XISL: An Attempt to Separate Multimodal Interactions from XML Contents

Tsuneo Nitta Kouichi Katsurada Hirobumi Yamada Yusaku Nakamura Satoshi Kobayashi

Graduate School of Engineering, Toyohashi University of Technology
1-1 Hibarigaoka, Tempaku-cho, Toyohashi 441-8580, Japan
nitta@tutkie.tut.ac.jp

Abstract

In this paper we outline a multimodal interaction description language XISL (Extensible Interaction-Sheet Language) that is developed to describe multimodal interactions (MMI), and to separate the description of interactions from XML contents. XISL makes an XML document independent of interactions that may differ between each terminal, and so enables such seamless services as web-browsing to be constructed easily. Since XISL also provides various combinatorial usage of modalities, a developer can describe a MMI scenario easily. We implemented an interpreter of XISL on prototype systems for multimedia lectures with different types of MMI and proved the viability of XISL by experiments using the systems.

1. Introduction

As the multimedia infrastructure is progressively developed, the user interface (UI) used in multimedia applications is becoming increasingly important. The multimodal interaction (MMI) scheme enables familiar, smooth, and robust UI to be developed, however, there are many issues concerning heterogeneity of users' terminals and difficulty in MMI design. In multimedia applications over networks, for example, a server system must provide input and output modalities that correspond to those of users' terminals. One solution is to construct multiple server systems with different combinations of modalities, however, this approach is inefficient and expensive.

Recently, XML documents are used for describing contents in many web services and are browsed through various types of terminals [1]. In the new web services, developers can easily ensure heterogeneity of users' displays by providing a style sheet (XSL [2]) suitable for each type of terminal together with common XML contents. On the other hand, VoiceXML has been proposed to describe interactions through mobile phones [3], however, interactions and XML contents are mixed in a VoiceXML document, and the targets of modalities are limited such as speech and DTMF, thus making it difficult to realise seamless services with MMI under heterogeneous UI environments.

We propose an interaction description language XISL (Extensible Interaction-Sheet Language) that separates the description of interactions from XML contents and enables the XML contents to be used independently and helps us to construct seamless services over networks.

In XISL, both the user operations and system actions are independently described from XML contents, while in the prior work [4], only system actions are separated. Figure 1 shows the concept of providing seamless services to such heterogeneous equipment as PCs, mobile terminals, digital TVs, and car navigators by installing two types of UI-specific sheets of XSL and XISL together with common XML contents on a server system.

The description and usage of modalities in MMI systems are complicated and it is time consuming for developers to learn them. XISL enables the developer to describe a scenario of MMI easily by providing various functions to represent combinatorial usage of modalities such as sequential, parallel, alternative, and co-ordinated usage.

This paper is organised as follows: Section 2 outlines the system architecture. Section 3 describes XISL. Section 4 then explains the implementation of XISL in prototype lecture systems. Finally, Section 5 discusses future works.

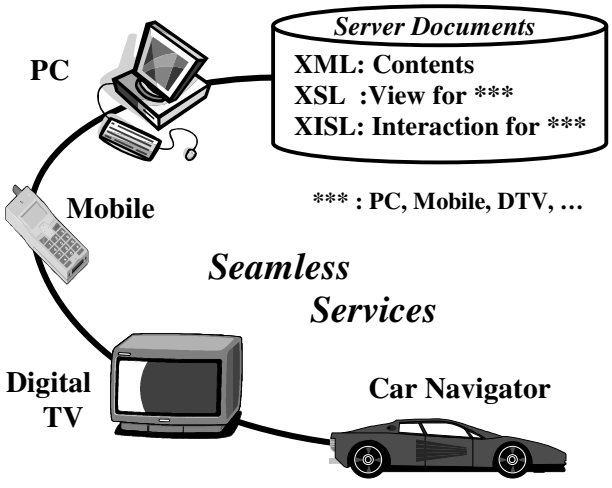

***Figure 1*: Seamless Services over Networks.**

2. System Architecture

Figure 2 shows a block diagram of an MMI system with input modalities of speech and pointing gestures, and output modalities of graphics and speech (TTS: Text-to-Speech). Speech input is recognised with an assigned grammar and the result is transferred to an input integrator. In the input integrator, after converting the recognition result into a semantic-code sequence such as an object sequence in a user operation, the semantic-code sequence and/or pointing information are input to an integration table. The integration table generated from XISL is represented in the form of a transition table that integrates input modalities, solves competitive input modalities, and handles time-out processing (see Table 1 [5]). A multimodal dialogue manager interprets information of input operations and output actions in XISL data, then sends the result on input operations to the input integrator in the form of an integration table, and transfers the information on output actions that corresponds to the result of input integration to a response controller. The response controller decides which output modalities should be used according to a decision tree of response strategies that contains pre-selected modalities by user, current input modalities, and category of output (system error, warning, notice, prompt for a user, etc.) [6], then sends response information to output engines of graphics and TTS that are used for agent actions, balloons, and speech of an anthropomorphic agent, etc.

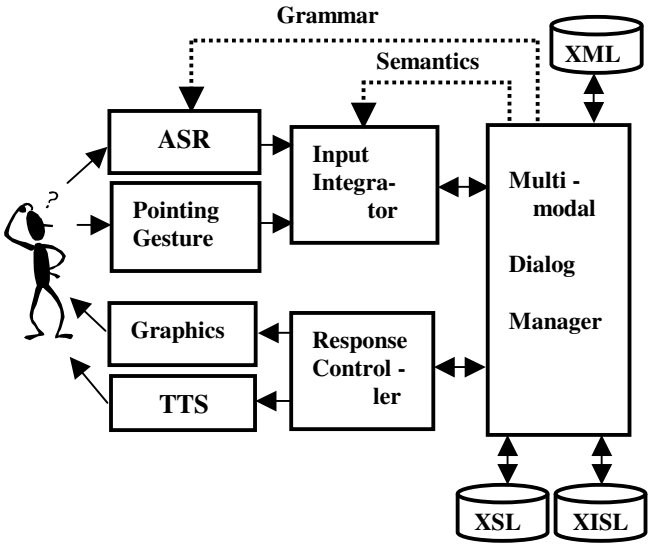

***Figure 2*: Block Diagram of an MMI System.**

3. Extensible Interaction-Sheet Language

3.1. What should be described in XISL?

Our first purpose is to separate interactions from XML contents. Since an interaction in an XML document is described for each object in the document, the interaction in an XISL document should be described for the corresponding object (including a page object) in the form of a pair of an input operation and an output action.

Our second purpose is to describe an MMI scenario in an XISL document easily. When multiple modalities are used in the interaction, they may be used in various combinations: some of them may be used in parallel, others sequentially, and still others in alternative combinations. In order to incorporate such combinatorial usage, some tags and their attributes that can be applied to control interaction flows should be introduced.

3.2. Elements and Document Structure of XISL

Figure 3 shows an example of an XML document, and Fig. 4 shows the corresponding XISL document. All the "< >"s in Fig. 4 are elements of XISL. The top-level of

***Table* 1 Input Integration Table**

op. / state	Pointing		…	Voice		
	command	object	…	command	object	…
0	s4	s1	…	s5	s2	...
1	r1	p1	…	r2	p2	...
2	r3	p3	…	r4	p4	...
…	…	…	…	…	…	...
…	…	…	…	…	…	...

s: shift r: accept & reduce
p: process (rejection/ overwrite/….)

the XISL document is an <xisl> element ((a) in Fig. 4) that is divided into two fields: a <head> field and a <body> field ((b) and (c) in Fig. 4). The <head> field holds the information about the XISL document, and the <body> field contains one or more sequences of dialogues delimited by <dialogue> tags ((d) in Fig. 4). Each dialogue is composed of one or more <exchange> elements ((e) in Fig. 4) that are the most primary units of interaction. The <exchange> element is composed of an <operation> element and an <action> element ((f) and (h) in Fig. 4) that are used to describe user operations and system actions, respectively.

An interaction flow is controlled by using a "comb" attribute of the <dialogue> element whose value is *par* (parallel), *seq* (sequential) or *alt* (alternative). When the "comb" is *par*, all the <exchange>s are executed in parallel. If the "comb" is *alt*, one of the <exchange>s is performed alternatively, and if the "comb" is *seq*, all the <exchange>s are performed in sequence. We can also control interaction flows by means of <par>, <seq> and <alt> tags. The elements bound by these tags are

```
<?xml version="1.0" encoding="Shift-JIS"?>
<!DOCTYPE lecture SYSTEM "TTS.dtd">
<lecture>
  <page id="1" next="2" xisl="1">
    <head>
      <title> Text to Speech </title>
    </head>
    <body>
      <list mark="none">
        <item id="1">
          Text: <text>"It's fine today."</text>
        </item>
        <item id="2">
          Morphological Analysis
        </item>
        <item id="3">
          Speech Synthesis
        </item>
      </list>
    </body>
  </page>
</lecture>
```

***Figure 3*: Example of XML Document.**

equivalent in functions to the corresponding attributes of <dialogue>. These tags and attributes can also be used in an <operation> tag and an <action> tag (*alt* and <alt> do not appear in an <action> tag).

In an <operation> element and an <action> element, we can describe <input> tags ((g) in Fig. 4) and <output> tags, respectively. The <input> tag has three attributes of "pattern", "modality" and "event" that correspond to the intended object of a user, category of modality, and an input event, respectively. The <output> tag has two attributes of "modality" and "value" that indicate an output modality and its content, respectively.

Data and values of attributes in an XML document can be transferred into an XISL document by using <get_value> and <get_attr> tags ((i) and (k) in Fig. 4), respectively. Common attributes of these tags are "pattern" and "var". A "pattern" attribute assigns the location of data or value, then the data or value is substituted into the variable specified by the "var" attribute. The <get_value> tag is used to obtain the data specified by the "pattern" attribute, while the <get_attr> tag gets the value of the attribute specified by an "attr" attribute.

To describe system actions, two types of tags are provided. An <exec> tag ((j) in Fig. 4) executes a program indicated by a "program" attribute with arguments assigned by a "src" attribute. A <jump> tag ((l) in Fig. 4) terminates the dialog, and jumps to other XML content indicated by a "goto" attribute.

```
<?xml version="1.0" encoding="Shift-JIS"?>
<!DOCTYPE lecture SYSTEM "xisl.dtd">
<xisl>                                                          (a)
  <head>                                                        (b)
    <content> TTS.xml </content>
    <name> TTS.xisl </name>
  </head>
  <body>                                                        (c)
    <dialogue id="first" comb="seq">                            (d)
      <exchange>                                                (e)
        <operation comb="par">                                  (f)
          <input pattern="page[@id='1']" modality="speech"
                 event="result"/>                               (g)
          <input pattern="page[@id='1']/body/list/item[@id='2']
                 modality="touch" event="click"/>
        </operation>
        <action>                                                (h)
          <get_value var="text"                                 (i)
           pattern="page[@id='1']/body/list/item[@id='1']/text"/>
          <exec program=" Morph_Analysis" src="$text$"/>        (j)
        </action>
      </exchange>
      <exchange>
        <operation comb="par">
          <input pattern="page[@id='1']/body/list/item[@id='3']"
                 modality="touch" event="click"/>
        </operation>
        <action>
          < get_value var="text"
           pattern="page[@id='1']/body/list/item[@id='1']/text"/>
          <exec program="Sp_Synth" src="$text$"/>
        </action>
      </exchange>
      <exchange>
        <operation comb="par">
          <input pattern="page[@id='1']" modality="touch"
                 event="click"/>
        </operation>
        <action>
          <get_attr var="next_page" pattern="page[@id='1']"
                    attr="@next"/>                              (k)
          <jump goto="$next_page$"/>                            (l)
        </action>
      </exchange>
        :
    </dialogue>
    <dialogue id="second" comb="seq">
        :
    </dialog>
  </body>
</xisl>
```

***Figure 4*: Example of XISL Document.**

3.3. Example of XML and XISL

The XML document and XISL document shown in Fig. 3 and Fig. 4 are a lecture content on "Text to Speech" and its interaction. The XML document in Fig.

3 contains a sentence “Text: It’s fine today.” and two buttons of “Morphological Analysis” and “Speech Synthesis”. In this example, a teacher interacts with an MMI system by using three types of operations. When he clicks the button “Morphological Analysis” while speaking “result”, the system executes a program “Morph_Analysis” with an argument “It’s fine today” and returns its result. If he clicks the button “Speech Synthesis”, the system outputs the synthesised speech after calculating the waveform with a “Sp_Synth” program. If he clicks another place on the screen, the next page is displayed and a new interaction is performed.

4. Prototype System

We implemented an XISL interpreter (Visual C++®) and confirmed the effectiveness of XISL on MMI lecture systems with various types of modalities. Figure 5 shows the outline of an MMI system in which lecture contents, their views, and interaction descriptions are represented as XML documents, XSL documents, and XISL documents, respectively. In the MMI systems, pointing gestures (pointing, gesture and digital ink), speech (speech recognition and TTS) and anthropomorphic agents (action, balloon and speech) are employed as input/output modalities.

We employed Microsoft® Agent as the anthropomorphic agents, and LaLaVoice™ as the speech recogniser / synthesiser. Lecture slides described in HTML documents that are converted from the XML and XSL documents are displayed on a web browser IE5.0. Pointing gestures were implemented by using a transparent window that is put on a lecture slide.

When multimodal inputs are given to the system, they are converted into a corresponding semantic-code sequence of the inputs, and are integrated by using the integration table shown in Table 1. If the inputs are accepted, the response controller decides the output modalities such as action and speech of agents, and display of a lecture slide.

5. Discussion

In this paper, we described an MMI description language XISL, its elements and document structure, and implementation in prototype lecture systems. A system designer can develop an MMI system more easily by using XISL, but we consider that a GUI-based prototyping tool is needed for designing such systems quickly [7]. The tag set of XISL must also be expanded to apply XISL to various domains. We are now trying to describe various applications by using XISL such as for tourist guidance, car navigation, on-line shopping, and interaction with a robot, and will reflect the results on the XISL elements and document structures. There are many issues cocerning the multimodal dialogue

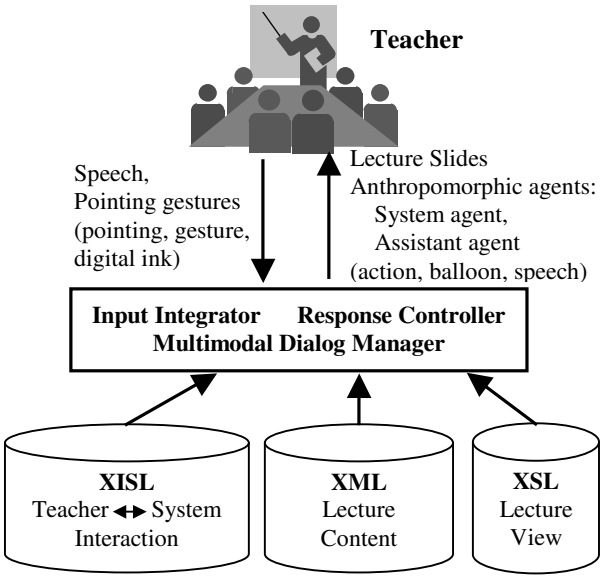

***Figure 5*: MMI Lecture System.**

manager; dialogue flow controll by using dialogue context and task switching in multitask applications are the next targets of our MMI study.

* Visual C++ and Microsoft are registered trademarks of Microsoft Corp.
* LaLaVoice is a trademark of Toshiba Corp.

References

[1] http://www.w3.org/XML/
[2] http://www.w3.org/Style/XSL/
[3] http://www.voicexml.org/
http://www.w3.org/Voice/
[4] Wang, K., “Implementation of a Multimodal Dialogue System using Extended Markup Languages”, ICSLP2000, II-138-141, 2000.
[5] Yamada, M., Murakami, T., Yamada, H., Katsurada, K., and Nitta, T., “Integrating Modalities in Multimodal Interface”, Proc. of 62nd meeting of IPSJ, 1U-04, 2001 (in Japanese)
[6] Ohtani, Y., Suzuki, H., Kobayashi, S., Katsurada, K., Nitta, T., “Response Strategies in Multimodal Interface”, Proc. of 62nd meeting of IPSJ, 1U-05, 2001 (in Japanese)
[7] Kamio, H., Amamiya, M., Matsu'ura, H., and Nitta, T., "Muse, a rapid prototyping tool", Advances in Human Factors/Ergonomics 21B - Design of Computing Systems (Proc. HCI International'97), pp. 569-572 (1997).