Development of a Toolkit for Spoken Dialog Systems with an Anthropomorphic Agent: Galatea

Kouichi Katsurada* Akinobu Lee† Tatsuya Kawahara‡ Tatsuo Yotsukura§
Shigeo Morishima¶ Takuya Nishimoto‖ Yoichi Yamashita** and Tsuneo Nitta*

* Toyohashi University of Technology, Toyohashi, Aichi 441-8580, Japan
E-mail: {katurada, nitta}@tutkie.tut.ac.jp
† Nagoya Institute of Technology, Nagoya, Aichi 466-8555, Japan
E-mail: ri@nitech.ac.jp
‡ Kyoto University, Kyoto 606-8501, Japan
E-mail: kawahara@i.kyoto-u.ac.jp
§ ATR MIS, Soraku-gun, Kyoto 619-0288, Japan
E-mail: tatsuo.yotsukura@atr.jp
¶ Waseda University, Shinjuku-ku, Tokyo 169-8555, Japan
E-mail: shigeo@waseda.jp
‖ The University of Tokyo, Bunkyo-ku, Tokyo 113-8656, Japan
E-mail: nishi@hil.t.u-tokyo.ac.jp
** Ritsumeikan University, Kusatsu, Shiga 525-8577, Japan
E-mail: yama@media.ritsumei.ac.jp

***Abstract*—The Interactive Speech Technology Consortium (ISTC) has been developing a toolkit called Galatea that comprises four fundamental modules for speech recognition, speech synthesis, face synthesis, and dialog control, that can be used to realize an interface for spoken dialog systems with an anthropomorphic agent. This paper describes the development of the Galatea toolkit and the functions of each module; in addition, it discusses the standardization of the description of multi-modal interactions.**

I. Introduction

Conventional human interface devices include hardware such as a keyboard, mouse, touch device with a stylus pen, display, etc., and the basic software required to control them. In recent years, developments in information technology and robotics have led to humans communicating with a wide variety of devices such as small devices, in addition to conventional PC devices. Since a keyboard cannot be integrated in a small device, a flexible and convenient interface is required to utilize such devices. A speech interface is one of the fundamental technologies that can help in using such devices efficiently. However, speech recognition and speech synthesis are expensive technologies that require a large amount of data for learning models, and the implementation of such systems involves a lot of work. Therefore, it is difficult for developers to develop a speech interface from scratch.

Considering this background, in November 2003, we organized the Interactive Speech Technology Consortium (ISTC)[1] [1] under Information Processing Society of Japan (IPSJ) Special Interest Group of Spoken Language Processing (SIG-SLP)[2], and we have provided fundamental software that is essential for developing speech interaction systems. ISTC has provided and maintained the Galatea toolkit that includes speech recognition, speech synthesis, face synthesis, and agent management modules for speech interaction using an anthropomorphic agent, and have standardized a description for Multi-Modal Interaction (MMI). This paper describes the Galatea toolkit developed in ISTC and describes efforts that are being made to standardize the MMI description.

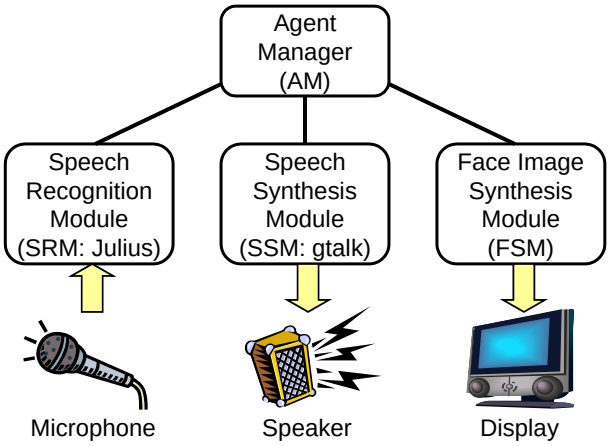

Fig. 1. Basic framework of Galatea toolkit.

[1] http://www.astem.or.jp/istc/index_e.html

[2] http://sig-slp.jp/ (In Japanese)

II. Framework and Features of Galatea

A. Basic Framework

Galatea is a toolkit that is used for developing speech interaction systems. As shown in Fig.1, it comprises four basic modules: Speech Recognition Module (SRM), Speech Synthesis Module (SSM), Face Synthesis Module (FSM), and Agent Manager (AM). The following subsections describe the functions of each module in detail.

B. Speech Recognition Module: SRM

SRM is implemented based on a speech recognition engine Julius[3] together with a wrapper program that manages communication protocol. The wrapper program connects to and communicates with the other modules; in addition, it configures the settings for Julius, activates/inactivates Julius, and controls the actions of Julius. The recognition results obtained by Julius are sent to the other modules through the wrapper program.

We have developed and published the latest versions of Julius and SRM at ISTC. Julius has been improved in terms of functions, performance, and usability for use in speech interaction systems. With regard to Windows Speech API (SAPI) version of Julius, an API for Speech Application Language Tags (SALT) has been developed for embedding a speech interface within a markup language such as HTML.

1) Extension of functions concerning language model: Various types of recognition tasks such as number recognition, dictation, and so on, are assumed in a speech interaction system. To deal with various tasks flexibly, the following extensions were made to its language model and some related sub-modules.

- Support arbitrary length of N-gram
- Support isolated word recognition
- Support word graph and outputting the confusion network
- Integration of Julius and Julian
- Simultaneous recognition using multiple grammars or multiple language models

In particular, in version 4.0 that was published in December 2007 [2], the source code was improved significantly; this included the modularization of word connection constraints. In this version, an engine can use either an N-gram, grammar, or isolated word recognition using only a dictionary. Along with this modification, Julian, a grammar-based engine, was merged with Julius. In addition, user-defined word connection constraints can be built in Julius, and recognition using multiple language models is available in this version. Although some of these functions are not provided by SRM, this version is expected to be useful for constructing flexible speech interaction systems using functions such as switching language models, simultaneous recognition on multiple tasks, and so on.

[3]http://julius.sourceforge.jp/en_index.php

2) Improvement of robustness in inputs: For the purpose of stable and good performance in an actual environment with noise, we improved the following functions.

- Improvement of Voice Activity Detector (VAD)
- Normalization of MAP-CMN and online energy coefficient
- Reduction of delay in speech input by improving the buffering functions

In particular, in VAD, a Gaussian Mixture Model (GMM)-based distinction between speech and non-speech inputs was implemented [3]. GMM-based VAD, in which the scores are calculated at each frame, and decoder-based VAD, which uses word hypotheses, are implemented in this version [4]. These modifications help in avoiding misrecognitions caused by non-speech inputs, and make the recognition engine robust in environments with a lot of noise.

3) Improvement of performance and stability: With regard to the software performance, the following improvements have been made in the new version of Julius.

- Faster execution
- Efficient memory utilization
- Stable performance on Windows
- Integration of source code, documentation

With regard to the faster recognition, a trigonometric function table is used in the calculation of the MFCC, and all divisions are replaced by multiplications in the calculation of the acoustic likelihoods. The latter reduces the process time by 20% and 40% by using a Phonetic Tied-Mixture (PTM) tri-phone and a general tri-phone, respectively. Moreover, the introduction of a lexicon tree, optimization of the N-gram structure, and efficient use of the working memory increases the stability of Julius. The new version of Julius can be compiled on both the Linux and Windows OSs from the same source code. It can also be compiled on the mingw environment. In addition, the documentation has been prepared using Doxygen; it is helpful for understanding the details of the recognition engine.

C. Speech Synthesis Module: SSM

1) GalateaTalk: The speech synthesis module (SSM) in Galatea works as a stand-alone Japanese TTS system and it is called GalateaTalk. GalateaTalk is composed of a main program, *gtalk*, and text processing sub-systems. The *gtalk* program has functions of input command analysis, synthetic speech generation, and speech output. The text processing includes morphological analysis, pronunciation assignment, and accentuation processing, and it is invoked by the *gtalk* program.

The speech synthesis engine in GalateaTalk uses an HMM-based speech synthesis system[5], [6]. The speaker adaptation techniques enable to synthesize various speech with different voice qualities and with rich emotional feelings for the HMM-based speech synthesis[7]. The morphological analysis for speech synthesis outputs pronunciation and accent information

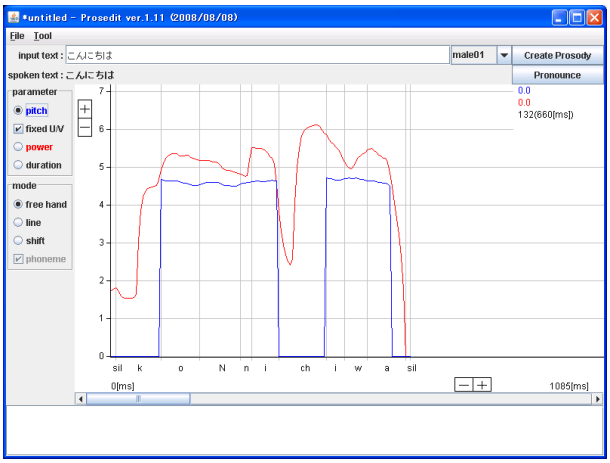

Fig. 2. A screen shot of *ProsEdit*.

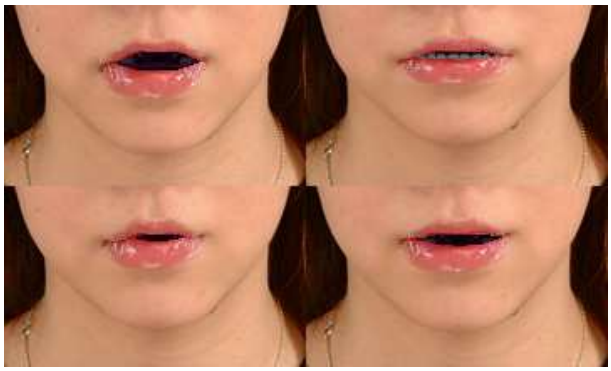

Fig. 3. Japanese vowel mouth shapes.

from input text, and uses a dictionary that is developed by the UniDic project[4].

The GalateaTalk implements the following functions toward flexible speech synthesis in spoken dialog systems.

(1) Stop of synthetic speech output: The GalateaTalk can interrupt the speech output process and stop the speech output when a user of a dialog system gives a barge-in utterance. The phoneme sequence that is presented to the user before the interruption is available in the GalateaTalk.

(2) Selection of speakers: The GalateaTalk can use multiple speakers in an utterance, and it can simply change the voice quality by controlling a frequency warping coefficient.

(3) Flexible control of prosody: Japan Electronic Industry Development Association Standard (JEIDA) recommended "standard of symbols for Japanese Text-To-Speech synthesizer (JEIDA-62-2000)" that includes tags embedded in input text for speech synthesis [8]. The GalateaTalk controls prosody based on this scheme.

(4) Synchronization with facial image output: In a Galatea framework, SSM and FSM share the time structure information of output sentences in terms of phoneme durations and realize the synchronization between speech and facial image. The GalateaTalk determines the phoneme durations and sends them to FSM.

2) Customization Tools: This toolkit provides two tools for realizing the synthesis of various speech: *VoiceMaker* and *ProsEdit*.

The *VoiceMaker* is a tool for automatically building speaker models for GalateaTalk using several tens or hundreds of sentence speech that speakers uttered according to instructions by the *VoiceMaker*.

The *ProsEdit* is a GUI tool for manual manipulation of prosodic parameters (F0, power, and duration) that are generated by GalateaTalk. Fig. 2 shows a screen shot of *ProsEdit*. Some utterances in a dialog consist of words that are independent of the dialog context. We can expect to make dialogs more lively and more active by manually manipulating prosodic parameters for such utterances a priori using *ProsEdit* to express intentions or emotional feelings.

D. Face Image Synthesis Module: FSM

1) Basic function: We developed the face image synthesis module (FSM) that only requires one frontal face image, and can be used by any skill level of users. A user's original agent can be generated by easy adjustment of the frontal face image and the generic wire-frame model using FaceMaker that is GUI-based face-fitting tool. To change the agent's mouth shape, facial expressions, and behaviors we need to set appropriate parameters of the models. Expression modification is made by combining basic expressions (for example: inner brow raiser, cheek raiser, etc.). Therefore, we employed the facial action coding system (FACS) [9], which is an objective method for quantifying the facial movement. FACS is a convenient coding scheme that codes the facial muscular movements by 44 action units (AUs) or their combinations (AU combinations). Mouth shape modification is made by a set of 13 VISEME parameters. VISEME is a generic facial image that can be used to describe a particular sound. It is the visual equivalent of a phoneme or unit of sound in spoken language. Typical Japanese vowel mouth shapes are shown in Fig. 3.

2) Function enhancement: Major function enhancements of FaceMaker and FSM during ISTC are as follows:

(1) English speech animation : FaceMaker and FSM increased the number of VISEME parameters from 13 to 17 and supported speech animation in English. the switching between Japanese and English is achieved by changing a configuration file in FSM.

(2) Accuracy improvement for lip-sync : It is important that the mouth animation made by the Face Synthesis Module and voice made by the Speech Synthesis Module should be synchronized. We developed the synchronization techniques used to provide a realistic

[4]http://www.tokuteicorpus.jp/dist/ (In Japanese)

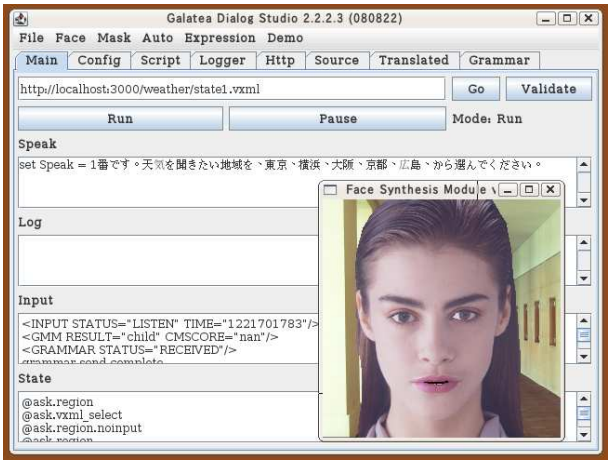

Fig. 4. A screen shot of Galatea Dialog Studio.

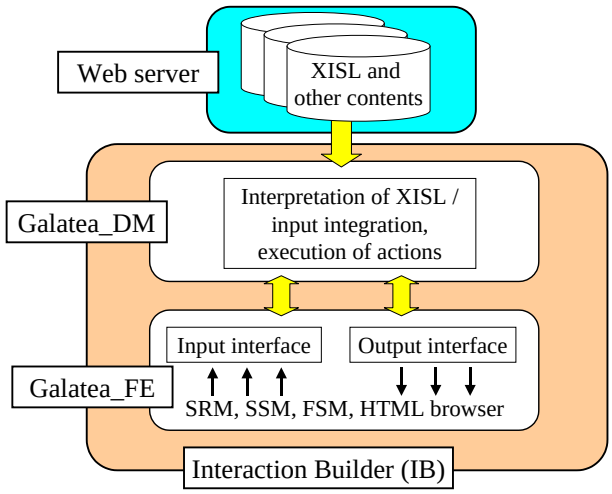

Fig. 5. Architecture of Galatea for Windows.

lip-synchronized animation. This technique provides the local time offset function of FSM and Speech Synthesis Module during a lip-sync animation. In addition, FSM has the function of offset time for starting speech animation.

(3) Output image capture : FSM now has an image capturing function for the lip-sync animation. This function is used to control the frame rate.

(4) Improvement of user interface (FaceMaker) : The GUI-based face-fitting tool that is developed by Galatea project requires complicated operation for a beginner user. The tool employs to introduce a new UI for improving usability using Qt[5]. Furthermore, we proposed a new fitting method that can create an optimally-matched agent's wire-frame model in a short period of time using RBF (Radial Basis Function).

E. Interaction Management Module: AM

1) Linux version: The integration system for Linux adopted a concept of architecture that the virtual machines communicate using its simple commands, so that the developer can easily understand. The Agent Manager, which communicates between sub-modules, was implemented with Perl language. The Dialog Manager (Galatea Dialog Studio) which controls the sub-modules was implemented with Java and Ruby language. It is developed based on VoiceXML, which is one of the standards of voice interface description. Fig. 4 shows a screen shot of Galatea Dialog Studio. Not only the improvements of the sub-modules (such as SSM, SRM and FSM), but also the following improvements were accomplished after the release of 'IPA final version' in 2003.

(1) Ubuntu, one of the latest Linux environments, is newly selected as the execution environment. For the ease of installation, 'deb' packages are provided.

(2) A command is newly introduced to generate a set of project-related files, which contain the configurations such as the options given to the sub-modules. In other words, the project-dependent customization can be done in a more elegant way.

(3) The management between DM and Julius-based SRM is modified to improve the compatibility with the VoiceXML standard and to improve the productivity of the application developers.

(4) More natural face movements and emotional expressions are realized. Utterances and changes of expressions can be performed in parallel.

(5) The graphical user interface for the developers is reinforced. System states, logs, and errors can be viewed more easily.

(6) Compatibility with the latest tools of web application development (such as Ruby on Rails) is improved.

Ruby on Rails (RoR) is expected to contribute greatly to the creation of VoiceXML contents. It gives a number of suggestions for designing multi-modal dialog systems in a sophisticated way, because the application framework of RoR assists the modern methodology including object-oriented design. Supposing an application system has strictly separated MVC (Models, Views and Controllers), the Models and Controllers, which are the modality-independent parts of the system, can be realized effectively, because we utilize the support of RoR for the parts. It is easy to add the VoiceXML-based elements to the original Views of RoR, which is designed for HTML.

It is important to excite the interest of researchers and developers in the tool. The details of the Linux version are shown in the web site[6].

2) Windows version: The Windows version of the AM comprises Galatea_FE that controls the engines, Galatea_DM that manages a dialog flow, and a rapid prototyping tool called Interaction Builder (IB) [10] that is used for developing dialog scenarios. Fig. 5 shows the architecture of the Windows version of the AM.

[5]Qt Cross-Platform Application Framework, http://trolltech.com/products/qt

[6]http://sourceforge.jp/projects/galatea/

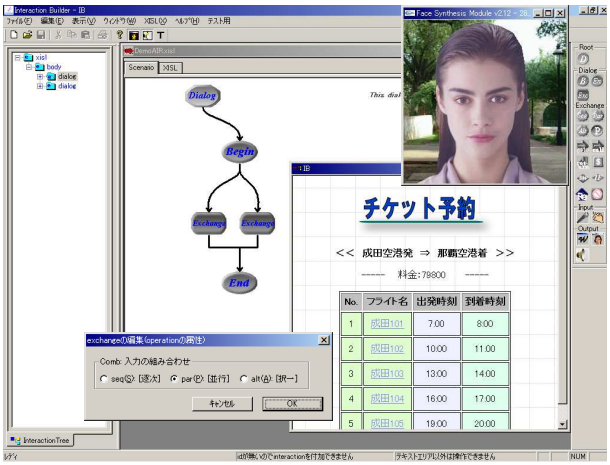

Fig. 6. A screen shot of IB.

Galatea_FE first invokes SRM, SSM, and FSM, and then communicates with them through sockets. It also has a facility to browse web pages, and it can accept multi-modal inputs using both a pointing device and a voice interface.

Galatea_DM controls a dialog flow that is described using the XISL language [11]. XISL is an XML-based MMI description language that controls a dialog flow, multi-modal inputs/outputs, arithmetic operations, and conditional branches. Galatea_DM downloads an XISL document from a web server and interprets the entire document, except for the multi-modal inputs/outputs that are sent to and processed by Galatea_FE.

IB is a tool that is used for generating XISL documents by means of GUI operations. A user can construct an interaction flow by dragging and dropping the dialog component icons. It also provides a wizard function for easily constructing dialogs for some typical flows. Fig. 6 shows a screen shot of IB.

III. Standardization of MMI Description Language for Spoken Dialog System

ISTC organized the MMI description language study WG and engaged in efforts aimed at the standardization of the MMI description language. The WG documented the use cases and requirements of MMI systems and proposed a standard architecture for MMI systems based on a six-layered model [12].

A. Use Cases for MMI System

Based on the MMI systems that have been developed by several companies/universities, we have selected the following eight use cases for consideration.

- Online shopping system
- Directory search using speech
- Site search
- Interaction with a robot
- Negotiation with a dialog agent
- Spoken guidance system
- Area guide
- Setting a destination on a car navigation

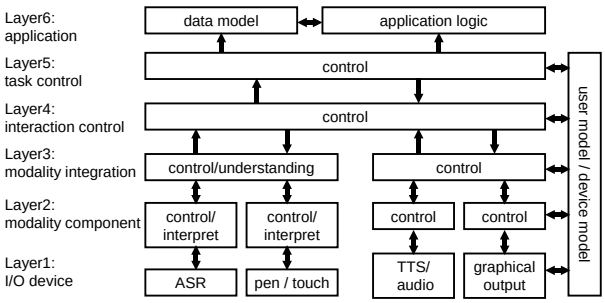

Fig. 7. Six layered model of MMI system.

Each use case contains an interaction scenario, user modality, and some other functions required by the system. The details of the use cases are provided online[7].

B. Requirements for MMI System

From the above use cases, we derived the following requirements for MMI systems.

(1) General requirements
(2) Input modality requirements
(3) Output modality requirements
(4) Architecture, integration and synchronization
(5) Runtimes and deployments
(6) Dialog management
(7) Handling forms and fields
(8) Co-ordination with applications and external modules
(9) User information, environment information
(10) Functions from developer's perspective
(11) Application and session
(12) Use of ECMAScript

(1) to (5) were also included in the requirements documented by the W3C[8], while (6) to (12) are original requirements derived from the above use cases. The requirements are listed online[9].

C. Architecture of MMI System

An MMI system handles various modalities, integrates them, and controls a dialog flow. These processes are too complex to deal with in a system based on a single-layered architecture. Therefore, a two or three-layered model has usually been investigated in some studies (such as the W3C ' s model). In ISTC, we have proposed a six-layered model to enable the development of detailed functions independently. Fig. 7 shows the architecture of this model.

When developing an MMI system based on this model, a developer does not need to implement each layer independently. He/she can combine two or three layers into a single module. Since the layers are segmentized into small modules, developers can construct an MMI system flexibly;

[7] http://www.astem.or.jp/istc/ISTC-SIG-MMI/index.html (In Japanese)
[8] http://www.w3.org/TR/mmi-reqs/
[9] http://www.astem.or.jp/istc/ISTC-SIG-MMI/meeting12/MMI-Requirement3.pdf (In Japanese)

therefore, a prototype of an MMI system can be built easily by implementing only a small part of the system.

IV. Conclusions

We presented the Galatea toolkit for spoken dialog systems using an anthropomorphic agent. In addition, we have described our efforts to standardize the MMI description.

In the future, we will investigate how to localize or internationalize the Galatea toolkit. We will also publish a live CD/DVD versions of Galatea based on Knoppix, and documents and licenses will be arranged to expand to the open-source community. With regard to the MMI architecture, we will publish a document describing its standardization at the Information Technology Standards Commission of Japan (ITSCJ), and plan to propose it as an international standard in collaboration with the W3C MMI-WG. Although the activities of ISTC terminated in March 2009, we would like to publish the toolkit online.

Acknowledgment

We thank the ISTC committee and the members of ISTC for their contributions toward the development of this software.

References

[1] T. Nitta, et al.: “Activitied of Interactive Speech Technology Consortium (ISTC) Targeting Open Software Development for MMI Systems,” Proc. of RO-MAN’04, 2B4 (2004).

[2] A. Lee, “Large Vocabulary Continuous Speech Recognition Engine Julius ver. 4,” IEICE technical report. Speech, 107, 406, pp.307–312 (2007). (In Japanese)

[3] A. Lee, et al., “Noise robust real world spoken dialogue system using GMM based rejection of unintended inputs,” ICSLP2004, pp.847–850 (2004).

[4] H. Sakai, et al., “Voice Activity Detection Applied to Hands-Free Speech Recognition based on Decoding using Acoustic and Language Models”, IEICE technical report. Speech, 107, 116, pp.55-60 (2007). (In Japanese)

[5] T. Masuko, K. Tokuda, T. Kobayashi and S. Imai: “Speech Synthesis Using HMMs with Dynamic Features,” Proc. 1996 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP’96), 1, pp.389–392 (1996).

[6] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, T. Kitamura: “Speaker Interpolation for HMM-based Speech Synthesis System,” J. Acoust. Soc. Jpn. (E), 21, 4, pp.199-206 (2000).

[7] J. Yamagishi, et al.: “Acoustic modeling of speaking styles and emotional expressions in HMM-based speech synthesis,” IEICE Trans. Inf. & Syst., **E88-D**, 3, pp.503–509 (2005).

[8] Japan Electronic Industry Development Association Standard(JEIDA) : ”Standard of Symbols for Japanese Text-To-Speech Synthesizer (JEIDA-62-2000)”, (2000).

[9] P. Ekman and W. V. Friesen: “Manual for the Facial Action Coding System and Action Unit Photographs,” Palo Alto, CA: Consulting Psychological Press (1978).

[10] K. Katsurada, H. Adachi, K. Sato, H. Yamada and T. Nitta, “Interaction Builder: A Rapid Prototyping Tool for Developing Web-Based MMI Applications,” IEICE Trans. Inf. & Syst., **E88-D**, 11, pp.2461–2468 (2005).

[11] K. Katsurada, Y. Nakamura, H. Yamada and T. Nitta, “XISL: A Language for Describing Multimodal Interaction Scenarios,” Proc. of ICMI-03, pp.281–284 (2003).

[12] T. Nitta, K. Katsurada, M. Araki, T. Nishimoto, T. Amakasu and S. Kawamoto, “Proposal of a Hierarchical Architecture for Multimodal Interactive Systems,” IPSJ SIG Technical Reports, 2007-SLP-68, pp.7–12 (2007). (In Japanese)