The original MMDAgent has a dedicated voice input system, with which MMDAgent first analyzes and recognizes voice input, and then generates a command as an input to the MMDA-FSM. However, the dedicated voice input was obstacle to achieve multi-modal controls of HNS. Therefore, we replaced the input system with a new Web service (called Virtual Agent Service), by which any external application can send commands directly to the MMDA-FSM.

B. External voice recognition system

Owing to Virtual Agent Service, MMDAgent no longer depends on the built-in voice recognition system. Therefore, using a variety of existing voice recognition engines, we implemented an external application for voice input. The application simply recognizes user's voice input, and sends an appropriate command to the Virtual Agent Service. The command is passed via Web-API of the service in a platform independent manner.

C. Revision of MMDA-FSM to execute Web-API

To invoke operations of HNS services and appliances, we revised MMDA-FSM so that it can execute Web services. CS27-HNS can be

Figure 1. Virtual Agent "Mei" of MMDAgent (Copyright 2009-2013 Nagoya Institute of Technology. Used with permission.)

operated by Web-API. Therefore, the revision allows MMDAgent to execute any appliance or service within CS27-HNS, in the same way as other operations specified in the MMDA-FSM.

Limitations of Prototype

After evaluation in the practical setting, we have found several limitations of the prototype system, which prevent future extensions for advanced user requirements. These limitations mainly came from the fact that the prototype strongly relied on the MMDA-FSM.

A. Limitation L1: Tight coupling of VA behaviors and HNS operations

In the prototype system, detailed motions of a VA and HNS operations (i.e., Web-API) were both specified statically within the same big MMDA-FSM. Thus, the behaviors of VA were tightly coupled with the invocation of HNS operations, which made the whole system quite complex to maintain. We could not use VA independently of the HNS, or could not modify the FSM dynamically from external applications.

B. Limitation L2: Low expressive power of MMDA-FSM

In the prototype system, HNS-VAUI worked according to a given MMDA-FSM. However, the expressive power of the MMDA-FSM was not high enough to describe rich and advanced interactions. For example, the original MMDA-FSM did not have a feature to obtain data from an external service and use it in another transition. Thus, it was difficult to include dynamic contexts in HNS-VAUI obtained from, for example, sensor services or external information services.

PROPOSED FRAMEWORK

In order to cope with the limitations of the prototype system, we propose a new framework for HNS-VAUI in this section. The new framework is supposed to provide more advanced interactions and higher usability.

Overview

To illustrate the key idea of the new framework, we first present architectures of the prototype and the new HNS-VAUI in Figure 2. As shown in the upper half of the figure, the prototype system used a single MMDA-FSM to specify both motions of a VA and external HNS services. This yielded the problem of tight coupling discussed in previous section.

In the proposed framework, we decompose the tightly-coupled system into two separate services: MMC service and MSM service. The MMC (Miku Miku Command) service concentrates on controlling detailed motions of a VA. As shown in the Figure 3, wrapping MMDAgent with the original MMDA-FSM, the MMC service publishes a stateless Web-API that simply commands a VA to speak or move. Thus, the MMC service implements a puppet VA that behaves as is just requested.

On the other hand, the MSM (Miku State Machine) service manages logic of HNS operations and dialog with the agent using richer state machines (called MSM-FSM). The MSM service publishes a stateful Web service that invokes external Web services based on a given input and the current state. Since the detailed motions of VA is delegated in the MMC service, the MSM service can deal with the VA motion and the HNS operation in the same way. The MSM-FSM can use the return value of Web-API for dynamic contexts, and has API for dynamic update of the FSM. Thus, behaviors of VA and invocation of HNS operations are decomposed, which achieves more flexible and sophisticated dialog in the HNS-VAUI.

In the following sections, we explain details of the MMC and MSM services.

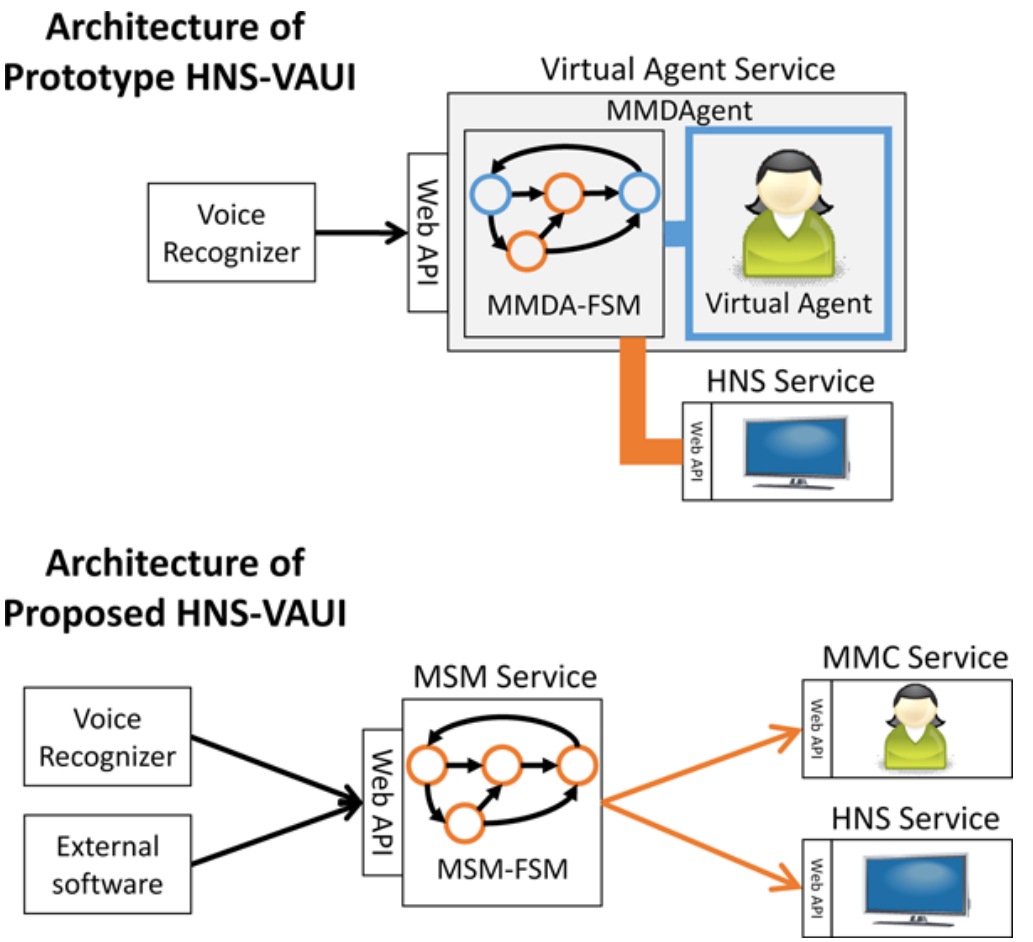
MMC (Miku Miku Command) Service

The MMC service provides a stateless Web service that directly sends commands to a VA of MMDAgent. It has two service API: say() and doMotion(). The method say() commands the VA to speak a given speech text, whereas doMotion() commands the VA to perform a given motion. When an external client application invokes the API, the MMC service internally communicates with the MMDAgent. The communication is encapsulated so that the client does not worry about the timing of message exchange or specification of the MMDA-FSM. Thus, the client can execute API at any time regardless the status of the VA.

Although the MMC service is supposed to provide stateless API, it takes some time for a VA to complete every single motion or speech. Therefore, when multiple service requests arrive within a short period, the MMDAgent would not be able to handle all the requests, simultaneously. Thus, it would fall into a panic state without proper flow control.

To cope with the problem, we implement a synchronization mechanism between the MMC service and the MMDAgent. Figure 4 shows the proposed mechanism, represented by two communicating FSMs handling the flow control of say() API. An MMC-FSM in the left side synchronizes a MMDAFSM in the right side, via message exchanges. In the

Figure 2. Architectures of prototype and proposed HNS-VAUI frameworks



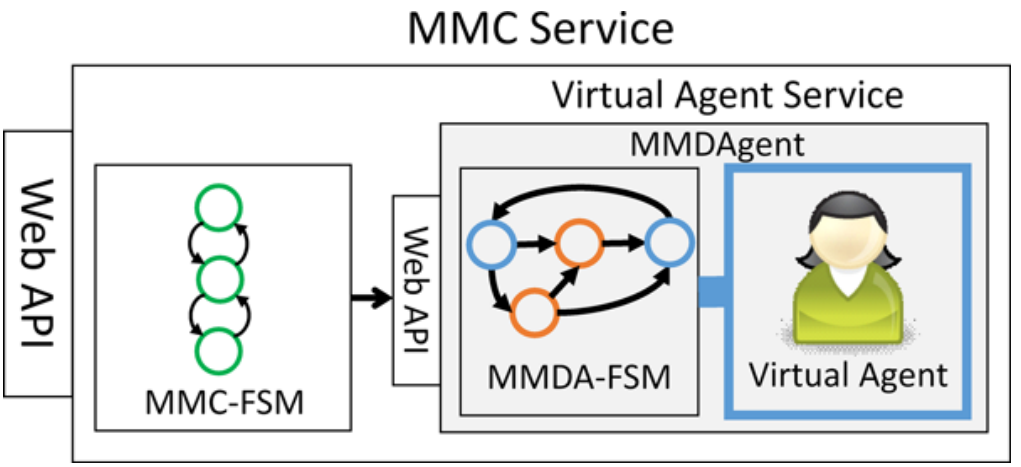
FSMs, "?" represents an input message, while "!" represents an output message. A message in lower-case letters with () represents an input message to the MMC service (i.e., API call of MMC), and a message in upper case is a command to the MMDAgent. A pair of input/output messages (with the same label) synchronizes state transitions of two FSMs.

The flow control works as follows. First, the MMC service receives a request say("Hello") at the wait state. The service sends a SPEECH command to the MMDAgent and moves to the execute state. Upon receiving the command, the MMDAgent tells a VA to start speaking "hello". When the VA finishes speaking, the MMDAgent executes over() method of the MMC service. Then, the MMC returns to the wait state.

If MMC receives another request say("Aloha") in the execute state, it sends a STOP SPEECH and moves to the cancel state. The MMDAgent tells the VA to stop speaking and executes over() method. When the speech of "hello" is canceled, MMC sends a SPEECH command with "aloha". Thus, two consecutive requests are successfully consumed.

Similarly, we implement the same mechanism for the doMotion() method.

Figure 3. Detail of MMC service architecture



MSM (Miku State Machine) Service

The MSM Service provides Web service that dynamically constructs a stateful logic of HNS-VAUI. The MSM service wraps a set of finite state machines (say MSM-FSMs) that receive input messages (from external voice recognition systems, sensors, context-aware services, etc.) and invoke Web services (including the MMC service, HNS services, generic Web services, etc). The MSM service has input() method that inputs a message to a MSM-FSM. When the MSM-FSM receives a message, it moves to a new state and execute actions associated with the state transition. The MSM service also has addTransition(), editTransition(), deleteTransition() methods to dynamically modify a MSM-FSM.

A MSM-FSM consists of a set of state transitions. Each transition is defined by five

Figure 4. Communicating FSMs to handle say() method of MMC service

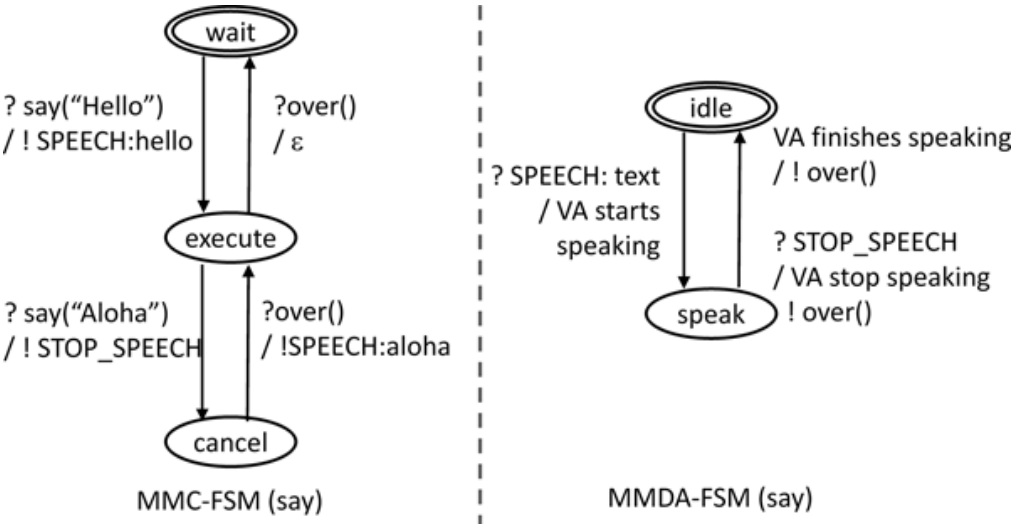


Table 1. Example of state transition

No.	Transition ID	Current State	Next State	Event	Action
1	Activate Voice Control Mode	Idle	Voice Ctrl Mode	ACTIVATE	MMC.say(May I help you?); MMC.doMotion(smile);
2	Deactivate Voice Control Mode	Voice Ctrl Mode	Idle	DEACTIVATE	MMC.say(Bye.); MMC.doMotion(bye);
3	Turn on TV	Voice Ctrl Mode	Voice Ctrl Mode	TV_ON	MMC.say(I turn on the TV.); TVService.on();
4	Ask Weather Report	Voice Ctrl Mode	Voice Ctrl Mode	ASK_WEATHER	[weather] = WeatherService.get(today); MMC.say(It is [weather], today.);
5	Recommend Fan	Idle	Recommend Fan	RECOMMEND_FAN	MMC.say(Shall I turn on the fan?);
6	Accept Fan	Recommend Fan	idle	YES	MMC.say(I turn on the fan.); FanService.on();
7	Reject Fan	Recommend Fan	idle	NO	MMC.say(Okey.);
8	Timeout Fan	Recommend Fan	idle	TIMEOUT	

elements: transition ID, current state, next state, event and action. A current state (or next state) specifies a state which the transition moves from (or to, respectively). An event corresponds to an input message making the transition fire. An action contains one or more Web services to be executed as feedback of the state transition. Table 1 shows an example of a MSM-FSM, and Figure 5 shows its schematic representation.

We explain how the MSM Service realizes the interaction as follows. First, an external application (e.g., a voice recognizer, a context-aware application, manual input of a user) executes input() API with an event string. Then, the MSM service checks the current state of the MSM-FSM, and finds a transition whose event is identical to the given event. Next, the MSM service change the current state to the next state, and executes Web services associated with the corresponding action. The Web services include behaviors of a VA (executed by MMC service) and HNS operations.

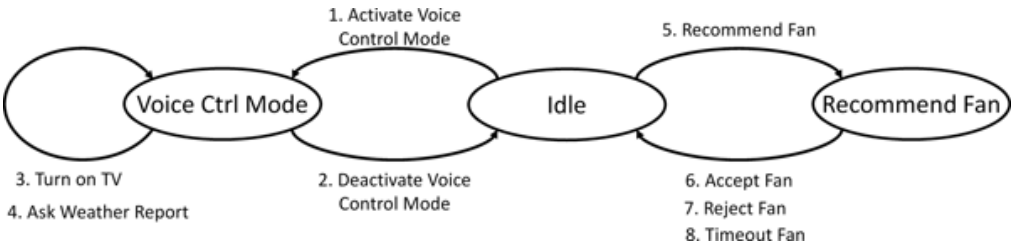
Implementation

The proposed framework has been implemented by using the following technologies.

a. MMC service: Java 1.7.0 45, Apache Tomcat 7.0.39, Apache Axis2 1.6.2, MMDAgent 1.4
b. MSM service: Java 1.7.0 45, Apache Tomcat 7.0.39, Apache Axis2 1.6.2, MongoDB 2.2.7
c. Voice Recognizer: Java 1.7.0 45, Julius 4.2.3

In addition, the 3D model for the agent have been borrowed from Lat, respectively (MikuMikuDance Wiki, 2010).

Figure 5. State transition diagram of table 1



The whole system comprised of about 4500 lines of code. The development effort was about 5 man-days. Note that the MMC and MSM services are deployed as Web service using Axis2 middleware. Every API is invoked in the form of URL. For example, MMCService.say("hello") can be executed as http://server addr:8080/MMCService/say?text=hello.

CASE STUDY

Outline

In this section, we conduct a case study that demonstrates three kinds of interaction scenarios between a VA and a user. The first scenario is the conventional command-based interaction, where the user asks the VA to turn on a TV. The second scenario involves advanced interaction, where the VA obtains data from an external weather service and tells the weather report to the user. The third one is the context-aware interaction, where the VA autonomously recommends to turn on a fan when room temperature is high.

To implement these interaction scenarios, we first build the MSM-FSM shown in Table I and Figure 5, using the MSM service. The initial state is set to Idle. Then, for the external voice recognizer, we define a mapping from user's voice to an input event of the MSM service, as shown in Table 2. For example, if the user says "Thank you" to the system, the voice recognizer sends DEACTIVATE to the MSM service. In the MMC service, we configure two motions smile and bye, by which the VA smiles and waves a hand, respectively. For the appliance control, we use TVService and FanService that operate a TV and a FAN in CS27-HNS.

Scenario 1: Turning on TV

This scenario starts when the user says "Hello, Ms. Agent!". According to Table II, event ACTIVATE is sent to the MSM service. The MSM service in Idle state receives ACTIVATE, and executes two actions of the MMC service (See transition no.1 in Table I). By the MMC service, the VA says "May I help you?" with a smile face. The MSM then moves to VoiceCtrlMode. Next, the user says "Turn on a TV.". Event TV ON occurs, and transition no.3 fires to execute two actions. The VA says "I turn on the TV" and the TV is turned on by TVService.on().

Scenario 2: Asking Today's Weather

Following Scenario 1, if the user says "What is today's weather?", event ASK WEATHER occurs at VoiceCtrlMode state. The MSM service then triggers transition no.4 and executes WeatherService to get weather report data. The data is temporarily stored in variable [weather], which is passed in the speech text for the MMC service. If the weather report is "fine", the VA says "it is fine today". Finally, the user says "Thank you" and event DEACTIVATE occurs. The state moves to Idle to wait for the next order.

Scenario 3: Suggesting Using Fan

Our research group has developed a framework, called RuCAS, for creating context aware services in CS27-HNS (Takatsuka et al., 2013). RuCAS helps users to define custom contexts using various sensors and services, and to create own context-aware services. Using RuCAS, we first define a context Hot to be a situation that the room temperature is higher than 28 degree. Then, we create a rule "When Hot is true, send event RECOMMEND FAN to the MSM service".

When the temperature actually becomes higher than 28 degree, RuCAS sends RECOMMEND FAN to the MSM service, and transition no.5 fires. The VA speaks "Shall I turn on the fan?", and the MSM service moves to state Recommend Fan. Now, if the user says "Yes", the VA says "I turn on the fan" and the fan is switched on, by transition no.6. If the user says "No" or event TIMEOUT occurs, no operation is performed.

Thus, RuCAS and the proposed framework are easily integrated to perform sophisticated context-aware interactions of HNS-VAUI.

Table 2. Voice-event mapping configured in voice recognizer

user's speech text	input event for MSM
"Hello, Ms. Agent!"	ACTIVATE
"Turn on a TV."	TV ON
"Thank you."	DEACTIVATE
"What is today's weather?"	ASK WEATHER
"Yes"	YES
"No"	NO

Discussion

Through the case study, it was shown that the proposed HNS-VAUI framework can implement richer and more advanced interactions, easily. However, as the size of HNS becomes larger, the MSM-FSM becomes more complex, since new transitions are required for each operation of home appliances. We believe that the management can be facilitated by introducing a dedicated management service and patterns of state transitions. Furthermore, using multiple MSM-FSMs may cause a functional conflict, which is known as a feature interaction problem. For this, the existing approach to detect and solve the interactions can be used (Inada et al., 2012).

Our next challenge would be adaptation or personalization of HNS-VAUI. Since every HNS environment is different from each other, and life style varies from one family to another. Moreover, every user has a favorite virtual agent and preferred interactions. In order to meet such individual needs, HNS-VAUI should be able to adapt to the user as the user keeps using it. An approach to implement the adaptation is to extract user's preference and contexts by applying data mining to large-scale log of VAUI operations and environment. This challenge will be left for our future work.

CONCLUSION

In this paper, we have proposed a new framework for implementing virtual agent user interface for home network system (HNS-VAUI). To cope with limitations of the previous prototype system, we decompose detailed behaviors of the virtual agent and logic of HNS operations within two separate services: MMC (Miku Miku Command) service and MSM (Miku State Machine) service. The two services are loosely coupled to enable more flexible and sophisticated dialog in the HNS-VAUI. A case study with three interaction scenarios illustrated the practical feasibility of the proposed framework.

Our future work include the personalization and adaptation of HNS-VAUI. We also plan to conduct experimental evaluation with subjects, to see how interactions with VA affect the human satisfaction and usability.

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.24500258), Scientific Research (B) (No.26280115), Young Scientists (B) (No.26730155)] and Kawanishi Memorial ShinMaywa Education Foundation.

REFERENCES

Axis.apache.org. (2004). Apache Axis2. Retrieved from http://axis.apache.org/axis2/java/core/

Cassell, J. (2000). Embodied conversational interface agents. *Communications of the ACM*, *43*(4), 70–78. doi:10.1145/332051.332075

Igaki, H., Nakamura, M., & Matsumoto, K. (2004). Design and evaluation of the home network systems using the service oriented architecture. In *1st International Conference on E-business and Telecommunication Networks* (pp. 62-69). Setúbal, Portugal: INSTICC Press.

Inada, T., Igaki, H., Ikegami, K., Matsumoto, S., Nakamura, M., & Kusumoto, S. (2012). Detecting Service Chains and Feature Interactions in Sensor-Driven Home Network Services. *Sensors (Basel, Switzerland)*, *12*(7), 8447–8464. doi:10.3390/s120708447 PMID:23012499

Julius.sourceforge.jp. (2002). Open-Source Large Vocabulary CSR Engine Julius. Retrieved from http://julius.sourceforge.jp/en_index.php

Li, X., & Zhang, W. (2004). The design and implementation of home network system using OSGi compliant middleware. Consumer Electronics. *IEEE Transactions on*, *50*(2), 528–534.

MikuMikuDance Wiki. (2010). Miku Hatsune (Lat). Retrieved from http://mikumikudance.wikia.com/wiki/Miku_Hatsune_(Lat)

Mmdagent.jp. (2009). mmdagent.jp. Retrieved from http://www.mmdagent.jp/

Nakamura, M., Igaki, H., Tamada, H., & Matsumoto, K. (2004). Implementing integrated services of networked home appliances using service-oriented architecture. In *2nd International Conference on Service Oriented Computing* (pp. 269-278). NY: ACM. doi:10.1145/1035167.1035206

Nakamura, M., Tanaka, A., Igaki, H., Tamada, H., & Matsumoto, K. (2008). Constructing home network systems and integrated services using legacy home appliances and web services. [IJWSR]. *International Journal of Web Services Research*, *5*(1), 82–98. doi:10.4018/jwsr.2008010105

Ochs, M., Pelachaud, C., & Sadek, D. (2008). An empathic virtual dialog agent to improve human-machine interaction. In The Seventh international Conference on Autonomous Agents and Multiagent Systems (pp. 89-96). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Papazoglou, M., & Georgakopoulos, G. (2003). Service-Oriented Computing. *Communications of the ACM*, *46*(10), 25–28.

Soda, S., Nakamura, M., Matsumoto, S., Izumi, S., Kawaguchi, H., & Yoshimoto, M. (2012). Implementing virtual agent as an interface for smart home voice control. In The 19th Asia-Pacific Software Engineering Conference (pp. 342-345). NY: IEEE. doi:10.1109/APSEC.2012.39

Takatsuka, H., Saiki, S., Matsumoto, S., & Nakamura, M. (2013). Implementing execution platform for managing context-aware services based on heterogeneous and distributed web services. *IEICE Technical Report*, *113*(327), 71–76.

Tokuda, K., Matsumoto, S., & Nakamura, M. (2012). Implementing personal home controllers on smartphones for service-oriented home network. In The 8th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (pp. 777-784). NY: IEEE. doi:10.1109/WiMOB.2012.6379162

Tomcat.apache.org. (1999). Apache Tomcat. Retrieved from http://tomcat.apache.org/