

A UI DESIGN SUPPORT TOOL FOR MULTIMODAL SPOKEN DIALOGUE SYSTEM

Hiroyuki Kamio , Mika Koorita , Hiroshi Matsu'ura ,
Masafumi Tamura and Tsuneo Nitta
Multimedia Eng. Lab., TOSHIBA Corporation
70, Yanagi-cho, Saiwai-ku, Kawasaki-shi,
Kanagawa-ken, 210 JAPAN

Abstract

In this paper, we describe a user-interface (UI) design support tool to design and evaluate multimodal UI easily. Multimodal UI designers meet the various requirement of system engineers and design panels (panel-data) and describe plan-goal scenarios (process-data). In the UI design support tool, first, designers construct panel-data by setting UI-objects on a card, and then describe plan-goal scenarios linking a UI-object with other cards. Designers are also able to test their multimodal UI on the tool. The UI design support tool offers devicemanager to handle multiple inputs and transforms all the multimodal inputs into an X windows events. The UI design support tool can be applied for various types of multimodal dialogue systems.

1 Introduction

In recent years, user-interface(UI) with speech input facility is applied to such products as a voice-activated ticket vendor, a voice dialing telephone and a telephone service system (ANSER) [1][2]. However, because most of the applications require alternative paths to complete user's demands, multimodal UI with multiple input channels of speech, touch and character, etc. is studied actively [3]. The authors have developed a multimodal keyword-based spoken dialogue system (MultiksDial) [4]. MultiksDial has multiple input channels as well as multiple output channels, and is applied to a directory guidance system.

UI design takes much time for evaluation and improvement. Moreover, because multimodal UI design requires description of a complicated plan-goal scenario, rapid-prototyping is a key-technology for a multimodal dialogue system. We have developed a rapid-prototype system, or a UI design support tool. The tool makes it easy to design multimodal spoken dialogue systems and to evaluate them.

In this paper, we describe newly developed UI design support tool. A UI design support tool adopts GUI to design panels and to describe plan-goal scenarios easily, and the tool can also test the multimodal UI. Section 2 introduces MultiksDial and UI development environment and section 3 explains the architecture and system behavior of the UI design support tool.

2 MultiksDial and UI development environment

Figure 1 shows a blockdiagram of a multimodal keyword-based spoken dialogue system (MultiksDial). MultiksDial offers multiple input channels of spontaneous speech and designation by touch, as well as multiple output channels of graphics and voice responses by text-to-speech [5]. MultiksDial also equippes photoelectric sensors to detect a user's situation, and to construct human-computer interaction.

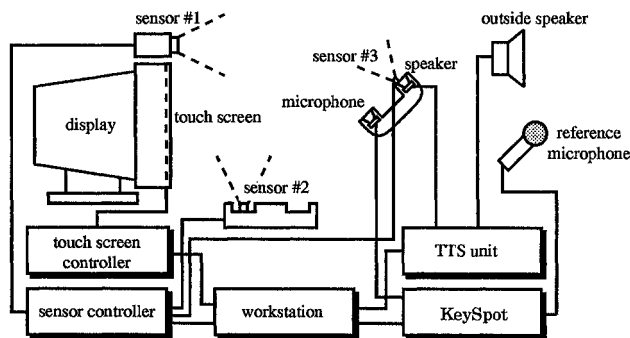


Figure 1: Blockdiagram of a multimodal keyword-based spoken dialogue system (MultiksDial).

A keyword-spotting unit (KeySpot) is based on SMQ/HMM (SMQ: Statistical Matrix Quantization) [6]. KeySpot recognizes 500 Japanese keywords in continuous speech or 1000 isolated Japanese words in real-time. Furthermore, KeySpot implements an adaptive noise canceler using two microphones, therefore KeySpot can recognize contaminated speech in noisy environments.

The photoelectric sensors have several roles (cf. Table 1). MultiksDial uses information from the sensors to grasp user's situation and to give a timely guidance to the user.

A multimodal-dialogue manager on a workstation receives speech recognition results, detection data from a touch screen controller and the user's situations from a sensor controller. The manager also refers to plan-goal scenarios, or data-frames composed of panel-data and process-data. Here, the process-data include a destination panel and preceding action. The manager then executes the action composed of graphic images, displayed instructions and voice responses.

Table 1: Roles of sensors.

	position	role
sensor #1	in front of MultiksDial	detects approaching or leaving
sensor #2	on handset's holder	detects lifting or putting on a holder
sensor #3	near a speaker on handset	detects putting a handset on or off user's ear

We have developed multimodal UI development tool to construct multimodal UI on MultiksDial easily [7]. This tool provides a script (UIScript) with which system developers can describe plan-goal scenarios. A text format file of the UIScript shown in Figure 2 is translated into data-frames and referred by the multimodal dialogue manager. Consequently, a developer can construct multimodal UI environment according to the following steps (cf. Figure 3).

1. A developer describes plan-goal scenarios for multimodal UI by editing a text in UIScript.
2. A UIScript translator converts the UIScript into data-frames which MultiksDial refers to.
3. The multimodal dialogue manager of MultiksDial proceeds in step with the dialogues along plan-goal scenarios described in data-frames.

System developers can also modify their multimodal UI easily with this tool. However, because the developers must understand the grammar of UIScript, a support tool with which they can design a multimodal UI easily without any special knowledge.

```

(colormap
  (color_name gray
   (RGB 54271, 54271, 54271))
  :
)
(panel
  (panel_name start_panel
   (background
    (size 1140, 800)
    (color gray))
   (button start_button
    :
   )
  )
  (process
   (process_name start_process
    (picture_name start_panel)
    (message
     (item_name message_board
      (text "Please touch the start button.")
      (color powder_blue)))
    (button
     (item_name start_button
      (next_process speech_input_process)
     )))
   :
  )
)

```

Figure 2: An example of text data (UIScript) of a plan-goal scenario.

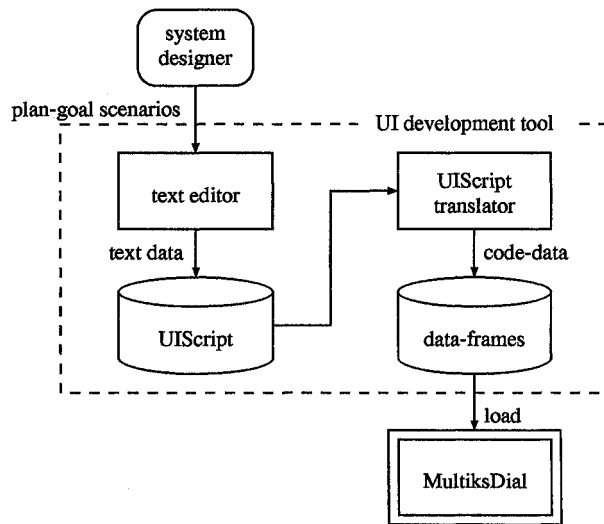


Figure 3: A flow of developing a multimodal UI.

3 A UI design support tool

System developers describe plan-goal scenarios with UIScript in UI development tool described at the preceding. However, because it is not easy to write down whole the scenarios with UIScript for the developers who are not familiar with the grammar of UIScript, we have developed a UI design support tool in which GUI is adopted to construct a multimodal UI easily. We can create both panel-data and process-data through GUI on the support tool and construct multimodal UI as if we use a well-designed authoring tool.

In the support tool, a panel is called a card. And also, elements of the panel like buttons, texts and images are called UI-objects. Designers puts some UI-objects on a card when they design the appearance of the panel (panel-data). Then they link one UI-object with another card or UI-object to determin panel transition (process-data). After constructing the UI, they can test their multimodal UI along scenarios with the tool. If they find some problems in the multimodal UI, they can modify it and reevaluate it easily.

Actual design steps of of multimodal UI with the UI design support tool is as follows.

3.1 the card appearance designing

The designer designs a card's appearance by putting UI-objects on a card (cf. Figure 4). UI-objects are composed of five types of objects: button-objects, text-objects, image-objects, sound-objects, and text-to-speech objects. Button-objects are composed of image-buttons and text-buttons. An image appears on the image-button and text appears on the text-button. Button-objects catch X-Events (Xevents) of clicking mouse button. If a button-object which is linked with other card catches Xevent, it sends a message to the destination card.

The external appearances of text-objects and image-objects are also texts and images, but these objects do not catch any Xevents. Sound-objects and text-to-speech objects output sounds and speeches when the objects receive a message to output from another UI-object.

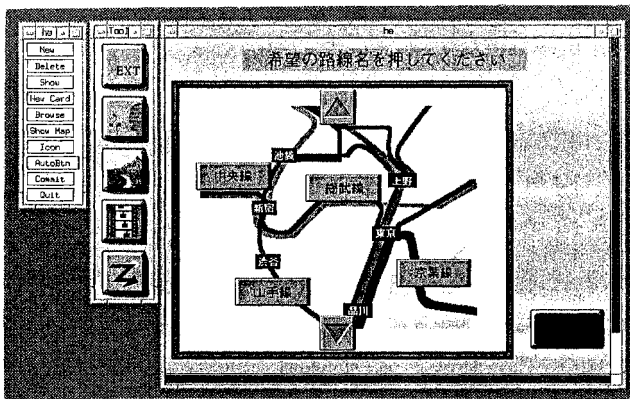


Figure 4: Panel-data are designed by putting UI-objects on a card.

3.2 describing plan-goal scenarios

After designing cards, the designer decides the plan-goal scenario of multimodal UI (cf. Figure 5). They construct the scenario by linking button-objects to other UI-objects or cards. When a button-object catches an Xevent, the UI-object sends a message to the destination UI-object or card along the link (cf. Figure 6). Here, there are several kinds of messages according to the function of the destination UI-object. For example, image-objects receive a message to show and to hide, sound-objects receive a message to play and to stop. When a UI-object sends a message to the destination card to show itself, the card shows the card itself on the display and then card-transition occurs.

3.3 testing the multimodal UI

The designer can test their multimodal UI after describing plan-goal scenarios. To test a multimodal UI, the UI design support tool offers devicemanager which transforms multiple inputs from KeySpot, touch screen controller, timer process and sensor controller into Xevents (cf. Figure 7).

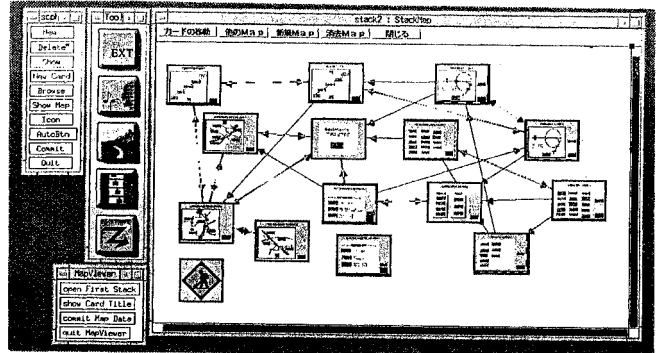


Figure 5: Plan-goal scenarios are described by linking UI objects with other cards.

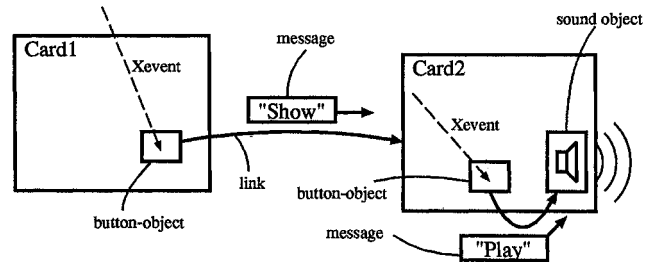


Figure 6: Button-object sends a message to the destination card or UI-object when it catches an Xevent, and then it gets into action according to the message.

If device-manager receives the input from the touch screen controller, it detects a contact spot and sends an Xevent of clicking mouse button to the corresponding UI-object directly. Therefore, the button-object can handle the input from the touch screen in same as an input from a mouse. On the other hand, if the devicemanager receives the input from KeySpot, timer process or sensor controller, it sends an Xevent to a corresponding exclusive object. These are, a speech exclusive object, timer exclusive object or sensor exclusive object respectively. Exclusive objects have many links, and each link corresponds to the result of speech recognition, the state of sensors and timer. When a user's utterance is recognized, first, the devicemanager gets the result from KeySpot, and then sends an Xevent including the speech recognition result to the speech exclusive object. Next, it sends a message to the registered object corresponding to the result (cf. Figure 8). In the same way, the timer exclusive object and the sensor exclusive object send Xevents to the corresponding registered objects when devicemanager gets an input from the timer process or the sensor controller. The UI design support tool can treat all the inputs from multiple input devices as Xevents using the above mechanism.

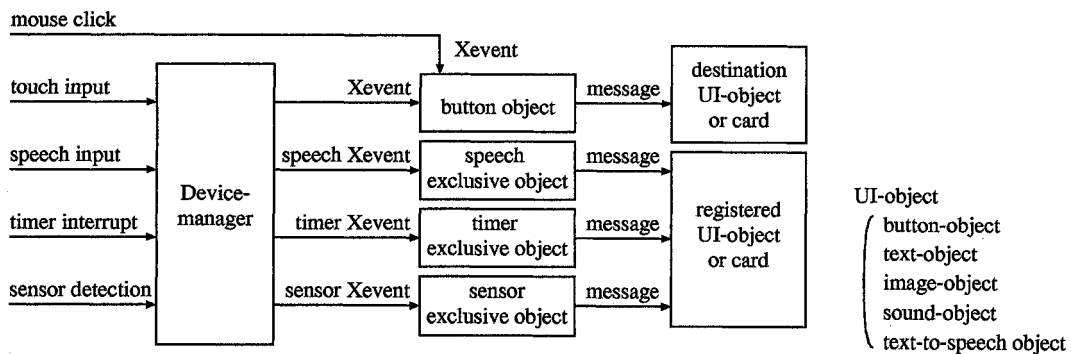


Figure 7: A devicemanager handles speech, touch, timer and sensors.

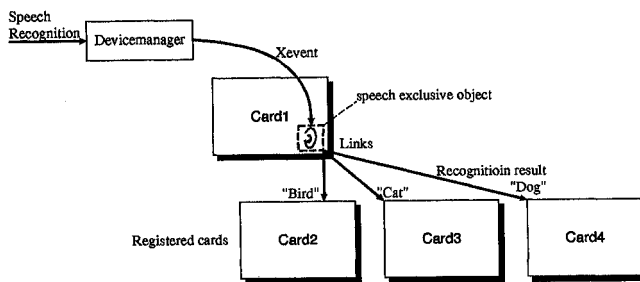


Figure 8: The speech object sends a message to registered objects when it gets an Xevent including a speech recognition result.

3.4 generating UIScript

After testing and reevaluating the multimodal UI, the UI design support tool generates a UIscript for the UI development tool. At this point, a developer can rewrite the UIscript to add special functions, which the UI design support tool does not equip, for example, a character recognition function. Then, a more complicated UI can be realized on MultiksDial.

4 Conclusion

We described a newly developed UI design support tool to construct multimodal UI. The tool enabled us to design a complicated multimodal UI easily as if we were using a well-designed authoring tool.

In future work, we have a plan to apply the tool to multimodal UI design of social-automation systems in the real world, and to improve UI environment of the tool itself.

References

- [1] N.Ohguchi, "Toshiba's Voice Activated Phone Incorporates Artificial Intelligence," Telecommunications, pp.94-96, 1989.
- [2] R.Nakatsu, N.Ishii, "Voice Response and Recognition System for Telephone Information Services," Proc. SPEECH TECH'87, pp.168-172, 1987.
- [3] L.Nigay, J.Coutaz, "A Design Space for Multi Modal Systems: Concurrent Processing and Data Fusion," Proc. INTERCHI'93, pp.172-178, 1993.
- [4] H.Matsu'ura, Y.Masai, J.Iwasaki, S.Tanaka, H.Kamio, and T.Nitta, "Applying a spontaneous speech recognizer, a touchscreen, a rule-based speech synthesizer, and photoelectric sensors to a multimodal dialogue system," Proc. ISSD-93, pp.105-108, 1993.
- [5] H.Saito, M.Kurihara, K.Kobayashi, Y.Hara, and N.Saito, "A Japanese Text-to-Speech System for Electronic Mail," Proc.ICSLP-90, pp.881-884, 1990.
- [6] Y.Masai, S.Tanaka, and T.Nitta, "Speaker Independent Keyword Recognition Based on SMQ/HMM," Proc. ICSLP-92, pp.619-622, 1992.
- [7] H.Matsu'ura, Y.Masai, J.Iwasaki, S.Tanaka, H.Kamio, and T.Nitta, "Multimodal, Keyword-based Spoken Dialogue System — MultiksDial," Proc. ICASSP'94, pp.II-33-36, 1994.