Multimodal Mobile Robot Control using Speech Application Language Tags

Michael Pucher, Marián Képesi

Telecommunications Research Center Vienna, Donau-City-Str. 1, 1220 Vienna, Austria
{pucher, kepesi}@ftw.at

Abstract. This paper describes the design and architecture of a multimodal interface for controlling a mobile robot. The architecture is build up from standardized components and uses Speech Application Language Tags. We show how these components can be used to build complex multimodal interfaces. Basic design patterns for such interfaces are presented and discussed.

1 Introduction

At our institute we develop and investigate multimodal interfaces for mobile devices for next generation telecommunication networks.

Multimodal interfaces provide a promising paradigm for next generation user interface design. These interfaces combine inputs from different modalities like voice, vison and gesture and produce output with different modalities. In a mobile context multimodal input can overcome several restrictions of the mobile input bottleneck. In mobile usage situations the user has to deal with small displays in terms of size and resolution, small and limited keyboards and no access to the visual display. A typical mobile usage situation is driving a bicycle. All these restrictions can be overcome by multimodal interfaces, which can also be used to build more robust speech recognition interfaces.

1.1 Mobile Robotics and Ambient Intelligence

As we see it, mobile robotics is relevant for ambient intelligence in at least two different points:

- Robots can be seen as devices having sensors (inputs) and actuators (outputs) which can be combined to behaviours [5]. The so conceived robots provide an analogy to certain forms of ambient intelligence. The intelligent home for example, can be seen as a device (the apartment) which has a set of sensors (temperature, videocam) and a set of actuators (door, window), which are combined into intelligent behaviours.
- Furthermore a mobile robot can turn a normal apartment into an intelligent home without the need to attach sensors and actuators to the "home" directly. If I move to a new apartment, I can place my personal robot there and turn my new apartment to an intelligent one. All functions, like measuring the temperature, checking if the lights are off, that are not apartment-specific can be used. The intelligence of the robot is transferred to the apartment in this way. This ambient intelligent solution is also mobile.

1.2 Multimodality and Ambient Intelligence

Ambient intelligence should allow the user to interact naturally with an intelligent interface. Because human face-to-face communication is multimodal, multimodal communication is perceived as natural and intelligent. Therefore well designed multimodal interfaces can have these required properties of ambient intelligence.

2 The Application

Our web interface allows a user to control a robot using a multimodal user interface which shows the map of that room where the robot is located including the robots location. It is possible to send the robot around and let it fulfill different tasks like measuring the temperature or checking if the windows are closed. For example the user could click on the room-icon and ask "What's the temperature in this room?".

At the moment we have only implemented dummy sensors, which produce a random sensor output, because our main interest was to investigate the relation between the multimodal interface and the (partially simulated) robot capabilities. The user can also ask the robot about it's functionality or use voice commands to control the robot remotely. The robot “answers” the question posed by the user, by using a Text to Speech (TTS) system. (A Text to Speech system converts a given text to synthesized speech)

Our application implements third order multimodality [9], which is the most advanced form of multimodality and allows coordinated, simultaneous multimodal input. This means that the input from different modalities produces multimodal events which drive the application.

We see the main advantage of our architecture in the use of standardized components and protocols which make the architecture flexible and maintainable.

3 Standardized Components

3.1 SALT

Speech Application Language Tags [8] (SALT) are a standardized set of EXtensible Markup Language [10] (XML) tags, which can be used inside of HyperText Markup Language [11] (HTML) pages. The standardization of SALT is driven by major companies in the field. At the moment SALT is available for desktop browsers and Pocket PC browsers [6].

The Microsoft Speech SDK (Software Development Kit) Version 1.0 Beta 3 already supports the Pocket PC platform which runs on mobile clients. Our application was implemented using an older version of the SDK which supported only desktop browsers. Several implementations of SALT browsers can be found on [8].

We will show how one can deal with the complexity of multimodal interface development using SALT. Any SALT enabled client can control the robot via the HyperText Transfer Protocol [12] (HTTP). To interpret the SALT pages the browser needs to have access to a local or remote Text to Speech (TTS) and Automatic Speech Recognition (ASR) systems. (Automatic Speech Recognition means the “correct” conversion of speech into text)

3.2 Java

Java is a programming language and platform developed by Sun Microsystems [2], which provides amongst other things, standardized interfaces for building web applications.

Java Server Pages (JSP) are programs which run on a server. They can be called by a client via HTTP, keeping track of the clients state and producing dynamic output for the client. The advantage of Java Server Pages for our application is, that we can include the dynamic and static HTML, and the control of the robot in one JSP.

4 Architecture

A complete voice platform consists of at least three components. The TTS and ASR, and the dialog management. The dialog management manages the applications dialog flow.

In Figure 1 the building blocks of our application are shown. The logic of the application is distributed between the SALT client, the Webserver and the Robot.

The dialog management is done by the SALT client. The SALT tags in combination with JavaScript functions allows us to implement the dialog. (JavaScript is a scripting language which can be used inside of webpages and is interpreted by the clients Webbrowser) The dynamic SALT pages are requested from the Webserver and are sent to the client via HTTP. They include a response message from the robot.

The Graphical User Interface of the SALT client presents a map including the location of the robot. It can be used to send the robot around, and track it’s movement.

The Webserver establishes the communication between the client and the robot. The robot’s default interface is an infrared transmitter/receiver. Because infrared receivers only work in line of sight, this restricts the robot’s mobility. A radio interface to the robot was implemented so that it could be controlled also when out of sight. The robot communicates through frequency-shift-keyed (FSK) data using an 868 MHz transceiver. (transmitter/receiver) The FSK is used to modulate the binary data in order to achieve errorless transmission of robot control commands. We call these two components radio/infrared proxy and infrared/radio proxy in analogy to the term’s use in IP networks.

The robot interprets the commands it gets from the client. The commands are divided into simple commands which translate into simple robot actions, such as “Turn right”, “Stop” etc. and complex commands, like “Is the light in this room on?” which trigger complex robot programs. There are some commands, like “What can you do?” which are processed directly by the Webserver. In case of this command a string with a description of the robots functionality is returned.

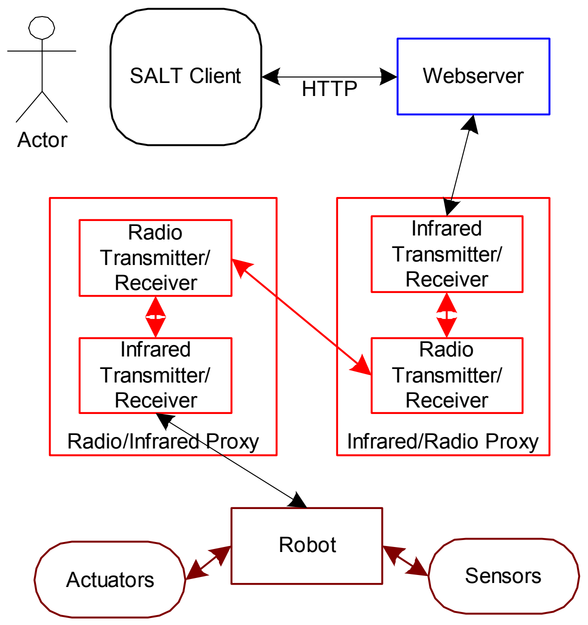
Actor
SALT Client
HTTP
Webserver
Radio Transmitter/ Receiver
Infrared Transmitter/ Receiver
Radio/Infrared Proxy
Infrared Transmitter/ Receiver
Radio Transmitter/ Receiver
Infrared/Radio Proxy
Robot
Actuators
Sensors

Fig. 1. Logical Architecture of a multimodally controlled robot

Figure 2 shows the system diagram of the SALT client and the Webserver. In our case a laptop computer is used as a client running a SALT multimodal browser, but a palmtop computer, a smartphone or any other mobile SALT enabled client could be used.

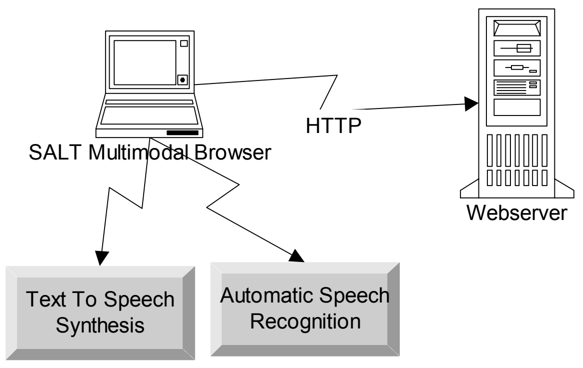

Fig. 2. System Architecture

The multimodal browser accesses the TTS and ASR, which runs on the same machine as the browser in our case, but could also be remote.

The Webserver returns SALT pages in the users language. Localization of applications, e.g. translation into other languages is easy from the application providers point of view, because the SALT client chooses the ASR and TTS to use.

5 Multimodal Dialog Management using SALT

5.1 Dialog Managment

Block schemes of finite-state machines describing the two different standard SALT tags are shown on Figure 3. The first tag is called SALT:prompt. It plays a specified prompt to the user, using the systems TTS.

The second tag is SALT:listen which starts a recognition process for a specified amount of time and goes into one of the end states afterwards. Both objects “start” methods can be called inside of JavaScript functions.

The description of the SALT tags as finite-state machines is convenient for the definition of the systems’s application logic. Finite-state machines are a common method for the design of dialog systems. These blocks can be combined to implement the dialog management of our application. Figure 4 shows a design pattern for an application that uses two standard SALT tags. This pattern could be called the "endless prompt and recognition dialog", where the recognition determines the next prompt. The application plays a prompt, tries to recognize the user's speech command (specified by a grammar) and loads a new Webpage. At this point the application starts again.

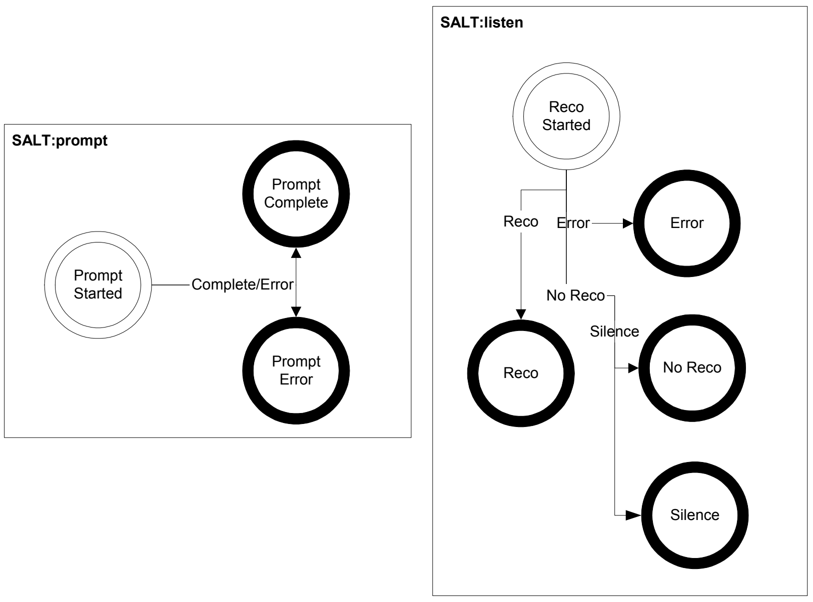

Fig. 3. Recognition and Synthesis Blocks

As one can see from Figure 4, the application runs endlessly unless the user closes the browser. The starting point is the starting of the prompt where the synthesized string depends on the dynamic SALT content generated by the JSP. If, for example, the last recognized command was the question "What can you do", which was sent to the robot, then the synthesis string will be "I can walk around, measure the temparature,…". When the SALT page is loaded for the first time the user is welcomed.

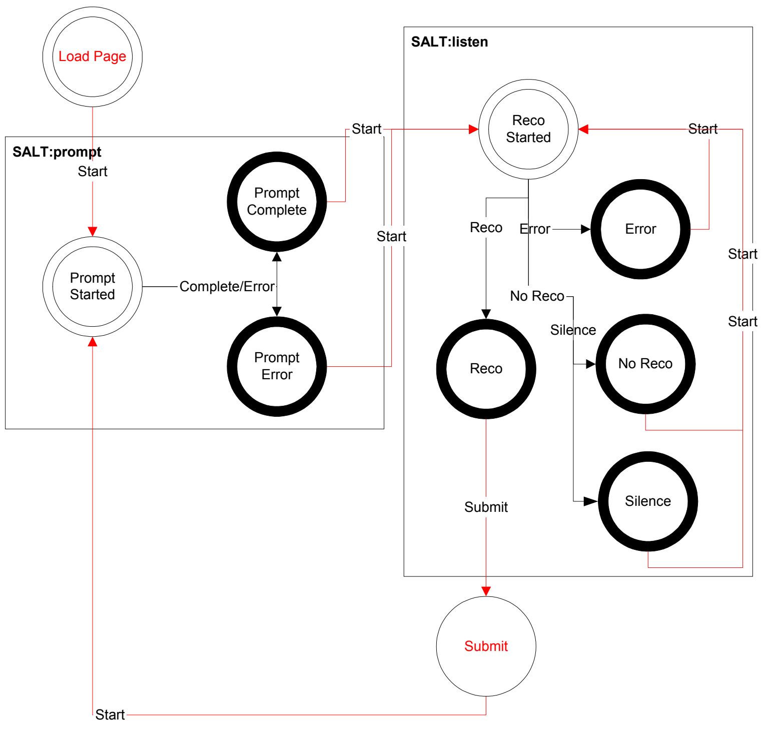

Fig. 4. Dialog Management of the application

The prompt has to terminate before the recognition is started, because otherwise it can happen that the robot gives itself commands. If the synthesis string is "I am going forward" and "forward" is also present in the list of commands, the robot will tell itself to walk forward, if the synthesis and recognition process overlap and the synthesis is played using the loudspeakers.

When the prompt ends the recognition process is started. While the recognition is running the user can see a progress bar showing the voice energy on the screen. If the recognition ends with an error it is started again, otherwise a JSP is called that generates a new SALT page.

In case of multimodal commands the JSP is only called if certain visual (mouseclicks) and speech commands occur at the same time.

This dialog can be seen as a design pattern for multimodal remote control dialogs. One important concept which is not mentioned in the above diagram, but is needed for the SALT:listen tags are grammars.

They can be specified in the HTML page or in a separate file.

```
<SALT:listen id="MainReco" onreco="HandleOnReco()"…>
<SALT:grammar …>
<grammar version="1.0" >
<rule id="numbers">
          <one-of>
          <item>forward</item>
          <item>back</item>
          <item>left</item>
          <item>right</item>
          <item>stop</item>
          <item>bye</item>
          …
          </one-of>
</rule>
</grammar>
</SALT:grammar>
</SALT:listen>

function StartRecognition()
{
          try{
          MainReco.Start();
          }catch(e){
          alert("Recognition error"); }
}
```

The above code shows the definition of a SALT:listen tag with an associated grammar and a JavaScript function which is called to start the recognition. In the first line of the SALT:listen tag the function is defined which is called when the object comes into the “reco” state, meaning that something was recognized.

5.2 Multimodal Integration

Under multimodal integration we understand the process of combining the input from the different modalities to a merged multimodal input and the generation of a multimodal output from this input. The first step is done by the SALT client in our case, while the second processing step is done at the server.

The dialog management and the multimodal integration are nicely separated in our application. The integration happens at one point in the applications dialog flow, namely before the submission of the webpage.

This separation is specific for our application, because SALT allows us to merge the dialog and the integration at any point in the above state diagram.

There are many different formalisms for multimodal integration like typed feature structures [3] or rule-based integration. We implemented a simple rule-based system using JavaScript functions. This method is sufficient to implement multimodal systems that support coordinated, simultaneous multimodality which is the most advanced form of multimodality.

The ontology of the application can be derived from the applications rules. “Ontology” in this context means the basic things or events that are represented in different modalities. The object/event “Hello World” could be represented as a text string or as a voice command. To know which representations refer to a common object one has to look at the rules of the system. W. Wahlster recently argued that a multimodal system should have an underlying ontology for the system to understand its own multimodal output [13]. To implement this form of “symmetric multimodality” using SALT one would have to add an ontology module at the server side.

One of our rules is, that if a room on the map is choosen with a mouseclick and the speech command “What’s the temperature here?” is uttered at the same time, then these inputs will be combined to a multimodal command.

```
if(submitForm.x1.value != "0" && submitForm.sp.value == "temperature")
submitForm.submit();
```

The code above shows the part of a function which implements this rule. Through the combination of multimodal rules complex interfaces can be created.

Although the multimodal integration can be triggered at any state of the dialog, the integration and dialog can be logically seperated using this rule-based integration design.

6 The Robot

Our robot was built with the Lego Mindstorms kit [7]. It can drive around, has a touch sensor and a simulated temperature and vision sensor. To enable the communication with the Webserver we installed the Lejos Java Virtual Machine on the robot, so that we can run small Java programs on the robot [4].

The behaviour of the robot, including the navigation was implemented according to the behaviour control theory from [1].

7 Conclusion

We showed how to build a sophisticated multimodal interface using standardized components. We also discussed the relevance of these kinds of interfaces for ambient intelligence. Such standardized components are of special importance for future interfaces for mobile devices. There are and will be many different types of such devices, leading to a crucial need of having standardized basic building blocks.

The work presented here is part of the project Speech&More carried out at the Telecommunications Research Center Vienna (ftw.) ftw. is supported by the Kplus program of the Austrian Federal Government.

References

1. Bagnall B., Core Lego Mindstorms Progamming, Prentice Hall 2002, pp 196
2. Java 2 Platform, Enterprise Edition (J2EE), http://java.sun.com/j2ee
3. Johnston M., Unification-based Multimodal Parsing, COLING-ACL, Montreal, Canada, 1998, pp 624-630
4. Lejos – Java for the RCX, http://www.lejos.org/
5. Maes, P. and R. A. Brooks, Learning to Coordinate Behaviors, AAAI, Boston, MA, August 1990, pp. 796–8
6. Microsoft .NET Speech SDK Version 1.0 Beta, http://www.microsoft.com/speech/getsdk
7. Mindstorms - Robotics Invention System 2.0, http://www.mindstorms.com/
8. SALTFORUM: Speech Application Language Tags, http://www.saltforum.org
9. W3C: Multimodal req. for Voice Markup Lang. Working draft 10. July, 2000, http://www.w3.org/TR/multimodal-reqs
10. W3C: http://www.w3.org/XML/
11. W3C: http://www.w3.org/MarkUp/
12. W3C: http://www.w3.org/Protocols/
13. Wahlster, W., SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell In: Krahl, R., Günther, D. (eds): Proceedings of the Human Computer Interaction Status Conference 2003, June 2003, Berlin: DLR, p. 47-62