

A Markup Language for Describing Interactive Humanoid Robot Presentations

*Yoshitaka Nishimura,
Shinichiro Minotsu,
Hiroshi Dohi and Mitsuru Ishizuka*
Graduate School of Information Science
and Technology, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo
113-8656, Japan
{nisshi@mi.ci., mino@mi.ci.,
dohi@mi.ci., ishizuka@}i.u-tokyo.ac.jp

*Mikio Nakano, Kotaro Funakoshi,
Johane Takeuchi, Yuji Hasegawa and
Hiroshi Tsujino*
Honda Research Institute Japan Co., Ltd.
8-1 Honcho, Wako-shi, Saitama
351-0188, Japan
{nakano, funakoshi, Johane.Takeuchi,
yuji.hasegawa,
tsujino}@jp.honda-ri.com

ABSTRACT

This paper presents a multi-modal presentation markup language for humanoid robots, MPML-HR ver. 3.0, which is able to describe presentation contents including speech-based interactions with audiences. Previous versions of MPML-HR do not feature any interaction functionality which dynamically changes the presentation according to the utterances by audiences, although such interaction makes the presentation more effective and understandable. Since MPML-HR ver. 3.0 inherits simple descriptions of previous versions of MPML-HR, the content designer can describe interactive presentations without configuring conventional complicated multi-modal interactive systems.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Voice I/O.

General terms: Design, Human Factors, Language.

Keywords: Presentation, interaction, multimodal system.

INTRODUCTION

Recently many kinds of robots receive much attention. Among them, humanoid robots are expected to be able to work as humans do. Humanoid robots are suitable for a tool for presentation, because they can use many modalities and are attractive. We have been developing Multimodal Presentation Markup Language for Humanoid Robots (MPML-HR) [1], for easily describing humanoid robot presentations. While complicated computer programs used to be required to operate humanoid robots, MPML-HR has made it possible for non-experts to create presentation contents for humanoid robots.

Previous versions of MPML-HR, however, do not feature

interactions with audiences. Interactions with audiences, which include spoken questions about presentation and speech-based control of the presentation flow, make humanoid robot presentations more efficient and understandable. Although MPML [2], based on which MPML-HR was designed, features simple interactive functions, the timing of the audience's speech needs to be specified a priori.

Applying speech-based or multi-modal human-machine interactive systems for a specific domain is one approach. However, it is not easy for people who do not have expertise to configure such systems. Several markup languages for such interactive systems have been therefore developed. Examples are VoiceXML [3] and XISL [4]. However, since these are not designed for presentations, scripts for interactive presentations would become complicated. Other attempts have been made to make it easy to configure multi-modal interaction systems for specific task domains (e.g., [5, 6]), they suffer from the same problem.

Since we focus on presentations, we take an approach different from those languages for interactive systems. We extend the previous versions of MPML-HR for interactive presentation, resulting in MPML-HR ver. 3.0. Since MPML-HR ver. 3.0 inherits simple descriptions of previous version of MPML-HR, content designers can describe interactive presentations without configuring conventional complicated multi-modal interactive systems. The interpreter for MPML-HR ver. 3.0 is built on a model of a conversational robot intelligence, RIME (Robot Intelligence based on Multiple Experts) [7]. This model features multiple experts, each of which is responsible for certain kinds of tasks such as a dialogue in a specific domain and a task involving physical behaviors. We have developed an expert which is specialized for MPML-HR presentations. An MPML-HR script is transformed into knowledge used by the expert. The speech interaction functionality of an MPML-HR expert is shared with other experts, although a speech recognition language model particular to each MPML-HR script is used. We found that RIME's control mechanism is suitable for implementing MPML-HR ver. 3.0. Thanks also to RIME, MPML-HR presentations can be easily combined with other tasks such as

task-oriented and non-task-oriented dialogues.

MARKUP LANGUAGE FOR INTERACTIVE PRESENTATION

This section describes MPML-HR ver. 3.0 after briefly explaining MPML which is for animated agents and previous versions of MPML-HR.

MPML and MPML-HR

MPML is a scripting language that allows non-experts to create the contents of multimodal presentations by on-screen animated agents. Since it is a middle-level language, it is independent of a low-level agent controller. It is based on XML, and it features many kinds of tags to control agents' positions, movements, gestures, and emotional expressions, as well as changing images on the screen. MPML-HR is an extension to MPML for humanoid robots, which performs presentation using a screen. An example of this extension is the `point` tag, which enables robots in the real space to point to a position on the screen.

Extensions for Interaction

In this research, we deal with audiences' utterances such as requesting to repeat previous explanations or to skip some contents. In order to cope with such interruptions, MPML-HR needs to be extended so that the content designer can specify interruption utterance patterns for speech recognition and which point the presentation should move to for each type of interruption.

To make it possible to specify the point to move to, the presentation content needs to be divided into short pieces so that the robot can move to the head of one of such short pieces. To divide contents, we use the `page` tag in MPML-HR scripts. Each part embraced by `page` corresponds to one slide on the screen. Since each slide can be considered to be one topic in the presentation, this tag is suitable for indicating the point to move to when an audience interrupts.

One issue in handling audience interruptions is that there can be speech recognition errors. If the robot reacts to erroneous speech recognition results, the presentation will move to an unintended point. We therefore incorporate two functionalities into MPML-HR ver. 3.0. One is to ask the audience back what he/she said when the speech recognition confidence is low. The other is to go back to the original point in the presentation when the presentation moved by a speech recognition error and the audience makes an utterance to correct it.

An MPML-HR ver. 3.0 script consists of two parts: the `body` part, which specifies the presentation content, and the `interaction` part (Figure 1 (1)), which specifies how to handle interruptions. The `grammar` tags (Figure 1 (2)) specify the recognition grammars (“[]” means optional and “(|)” means alternative). When an audience interruption utterance matches one of the grammars, the robot executes the inside of the `grammar` tag. The `recog` tags embracing `grammar` tags (Figure 1 (3)) specify in which part of the presentation these grammars are effective; in other words, when interruption utterances matching these grammars are handled (i.e., not neglected). The `grammar` tags not embraced by any `recog` tags are effective throughout the pre-

```
<?xml version="1.0" encoding="UTF-8" ?>
<mpml>
<body>
  <page ref="/slide.ppt#1" id="first">
    <synch>
      <speak>Tomorrow, it will be fine in Tokyo.</speak>
      <play act="victory" />
    </synch>
  </page>
  <page ref="/slide.ppt#2" id="second">
    <speak>The day after tomorrow, it will rain in Tokyo.</speak>
  </page>
  <page ref="/slide.ppt#3" id="last">
    <speak>Yesterday, it was cloudy in Tokyo.</speak>
  </page>
</body>
<interaction>.....(1)
  <grammar regram="No [it is not]">.....(2)
    <goback />
  </grammar>
  <grammar regram="How was yesterday">
    <do page="last" />
  </grammar>
  <recog page="second">.....(3)
    <grammar regram="[Please] (explain talk) the first slide">
      <askback utterance="Did you say something" />.....(4)
      <jump page="first" />
    </grammar>
  </recog>
</interaction>
</mpml>
```

Figure 1: Example MPML-HR ver. 3.0 Script

```
Robot: Tomorrow, it will
User: How was yesterday ?
Robot: [Go to "second" page because of
       a speech recognition error] OK.
Robot: The day after tomorrow,
User: No.
Robot: [Go back to "start" page] Tomorrow, it will be fine
User: How was yesterday ?
Robot: [Go to "last" page] OK.
Robot: Yesterday, it was cloudy in Tokyo.
Robot: [Return to "first" page] The day after tomorrow,
User: Please go to the first slide.
Robot: [Ask back] Did you say something ?
User: Please explain the first slide.
Robot: [Jump to first page] OK.
Robot: Tomorrow, it will be fine in Tokyo.
.....
Robot: Yesterday, it was cloudy in Tokyo.
```

Figure 2: Transcription of an Example Presentation

sation. The `jump` and `go` tags specify how the presentation moves when an interruption utterance is detected. A `jump` tag makes the presentation move to the page specified by the `page` attribute and does not come back to the interruption point, while a `go` tag executes the page specified by the `page` attribute and goes back to the interruption point. The `goback` tag is a command for going back to the original point in the presentation when the presentation moved by a speech recognition error and the audience makes an utterance to correct the erroneous move. The correction utterance is specified by the `grammar` tag embracing the `goback` tag. The `askback` tag (Figure 1 (4)) indicates the robot asks back the audience with its `utterance` attribute when the speech recognition confidence score of the interruption utterance is higher than a threshold (called *asking back threshold*), Figure 2 shows an example interaction enabled by the script in Figure 1.

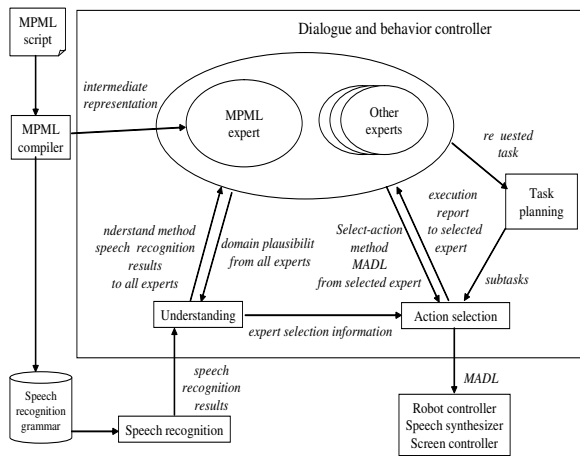


Figure 3: System Architecture

INTERACTIVE PRESENTATION SYSTEM BASED ON A MULTI-EXPERT MODEL FOR CONVERSATIONAL ROBOTS

System Overview

Figure 3 shows the architecture of the MPML-HR ver. 3.0 system. It is built as part of the symbol-level dialogue and behavior controller of the conversational robot. The dialogue and behavior controller selects actions for performing tasks as well as understands recognition results of human speech and decides how to react to it. Its output is in the form of what we call MADL (Multi-modal Action Description Language), which includes one or a combination of speech synthesis commands with text (e.g., “hello”) and symbolic representations of physical action commands (e.g., “gesture hello” and “approach john”), commands to show a slide, and others. The commands in one MADL are performed simultaneously. The dialogue and behavior controller is connected with a speech recognizer, a speech synthesizer, a slide controller, and a robot controller. The current implementation uses Julius/Julian [8] as the speech recognizer, NTT-IT’s FineVoice as the speech synthesizer, and Honda ASIMO as the robot.

Our dialogue and behavior controller is built on RIME (Robot Intelligence based on Multiple Experts) (formerly called MEBDP in [7]), which is a model for the dialogue and behavior controller of conversational robots. This model features multiple experts, each of which is responsible for certain kinds of tasks such as a dialogue in a specific domain and a task using physical behaviors. We have developed an expert which is specialized for MPML presentations. (Hereafter we write MPML instead of MPML-HR ver. 3.0 when it is clear from the context.) An MPML script is compiled by a module called the MPML Compiler into an intermediate representation used by the expert.

Behavior and Dialogue Controller based on Multiple Experts

The dialogue and behavior control subsystem features modules called *experts*, which are specialized for performing certain kinds of subtasks by performing physical actions and engaging in dialogues. Each expert corresponds to a type of

```
<block id="1" next="2">
<interruption-handle recgram="gram_0_1" goto="goback"/>
<interruption-handle recgram="gram_0_2" goto="3_r"/>
<madl>
  <show-slide uri="/slide.ppt#1"/>
</madl>
<madl>
  <utterance>Tomorrow, It will be fine in Tokyo.</utterance>
  <play name="victory"/>
</madl>
</block>
```

Figure 4: An Example of Intermediate Representation

subtasks. When the robot is trying to perform a subtask, the expert for its type is used for selecting actions. Such an expert is called *being in charge*. When a human utterance is received, the expert to become in charge is determined by its understanding results and the context. An expert is a kind of object in the object-oriented programming framework. Each expert has its own internal state, which includes understanding results, grounding status, local action plans, and other information.

There are three modules that coordinate these experts. The *understanding* module dispatches the speech recognition results to each expert and selects the most appropriate expert to take charge. The *action selection* module asks the selected expert to decide actions to perform. The *task planning* module decides which expert should take charge when the robot is performing a task. These three modules run in parallel to handle human interruptions.

Each expert must support several methods for accessing its internal state. We will explain only methods related to MPML Experts. The *initialize* method, which is called when the expert is created, initializes its internal state. The *understand* method is called by the understanding module when the speech recognition result is received and it updates the information state based on human speech recognition results, using domain-dependent sentence patterns for utterance understanding. This method returns a score between 0 and 1 that indicates the plausibility the human utterance should be dealt with by the expert. The *select-action* method, called by the action selection module outputs one action based on the content of the information state. This method is called when the execution of an MADL is finished. If the MADL has a flag to wait for a human utterance, the method call is suspended until a human utterance is detected.

MPML Compiler

The MPML Compiler compiles an MPML script into an intermediate representation. This representation is composed of several `block` elements. Each `block` element corresponds to one of the `page` elements in the `body` in the original MPML script. Several `block` elements for each `page` are created according to the context where the block is executed. For example, the subsequent block is different according to which tag invokes this block either a jump tag or a go tag. Figure 4 shows an example of this representation, which is the compilation of the first page of Figure 1.

The `interruption-handle` elements are created for each grammar for interaction, and specify the grammar ID's and the blocks to move to.

Each of the `madl` elements is MADL, a multi-modal expression which is in the form of the output of the dialogue and behavior controller. A `show-slide` element is a command to show a slide on the screen. An `utterance` element, a `play` element, and a `go-to` element respectively represent commands for making an utterance, performing a gesture, and physically moving to a specified location.

As well as the intermediate representation, a configuration file for the MPML expert and grammar definition files are created. The configuration file holds the list of the grammars, based on which, speech recognition grammar is created. The current version of the system uses one network grammar for speech recognition that combines all the grammar used in all the experts.

MPML Expert

An MPML expert should be created for each presentation. However, all MPML experts share their methods, and only the intermediate representation and grammars are different. That is, all MPML experts are in the same class in the object-oriented programming framework. The MPML expert class was created by implementing methods described above.

The `initialize` method converts the intermediate representation into a stack which stores pairs of an MADL and a list of interruption handling information (grammar ID and `go/jump` destination). It also converts grammars to finite-state automata.

The `select-action` method works as follows. It pops a pair of an MADL and an interruption handling information list from the stack, stores interruption handling information, and sets the active grammar ID list. It then returns the MADL.

The `understand` method checks if the confidence score of the speech recognition result is over a threshold (we call this threshold *rejection threshold*). The rejection threshold is lower than the threshold to determine if the robot should ask back to the audience (*asking back threshold*). If the confidence score is lower than the rejection threshold, the interruption is neglected. Otherwise this method applies the automata for the active grammars to the speech recognition result. If one of the automata successfully parses it, and if the confidence is lower than the asking back threshold, an MADL that contains utterance for asking back is pushed to the top of the stack. In this case, a flag to wait for a human utterance after finishing executing the MADL is set. If the confidence is greater than the asking back threshold, the stack is changed according to the `go/jump` destination corresponding to the grammar. It always returns the score of 1.0 so that the MPML expert takes charge until the presentation finishes or an audience requests to stop the presentation.

Example Presentation

We implemented a system that can perform a guide to Tokyo in Japanese based on an MPML-HR ver. 3.0 script. It features two experts, one is for understanding request to start the presentation, the other is an MPML expert that performs

the presentation. The MPML script has about 160 tags, features six pages, and explains four main places in Tokyo. The number of recognition grammar is 11. Although the interrupting utterance patterns are limited, we confirmed that they are enough to move to any other pages, and that moves are possible any time during the presentation.

CONCLUSIONS AND FUTURE WORK

This paper presented a markup language for interactive humanoid robot presentation, MPML-HR ver. 3.0, and its implementation. It can describe simple interactions during presentations such as audiences' requesting to repeat previous explanations or to skip some contents without configuring conventional complicated multi-modal interactive systems. We found the multi-expert-based robot dialogue and behavior controller is suitable for building the MPML-HR ver. 3.0 interpreter. In addition, integrating presentations with other kinds of tasks is simple. Although the current implementation works only with a robot, it is possible to augment it so that an on-screen animated agent can be used.

Since the current implementation is based on confidence scores for speech recognition results, we are planning to improve these scores for more accurate rejection of unreliable recognition results. We are also planning to expand the kind of interactions, while keeping the MPML's simplicity.

REFERENCES

1. Y. Nishimura, K. Kushida, H. Dohi, M. Ishizuka, J. Takeuchi, and H. Tsujino, "Development and psychological evaluation of multimodal presentation markup language for humanoid robots," in *Proc. of IEEE RAS Humanoids*, 2005, pp. 393–398.
2. H. Prendinger, S. Descomps and M. Ishizuka, "MPML:A Markup Language for Controlling the Behavior of Life-like Characters," *Journal of Visual Languages and Computing*, 2004, Vol.15, No.2, pp. 183–203.
3. Voice XML Forum homepage: <http://www.voicexml.org/>
4. K. Katsurada, Y. Nakamura, H. Yamada, T. Nitta, "XISL:A Language for Describing Multimodal Interaction Scenarios," *Proc. of ICMI*, 2003, pp. 281–284.
5. J. Glass and E. Weinstein, "SPEECHBUILDER: Facilitating spoken dialogue system development," in *Proc. Eurospeech*, 2001, pp. 1335–1338.
6. S. Sutton et al., "Universal Speech Tools: The CSLU Toolkit" in *Proc. ICSLP*, 1998, pp. 3221–3224.
7. M. Nakano, Y. Hasegawa, K. Nakadai, T. Nakamura, J. Takeuchi, T. Torii, H. Tsujino, N. Kanda, and H. G. Okuno, "A two-layer model for behavior and dialogue planning in conversational service robots," in *Proc. IEEE/RSJ IROS*, 2005, pp. 1542–1548.
8. T. Kawahara, A. Lee, K. Takeda, K. Itou, and K. Shikano, "Recent progress of open-source LVCSR engine Julius and Japanese model repository," in *Proc. ICSLP*, 2004, pp. 3069–3072.