XISL: A Language for Describing Multimodal Interaction Scenarios

Kouichi Katsurada Yusaku Nakamura Hirobumi Yamada Tsuneo Nitta
Toyohashi University of Technology
1-1 Hibarigaoka, Tempaku-cho, Toyohashi, Aichi 441-8580, JAPAN
+81-532-44-6884
{katurada, nitta}@tutkie.tut.ac.jp {nakamura, yamada}@vox.tutkie.tut.ac.jp

ABSTRACT
This paper outlines the latest version of XISL (eXtensible Interaction Scenario Language). XISL is an XML-based markup language for web-based multimodal interaction systems. XISL enables to describe synchronization of multimodal inputs/outputs, dialog flow/transition, and some other descriptions required for multimodal interaction. XISL inherits these features from VoiceXML and SMIL. The original feature of XISL is that XISL has enough modality-extensibility. We present the basic XISL tags, outline of XISL execution systems, and then make a comparison with other languages.

Categories and Subject Descriptors
I.7.2 [**Document Preparation**]: Markup Languages

General Terms
Languages

Keywords
Multimodal interaction, XISL, XML, Modality extensibility

1. INTRODUCTION
Using multimodal interaction (MMI) for accessing web has actively been discussed. The WWW Consortium organized a multimodal working group [6], and started standardization of MMI description languages. Some other organizations have been specifying their own descriptions to deal with multi-modalities [1][5][8]. The common purpose of these efforts is to provide seamless web services on various types of terminals such as mobile phones, PDAs, and other terminals in addition to ordinary PCs. In the near future, many other types of terminals will be used for accessing the web, and they will provide more advanced web services.

As various terminals are introduced for web access, the MMI description language will be required to deal with new modalities supported by those terminals. The terminals might handle some innovative modalities such as physical gesture inputs or haptic inputs/outputs as well as currently discussing modalities such as speech, ink, and so on. It is desirable to handle these modalities without changing MMI description languages drastically. However, previous languages need to change their specifications when dealing with these modalities because they contain modality dependent specifications.

With this background, we propose an MMI description language XISL (eXtensible Interaction Scenario Language) [2][3][9] that has enough modality-extensibility. XISL is an XML-based language that inherits a lot of tags from VoiceXML [4] and SMIL [7]. These tags enable XISL to control dialog flow/transition, synchronization of multimodal inputs/outputs, and other descriptions required for multimodal interaction. However, extensibility of modalities is not provided by them. XISL realizes it by consolidating modality-dependent descriptions into two tags and giving flexibility to their descriptions. This enables to extend modalities without changing the specification of XISL. Therefore, developers can easily construct seamless services using various terminals by means of XISL.

This paper is organized as follows. In section 2, we outline the features and tag set of XISL. In section 3, we explain the XISL execution systems. After comparing XISL with the other languages in section 4, we conclude our work in section 5.

2. MMI DESCRIPTION LANGUAGE XISL

2.1 Goal of XISL
The purpose of XISL is to provide a common language for web-based multimodal interaction systems on various types of terminals. For this purpose, XISL is specified to satisfy the following conditions.

1. It can control dialog flow/transition.
2. It can synchronize input/output modalities.
3. It has enough modality-extensibility.

As for conditions 1 and 2, some previous languages have desirable features. VoiceXML is a language for describing interaction scenarios executed in spoken dialog systems. It provides various notations and concepts such as dialog transitions, conditional branches, arithmetic operations, and so on. Since all of these notations and concepts are useful in interaction systems generally, we employed them in XISL. However, VoiceXML does not take account synchronization of multiple modalities because it is developed for spoken dialog systems that handle only speech

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICMI'03, November 5–7, 2003, Vancouver, British Columbia, Canada.
Copyright 2003 ACM 1-58113-621-8/03/0011…$5.00.

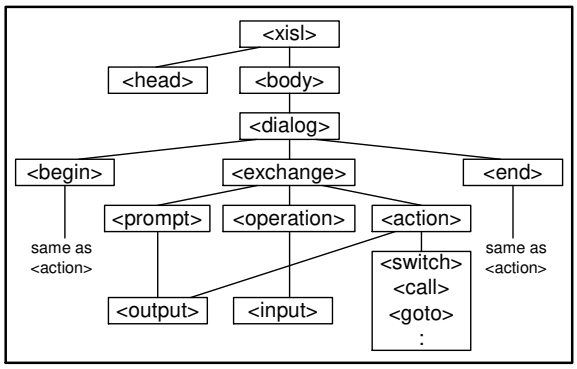

Figure1. Tree structure of XISL elements

modality. For this problem, we introduce SMIL as a solution because it can control synchronization of output media.

Although VoiceXML and SMIL bring solutions for the above conditions 1 and 2, they do not give any solution for condition 3. We cope with it by consolidating modality-dependent description of XISL into <input> and <output> tags, and giving flexibility to their attributes and contents. This is a good solution because developers can introduce a terminal with new types of modalities without changing the specification of XISL. The details of their specifications are described in section 2.4.

2.2 XISL Tag Set

XISL is an XML-based language that composes a tag tree described in figure 1. Figure 2 shows a fragment of an XISL document for an online shopping system. In this example, the system first outputs an HTML page and a voice prompt. Then, it requires a user to select an item that he/she wants to buy by speech or pointing. Upon accepting an input, the system outputs the name of the item for confirmation, and goes to the next dialog. We explain the outline of XISL tags by referring to this example.

An <xisl> element, a <body> element, <dialog> elements, and <exchange> elements ((a)-(d) in figure 2) correspond to a whole document, interaction contents, a set of exchanges between a user and a system, and an exchange, respectively. The <dialog> elements and the <exchange> elements correspond to the elements <form> and <field> of VoiceXML. The <dialog> element can contain a <begin> element and an <end> element to describe initial and terminal process of the <dialog>, respectively.

The <exchange> element is composed of at most one <prompt> element, an <operation> element, and an <action> element ((e), (h), (k) in figure 2) that represent a prompt to a user, a user's operation and a system action, respectively. Of these elements, <prompt> and <operation> elements have the same goal as the <prompt> and <grammar> elements of VoiceXML. The critical differences are that the <prompt>/<operation> elements of XISL can contain multiple <output>/<input> elements (outputs for a user ((f), (g) in figure 2) / inputs from a user ((i), (j) in figure 2)), while the <prompt>/<grammar> elements of VoiceXML can contain single prompt/grammar. This extension enables to describe multimodal simultaneous inputs and outputs in XISL. The <action>, the <begin>, and the <end> element contain some elements such as <output> elements, <switch> elements, <call> elements, <goto> elements ((l) in figure 2), and so on. Most of these tags are introduced from VoiceXML.

```
<?xml version="1.0" encoding="Shift-JIS"?>
<!DOCTYPE xisl SYSTEM "xisl.dtd">
<xisl version="1.0"> ---------------------------------------------------------- (a)
  <body> ------------------------------------------------------------------- (b)
    <dialog id="order"> ---------------------------------------------------- (c)
      <exchange> ----------------------------------------------------------- (d)
        <prompt> ----------------------------------------------------------- (e)
          <output type="browser" event="navigate">
            <![CDATA[
              <param name="id"> item_list </param>
              <param name="uri">
                www.vox.tutkie.tut.ac.jp/OLS.html
              </param>  ]]> ------------------------------------ (f)
          </output>
          <output type="tts" event="speech">
            <![CDATA[ <param name="text">
              Please select an item.
              </param>  ]]> ------------------------------------ (g)
          </output>
        </prompt>
        <operation comb="alt"> ------------------------------------ (h)
          <input type="touch" event="click"
                 match="[item:=@id]"/> ------------------------- (i)
          <input type="speech" event="recognize"
                 match="./grammar.txt#items"
                 return="item"/> ------------------------------------ (j)
        </operation>
        <action> ----------------------------------------------------------- (k)
          <output type="tts" event="speech">
            <![CDATA[ <param name="text">
              You ordered <value expr="item">.
              </param>  ]]>
          </output>
            :
          <goto next="how_many.xisl" namelist="item"/> -- (l)
        </action>
      </exchange>
        :
    </dialog>
  </body>
</xisl>
```

Figure 2. Example of XISL document

2.3 Synchronization of Tags in XISL

To enable interaction level and modality level synchronization, we employed synchronization mechanism like the one used in SMIL for the <exchange> elements and <input>/<output> elements. The "comb" attribute in a <dialog> tag controls synchronization of <exchange> elements in it. If "comb" is "par" (parallel), all <exchange> elements are executed in parallel. If "comb" is "alt" (alternative), one of the <exchange> elements is executed alternatively, and if "comb" is "seq" (sequential), all <exchange> elements are executed in document order. The <exchange> elements can be also controlled by means of <par_exchange>, <seq_exchange>, and <alt_exchange> tags. The <exchange> elements bound by one of these tags are executed as if they are bound by the <dialog> element whose "comb" attribute is "par", "alt", or "seq". This type of synchronization is also available for controlling <input> element. The <output> elements are synchronized by <par_output> or <seq_output> tags.

The tags (h), (i), and (j) in figure 2 presents an example of <input> synchronization. In this example, the "comb" attribute of an <operation> tag ((h) in figure 2) is set to "alt". Therefore two

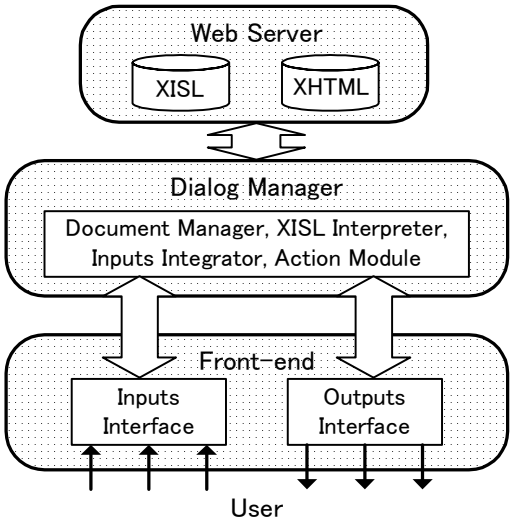

Figure 3. XISL execution system

inputs ((i), (j) in figure 2) are alternatively accepted in this example.

2.4 Input/Output Description

The <input> tag includes four attributes: "type" (input modality), "event" (input event), "match" (path to a target XHTML element), and "return" (return values). The <output> tag contains two attributes: "type" (output modality), and "event" (output event) in XISL specifications. The author can set some parameters in the <param> tags in these elements.

In order to make descriptions of modalities more flexible, strict attribute values and contents of <input> and <output> elements are not specified by XISL. The descriptive details are specified by terminal developers, who can therefore expand or modify modalities easily.

3. XISL EXECUTION SYSTEM

3.1 Outline of XISL Execution System

Figure 3 shows the outline of an XISL execution system composed of two modules: a front-end module and a dialog manager module. The dialog manager interprets XISL documents, manages dialog flow, and controls inputs and outputs. It is independent of both application and terminal. On the other hand, the front-end is a user interface terminal that handles various modalities. It depends on applications and terminals, respectively. This division enables us to reuse the dialog manager when introducing new terminals with different types of modalities because it is independent of modalities.

The XISL document is firstly sent to an XISL interpreter in the dialog manager module. The XISL interpreter interprets the XISL document and divides it into <prompt>s, <operation>s, and <action>s. Then, it sends <operation>s to the input integrator, and <prompt>s and <action>s to the action module. The input integrator extracts <input> tags from <operation>s, and sends them to the front-end. It also makes an integration table based on the <operation>s and uses it to match the user's inputs from the front-end module. The action module extracts <output>s from <prompt>s and <action>s, and sends them to the front-ends. The other tags (such as <switch>s, <goto>s, etc.) in the <action> tags are executed inside the action module.

```
<?xml version="1.0" encoding="Shift-JIS"?>
<!DOCTYPE xisl SYSTEM "xisl.dtd">
<xisl version="1.0">
  <body>
    <dialog id="order">
      <exchange>
        <prompt>
          <output type="speech" event="play"> ---------- (1)
            <![CDATA[
              <param name="mode"> tts </param>
              <param name="speech-text">
                This is an online shopping store.
                Please select an item.
              </param> ]]>
          </output>
        </prompt>
        <operation comb="alt">
          <input type="speech" event="recognize"
                 match="./grammar.txt#items"
                 return="item"/> ---------- (2)
          <input type="dtmf" event="push"
                 match="./grammar_dtmf.txt#items"
                 return="item"/> ---------- (3)
        </operation>
        <action>
          <output type="speech" event="play">
            <![CDATA[
              <param name="mode"> tts </param>
              <param name="speech-text">
                You ordered <value expr="item">.
              </param> ]]>
          </output>
            :
          <goto next="how_many_tel.xisl" namelist="item"/>
        </action>
      </exchange>
        :
    </dialog>
  </body>
</xisl>
```

Figure 4. Example of XISL document for a telephone-based system

3.2 Example of XISL Document for Different XISL Execution System

Here, we show XISL examples for two types of XISL execution systems, namely a PC-based system and a telephone-based system, to show modality-extensibility of XISL.

The PC-based system is developed as an information kiosk put at a station or a convenience store. The XISL document shown in Figure 2 (in section 2.2) is described for a PC-based online shopping system. It shows how to describe outputs/inputs in XISL. As for outputs, web page browsing is described in (f) of figure 2. It requires a URL and an id as parameters. Synthesized speech output is described in (g) of figure 2 with a speech text as a parameter. Click and speech inputs are described as shown in (i) and (j) of figure 2. In addition to these modalities, the PC-terminal can handle other touch operations (double-click, drag, etc.), anthropomorphic agent actions (speech, move, etc.), and other operations and actions.

The telephone-based system is developed as a voice-portal system. Figure 4 shows an example of an XISL document for an online shopping system. As same as the example for the PC-terminal, the outline of dialog is composed of a prompt, an operation, and an action. However, the modalities used in <input>s and <output>s ((1)-(3) in Figure 4) are different from the ones used in the PC-terminal. The descriptions of these tags are specified for the telephone-based system that can handle speech and DTMF modalities. If a developer wants to add some new modality in some front-end, he/she has only to specify descriptions of <input>/<output> tags and modify the front-end module to interpret them.

4. COMPARISON WITH OTHER APPROACHES

Some organizations have specified their own descriptions to deal with multi-modalities. In this section we compare XISL with SALT, XHTML+Voice, and a SMIL based approach.

4.1 Comparison with SALT

SALT [5] defines additional tags for describing speech recognition and TTS that are attached to an HTML document. The most remarkable feature of SALT is that it is highly suitable to HTML documents. SALT speech grammar tags are embedded in an HTML document, and are bound to <input> tags in the HTML document. This enables to attach speech interface to current HTML documents by minimal addition. XISL needs much description than SALT tags. Therefore, SALT is good choice for adding simple speech interaction to an HTML document.

On the other hand, XISL is better suited for describing complex MMI using sequential, parallel, or alternative combination of multiple modalities. Since XISL (interaction scenario) is explicitly separated from XML or HTML contents, the application developer can easily read and trace interaction flow. It will increase reusability of XISL in complex multimodal interaction.

4.2 Comparison with XHTML+Voice

XHTML+Voice [8] attempts to add VoiceXML [4] to XHTML pages. A VoiceXML source code is added into a <head> element of an XHTML document, and bound to some XHTML tags like SALT. Since some interaction flow is described in VoiceXML code, XHTML+Voice can control more complex MMI than that SALT can control. However, XHTML+Voice assumes only voice interaction as additional modalities to web pages. On the other hand, our approach has more flexibility to add modalities than XHTML+Voice approach.

4.3 Comparison with a SMIL based approach

The SMIL [7] based approach [1] realizes event-based multimodal dialog control by SMIL and ReX (Reactive XML). In this approach, temporal control is described in SMIL, and event control is described in ReX. Since this approach is based on event control, it is easy to extend modalities. However, in this approach, the modality-dependent descriptions are not explicitly separated from the modality-independent descriptions, whereas our approach explicitly separates them. Therefore, XISL is easier to read and reuse than the SMIL based approach.

5. CONCLUSION

In this paper we presented an MMI description language XISL, its execution system. We made XISL a modality-extensible language by consolidating modality-dependent description of XISL into <input> and <output> tags, and giving flexibility to their descriptions. However, writing XISL is a hard task for authors because it requires a lot of parameters for describing individual <input>/<output> tags. In order to solve this problem, we will provide a GUI-based prototyping tool that relieves the author from writing the XISL documents.

Future problems are conformation of XISL into the W3C MMI requirement and modification of its execution system for the conformation.

6. ACKNOWLEDGMENTS

This work was supported in The 21st Century COE Program "Intelligent Human Sensing" and Grant-in-Aid for Young Scientists (B) 14780323 2003, from the ministry of Education, Culture, Sports, Science and Technology.

7. REFERENCES

[1] Beckham, J.L., Fabbrizio, G.D., and Klarlund, N. Towards SMIL as a Foundation for Multimodal, Multimedia Applications. In Proceedings of EUROSPEECH 2001 (Aalborg, Denmark, September 2001), 1363-1366.

[2] Katsurada, K., Nakamura, Y., Kobayashi, S., Nitta, T. XISL: An Attempt to Separate Interactions from Data. In Proceedings of INTERACT'01 (Tokyo, Japan, July 2000), 763-764.

[3] Nitta, T., Katsurada, K., Yamada, H., Nakamura, Y., Kobayashi, S. XISL:An Attempt to Separate Multimodal Interactions from XML Contents. In Proceedings of EUROSPEECH 2001 (Aalborg, Denmark, September 2001), 1197-1200.

[4] VoiceXML Forum http://www.voicexml.org/

[5] Wang, K. SALT: A Spoken Language Interface for Web-based Multimodal Dialog Systems. In Proceedings of ICSLP'02 (Denver CO. September 2002), 2241-2244.

[6] W3C multimodal interaction activity. http://www.w3.org/2002/mmi/

[7] W3C SMIL Specifications http://www.w3.org/AudioVideo/

[8] XHTML+Voice Profile 1.1 http://www-3.ibm.com/ software/pervasive/multimodal/x+v/11/spec.htm

[9] XISL homepage http://www.vox.tutkie.tut.ac.jp/XISL/XISL-E.html