Reducing the Description Amount in Authoring MMI Applications

Kouichi Katsurada, Kazumine Aoki, Hirobumi Yamada and Tsuneo Nitta

Graduate School of Knowledge-based Information Engineering
Toyohashi University of Technology, Japan
{katurada, nitta}@tutkie.tut.ac.jp, {aoki, yamada}@vox.tutkie.tut.ac.jp

Abstract

Although an MMI description language XISL1.1 (eXtensible Interaction Scenario Language) developed for web-based MMI applications has some desirable features such as high extensibility and controllability of modalities, it has some issues concerning capabilities for describing slot-filling style dialogs, a large amount of description for modality combination of inputs, and complicated authoring when handling XML contents. In this paper we provide a new version of XISL (XISL2.0) that resolves these issues by using VoiceXML-based syntax, modality-independent input description, and XForms-like data model. The results of comparison between XISL1.1 and XISL2.0 showed that XISL2.0 reduced the amount of description by 40% as compared with XISL1.1, and XISL2.0 succeeded to mitigate the difficulties in the authoring work of developing MMI applications.

1. Introduction

There has been proposed various types of multimodal interaction (MMI) systems and fundamentals for constructing MMI applications. The WWW consortium organized a multimodal working group [1], and has standardized some fundamentals such as an MMI framework, EMMA (a markup language to represent semantics of inputs), and so on. Some other organizations have been developing their own frameworks to deal with MMI on web applications [2][3]. We have also proposed MMI architecture [4] together with an MMI description language XISL1.1 (eXtensible Interaction Scenario Language) [5][6] that has extensibility of modalities, as well as controllability of them.

Although XISL1.1 has desirable features, it has some issues concerning capability for describing slot-filling style dialogs, a large amount of description for modality combination of inputs, and complicated authoring when handling XML contents. In this paper we provide a new version of XISL (XISL2.0) to resolve these issues. Firstly, we change the syntax of XISL to that of VoiceXML to simplify the description of slot-filling style dialogs and to introduce the FIA algorithm. Secondly, we modify input description into modality-independent one to avoid the complex description of modality combination. And lastly, we introduce an XForms-like data model which enables to handle XML contents from XISL. These resolutions can mitigate the difficulties of authoring work when developing web-based MMI applications.

This paper is organized as follows: section 2 outlines the old version of XISL (XISL1.1) and its issues. Section 3 describes outline of XISL2.0 and comparison with XISL1.1. After discussing some future issues in section 4, we conclude our work and show some future work in section 5.

2. XISL1.1 and Its Issues

2.1. Outline of XISL1.1

XISL1.1 [5][6] is an XML-based language for describing MMI between a user and a system. In XISL1.1, a scenario is composed of a sequence of exchanges that contains a set of prompts for a user, a set of user's multimodal inputs and a set of system's actions. Actions include outputs to a user, simple arithmetic operations, conditional branches, dialog transition, and so on.

Figure 1 shows a fragment of an XISL1.1 document for an online shopping system. This example describes a scene in which the system asks a user which item to buy and asks how many items to buy. The user can reply with some input patterns (selecting modalities from speech and pointing, and whether inputs both the item and quantity or inputs only the item first). Hereinafter, we outline the syntax of XISL1.1 along this example.

An <exchange> element and a <dialog> element ((b) and (a) in figure 1) correspond to an interaction turn and its set, respectively. The <exchange> element is composed of at most one <prompt> element, an <operation> element, and an <action> element ((c), (d), (g) in figure 1) that represent prompts for a user, user's multimodal inputs and system actions, respectively. User's inputs are accepted by the <input> elements ((f) in figure 1) in an <operation> element. An <action> element contains <output> elements ((i) in figure 1) for outputs and some other elements for simple arithmetic operations, conditional branches, dialog transition, data submission to a server ((h) in figure 1), and so on. Note that this example omits some internal contents of <prompt> and <output> elements for simplification.

XISL provides some notations to synchronize input/output modalities. A "comb" attribute in a <operation> tag controls synchronization of <input> elements in it. If "comb" is "par" (parallel), all the <input>s are required to go to the next <action>. If "comb" is "alt" (alternative), one of the <input>s is required, and if "comb" is "seq" (sequential), <input>s are processed in document order. The <input> elements can be also controlled by <par_input> ((e) in figure 1), <seq_input>, and <alt_input> tags. For <output> elements, two tags (<par_output> and <seq_output>) are prepared. For further details of XISL tags, the reader can refer to the articles [5] or the XISL web site [6].

```
<?xml version="1.0" encoding="Shift-JIS"?>
<!DOCTYPE xisl SYSTEM "xisl.dtd">
<xisl version="1.0">
  <head> … </head>
  <body>
    <dialog> …… (a)
      <exchange> …… (b)
        <!--An exchange to accept both an item
            and its quantity-->
        <prompt> …… (c)
          <!-- A prompt to ask an item and its quantity -->
        </prompt>
        <operation comb="alt"> …… (d)
          <par_input> …… (e)
            <input type="touch" event="click"
                match="item" return="ITEM"/> …… (f)
            <input type="speech" event="recognize"
                match="./sp.grm#quantity" return="QTY"/>
          </par_input>
          <par_input>
            <input type="touch" event="click"
                match="quantity" return="QTY"/>
            <input type="speech" event="recognize"
                match="./sp.grm#item" return="ITEM"/>
          </par_input>
          <input type="speech" event="recognize"
              match="./sp.grm#item_qty"
              return="ITEM_QTY"/>
        </operation>
        <action> …… (g)
          <submit next="set_item.cgi"
              namelist="ITEM"/> …… (h)
          <submit next="set_quantity.cgi"
              namelist="QTY"/>
          <output> …… (i)
            <!--You ordered QTY ITEMs. -->
          </output>
            :
        </action>
      </exchange>
      <exchange>
        <!-- An exchange to accept only an item -->
        <operation comb="alt">
          <input type="touch" event="click"
              match="item" return="ITEM"/>
          <input type="speech" event="recognize"
              match="./sp.grm#item" return="ITEM"/>
        </operation>
        <action>
          <submit next="set_items.cgi" namelist="ITEM"/>
          <output>
            <!--Please input quantity of ITEM.-->
          </output>
            :
      </exchange>
    </dialog>
  </body>
</xisl>
```

Figure 1: An example of MMI description with XISL1.1

2.2. Issues concerning XISL1.1

Although XISL1.1 has some desirable features in its extensibility and controllability of modalities, some issues became clear through developing many applications. The first issue is the problem appeared at describing slot-filling style dialogs. In the example shown in figure 1, the first <exchange> is waiting for both an item and its quantity, while the second one is waiting for only an item. Although these

```
<?xml version="1.0" encoding="Shift-JIS"?>
<!DOCTYPE xisl SYSTEM "xisl.dtd">
<xisl version="1.0">
  <head> … </head>
  <body>
    <form id="SHOP"> …… (1)
      <fe> <!-- Display an HTML content --> </fe> …... (2)
      <fe> <!-- Set a speech
          recognition grammar --> </fe> …… (3)
      <initial> ……..…… (4)
        <prompt>
          <fe><!--A prompt to ask item
              and quantity --></fe>
        </prompt>
      </initial>
      <field name="ITEM"> …… (5)
        <prompt>
          <fe> <!--A prompt to ask an item --> </fe>
        </prompt>
        <filled> ……..... (6)
          <backend action="set_item.cgi"
              namelist="ITEM"/> …… (7)
            :
        </filled>
      </field>
      <field name="QTY">
        <prompt>
        <fe> <!--A prompt to ask quantity
            of the item --> </fe>
        </prompt>
        <filled>
          <backend action="set_quantity.cgi"
              namelist="QTY"/>
            :
        </filled>
      </field>
      <filled namelist="ITEM QTY">
        <fe> <!--You ordered QTY ITEMs. --> </fe>
          :
      </filled>
    </form>
  </body>
</xisl>
```

Figure 2: An example of MMI description with XISL2.0

two <exchange>s are waiting for different input patterns, they partially include common description of <inputs>s. These <input>s may be integrated if some efficient description is introduced. This issue arises when describing slot-filling style dialogs. Therefore, the introduction of some syntax and prompting algorithm for handling slot-filling style dialog is expected.

The second issue is a large amount of description for multimodal input combination in MMI systems. As shown figure 1, the description becomes very long if all the types of modality combinations are described. One solution is to make input descriptions modality-independent so as to avoid this difficulty.

The third issue is that XISL1.1 needs a complicated CGI program for handling contents used in application. In figure 1, some <submit> elements are used to activate server-side CGI that modifies XML contents. If some XML description and its control methods are introduced into XISL, the heavy load of CGI description is reduced, as well as the amount of server-client communication is decreased.

```
<emma version="1.0">
  <interpretation id="SHOP">
    <ITEM> Orange </ITEM>
    <QTY> 2 </QTY>
  </interpretation>
</emma>
```

Figure 3: EMMA result for application shown in figure 2

```
<xisl version="1.0">
  <head>
    <model>
      <instance>
        <my:Shop>
          <my:ShoppingCart>
            <my:Item/>
            <my:Quantity/>
          </my:ShoppingCart>
          <my:Tax> 1.05 </my:Tax>
          <my:TotalPayment/>
        </my:Shop>
      </instance>
      <bind nodeset="/my:Shop/my:Tax" readonly="true"/>
    </model>
  </head>
    :
</xisl>
```

Figure 4: A data-model for online shopping application

```
<form id="SHOP">
  <field name="ITEM">
    <filled>
      <set_value expr="'Orange'"
        select="/my:Shop/my:ShoppingCart/my:Item"/>
    <filled>
  </field>
      :
```

Figure 5: A model control tag to set a value

3. Proposal of XISL2.0

In order to resolve the above issues, we introduce VoiceXML-based syntax, modality-independent input description, and XForms-like data model. We call the new version of XISL XISL2.0. In this section, we first describe the outline of XISL2.0 and then compare it with XISL1.1.

3.1. Outline of XISL2.0

3.1.1. VoiceXML-based syntax

VoiceXML [7] is a well-known language for describing speech application that is standardized at W3C. It has simple description but strong algorithm for slot-filling style dialogs. Therefore, we used VoiceXML as a base of our new XISL. Figure 2 shows a fragment of an XISL2.0 document for the same online shopping application described in Figure 1.

XISL2.0 is processed along almost the same algorithm as the one used in VoiceXML, the so-called FIA (Form Interpretation Algorithm). When a <form> element ((1) in figure 2) is visited, two <fe> elements ((2) and (3) in figure 2) are executed first. <fe> elements are XISL2.0 original elements that is used for setting some parameters to

Table 1: Amount of description for online shopping

	XISL1.1	XISL2.0
Elements/attributes	412	246
File size (byte)	14442	7202

Table 2: Contribution of each factor

VoiceXML syntax	9.5%
Modality independent description	25.7%
XForms data-model	-4.1%
Other modification	9.2%
Total	40.3%

Table 3: CGI programs

	XISL1.1	XISL2.0
Number of programs	7	2

modalities/media in the MMI system. Then, an <initial> element ((4) in figure 2) is selected and a prompt is output. The <field> element ((5) in figure 2) is an input slot that accepts a user's input. In this example two <field> elements are prepared for inputs concerning an item and its quantity. If a <field> is filled, the <filled> element ((6) in figure 2) in the <field> is executed. The <backend> element ((7) in figure 2) is an XISL2.0 original tag which plays the same role as the <submit> element in XISL1.1. It is renamed because XISL2.0 introduces a new <submit> element which is used in VoiceXML. If there are unfilled elements, the FIA algorithm automatically selects a <field> element, outputs a <prompt> in the <field> element, and asks a user to input some data for the <field>.

3.1.2. Modality-independent input description

In VoiceXML, a <field> tag can accept semantic interpretation results of speech input represented by NLSML [8] or SISR [9], and the results are hold by a variable given as the "name" attribute of a <field> tag. In the near future, VoiceXML is expected to support EMMA [10] for representing input interpretation results. EMMA is a standard specified by W3C MMI working group [1] formatting interpretation results in MMI systems. Figure 3 shows an example of EMMA description. The interpretation results are embedded in the <interpretation> element. In addition to representing interpretation results, EMMA can add some additional information (such as input mode, confidence score, time stamps, and so on) to the results. We also used EMMA as the representation of interpretation results and introduced the same usage of attributes of <field> element.

3.1.3. XForms-like data model

In order to get rid of complicated CGI programs, we introduced XForms-like data model [11] and its control tags in XISL2.0. Figure 4 and figure 5 show examples of a data-model and a control tag. As shown in figure 4, a data-model is described in the <head> element in an XISL document. An application developer can define arbitrary XML contents in an <instance> element in a <model> element. As same as XForms description, <bind> element can control restriction and dependency of contents. In order to modify and control this model from XISL2.0 forms, we prepared some elements to set a content (or an attribute), get a content from an element

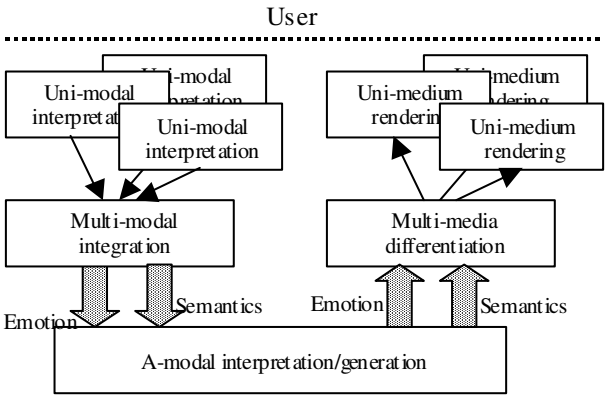

Figure 6: UMA three-layered model

(or an attribute), delete a part of XML tree, and add a part of XML tree. Figure 5 shows an example to set a content "Orange" to the element <my:Item> in the model shown in figure 4.

3.2. Comparison with XISL1.1

We have developed an online shopping application with XISL1.1 and XISL2.0, and compared the amount of description. The application is developed to be executed on a PC with a pointing device, a keyboard, a microphone, a screen and a speaker. We compared the number of elements/attributes, file size and the number of CGI programs.

Table 1 shows the result of comparison between the amount of XISL1.1 description and XISL2.0 description. The result shows XISL2.0 can reduce the number of elements/attributes by 40.3%, and the file size by 50.2%. Therefore, it is shown that the modifications enormously reduce the amount of description. Table 2 shows the contribution of each factor of this reduction. It shows that modality-independent description is the most effective modification. The reason is that modality combination is described all over the application, and all of them are removed by this modification. Table 2 also shows that introduction of XForms model increases the amount. However, this introduction concurrently reduces the number of CGI programs from 7 to 2 as shown in table 3. Therefore the total amount of description is reduced by this introduction.

4. Discussion

In section 3.1.2, we changed XISL description to modality-independent style. At the same time, we also modified the framework of our MMI system. We reconstructed the system based on UMA three-layered model [12]. UMA three-layered model is an engineering model for handling semantics and emotion in near future MMI systems. Figure 6 shows the structure of the model. In this model, each layer performs different role in the process of understanding multimodal inputs or rendering multimedia outputs.

The uni-modal layer is the lowest layer in the model. It is an interface layer, and handles each modality/medium individually. In other words, it interprets the inputs from each modality without the information from the other modality, and renders output for each modality one by one. Modality fusion and media fission are performed in the multi-modal layer. This layer integrates multimodal inputs or differentiates multimedia outputs. It derives semantic /emotional information from the inputs delivered from various modalities, and generates output contents from semantic/emotional information. The semantic/emotional information is sent from/to the a-modal layer. The a-modal layer is the highest layer which receives/delivers semantic /emotional information. This layer controls interaction based on some intension given as some method.

As described above, this model is designed to handle semantics/emotions in inputs/outputs. However, in our current MMI system, we only implemented semantic input interpretation part together with EMMA handling. In the near future, we will incorporate emotion and output differentiation that enables us to share semantic/emotional information among modalities in our MMI system.

5. Conclusions

This paper provides a new version of XISL which mitigates the difficulties in the authoring work of developing MMI applications. As shown in section 3, it reduced the amount of description by 40% as compared with XISL1.1. The most effective modification was modality-independent description which reduces the amount by about 26%. In addition to these modifications we also proposed the framework of MMI systems based on UMA three-layered model. The future work is to incorporate not only semantics but also emotions and output differentiation into MMI systems based on this model.

6. Acknowledgements

This work was supported in The 21st Century COE Program "Intelligent Human Sensing" and Grant-in-Aid for Young Scientists (B) 14780323 2004, from the ministry of Education, Culture, Sports, Science and Technology.

7. References

[1] http://www.w3.org/2002/mmi/
[2] http://www.saltforum.org/
[3] http://www.w3.org/TR/xhtml+voice/
[4] K. Katsurada, Y. Otani, Y. Nakamura, S. Kobayashi, H. Yamada, and T. Nitta, "A modality-independent MMI system architecture", Proc. ICSLP'02, Denver, United States, pp.2549-2552, Sept. 2002.
[5] K. Katsurada, Y. Nakamura, H. Yamada, and T. Nitta, "XISL: A language for describing multimodal interaction scenarios", Proc. ICMI'03, Vancouver, Canada, pp.281-284, Nov. 2003.
[6] http://www.vox.tutkie.tut.ac.jp/XISL/XISL-E.html
[7] http://www.w3.org/TR/voicexml20/
[8] http://www.w3.org/TR/speech-grammar/
[9] http://www.w3.org/TR/semantic-interpretation/
[10] http://www.w3.org/TR/emma/
[11] http://www.w3.org/TR/xforms/
[12] T. Nitta, "Consideration towards next-generation multi-modal interaction and MMI description language", IPSJ SIG Technical Report 2004-SLP-50, pp.15-22, Feb. 2004 (In Japanese).