Developments of Anthropomorphic Dialog Agent: A Plan and Development and its Significance

Shin-ichi Kawamoto[*1], Hiroshi Shimodaira[*1], Shigeki Sagayama[*1,2], Tsuneo Nitta[*3], Takuya Nishimoto[*4], Satoshi Nakamura[*5], Katsunobu Itou[*6], Shigeo Morishima[*7], Tatsuo Yotsukura[*7], Atsuhiko Kai[*8], Akinobu Lee[*9], Yoichi Yamashita[*10], Takao Kobayashi[*11], Keiichi Tokuda[*12], Teikichi Hirose[*2], Nobuaki Minematsu[*2], Atsushi Yamada[*13], Yasuharu Den[*14], Takehito Utsuro[*3]

[*1] JAIST, [*2] Univ. Tokyo, [*3] Toyohashi Univ. of Tech., [*4] Kyoto Inst. of Tech, [*5] ATR, [*6] AIST, [*7] Seikei Univ., [*8] Shizuoka Univ., [*9] NAIST, [*10] Ritsumeikan Univ., [*11] Tokyo Inst. of Tech., [*12] Nagoya Inst. of Tech., [*13] ASTEM, [*14] Chiba Univ.

1 Abstract

With financial support from Japan's Information-technology Promotion Agency (IPA), a three-year project was launched in 2000 to develop the basic software for an anthropomorphic spoken dialog agent. The basic software consists of four modules for speech recognition, speech synthesis, facial image synthesis, and multi-modal dialog integration. This interim report describes the basic design approach and an implementation of the software focusing on efforts to ensure the interactive capability of the spoken dialog agent.

1.1 Keywords

2 Introduction

With financial support provided by the Information-technology Promotion Agency (IPA), a three-year project was launched in April 2000 to develop the basic software for an anthropomorphic spoken dialog agent[1]. The basic software consists of four modules for speech recognition, speech synthesis, facial image synthesis, and multi-modal dialog integration.

System control and data management capabilities in a dispersed environment are essential in order for these various modules to interoperate smoothly as a single dialog system, and several systems exhibiting these capabilities have been developed including DARPA's Communicator Program [7] based on MIT's Galaxy-III [6], and the Open Agent Architecture (OAA) developed by SRI [8].

This paper describes the basic design and an implementation of the project software that is intuitive, easy to understand, and ensures fully interactive spoken dialog with the agent.

2. Development of Basic Software for Anthropomorphic Spoken Dialog Agents

2.1 Development Overview

Considering how much faster computers are able to crunch numbers and deal with complex calculations than people, why is that they are incapable of communicating by speech with humans? Just what would it take to enable computers to speak in the same way that people talk among themselves? As the technological foundation for such communications, research so far as focused on such dream technologies as "apprehending speech," "synthesizing speech," and "generating computer graphic representations of real people." Practical success has recently been achieved in applying speech recognition and speech synthesis technologies to synthetic voice reading. It is also now possible as we are already beginning to see in movies to render the realistic movement of actors and actresses with computer graphics. Basic technologies to achieve these sorts of human interfaces have been advanced to a certain level, and research is now concentrating on refining and improving the quality of these capabilities even more. Finally, considerable R&D is also seeking to integrate these technologies to create interfaces and systems that are capable of sustained dialog similar to that between people.

Being actively involved in these developments, the Multi-modal Tool Working Group of the Special Interest Group on Spoken Language Processing of the Information Processing Society of Japan (IPSJ SIG-SLP) over the three-year period from 1998 to 2000 identified anthropomorphic agents as a target of next-generation research, and they have developed a plan to build and make publicly available a research platform through collaborative efforts of researchers.

This conception received support of the Information-technology Promotion Agency (IPA), and more than ten research institutes began a cooperative effort to develop the basic software in March 2000.

2.2 Software Configuration

The anthropomorphic spoken dialog agent that is now under development consists of four basic software modules, all of which will be made available in the form of freeware. By implementing the software as separate modules, this is not only an effective tool for assessing the various constituent technologies, it also provides a versatile R&D platform making it easy to build original dialog systems by simply plugging in different software modules developed independently by the different R&D institutes involved in the project as required.

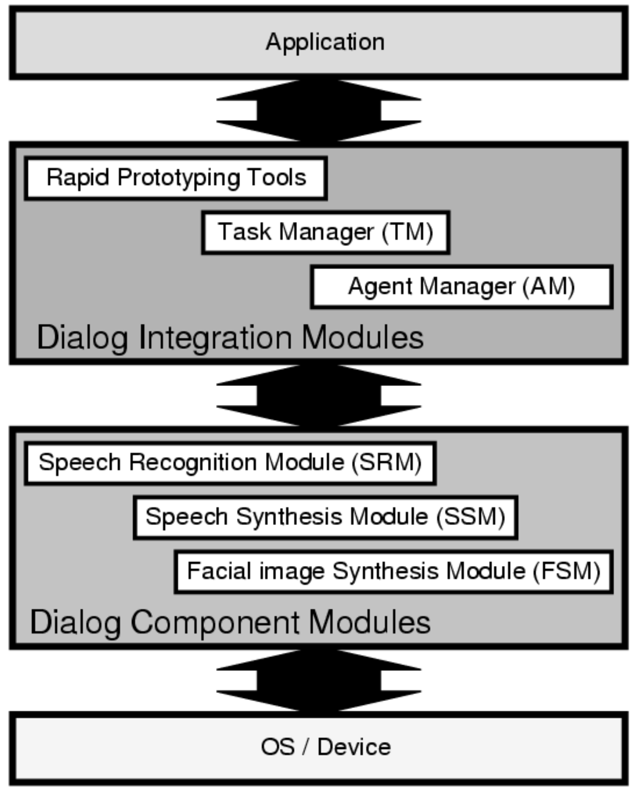

Figure 1 Anthropomorphic spoken dialog agent platform

2.2.1 Basic Software for Integrating Anthropomorphic Spoken Dialog Agents

New basic software is being developed to integrate and control the dialog component modules and to manage the dialog. Some of the specific projects that are currently under way include (a) an Agent Manager (AM) providing low-level control of the speech recognition, speech synthesis, facial image synthesis, and other modules; (b) the capability to interpret VoiceXML -based high -level dialog descriptions; (c) a Task Manager (TM) for controlling dialog using the functions provided by the AM; and (d) a prototyping tool to provide a GUI environment s upporting the setting of parameters and the description and control of scenarios, all things that are necessary to construct dialog systems.

In this paper we will address the issues involved in designing a dialog system from the standpoint of the Agent Manager, and present an implementation example.

2.2.2 Basic Software of Synthesizing Dialog Speech

New basic speech synthesis software is being developed that not only clearly reads sentences of mixed kanji and kana (Chinese characters and phonetic script) , but also shares data to enable synchronization with a facial image. This enables lip -sync, the synchronization of sound and motion so the facial movements of speech coincide with the sounds.

Furthermore, anticipating changes in the nature of the speech to reflect different circumstances or the intent of the speaker, we are also seeking ways to control a range of different emotions and speech rhythms.

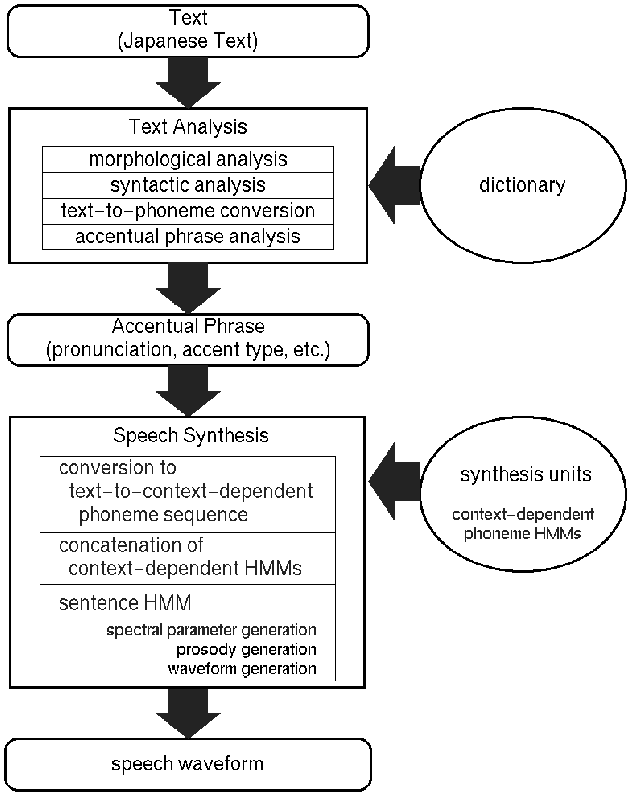

Figure 2 Speech synthesis module

2.2.3 Basic Software for Dialog Speech Recognition

Building on the software that came out of the IPA's basic Japanese dictation software development project from April 1997 to March 2000, it should be easy enough to extend the capabilities of that software package to accommodate dialog processing and implement flexible co ntrol. Specifically, we are doing away with grammar -based recognision and recognition results, and developing functions that can deal with unnecessary words and poses, and can provide dynamic control of recognition processing.

2.2.4 Basic Software for Fac ial Image Synthesis and Control

Starting with the IPA's facial image processing system for the human-like kansei agent that was developed from June 1995 to March 1998, we are enhancing the software package to support higher quality agent facial image synthesis, animation control, and precise lip -sync with synthetic and natural speech. Some of the specific enhancements include a GUI able to map standard wire frames to images of heads shot from different angles to easily generate 3D models of human heads, sh aring of data with the speech synthesis module, more precise lip -sync, the ability to add any facial expressions, and the ability to control nodding and blinking.

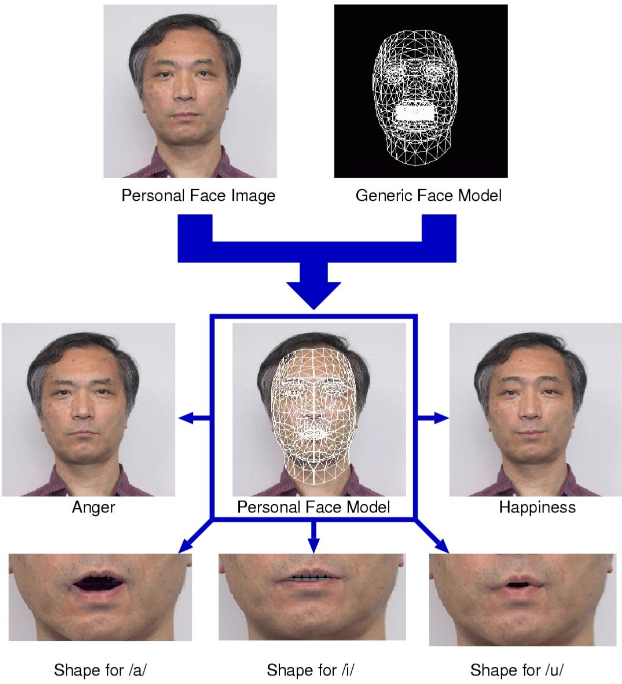

Figure 3 Facial image synthesis module

3. Basic Design of Module Integration Processing

3.1 Relationship Between the Agent Manager and Modules

The Agent Manager (AM) consists of two functional layers: the Direct Control Layer and the Macro Control Layer. Figure 4 shows a schematic representation of the relationship between the AM and the various modules.

The Direct Control Layer (AM -DCL) directly controls the sets of commands that are defined for each module, and the various modules are able to communicate with other modules through this layer. The Macro Control Layer (AM-MCL) interfaces mainl y with the Task Manager (TM). By redefining frequently used sequential command sets as macro commands and by taking on inter -module synchronization management and similar low -level module control, the AM -MCL markedly improves operation of the system from the standpoint of the Task Manager.

In principle, the Speech Recognition Module (SRM), and Speech Synthesis Module (SSM), and the Facial Image Synthesis Module (FSM) communicate through the AM - DCL. This means that, in developing a module, one only needs to worry about communication with the AM. This is a major benefit for the present project, because it allows the various modules to be developed at separate locations of the participating R&D firms.

The Task Manager (TM) mainly communicates through the AM-MCL, but if necessary can also communicate with the AM-DCL just like the other modules. All the output from the various modules is supplied to the TM. The TM is thus able selectively adopt whatever data that it needs from among the totality of data that it receives from all the modules.

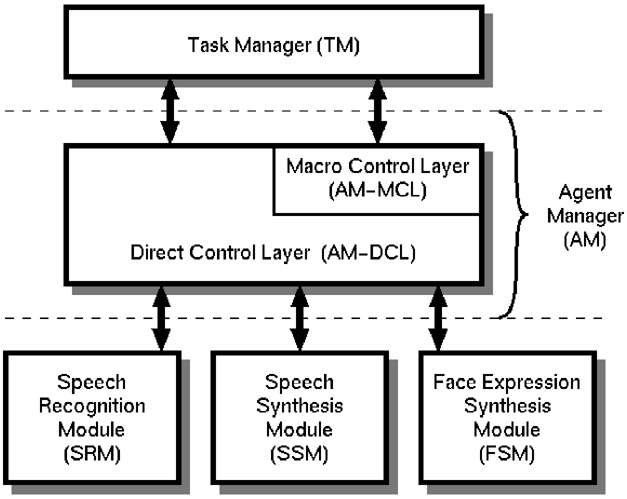

Figure 4 Basic configuration of the Agent Manager and Modules

3.2 Virtual Machine Models

In defining the command specifications for communicating between the dialog integration module and the various dialog component modul es, the dialog component modules are treated as virtual machine models. For example, Fig. 5 illustrates the relationship between the Agent Manager (AM-DCL) and a virtual machine model.

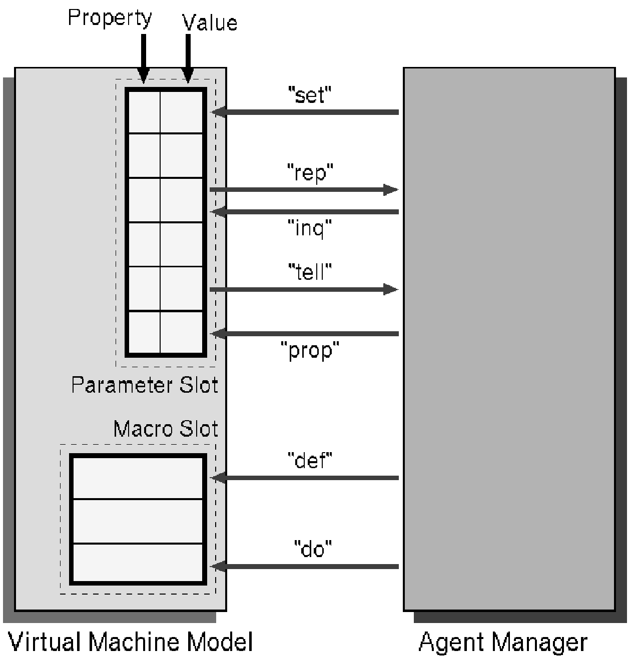

Figure 5 Relationship between the Agent Manager and a virtual machine model

The dialog component modules are managed by defining a parameter slot for each input and output parameter. The macro command definitions in the dialog component modules are managed in a similar way using macro slots. These slots are shared with the AM-DCL, and it is through the slots that the AM-DCL communicates with the dialog component modules.

Each slot possesses a value and a property that are treated like virtual machine model switches, and the slots are activated by common commands. The slot values can effect various actions; they can monitor an operating state, direct that an action start or stop, set a particular operating environment, and so forth. Changing the value of slot is immediately reflected in the form of a different action. In other words, changing a slot value is instantly associated with a particular action.

This makes it possible to manipulate all the dialog component modules in a centrally coordinated fashion. Commands are communication specifications that are not dependent on any particular module, while the parameter slots are module-dependent specifications that are not dependent on communication. This means that the difference between dialog component modules is nothing more than the different functions programmed in their parameter slots. And by abstracting and treating the dialog component modules as virtual machine models, this makes it very easy to either expand functional capabilities or add more dialog component modules. For example, a new function could be added by simply adding a new parameter slot to the virtual machine model. And like the dialog component modules that are already defined, new dialog component modules are added based on the concept of the virtual machine model by simply defining parameter slots.

Imagine, for example, that we want to add a facial expression recognition module. The minimal framework for incorporating such a module into the system would involve the definition of a number of parameter slots: (a) a parameter slot to start the module that is based on common specifications, (b) a parameter slot relating to the output of facial expression recognition results, (c) a parameter slot specifying a recognition algorithm, and (d) a parameter slot specifying the model. Or suppose we want to extend the speech analysis capabilities of a speech recognition module by enabling it to acquire a basic frequency of F0. This could be easily accomplished using the virtual machine model by adding a parameter slot to set the basic frequency output and a parameter slot to specify a basic frequency sampling algorithm.

3.3 Basic Operating Commands for a General-Purpose Virtual Model

Table 1 shows a list of commands that could be used in the basic operation of a general-purpose virtual module.

Two types of identifiers listed in Table 2 can be appended to slot values, etc. that are output from modules.

3.4 Problem of Synchronization Between Modules and an Implementation Example

Basically, each module is designed to operate independently of other modules, and is managed using the basic commands that were discussed earlier in the section on the Agent Manager. However, processing in some cases requires synchronization between modules. The most obvious example is lip-sync; when an anthropomorphic agent speaks, it is essential that the animated movement of the lips coincide with the synthesized or natural speech. In order to achieve such synchronization, data must be shared by two modules. Taking lip-sync as an example, here we will explain in greater detail how synchronization is achieved.

Table 1 Names and functions of basic operating commands for a general-purpose virtual module

Name	Function
set	Set a parameter slot value.
def	Set a macro slot value.
inq	Inquire a slot value.
prop	Set a slot property.
save	Apply a different name to the current slot and save.
rest	Restore the slot value that was saved with the save command.
del	Delete the slot value that was saved with the save command.
do	Evaluate macro slot value or file contents.

Table 2 Names and functions of identifiers that can be appended to slot values that are output from modules

Name	Function
rep	Slot value output.
tell	Output of value not defined as a slot.

3.5 Communication Between Modules to Achieve Synchronization

Synchronization between a speech synthesis module and a facial image synthesis module might be achieved

- by communication over a direct connection that is set up between the two modules, or
- in the same way as other communication, via the Agent Manager.

Note that while this example only involves synchronization between two modules, the first method would require a synchronization management capability in which every module was aware of every other module. This would make the modules more interdependent while at the same time diminishing their autonomy.

In the second method, synchronization between the two modules is managed by the AM. While this increases the processing costs, it makes it easier to maintain the autonomy of the modules since designers only have to concern themselves with communication between the module and the AM.

Because our current priority is to make it easier to ensure the autonomy of modules, we are proceeding on the basis of the second approach.

The actual synchronization indicator that defines the exact timing when speech begins and so on is achieved by conveying the system time to the two modules. Very accurate synchronization is achieved using the Network Time Protocol (NTP) that was developed for system time synchronization across networks.

3.6 Data Sharing Between Modules for Synchronization

Management of the synchronization between the two modules might be implemented by a higher level module that is separate and distinct from the AM. We are also considering defining and implementing a new type of module that is dedicated exclusively to synchronization. However, considering the importance of the lip-sync capability for agents and how frequently such capability is used in spoken dialog, we have currently implemented this function using a macro command provided by the Agent Manager.

The essential data that is needed for lip-sync when an agent speaks is the durations of each phoneme making up the speech. This information is obtained by interrogating the speech synthesis module. One might be able to think of other kinds of information that would be useful in this context, but for the time being we only use the duration of each phoneme.

It is also necessary to verify that the two modules are ready to speak before speaking can actually begin. This information is obtained by the following procedure. The speech process is divided into two parts: prepare to speak and begin to speak. The modules are designed to automatically generate a message indicating that they are ready to speak. As soon as the Agent Manager detects this information from the two modules, the AM directs that they can actually begin speaking.

Figure 6 shows the sequence of commands involved in this process. Note that the commands triggering these sequential processes are actually implemented by the macro command function in the AM-MCL.

4. Conclusions

This paper described the design and an implementation of the basic software for an anthropomorphic spoken dialog agent that ensures interactiveness, a project sponsored by Japan's Information-technology Promotion Agency (IPA).

Considering that this is an interim report on a system that is currently under development, there are a number of unresolved issues that still need to be worked out. As the project unfolds, we will further expand and enhance the functions of the dialog component modules and the multi-modal dialog integration module, and explore the feasibility of incorporating a standard distributed object environment architecture such as CORBA.

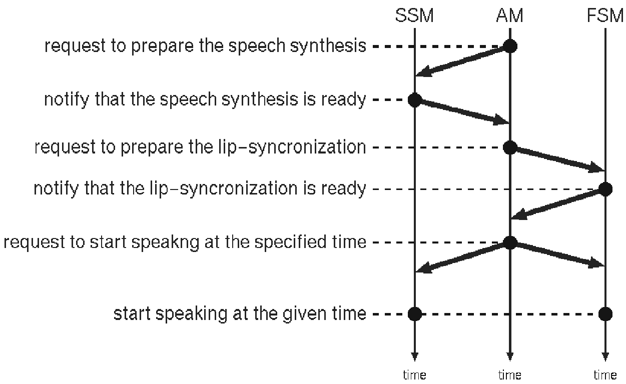

Figure 6 Processing flow between AM, SSM, and FSM when an agent speaks

Acknowledgments

Part of this work is supported by the Information-technology Promotion Agency's program to support original information technology.

REFERENCES

1. Shigeki Sagayama and Satoshi Nakamura: Development of Anthropomorphic Dialogue Agent: a Plan and Its Significance. Information Processing Society of Japan, Technical Report 2000-SLP-33-1, Oct. 2000 (In Japanese).
2. Stephenie Seneff, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid and Victor Zue: GALAXY-II: A Referece Architecture for Conversational System Development. In ICSLP-1998, pp.931--934, 1998.
3. DARPA Communicator Program, 1998. http://fofoca.mitre.org/.
4. OAA (The Open Agent Architecture). http://www.ai.sri.com/~oaa/.
5. CORBA (The Common Object Request Broker Architecture). http://www.corba.org/.