Galatea: Open-source Software for Developing Anthropomorphic Spoken Dialog Agents

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

[1] Japan Advanced Institute of Science and Technology
[2] The University of Tokyo
[3] Toyohashi University of Technology
[4] Advanced Telecommunications Research Institute International
[5] National Institute of Advanced Industrial Science and Technology
[6] Seikei University
[7] Shizuoka University
[8] Nara Institute of Science and Technology
[9] Ritsumeikan University
[10] Tokyo Institute of Technology
[11] Nagoya Institute of Technology
[12] The Advanced Software Technology and Mechatronics Research Institute of Kyoto
[13] Chiba University
[14] Kyoto University

Summary. Galatea is a software toolkit to develop a human-like spoken dialog agent. In order to easily integrate the modules of different characteristics including speech recognizer, speech synthesizer, **facial animation synthesizer**[facial-image synthesizer] and dialog controller, each module is modeled as a virtual machine having a simple common interface and connected to each other through a broker (communication manager). Galatea employs model-based speech and **facial animation**[facial-image] synthesizers whose model parameters are adapted easily to those for an existing person if his/her training data is given. The software toolkit that runs on both UNIX/Linux and Windows operating systems will be publicly available in the middle of 2003 [1, 2].

1 Introduction

Anthropomorphic spoken dialog agent (ASDA), behaving like humans with facial animation and gesture, and making speech conversations with humans,

is one of the next-generation human-interface. Although a number of ASDA systems [3–8] have been developed, communication between the ASDA system and humans is far from being natural, and developing a high quality ASDA system is still challenging. In order to activate and progress the researches in this field, we believe that an easy-to-use, easy-to-customize, and free software toolkit for building ASDA systems is indispensable. For examle, it would be nice if the toolkit provides unlimited number of life-like agent characters having different faces and voices as human beings in the real world.

We have been developing such an ASDA software toolkit named Galatea since 2000, aiming to provide a platform to build next generation ASDA systems. The features of the toolkit are as follows: (1) high customizability in text-to-speech synthesis, realistic face animation synthesis, and speech recognition, (2) basic functions to achieve incremental (on-the-fly) speech recognition, (3) mechanism for "lip synchronization"; synchronization between audio speech and lip image motion, (4) "virtual machine" architecture to achieve transparency in module to module communication.

If compared to the related works such as CSLU toolkit [9] and DARPA Communicator Program [10], our toolkit is still **preliminary**[germinal]. However, it is compact, simple, easy-to-understand and thus suitable for developing ASDA systems for research purposes, and of course it the first Japanese toolkit of life-like agents. One of the outstanding features of Galatea is that it uses a snap shot of an existing person to synthesize face images of an agent. Therefore, it can synthesize an unlimited number of agents having different faces as far as the snap shots of different people are provided. At present, simple ASDA systems have been successfully built with the toolkit under UNIX/Linux and Windows operating systems, and the subset of the toolkit will be publicly available in the middle of the year 2003.

This paper is divided into six sections. In section 2, design concepts for the Galatea software toolkit are discussed. Brief explanations of each functional module of the toolkit are given in section 3. Prototype systems developed by the toolkit are shown in section 4 followed by discussions in section 5. Finally the last section is devoted to conclusions.

2 Features for the Toolkit

In this section, we discuss the features of Galatea to build ASDA systems which speak, listen, and behave like humans.

2.1 Configuration for the easy-to-customize

In Galatea, synthesized facial images and voices are customizabile easily depending on the purposes and applications of the toolkit users. This customiazability is achieved by employing model based approaches where basic model parameters are trained or determined with a set of training data derived from

an existing person. Once the model parameters are trained, facial expressions and voice quality can be controled easily.

2.2 Key techniques for achieving natural spoken dialog

If compared to the keyboard-based conversation, typical phenomena are observed in speech-based conversation. These include the case that human listeners nod or say "uhmm" during a conversation, and the case that the speakers control the prosody to indicate types of utterances such as questions, statements, and emotions. Galatea provides basic functions to study those phenomena for human-like speech-based conversation. For example, Galatea provides the functions of incremental speech recognition, interruption over synthesized speech, and so on. In addition, Galatea provides a simple function of synchronization between the synthesized speech and the facial animation. This function will be useful to realize natural speech-based conversation.

2.3 Modularity of functional units

Naturally, Galatea provides a simple architecture to manage each functional unit, and to work in parallel. In some situations, system creators or toolkit users will not be satisfied with the performance of the original modules in the toolkit and they would like to replace them with the new ones or add new ones to the system. In such cases, it would be **desirable**[desired] that each functional unit is well modularized so that the users can develop, improve, debug and use each unit independently from the other modules. Galatea provides a basic module management architecture to satisfy these requirements for research and development.

2.4 Open-source free software

The technology used for creating the toolkit is still **insufficient to**[not enough to] achieve human-like conversation. Therefore it is desired that not only the creators of the toolkit but also the researchers and developers who use the toolkit would contribute to improve the toolkit further. In that sense, the toolkit should be released as a free software along with the program source code.

There **are**[have been] no existing ASDA softwares so far satisfying all of the requirements described above.

3 Toolkit Design and Outline

The basic agent system using Galatea consists of five functional modules: speech recognizer, speech synthesizer, facial animation synthesizer, agent manager which works as an inter-module communication manager, and task (dialog) manager. In addition, Galatea prepares the prototyping tool for coding

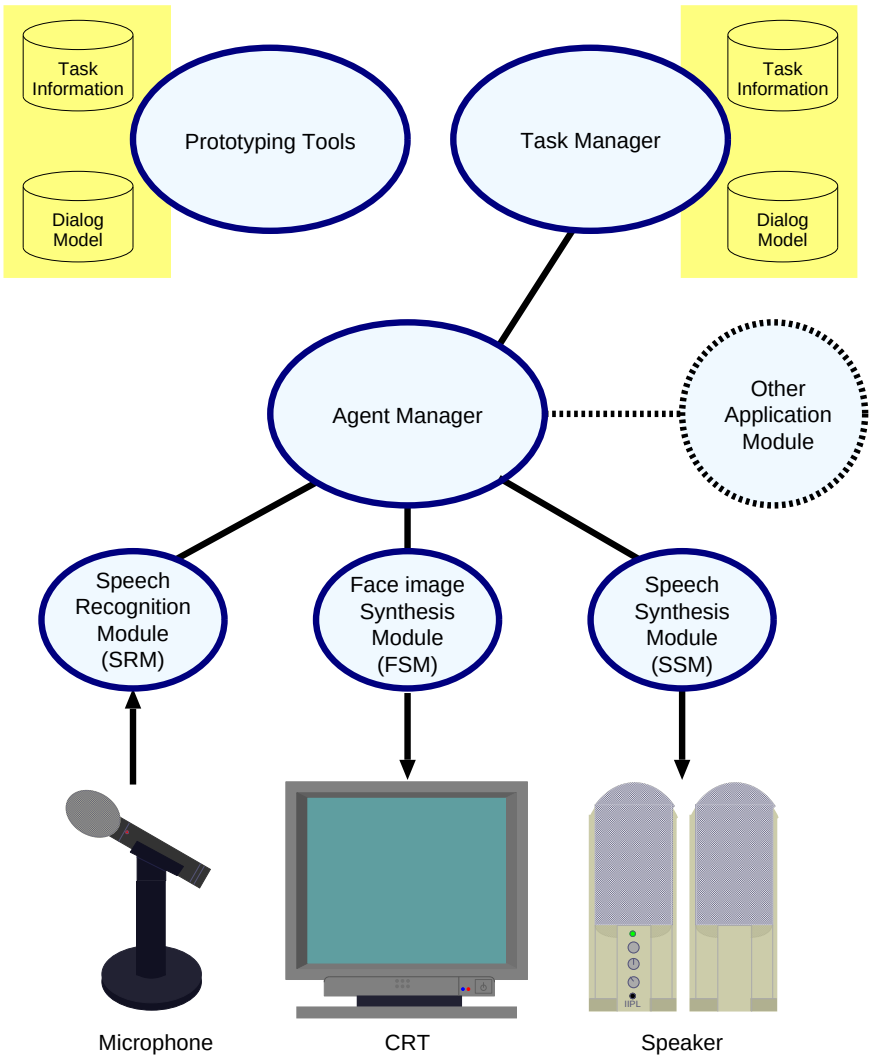


Fig. 1. System Architecture of Galatea

the dialog scenario easily. Fig. 1 shows the basic module architecture of the Galatea toolkit. **In Galatea toolkit, the functional units are independently modularized. Input/Output devices are directly managed in the module. The agent manager controls inter-module communication. If you want to add a new function, you implement a new module with a new function and a new module connects to the agent manager. The dialog manager communicates the agent manager to achieve the dialog tasks based on the database of dialog**

scenario. The prototyping tool, which supports the database creation of dialog scenario, works independently of the agent manager. In this section, we discuss the design of Galatea and **the functionality of its modules**[its module functionality].

3.1 Speech recognition module (SRM)

When constructing an ASDA system, the speech recognition module (SRM) used is required to have the following functions:

- Be able to accept various styles of input and output; For example, accepting a multiple format for grammar representation and outputting incremental recognition results
- Be able to change parameters and resources for recognition flexibly and dynamically; For example, changing grammar by request from external modules during dialog sessions
- Be able to control a recognition engine flexibly and dynamically; For example, stopping to recognize user's utterance, and then restarting

To meet the above requirements, we implemented SRM in the configuration shown in Fig. 2.

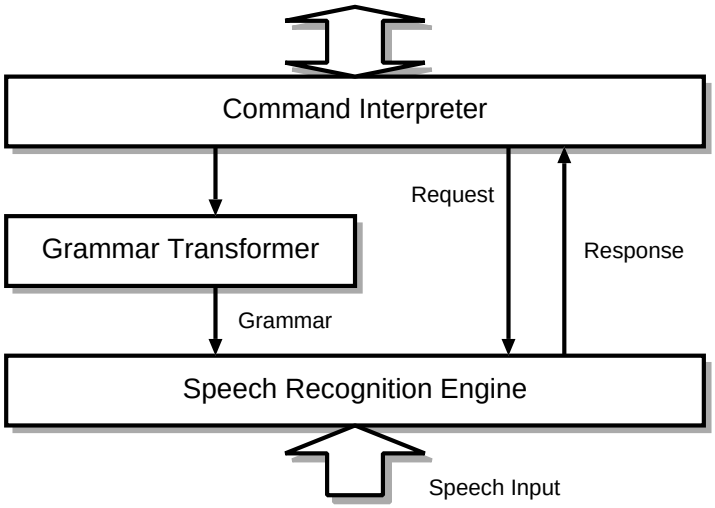


Fig. 2. Speech Recognition Module (SRM)

SRM consists of three submodules: the command interpreter, the speech recognition engine, and the grammar transformer. This configuration was designed not to drop the communication events and speech input events that

occur asynchronously by dividing the command processing and speech processing and dispatching them to exclusive processes. The configuration also contributes to concealing the speech recognition engine from other modules.

We prepared "Julian" as the standard speech recognition module, but all the modules are changeable if a module implements the interface and meets requirements such as accepting grammar written in context free grammar(CFG) or the same class language.

Major interfaces of SRM are as follows:

- Outputs
  - Recognition result
    SRM returns N-best multiple results after an entire utterance. Also, Julian supports the incremental output of recognition results during utterances. Recognition results are formatted in XML and include word sequences, a time stamp, score, lexical information of each word, such as parts of speech, phoneme sequence, and acoustic information of each word, such as duration, and average likelihood of acoustic models.
  - Engine Status
    SRM returns the engine status of speech input, such as "busy", "waiting", if requested by other modules.
- Control Command
  SRM can reload grammar through the Command Interpreter at other modules' request. If SRM is busy, it inserts sent grammar into a queue, and it load the grammars into the recognition engine when it finishes the recognition of each utterance. SRM can also change the settings of the speech recognition engine at any time.
- Grammar Representation
  The grammar that SRM accepts is specified by the XML-based representation. The representation **complies to the**[is made referred as] Speech Recognition Grammar Specification, which was specified by W3C.
  The syntax consists of definitions and sequences of "tokens" and "rules". We extend the token tag by adding a "phoneme" or "syllable" tag which represents the pronunciation of a word. The Grammar transformer transforms the XML grammar into a format that is accepted by the Speech Recognition Engine. It is developed by using XSLT [11], which is a XML transformation technology. Therefore it is easy to exchange the speech recognition engine.

3.2 Speech synthesis module (SSM)

Speech synthesis module (SSM) consists of four sub-modules and its configuration is shown in Fig. 4. The command interpreter receives an input

command from the agent manager and invokes sub-processes according to the command. The text analyzer looks up the dictionary to decompose input text data into morphemes, and provides the waveform generation engine with linguistic information including pronunciation, accent type, part of speech, and so on. The waveform generation engine produces sequences of speech parameters and converts them into synthetic speech waveform. The speech output sub-module outputs the synthetic speech waveform.

To realize customizable speech synthesis module, the module has to accept arbitrary Japanese texts including both "Kanji" (Chinese) and "Kana" characters, and synthesize speech with a human voice clearly in a specified speaking style. Tags embedded in the text specify the speaking style according to the JEIDA-62-2000, which is a description scheme of text for Japanese speech synthesis and is standardized by the Japan Electronic Industry Development Association (JEIDA) [12]. Fig. 3 is a sample text described with JEIDA-62-2000. The speech synthesis for a spoken dialog system is required to generate various types of prosody according to the **user's intention**[system's intention]. The task manager can describe spoken messages using the JEIDA-62-2000 tags to control prosodic parameters. For example,
`<RATE SPEED="`n`"> ... </RATE>`
lengthens the duration of tagged words by n times. For power and F0,
`<VOLUME LEVEL="`n`"> ... </VOLUME>`
`<PITCH LEVEL="`n`"> ... </PITCH>`
change the power and F0 in the same manner, respectively. The pronunciation and the accent type can also be assigned to words that are not found in the dictionary, such as task-specific proper nouns. Input text written with "Kanji" and "Kana" characters and optional embedded tags is analyzed by the text analyzer, which is implemented with a free Japanese morphological analysis system, ChaSen [13], and a newly developed dictionary.

The waveform generation engine in SSM is an HMM-based speech synthesizer, that simultaneously models spectrum, F0 and duration in a unified framework of HMM (Hidden Markov Model) [14, 15]. HMM is one of modeling techniques for a time sequence of the parameter vector. An HMM model probabilistically generates the parameter vector based on the state transition. HMM can be used for a pattern recognition, especially for the speech recognition, by selecting the most probable model among models of the class for observed parameter vectors. On the other hand, HMM can be also a generator of a time sequence of the feature vector. In the speech synthesis, a HMM sequence represents the phoneme sequence of a sentence, and it generates the most probable time sequence of the feature vector. The HMM-based speech synthesis has an advantage of voice quality control over waveform concatenation approaches. Speaker adaptation techniques in HMM-based speech recognition can be utilized for voice conversion in the HMM-based speech synthesis [16]. Such techniques enable us to easily prepare various types of speakers in the speech synthesis system. The `<VOICE>` tag changes the speaker of the SSM synthesizer even for partial words in an utterance.

The SSM module serves another important function to provide a mechanism for synchronizing the lip movement with speech, which is called "lip-sync". The employed mechanism is based on the sharing of each timing and duration information of phoneme in the speech, which is going to be uttered, between the SSM and the FSM (facial image synthesis module).

Finally, SSM can interrupt speech output to cope with the barge-in by the user of the dialog system. This is also important to realize natural dialog between the human and the machine. When the speech output is interrupted, SSM reports the phoneme sequence of words, which the user is expected to listen, to the agent manager.

```
<SPEECH> <VOICE OPTIONAL="male1">
Kore wa <PRON SYM="ai pi: e:">IPA</PRON>no purojekuto de
('This is')                              ('of')('in the project')
kaihatsusareta <EMPH>taiwa</EMPH>onsei gousei sisutemu desu.
('developed')   ('dialogue') ('speech synthesis system')
</VOICE> </SPEECH>
```

Fig. 3. A sample of input text for the speech synthesis module. (The input text is originally written in *Kanji* and *Kana* characters. Note that this example is rewritten in roman characters with English translation in the parentheses just for the readability.)

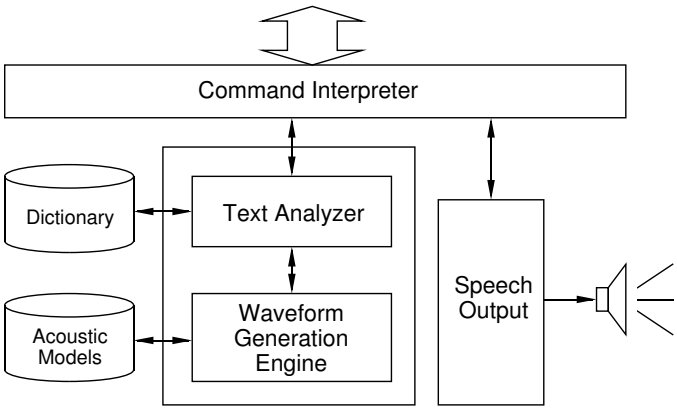


Fig. 4. Speech synthesis module

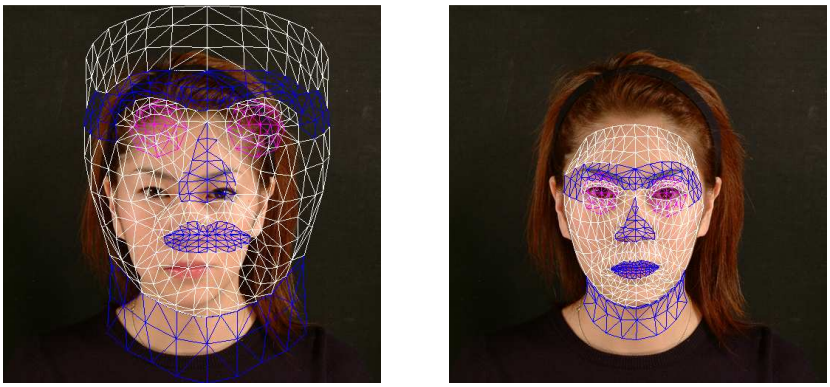
3.3 Facial image synthesis module (FSM)

FSM is the software package to support high quality facial image synthesis, animation control and precise lip-synchronization with synthetic and natural

voice. To customize the face model, a graphical user interface is equipped to fit a generic face wire frame model onto a frontal face snap shot image. Each action unit of FACS [17] is defined on this generic model and stereo type facial expression can be synthesized by combination of these action units. **FACS is an objective method for quantifying scheme that codes the facial muscular movements in terms of 44 action units.** Also **idle**[autonomous] actions like blinking and nodding can be generated. Lip movement in an utterance is controlled by VISEME and duration. Facial animation is expressed easily by a simple script.

Customizing the Face Model

To customize the face model only by snap shot, a generic face model is manually adjusted to the frontal face image. A graphical user interface helps to shorten the time to complete this fitting process. Fig. 5 shows the image before fitting and after fitting.



a) Before fitting **b) After fitting**

Fig. 5. Model fitting by GUI tool

Firstly, four points located on two corners of the sides around temple, bottom of nose and top of chin are adjusted and then face features are decided roughly. Secondly, four points around each eye and center of eye ball are decided, contour of eyelid and mouth and nose position are decided by moving control points by manual operation. Finally, the outline of the face is decided and the hair model is fitted. Then the personal face model is completely generated. In the preview window, fitting status of the face model is confirmed by rotating face and make a facial expression (Fig. 6). The eyeball can be selected in color and size.

This model has a generic oral and teeth tongue model and they are controlled in the utterance process. After a 5 minutes fitting process, any facial

Fig. 6. Preview window

expression with texture mapping can be synthesized by combination of action units of FACS which is predefined in the generic face model.

Facial Action Control

To control facial action, action units of FACS and basic mouth shape of VISEME are predefined in the generic face model.

Designing Mouth Shape

A typical mouth shape can be easily edited by the mouth shape-editing tool. A specific mouth shape is decided by controlling 17 parameters about the positions of lip parts. These parameters are controlled by a slider on the screen and mouth shape can be checked interactively in a preview window. Typical

vowel mouth shapes are shown in Fig. 7. All mouth shapes for VISEME in English and Japanese are already predefined.

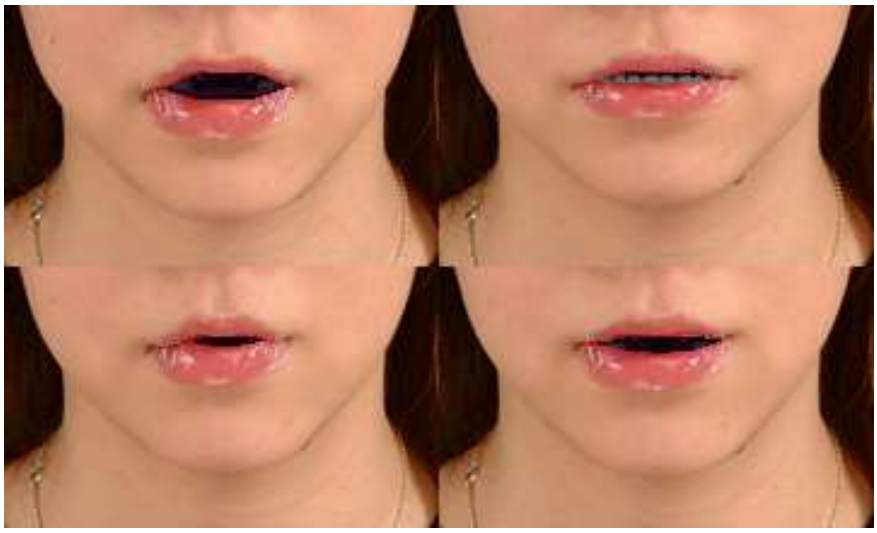


Fig. 7. Example of typical vowel mouth shapes (upper-left: vowel of "a", upper-right: vowel of "i", lower-left: vowel of "u", lower-right: vowel of "e")

Designing Facial Expression

Facial expression is generated by the combination of action units (AU). These AUs **control the**[are] basic movement of face like inner brow raise (AU1), upper lip raiser (AU10) etc. and are composed of 44 units corresponding to each facial muscle movement. Fig. 8 shows examples of typical expressions.

3.4 Module integration and customization tools

Agent manager

The Agent Manager (AM) serves as an integrator of all the modules of the ASDA system. One of its main functions is to play a central role of communication where every message from a module is sent to another module with the help of the AM. Here, the AM works like a hub in the Galaxy-II system [18]. Another essential function of the AM is to work as a synchronization manager between speech synthesis and facial image animation to achieve the precise lip-sync.

The AM consists of two functional layers: the Direct Control Layer (AM-DCL) and the Macro Control Layer (AM-MCL). Fig. 9 shows a schematic

Fig. 8. Example of typical expressions (upper-left: happiness, upper-right: sadness, lower-left: anger, lower-right: fear)

representation of the relationship between the AM and the various modules. The AM-DCL works as a dispatcher receiving commands from a module and forwarding them to the designated module. On the other hand the AM-MCL is a macro-command interpreter processing the macro commands mainly issued by the Task Manager (TM). There are mainly two functions for the AM-MCL. The first one is to simply expand each received macro-command in a sequence of commands and send them sequentially to the designated module. The second function is to process macro-commands that require more complicated processing than just expanding the commands. This happens in the case where more than one module is involved. Currently, the lip synchronization process is realized by a macro command and an example will be given in Section 4

Virtual Machine model

As previously described, the AM works as a hub through which every module communicates with each other. It is desired that every module has a com-

mon communication interface so that the AM can **be connected to**[make connection with] each module regardless of the interface used in the module. Furthermore, having a common interface reduces the effort of understanding and developing module dependent interfaces. For this purpose a virtual machine (VM) model is employed, where the module interface is modeled as a machine with slots, each of which has a value and attribute controlled by a common command set. Each slot can be regarded as a switch or dial to control the operation or a meter to indicate machine status. Fig. 10 illustrates the communication between the AM and a virtual machine model. Changing the slot values by a command corresponds to check or control the running status of the module or the function. For example, **issuing a**[following] command to the speech synthesis module means starting voice synthesis of a given text right now "`set Speak = Now`".

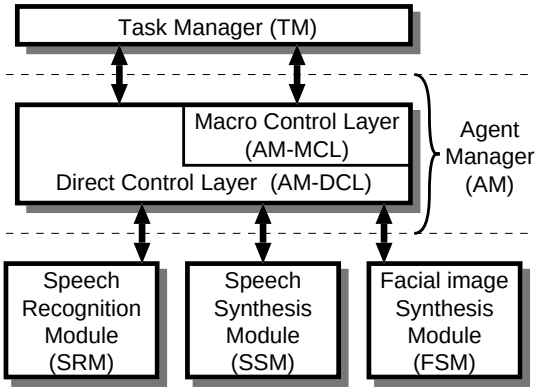


Fig. 9. Basic configuration of the AM and Modules

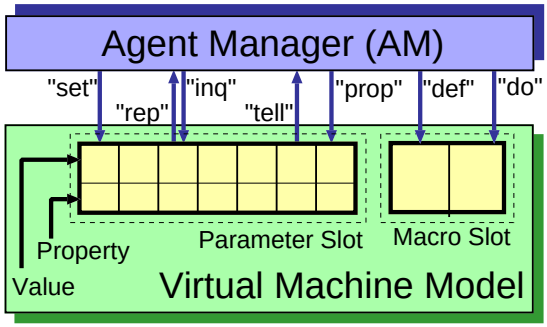


Fig. 10. Relationship between the AM and a virtual machine model

Task manager (TM)

To achieve the better interactions between agent and human, we must learn more about the human's behavior when using the dialog systems. Because the machines' abilities of recognizing and understanding speech or image cannot be compared with the human, imitating the human-to-human interactions such as speech, facial expressions and gestures is not always the royal road. First we will make the definition of the dialog as a set of interactions which can be represented with a dialog description language. The initial specification may have many limitations, but we can build a dialog system based on the specification. Using the system, we can obtain the corpus of human-to-machine dialogs and interactions. Investigation of the corpus may bring the better models of speech understanding, artificial mind, intelligence and the successive interactions. Repetitions of such study can contribute to the better design of the dialog description language, whose capability may gradually increase. Here we discuss the bootstrap design of dialog modeling and its description language which can represent the interactions with spoken language.

Although our VM model can manipulate the conversational input and output events in real-time, it is difficult to write or analyze the time sequence data of such events manually. As a software toolkit, therefore, it is crucial to use a language that can help writing dialogue patterns without concern to the background details of the device controls. It is possible to use the sequence of VM controls to show the time of each output event, the content of the utterances, the changes of facial expressions, etc. Higher level dialogue description language, however, can give the meanings to the series of events, such as "Repeat the question until the user answers to the confirmation."

Conversational phenomena can be explained with the three models as follows: (a) task descriptions, which include the intentions of the participants such as question or giving information, etc., (b) characters of the participants which include the differences of voice and face as well as the differences of the non-verbal communication styles, and (c) the variations among the dialogue sessions. Task description is the most important part in designing and analyzing the human-machine dialogues and there are several de-fact standards in this area. VoiceXML [19] is one of such options.

VoiceXML can cover two types of dialogues: (1) slot-filling type can be a simplified machine initiated dialogue, and (2) database search type can be a mixed-initiative dialogue. There remains, however, another type of the dialogue that cannot be covered with VoiceXML well: (3) explanation type can include navigation of the contents initiated by the user. For this type, we are investigating new style of interface and description language for user initiated interaction [20].

To meet the both demands of convenience for dialogue task designers and the usability for the dialogue system users, it is important to choose the appropriate language for the task description that fits for the dialogue type.

Our goal in developing the Task Manager is that the system can use the several types of dialogue description languages including VoiceXML. This is enabled by dividing the system into the translator, from VoiceXML documents to the intermediate language (Primitive Dialogue Operation Commands, PDOC), and the dialogue controller that interprets the PDOC documents. We also extended the original specification of VoiceXML to add some commands, including the facial expression controls of anthropomorphic dialogue agents. In our Task Manager, PDOC plays the role of low-level language that are close to the device events and sequence control, while the VoiceXML plays the role of the high-level language that handles the task-oriented information and the intentions of the participants. To analyze and model the time sequence data in conversational phenomena, this low-level description is also expected to be useful. Current implementation of the system is tested with tasks of the system-initiative type dialogue.

Making the dialog system which can understand natural language and multimodal input, a Semantic Interpretation Module (SIM) plays the important role. Although there are no such modules at this stage of our development, our toolkit design allows the module to be incorporated. There may be various approaches of SIM implementation, including the statistical models and the semantic parse tree. While the Task Manager concentrates on the management of state transitions or slot-fillings, the SIM can interpret the speech or multimodal input into the dialog acts.

Prototyping Tool

The rapid-prototyping tool named "Galatea Interaction Builder (IB)" runs on a PC and can handle the input modalities of speech, mouse, and keyboard as well as the output modalities of speech (TTS), facial expression, and window display. System developers can implement these input and output modalities without the knowledge of Multi-Modal Interface (MMI) description language by the support of IB [21].

MMI Description Language XISL [22]

XISL is a language for describing MMI scenarios between a user and a system. In principle, a scenario is composed of a sequence of exchanges that contains a set of user's multi-modal inputs and the system's actions corresponding to the inputs. Actions include outputs to a user, simple arithmetic operations, conditional branches, and so on. The details of the XISL specifications are on the web site [23].

Outline of Galatea Interaction Builder (IB)

Fig. 11 shows the workflow of prototyping using Galatea IB. Galatea IB is composed of three modules: a document-server module, a dialog manager, and a front-end module. The document server module holds MMI scenario

(XISL), data (XML), and view style (XSL). The dialog manager interprets an XISL document and controls the flow of dialog by integrating user's input from front-end and executing the system's action corresponding to the inputs. The front-end has an Automatic Speech Recognition (ASR) engine, a facial expression synthesis engine, and a TTS engine developed by Galatea project, as well as a pointing device (mouse) and keyboard.

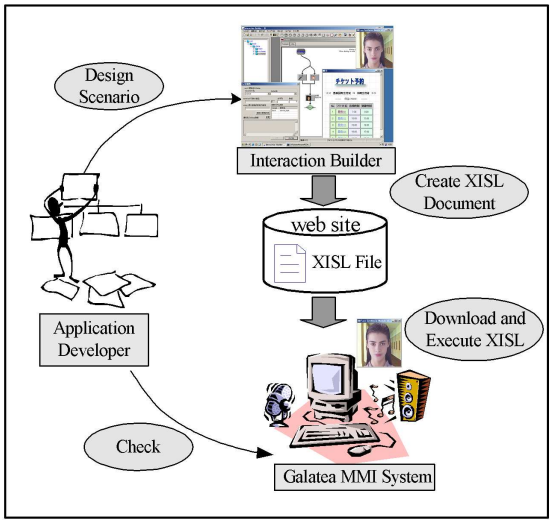


Fig. 11. Workflow of Prototyping Using Galatea Interaction Builder

Rapid-prototyping using Galatea IB

Galatea IB provides GUI designed for domain-specific prototyping that includes applications of airline ticket reservation and secretary services. Fig. 12 shows a screen in prototyping operation. In the following, we describe the facilities of IB according to the assigned numbers in Fig. 12.

The window shown in (1) of Fig. 12 is a scenario view window that presents a state transition diagram of an MMI application. Nodes of the diagram, or MMI components, which correspond to elements of XISL are connected with links. An application developer can easily construct and comprehend the structure of an MMI scenario on this window. The tool bar shown in (2) of Fig. 12 provides all the components such as speech input and output, mouse, and face etc. used in MMI applications. Each button corresponds to a node of state transition diagram. Fig. 13 shows the expanded view of the tool bar. The developer has only to drag one of these buttons and to drop it onto the scenario view window to add a node to the MMI scenario.

The dialog box, shown in (3) of Fig. 12, is popped up when the application developer drops a button on an MMI component of the scenario view window.

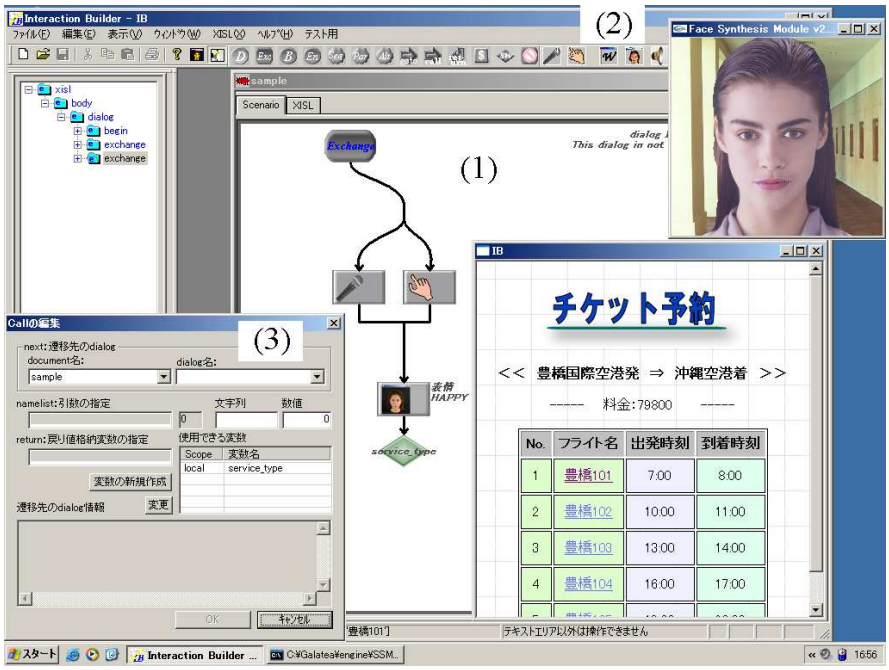


Fig. 12. An example of an IB screen

Fig. 13. Tool bar of IB

The developer has to assign some attributes and values to set up parameters for the MMI component. The developer can confirm the XISL documents by clicking the XISL tab of the scenario view window as shown in Fig. 14. After the confirmation, he/she saves the document and uploads it to a document server module, then tests a prototype system with MMI.

4 Prototype Systems

Using the software toolkit, we have built several experimental ASDA systems to evaluate the toolkit. A screen-shot of the system and an example of a user-system interaction are shown in Fig. 15 and Fig. 16 respectively.

All the tasks employed were very basic, small vocabulary where the number of uttered words is less than 100 and the perplexity is less than 10. The tasks include (1) an echo-back task which repeats what it heard using speech recognition and synthesis, (2) a simple appoint-arranging task which changes facial expressions as the conversation goes on, (3) a fresh food ordering task

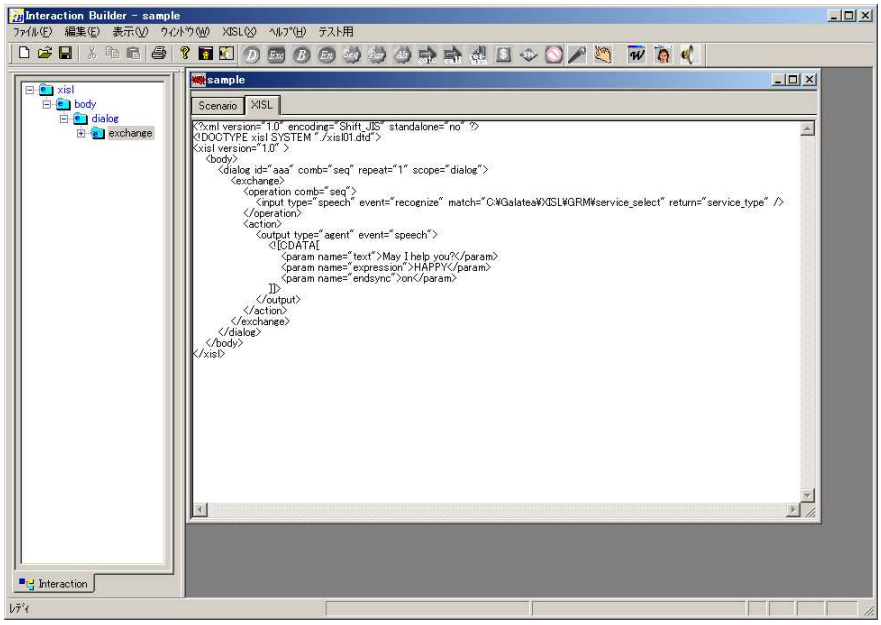


Fig. 14. A generated XISL document

Fig. 15. Screenshot of ASDA

Fig. 16. An example of user-system interaction

that takes orders from customers and **responds**[responses] with "yes" and nodding on the fly.

Those systems consist of the SRM [24], the SSM [14], the FSM [25], the AM, and a simple task-specific TM which was programmed directly with the command set of the toolkit. We implemented the systems on several platforms with different configurations. Fig. 17 shows the hardware configurations. Some of the demonstration movies (in Japanese, unfortunately) are available in our web site [1, 2].

Fig. 18 shows an example of how the AM and related modules work in the echo-back task. However, the FSM and lip-synchronization mechanism have been omitted in the figure for **simplicity**[brevity]. Here, the macro commands, [which is] introduced in 3.4, are used in the procedures 3 and 4 to achieve lip-synchronization between the speech and animation. Fig. 19 shows the sequence of commands involved in this lip-synchronization process.

Note that the modules operate in parallel and thus the speech recognition process is active while the agent is speaking. As a result, we confirmed that the system responded to the users quickly, [at the same time] face animation and synthesized voice were synchronized. However, in this case, we assumed **an ideal**[that the ideal] environment **where**[that] the results of speech recognition are not influenced by the output of speech synthesis.

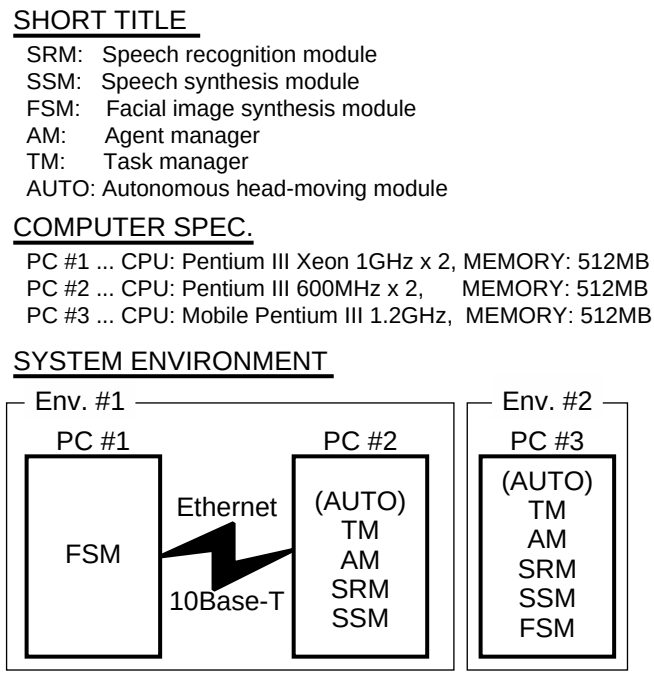


Fig. 17. Hardware configuration of the ASDA

5 Discussion

This section describes the current **development**[developing] status of the software toolkit and discusses further improvement.

5.1 Customization features

In SRM, multi-grammar support has been realized where grammars can be changed instantly, and those grammars are easy to customize by means of a supporting software tool.

The SSM can synthesize speech from arbitrary text sentences of mixed Kanji and Kana (Chinese characters and phonetic script), with customizable prosody. Though speaker adaptation has not been implemented, the employed HMM-based approach is promising in case of speaker adaptation [16, 26].

The FSM synthesizes 3D realistic facial animations from a single snapshot of a person's face by fitting a wire-frame model to a 2D picture. A software tool is provided to help fitting a standard wire-frame model to the input picture, whose manually fitting operation takes normally 10 minutes. Once the fitting is completed, one can get realistic 3D facial animation of the person whose motion, including blinking and facial expression, is easily and precisely controllable by commands in real time. Comparing to the **existing cartoon**

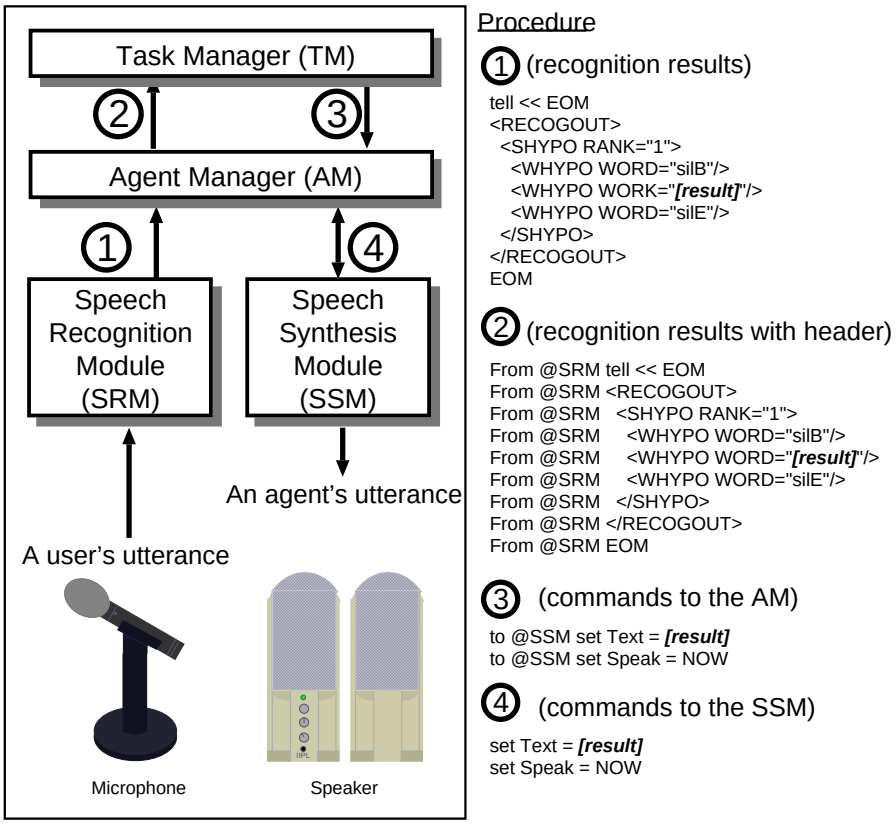


Fig. 18. An example of echo-back processing task

based[cartoon based existing] approaches where the number of characters is very limited, the proposed framework enables to generate facial animations of an almost unlimited number of characters as far as facial pictures are provided.

5.2 Software Modularity of functional units

As is described in the previous section, the virtual machine model enables highly modularity of each functional units such as SRM, SSM and FSM. Furthermore, the communication interface based on the UNIX standard I/O stream helps to develop and debug software modules easily.

5.3 Achievement of natural spoken dialog

Although the implemented mechanism for lip-sync contributes to enhance the naturalness of the synthetic facial animation, a number of issues are yet to be implemented to make the agent behave like a human. For example, humans

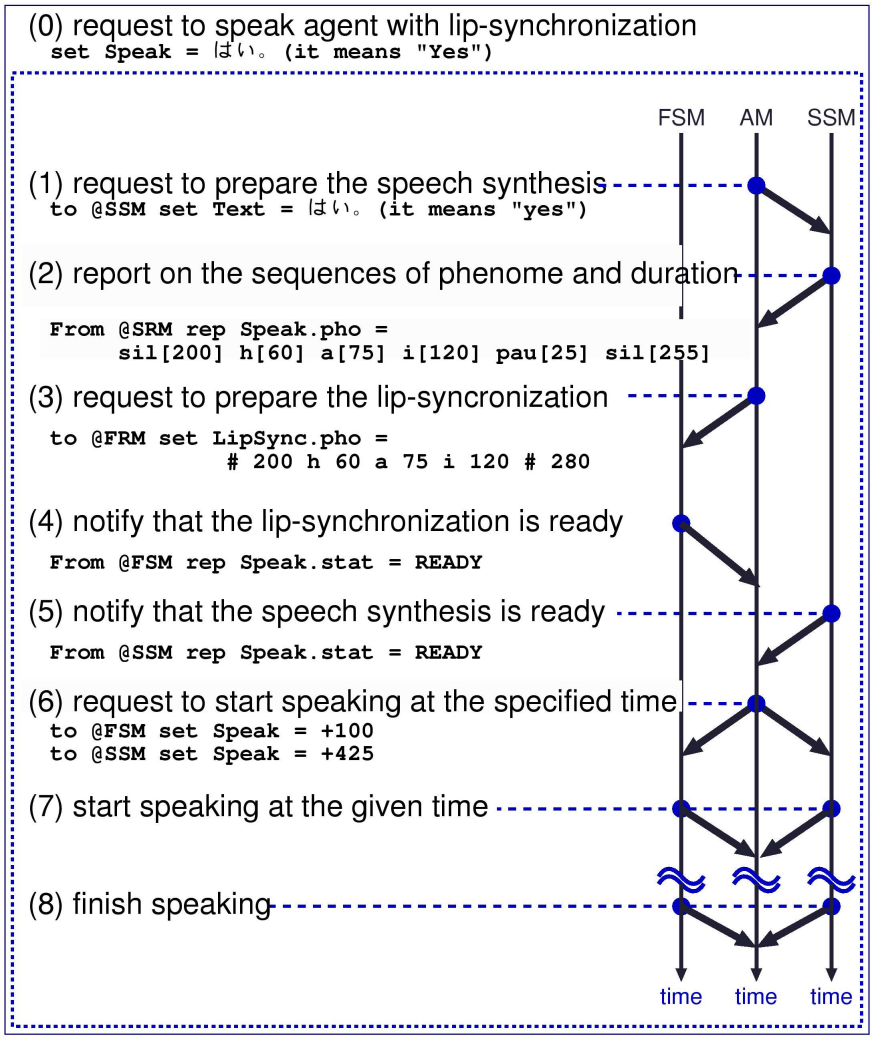


Fig. 19. Processing flow among the AM, the SSM, and the FSM when agent speaks (an example of processing in the AM)

move their heads while they are speaking. Besides the facial animation, **real-time**[realtimeness of] conversation is another crucial factor for the agent's naturalness as described in Section 2.2. A simple mechanism for incremental speech recognition has been implemented in the SRM. The mechanism provides frame-synchronous temporal candidates giving maximum scores at the moment before observing the end of utterance. These incremental recognition results will help to achieve interactive spoken dialog including nodding.

5.4 Related Works

Several attempts have been made to develop ASDA toolkits. Among them, the CSLU toolkit [9] is most similar to our toolkit. The CSLU toolkit provides a modular, open architecture supporting distributed, cross-platform, client/server-based networking. It includes interfaces for standard telephony, audio devices, and software interfaces for speech recognition. It also includes text-to-speech and animation components. This flexible environment makes it possible to easily integrate new components and to develop scalable, portable speech-related applications. Although the target of both of the toolkits is similar, function-wise and implementation-wise they are different. Compared to the speech recognizer and speech synthesizer of the CSLU toolkit that support several European languages, our toolkit supports Japanese language. The TTS in the CSLU toolkit is based on "unit selection and concatenation synthesis" from natural speech. It is a data-driven and *non* model-based approach. However, the TTS in our toolkit employs the HMM-based synthesis that is a data-driven and model-based approach. The different approaches give different characteristics to TTS. Generally speaking, the model-based TTS requires less training samples and it can control speech more easily than the non model-based TTS at the **cost**[expense] of speech quality.

Similar system architectures for distributed computing environments are employed in the Galaxy-II [18] of DARPA Communicator [10], the SRI Open Agent Architecture (OAA) [27], and our toolkit. Each of them have a central module called "Hub", "facilitator" and Agent Manager (AM), respectively. If compared to the existing systems which employ a large number of commands, our toolkit is more compact and simpler and it has only eight commands and two identifiers so that the programmers can understand and use the toolkit easily.

6 Conclusions

The design and architecture of a software toolkit for building an easy to customize anthropomorphic spoken dialog agent (ASDA) has been presented in this chapter. A human-like spoken dialog agent is one of the promising man-machine interfaces for the next generation. The beta-version of the software toolkit described in this paper will be released publicly in the middle of 2003. However, a number of factors are to be improved. Because of the high modularity and simple communication architecture employed in the toolkit, we hope that it would speed up the researches and application development based on ASDA, and as a result the toolkit would be upgraded.

References

1. Galatea Toolkit. http://iipl.jaist.ac.jp/IPA/.

2. Galatea Toolkit. http://hil.t.u-tokyo.ac.jp/~galatea/.
3. Gustafson, J., Lindberg, N., Lundeberg, M.: The August Spoken Dialogue System, EuroSpeech, pp. 1151–1154 (1999).
4. Julia, L., Cheyer, A.: Is Talking To Virtual More Realistic?, EuroSpeech, pp. 1719–1722 (1999).
5. Dohi, H., Ishizuka, M.: Visual Software Agent: A Realistic Face-to-Face Style Interface connected with WWW/Netscape, IJCAI Workshop on Intelligent Multimodal Systems, pp. 17–22 (1997).
6. Ushida, H., Hirayama, Y., Nakajima, H.: Emotion Model for Life-like Agent and its Evaluation, AAAI-98, pp. 62–69 (1998).
7. Sakamoto, K., Hinode, H., Watanuki, K., Seki, S., Kiyama, J., Togawa, F.: A Responce Model for a CG Character Based on Timing of Interactions in a Multimodal Human Interface, IUI-97, pp. 257–260 (1997).
8. Cassell, J., Bickmore, T., Campbell, L., Chang, K., Vilhjálmsson, H., Yan, H.: Requirements for an architecture for embodied conversational characters, Proceedings of Computer Animation and Simulation '99 (Eurographics Series) (Eds. by Thalmann, D., Thalmann, N.), pp. 109–122 (1999).
9. Sutton, S., Cole, R.: Universal speech tools: the cslu toolkit, Proceedings of the International Conference on Spoken Language Processing(ICSLP), pp. 3221–3224 (1998).
10. DARPA, : DARPA Communicator Program (1998). http://fofoca.mitre.org/.
11. XSLT, : XSL Transformations (XSLT) Version 1.0 (1999). http://www.w3.org/TR/xslt.
12. JEIDA, : Standard of symbols for japanese text-to-speech synthesizer, JEIDA-62-2000 (2000).
13. Morphological Analyzer ChaSen. http://chasen.aist-nara.ac.jp/index.html.en.
14. Yoshimura, T., Tokuda, K., Masuko, T., Kobayashi, T., Kitamura, T.: Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis, EuroSpeech, Vol. 5, pp. 2347–2350 (1999).
15. HMM-Based Speech Synthesis Toolkit (HTS). http://hts.ics.nitech.ac.jp/.
16. Tamura, M., Masuko, T., Tokuda, K., Kobayashi, T.: Adaptation of pitch and spectrum for HMM-based speech synthesis using MLLR, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Vol. 2, pp. 805–808 (2001).
17. P.Ekman, , W.V.Friesen, : "Facial Action Coding System(FACS): A Technique for the Measurement of Facial Action", Consulting Psychologists Press (1978).
18. Seneff, S., Hurley, E., Lau, R., Pao, C., Schmid, P., Zue, V.: GALAXY-II: A Referece Architecture for Conversational System Development, ICSLP-1998, pp. 931–934 (1998).
19. VoiceXML, : Voice eXtensible Markup Language VoiceXML Ver1.0 (2000). http://www.voicexml.org.
20. Nishimoto, T., Araki, M., Niimi, Y.: RadioDoc: A Voice-Accessible Document System, ICSLP2002, pp. 1485–1488 (2002).
21. Adachi, H., Katsurada, K., Yamada, H., Nitta, T.: Development of a Prototyping Tool for MMI Systems, Information Processing Society of Japan, Technical Report 2002-SLP-43 (In Japanese), pp. 7–12 (2002).
22. Katsurada, K., Otani, Y., Nakamura, Y., Kobayashi, S., Yamada, H., Nitta, T.: A modality-independent MMI system architecture, ICSLP2002, pp. 2549–2552 (2002).

23. MMI Description Language XISL. http://www.vox.tutkie.tut.ac.jp/XISL/XISL-E.pdf.
24. Kawahara, T., Kobayashi, T., Takeda, T., Minematsu, N., Itou, K., Yamamoto, M., Utsuro, T., Shikano, K.: Sharable software repository for Japanese large vocabulary continuous speech recognition, ICSLP-98, pp. 3257–3260 (1998).
25. Morishima, S.: Face Analysis and Synthesis, IEEE Siginal Processing Magizine, **18**, 3, pp. 26–34 (2001).
26. Tamura, M., Masuko, T., Tokuda, K., Kobayashi, T.: Text-to-speech synthesis with arbitrary speaker's voice from average voice, Proceedings of European Conference on Speech Communication and Technology, Vol. 1, pp. 345–348 (2001).
27. OAA (The Open Agent Architecture) (2001). http://www.ai.sri.com/~oaa/.