Telephony Applications using Microsoft Speech Server

Hao Shi and Sebastian Auer[*]

School of Computer Science and Mathematics
Victoria University, Melbourne, Australia

Summary
Many people call support hotlines every day as they need assistance with products provided by companies or they just want to check their account balance. The calls like this quite often placed in a queue for a long time before it is answered. It frustrates both the users and the company. In many cases this kind of calls should be automatically answered and support staff is only required if the call cannot be processed by the system. This helps people to get their information faster and also reduces the significant amount to companies. The waiting times will be much shorter and therefore will be less frustrated. However, the existing telephony technologies are very experience and small to medium companies cannot afford. Microsoft has built its Speech Server which can give these companies a chance to create a low cost telephony interface for their existing websites. This project aims to build a simple application which demonstrates what the latest Microsoft Speech Server can offer and how it can interact with the user by converting text to speech and by recognizing the users voice and react to user's input. In this paper, the detail requirements, step-by-step design and implement are addressed and a fully working telephony application is presented.

Key words:
Telephony, Applications, Microsoft Speech Server, eCommerce

1. Introduction

Many companies have already used telephony applications for various reasons, however creating and deploying such an application can get very difficult [1]. Microsoft Speech Server 2004 should make things easier and gives even small to medium companies or organisations the chance to use speech based interfaces. Accuracy of speech recognition was under constant development in the last years [2] and is now ready for productive use. A few researches recently have been carried out using VoiceXML ([3], [4]). However in this project, Microsoft Speech Server as well as SALT technology is used [5]. In this paper, the main goals are to demonstrate what is possible with the Microsoft Speech Server and to provide a showcase for future development.

2. Requirements and Software Installation

In order to develop and run telephony applications based on the Microsoft Speech Server the basic requirements for software and hardware are:

- Microsoft Windows Server 2003 with SP 1
- Microsoft .NET Framework 1.1 with SP 1
- Microsoft Speech Server 2004
- Microsoft Visual Studio .NET 2003
- A suitable Telephony Interface Manager
- Telephony board, otherwise only Voice-Over-IP

Of course a recent computer is needed to run the software although the speed of this computer is not very important. However if a solution like this is deployed in a company which receives a lot of simultaneous calls the server needs to be capable of handling the workload created by the Speech Server. First Telephony Interface Manager (TIM) has to be installed before the Speech Server and should not be replaced without a new installation. The Speech Server setup itself emerged as quite a challenge, it took a lot of time to get it running properly, even after a successful installation (according to the setup program). In this project only Voice-Over-IP was used for simulation purpose. Once a telephony board is installed on the server computer, the system should be able to communicate with the normal telephone lines.

3. Telephony Application Design

To demonstrate how a telephony application works, a ported version of the GrocerToGo eCommerce shop [7] was chosen and implemented. The GrocerToGo is a sample application shipped by Microsoft with Visual Studio .Net which is a simple web browser-based shop interface. The products of the shop have been customized to Australian local products and it was implemented for telephony only.

The flowchart of the system is shown in Fig. 1 and the telephony application set to achieve the following goals:

- interacting with the user by telephone/VOIP-software
- ordering items and checkout by voice
- two modes: beginner mode and advanced mode
- basic error handling for no input and wrong input

[*] Sebastian Auer was on internship at Victoria University, Australia.

Manuscript received August 5, 2006.
Manuscript revised August 25, 2006.

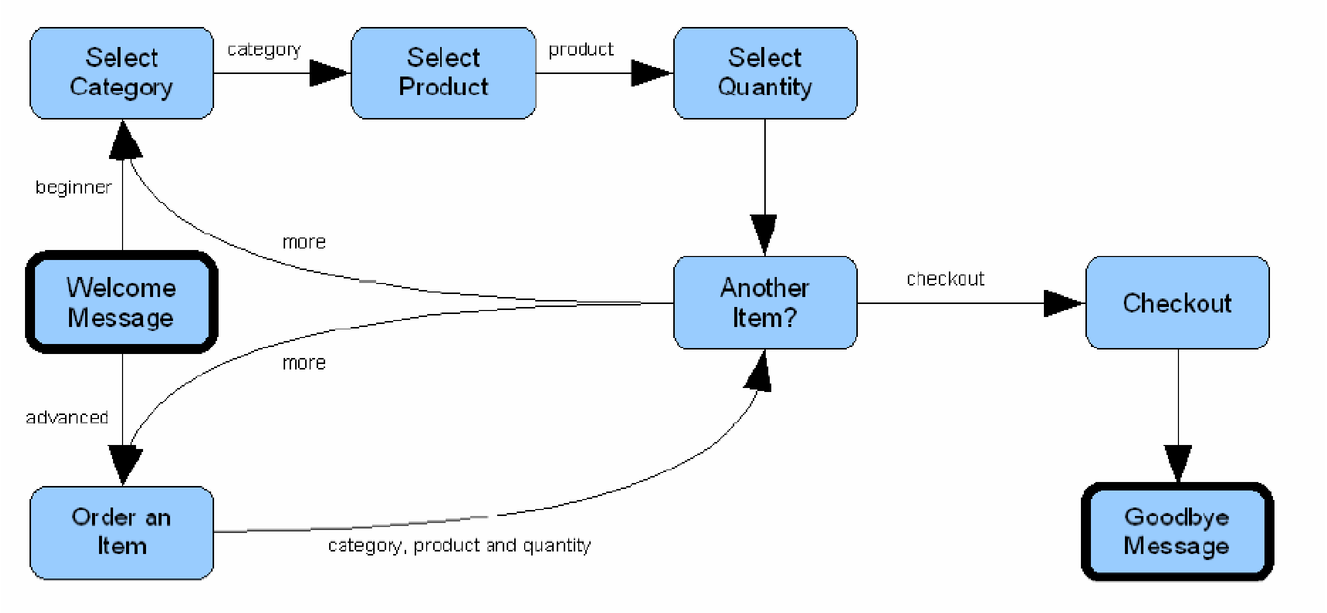


Fig. 1 Flowchart of the System

When the user calls, the first thing he hears is a welcome message and the ability to choose from beginner or advanced mode. In advanced mode the caller can order an item in one single step, he/she can tell the system the category, the type and the quantity together. A sample communication is shown in Table 1.

Table 1 Sample interaction in advanced mode

Computer:	Welcome to the GrocerToGo telephone shop. If you are new to this system please say 'beginner' to get a step-by-step introduction, advanced users please say 'advanced' to directly order items. Say 'beginner' or 'advanced' now.
User:	"Advanced"
Computer:	You have chosen 'advanced'. Please order an Item.
User:	"Seven Farmland Milk"
Computer:	You have ordered seven Farmland milk. Please say 'more' to order another item, or 'checkout' to finish shopping.
User:	"Checkout"
Computer:	You have chosen to checkout. The total price of your ordered items is nine dollars and three cents. Thank you for using the GrocerToGo telephone shop!

In beginner mode, the caller is guided through the process step-by-step. At first he/she chooses the product category (milk, soda or cereals), after that he/she will be asked what type of this product he/she wants and finally how many he wants to buy. At the end it is possible to checkout, as well as return to the beginning and more items can be ordered. A sample communication is recorded in Table 2.

Table 2 Sample interaction in beginner mode

Computer:	Welcome to the GrocerToGo telephone shop. If you are new to this system please say 'beginner' to get a step-by-step introduction, advanced users please say 'advanced' to directly order items. Say 'beginner' or 'advanced' now.
User:	"Beginner"
Computer:	You have chosen 'beginner'. You can order milk, cereals or soda. What do you want?
User:	"Milk"
Computer:	We have Skinny, Paul's, Farmland and Pura milk. What kind of milk do you want?
User:	"Farmland"
Computer:	Our price for Farmland milk is one dollar, twenty-nine cents. How many would you like?"
User:	"Seven"
Computer:	You have ordered seven Farmland milk. Please say 'more' to order another item, or 'checkout' to finish shopping
User:	"Checkout"
Computer:	You have chosen to checkout. The total price of your ordered items is nine dollars and three cents. Thank you for using the GrocerToGo telephone shop!

The call is automatically disconnected after the goodbye message at the end.

4. Implementation

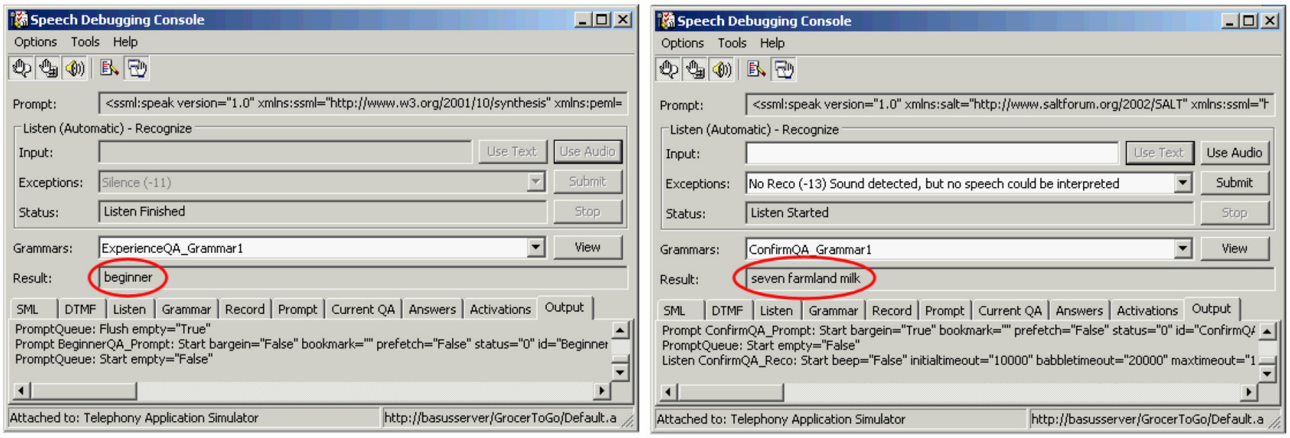
The whole application is coded in C# and JavaScript (ECMA standard) using Microsoft Visual Studio 2003.

After installing the Speech Application SDK (not to be mixed up with “Speech SDK”) a special kind of project named “Speech Application” is created. This project aims to reuse the existing company website so that the Telephony Application Server always loads its input from a web server. For that reason a web server with IIS has to be accessible from the development PC. After creating the project a new section can be seen in the toolbox and speech related objects can then be dragged onto the design interface.

QA (Question-Answer) Controls handle the prompts to the user and the recognition of his response. The prompt can be specified in the QAs property dialog if it is fixed, it can also be determined dynamically at runtime by using a prompt function. A prompt function is a JavaScript function which is called at runtime just before the speech output and returns the text to be spoken. Normally all prompt functions for one project (or one part of a project) are located in a prompt file which can be edited using Visual Studios built-in editor and is automatically included in the source. Prompt files are very useful, for example if a prompt is called again because of a recognition error it would be helpful to play a prompt that indicates the recognition error to the user instead of the same prompt again. Sample codes for selecting the right prompt are listed as below:

```
switch(lastCommandOrException) {
   case "Silence":
       return "Sorry, i didn’t hear you. Please order an item."
       break;
   case "NoReco":
       return "Sorry, i didn’t understand that. Please order an item."
       break;
   case "Repeat":
   default:
       return "Please order an item.";
       break;
}
```

Recognition is also a part of a QA; it uses grammars to determine what the user says. Grammars can be easily created by using Visual Studios grammar editor. When the users’ speech input equals one of the active rules, the QA returns a value specified in the grammar and saves this value into a semantic object, from where it can be accessed by prompt functions and activation functions as shown Fig. 2.



Prompt: <ssml:speak version="1.0" xmlns:ssml="http://www.w3.org/2001/10/synthesis" xmlns:peml=

Input:

Exceptions: Silence (-11)

Status: Listen Finished

Grammars: ExperienceQA_Grammar1

Result: beginner

```
PromptQueue: Flush empty="True"
Prompt BeginnerQA_Prompt: Start bargein="False" bookmark="" prefetch="False" status="0" id="Beginner
PromptQueue: Start empty="False"
```

Attached to: Telephony Application Simulator | http://basusserver/GrocerToGo/Default.a

Prompt: <ssml:speak version="1.0" xmlns:salt="http://www.saltforum.org/2002/SALT" xmlns:ssml="h

Input:

Exceptions: No Reco (-13) Sound detected, but no speech could be interpreted

Status: Listen Started

Grammars: ConfirmQA_Grammar1

Result: seven farmland milk

```
Prompt ConfirmQA_Prompt: Start bargein="True" bookmark="" prefetch="False" status="0" id="ConfirmQ/
PromptQueue: Start empty="False"
Listen ConfirmQA_Reco: Start beep="False" initialtimeout="10000" babbletimeout="20000" maxtimeout="1
```

Attached to: Telephony Application Simulator | http://basusserver/GrocerToGo/Default.a

Fig. 2 Speech Debugging Console showing recognized values

Another important point is to determine whether a QA should be executed or not, in order to realise a dialog flow as intended. Activation functions are used to achieve this, they are client-side JavaScript functions like the prompt functions 4 and are executed just before the QA itself is executed. If an activation function returns true the QA will be run, if it returns false the QA will be skipped. Sample activation function is listed below:

```
function activateAdvancedQA()
{
    if (siExp.value == "advanced") return true;
     else return false;
}
```

Commands are also used in the demonstration, for example the ”repeat” command which allows the user to have the last prompt played again in case he/she didn’t completely understand it. Commands are grammar rules which are always active while the execution is in the selected context. The user can say “repeat” at any time, if he does so the current output will be stopped if running and replayed. A lot of things are handled by client-side JavaScript functions and commands in a telephony application, so basically the properties in design mode and program JavaScript need to set.

5. Conclusions and Future Work

It is certain that telephony applications already play an important role in eBusiness environments and their influence will continue to grow. Telephony applications can be used in various areas and can easily handle simple tasks like ordering a pizza, recovering a lost password or just redirecting the calling user to the right support representatives. Microsoft Speech Server is a powerful tool which allows creating such applications with a moderate amount of effort. A few things could definitely be improved about Microsoft Speech Server, the lately announced new version MS Speech Server 2007 could make it even more comfortable to create and deploy telephony Applications. Another thinkable scenario is to use existing database structures, for example the database of a web application, and use the same data for the new telephony application. With this approached the data are synchronised to the existing database and the telephony application will have the same functionality as the web application. This would be a great benefit for visually impaired people and for those who do not have access to the Internet but still are able to place telephone calls.

References

[1] Li Deng and Xuedong Huang, 2004, Challenges in Adopting Speech Recognition, Communications of the ACM, vol. 47, no. 1, pp. 69-75

[2] Microsoft Corporation, Speech Technology Home Page, http://www.microsoft.com/speech/default.mspx .

[3] Mecanovic, Daniel, and Shi, Hao (2005) 'Voice User Interface Design for a Telephone Application Using VoiceXML', Lecture Notes in Computer Science, No. 3399, Web Technologies Research and Development, pp. 1058-1061.

[4] Vankayala, Rahual R and Shi, Hao (2006) 'Dynamic Voice User Interface Using VoiceXML and Active Server Pages', Lecture Notes in Computer Science, No. 3841, Frontiers of WWW Research and Development, pp. 1181-1184.

[5] Speech Application Language Tags (SALT) Forum, http://www.saltforum.org

[6] Vail Systems Inc., Vail SIP TIM Home Page, http://www.vailsys.com/mss/index.html .

[7] GroceToGo, An eCommerce Storefront http://samples.gotdotnet.com/quickstart/aspplus/doc/ecommerceapp.aspx ,

Dr. Hao Shi obtained her BE in Electronics Engineering from Shanghai Jiao Tong University, China in 1986. She joined the then Department of Electrical and Electronic Engineering, Victoria University as a Lecturer after completion of her PhD at University of Wollongong in 1992 and was promoted to a Senior Lecturer in 2001. She joined School of Computer Science and Mathematics in March 2003. She has been actively engaged in R&D and external consultancy activities. Her research interests include p2p Network, Location-Based Services, Web Services, Computer/Robotics Vision, Visual Communications, Internet and Multimedia Technologies.

Mr. Sebastian Auer is a 2nd year Computer Science student at University of Applied Sciences (http://www.fh-landshut.de), Germany. He was supervised by Dr. Hao Shi as an international intern student at Victoria University from February 2006 to July 2006 and successfully completed speech applications using Microsoft Speech Server.