A Rapid Prototyping Tool for Constructing Web-Based MMI Applications

Kouichi Katsurada, Kunitoshi Sato, Hiroaki Adachi, Hirobumi Yamada and Tsuneo Nitta

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

10.21437/Interspeech.2005-587

Abstract

We have developed Interaction Builder (IB), a rapid prototyping tool for constructing web-based Multi-Modal Interaction (MMI) applications. The goal of IB is making it easy to develop MMI applications with speech recognition, life-like agent, speech synthesis, web browsing, etc. For this purpose, IB supports the following interface and functions: (1) GUI for implementing MMI systems without details of MMI and MMI description language, (2) functionalities for handling synchronized multimodal inputs/outputs, (3) a test run mode for run-time testing. The results of evaluation tests showed that the application development cycle using IB was significantly shortened in comparison with the time using a text editor both for MMI description language experts and for beginners.

1. Introduction

Introducing Multi-Modal Interaction (MMI) to web applications has actively been discussed. The WWW consortium organized a multimodal working group [1], and 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 an MMI architecture [4] together with an MMI description language XISL (eXtensible Interaction Scenario Language) [5][6] that has extensibility of modalities, as well as controllability of them. However, since the development of web-based MMI applications needs various types of knowledge concerning modalities and application tasks, it is a hard task and takes a lot of time for evaluation-improvement cycles. For these reasons, a rapid prototyping tool is indispensable in the development of web-based MMI applications.

We have developed "Interaction Builder (IB)", a rapid prototyping tool for constructing web-based MMI applications. IB was developed as a part of Galatea toolkit [7] provided by Galatea Project (Open-source Software for Developing Anthropomorphic Spoken Dialog Agents Project) [8] and has been maintained by its successor consortium ISTC (Interactive Speech Technology Consortium) [9][10]. IB supports the following interface and functions: (1) GUI for implementing MMI systems without the details of MMI and MMI description language, (2) functionalities to deal with synchronized multimodal inputs/outputs, and (3) a test run mode for run-time testing. These facilities help developers to construct web-based MMI applications within a short span of time. IB has a similar interface and functions as a well-known RAD (Rapid Application Development) tool, the CSLU RAD tool [11][12] has. The CSLU RAD tool is not designed for web-based MMI applications but for agent-based speech applications. However, it has user-friendly GUI with various types of useful facilities. Therefore, we take it as a model of IB.

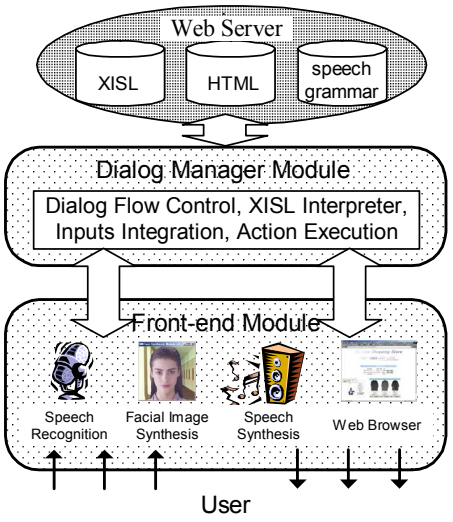

Figure 1: Galatea MMI system

This paper is organized as follows: Section 2 outlines the Galatea toolkit and its MMI prototype system. Section 3 describes facilities of IB and an example of rapid prototyping. After evaluation test in section 4 and comparison with other tools in section 5, we conclude our work and show some future work in section 6.

2. Galatea Toolkit and MMI System

2.1. Outline of Galatea MMI system

Galatea toolkit is a platform to build next generation life-like spoken dialog agent systems. It contains a speech recognition engine [13], a speech synthesis engine [14], a facial image synthesis engine [15], and some other modules. Figure 1 shows the architecture of the MMI system provided by Galatea Project. We call it the Galatea MMI system. Galatea MMI system is divided into two main modules: a front-end module and a dialog manager module.

The front-end module is a UI controller that actuates the above engines and a web browser. It receives speech grammar files and HTML files from the dialog manager module, and delivers them to engines and the browser.

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

Figure 2: An example of MMI description with XISL

The dialog manager is a core module of the MMI system. It controls dialog flows and sends input/output contents to the front-end module. The dialog flows are given as a document written in XISL. The dialog manager first downloads an XISL document from a web server, interprets it, and then executes actions along the scenario of the XISL document. It also has a facility to integrate multimodal inputs sent from the front-end module.

2.2. MMI description language XISL

XISL [5][6] is an XML-based language for describing MMI between a user and a system. In general, an MMI 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 2 shows a fragment of an XISL document[1] for an online shopping system. In this example, the system first outputs an HTML page and a voice prompt ((a) in Figure 2), and then asks the user to select an item. The user's inputs are accepted by the <input> tags in an <operation> tag. In this example, the user can select an item by pointing ((c) in Figure 2) or speech ((d) in Figure 2). Upon accepting the input, the system outputs the name of the item for confirmation ((e) in Figure 2), and goes to the next dialog ((f) in Figure 2).

[1] In this paper, XISL means version 1.1 of XISL.

XISL provides some notations to synchronize input/output modalities. A "comb" attribute in a <operation> tag ((b) in Figure 2) 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>, <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].

3. Interaction Builder

As shown in the previous section, XISL has enough power to describe MMI scenarios including synchronized multimodal inputs/outputs. However, since the description of XISL documents from scratch is not an easy task, some tool is required to reduce this task.

Interaction Builder (IB) is a RAD tool to construct MMI applications written in XISL. When using IB, a developer can design an MMI application without using a text editor. IB generates an XISL document internally. Then the developer tests the MMI application by executing it on the Galatea MMI system. After iterating this cycle, the XISL document is put on a web site. In this section, we discuss design policies for IB first, then present facilities of IB, and lastly we show an executive example of IB.

3.1. Design policies for IB

We set the following policies to design IB.

1. IB should have a GUI-based interface.
2. IB can deal with synchronized multiple inputs/outputs.
3. IB can execute rapid evaluation-improvement cycles.

Policy 1 is introduced for providing a user-friendly interface. Policy 2 is defined for the development of MMI applications using coordinated modalities. Policy 3 is set for constructing an application within a short span of time.

To satisfy policy 1, we employed a similar interface and functions as the RAD tool provided by CSLU toolkit [11][12], which is a well-known tool to construct agent-based speech applications. For policy 2, we introduced the inputs/outputs control mechanism provided by XISL. For policy 3, we incorporated a test run mode into IB.

3.2. GUI-based Interface of IB

The CSLU RAD tool has some dialog components on its main window. The dialog components include agent speech, conditional branch, transition to a subdialog, user's speech input, and so on. An application developer has only to drag one of them, drops it on the center of the main window, and connects with another component. Since this interface design approach is also useful for constructing web-based MMI applications, we employed the approach for designing IB.

Figure 3 shows a screen shot of our prototyping tool IB. The tool bar shown in (1) of Figure 3 contains dialog components to construct web-based MMI applications. Each button corresponds to a dialog component that represents one of an XISL element. An application developer has only to drag one of them and drop it onto a scenario view window

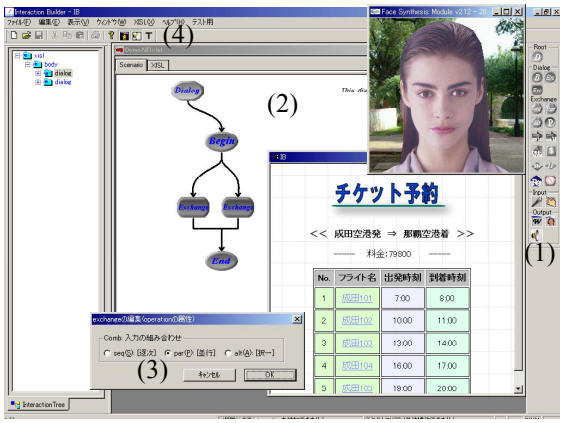

Figure 3: A screen shot of Interaction Builder

((2) of Figure 3) when developing MMI applications. Thus, the developer can easily comprehend the flow of an MMI scenario on this window. The difference between the CSLU RAD tool and our tool will be discussed in section 5.

3.3. Synchronization of multimodal inputs/outputs

As described in section 2.2, XISL provides various notations to synchronize multimodal inputs/outputs. IB supports dialog components corresponding to them. We prepared "par" component, "alt" component, and "seq" component to control synchronization. The synchronization pattern is also set up using a dialog box ((3) of Figure 3) that is popped up when the application developer drops an "Exchange" dialog component onto the scenario view window. The "Exchange" component is a component to set up a turn of interaction between a user and a system. The application developer can select "par", "alt" or "seq" on the dialog box.

3.4. Test run mode

The test run mode is introduced to IB to realize rapid evaluation-improvement cycles. When an application developer pushes a test mode button ((4) of Figure 3), the tool automatically starts up a front-end module and a dialog manager module, and then executes the underdeveloping MMI application. The application developer can confirm system behavior by checking it out on the front-end module and engines. The developer can also check the behavior by watching the dialog manager window that shows a flow of XISL execution.

4. Experimental Evaluation

4.1. Experimental setup

We made experimental evaluation of validity and usability of IB. In the evaluation, eight subjects were asked to develop two MMI applications using IB and a text editor. The application domain is online air ticket reservation. The interaction contains alternative inputs using speech/pointing, conditional branch according to inputs, agent's action and speech output, and web page presentation. The XISL files programmed by subjects are estimated to be almost 50 lines. Four of the subjects are XISL experts who know XISL syntax well, and the others are XISL beginners who have never developed an MMI application with XISL.

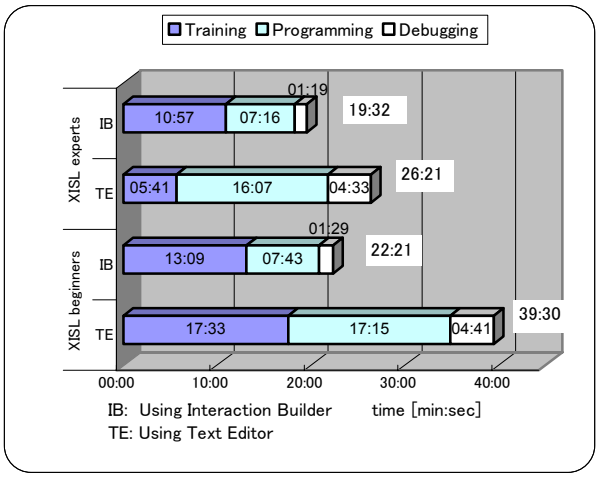

Figure 4: Experimental results of development time

Table 1: The result of questionnaires

Items	XISL Experts	XISL Beginners
Scenario flow comprehension	4.4	3.8
Speech input programming	4.3	4.5
Touch input programming	4.2	3.5
Browser output programming	4.7	4.2
Agent output programming	4.8	4.3
Overall usability	4.6	4.0

1 2 3 4 5

TE is good ◄——► IB is good

We timed the duration of (1) training, (2) programming, and (3) debugging. The training duration is the time to understand how to use IB (when using IB) or how to describe XISL tags (when using a text editor). Programming means the time to develop MMI applications using IB (or a text editor). Debugging is the time to confirm the behavior of the program and modify it. We also carried out questionnaires about functions and usability of IB and a text editor.

4.2. Experimental results

Figure 4 shows the averaged time needed for the application development. Four bars represent the time when XISL experts/beginners use IB/a text editor. The numerical symbols in a bar represent the duration of training, programming, debugging, and total development (from left to right).

The result shows that IB reduces experts' total time by 6 minutes 49 seconds (25.9%), and beginners' total time by 17 minutes 9 seconds (43.4%) as compared with the text editor. Almost all the durations are reduced by IB, however, only the training time of experts increases as compared with the text editor. The reason is that experts have skill at writing XISL documents on a text editor because they know XISL syntax well, but they have never used IB and so have to learn the usage of IB from scratch.

The result of programming time shows that IB reduces it by more than 50%. The developers who use text editor have to type XISL tags with complex properties. On the other hand, the developers with IB have only to drag a dialog component, drop it on a scenario view window, and set up some properties on a dialog box. This difference of usability brought the reduction of programming time.

As for debug duration, IB reduces the time by almost 70%. The reason is that IB does not generate a code containing syntax error, while a text editor often generates erroneous code that increases the debug duration.

4.3. Results of usability questionnaires

Table 1 shows the result of the questionnaires. The subjects are asked to give five-level rating to the functions and usability of IB compared with a text editor (1:TE is very good, 2:TE is good, 3:same, 4:IB is good, 5:IB is very good). The table shows that IB gets higher scores in all the items. However, beginners give a little worse scores than those from experts. The reason is that the dialog components of IB are designed based on XISL, and so beginners have to understand XISL tags when using IB. In addition, the dialog flow of IB is represented as layered style according to the structure of XISL tags. So the developers have to step down several layers to check the dialog flow. We have to redesign the dialog components of IB so that developers can construct applications without the knowledge of XISL.

5. Comparison with Other Prototyping Tools

CSLU RAD tool [11][12] is a model of IB as described in section 3.2. It provides a lot of dialog components for constructing agent-based speech applications using speech recognition, speech synthesis, and life-like agent. The biggest difference between the tool and IB is that CSLU RAD tool does not support synchronized multimodal inputs because its input modalities are speech and DTMF, and no need to support coordination of multimodal inputs. In contrast, IB is designed to develop web-based MMI applications, and so supports alternative/parallel/sequential inputs from microphone, web browser, and so on. Another difference is that CSLU uses animated life-like agent, while our IB uses facial image of a real human.

MPML Visual Editor [16] is a visual editor for constructing web-based MMI applications written in MPML (Multimodal Presentation Markup Language). The tool's interface is different from IB, however, it has almost the same functions as IB. It provides GUI, a function of media synchronization (parallel, sequential, and branching that corresponds to alternative), and so on. The goal of the tool is constructing rich contents using agent presentation. For this purpose, the tool prepares facilities for agent actions such as synchronization of multiple agents, which is not supported in our tool. On the other hand, our tool provides easy way to set up synchronization of multimodal inputs as explained in section 3.3. MPML Visual Editor can also control simple synchronization of multimodal inputs. However, it may be hard to construct complex synchronization of multimodal inputs for the developers.

6. Conclusions

This paper provided a rapid prototyping tool IB for constructing web-based MMI applications. IB has desirable features that (1) GUI-based interface like CSLU RAD tool is provided, (2) synchronization of alternative/parallel/ sequential inputs/outputs is supported, and (3) a test run mode is implemented. The results of evaluation tests showed that the application development time using IB was significantly shortened in comparison with the time using a text editor both for MMI description language experts and for beginners.

Some application developers unfamiliar with the internal language XISL tend to evaluate the usability of IB a little worse, because current IB has some XISL dependent GUI components, and these components give the difficulty of using IB for the application developers. Future work is to develop more useful interface for the developers unfamiliar with MMI description language.

7. Acknowledgements

This work was supported in The 21st Century COE Program "Intelligent Human Sensing" and Grant-in-Aid for Young Scientists (B) 17700185 2005, from the ministry of Education, Culture, Sports, Science and Technology.

8. References

[1] http://www.w3.org/2002/mmi/
[2] http://www.saltforum.org/
[3] http://www.w3.org/TR/xhtml+voice/
[4] Katsurada, K. et al., "A modality-independent MMI system architecture", Proc. ICSLP'02, Denver, United States, pp.2549-2552, Sept. 2002.
[5] Katsurada, K. et al., "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] Kawamoto, S. et al., "Galatea: Open-source software for developing anthropomorphic spoken dialog agents", in Life-Like Characters, ed. Prendinger, H. and Ishizuka, M., pp.187-212, Springer-Verlag, Berlin, 2004.
[8] http://hil.t.u-tokyo.ac.jp/~galatea/index.html
[9] http://www.astem.or.jp/istc/index_e.html
[10] Nitta, T. et al., "Activities of interactive speech technology consortium (ISTC) targeting open software development for MMI systems", Proc. RO-MAN'04, 2B4, Kurashiki, Japan, Sept. 2004.
[11] http://cslr.colorado.edu/toolkit/main.html
[12] Sutton, S. et al., "Universal speech tools: The CSLU toolkit", Proc. of ICSLP'98, pp.3221--3224, Sydney, Australia, Dec. 1998.
[13] Lee, A. et al., "Large vocabulary continuous speech recognition parser based on A* search using grammar category-pair constraint", J. IPS Japan, Vol.40, No.4, pp.1374-1382, Apr. 1999. (in Japanese)
[14] Yoshimura, T. et al., "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis", Proc. EUROSPEECH'99, pp.2347-2350, Budapest, Hungary, Sept. 1999.
[15] Morishima, S. et al., "Better face communication", Visual Proc. ACM SIGGRAPH'95, pp.117, Los Angeles, United States, Aug. 1995.
[16] Prendinger, H. et al., "MPML and SCREAM: Scripting the bodies and minds of life-like characters", in Life-Like Characters, ed. Prendinger H. and Ishizuka, M., pp.213-242, Springer-Verlag, Berlin, 2004.