PharmaSUG 2014 - Paper AD18

Don't Type, Talk SAS®

Vikas Gaddu, Anova Groups, Cary, NC, USA
Smitha Madhu, Anova Groups, Cary, NC, USA

Abstract

Are you a lazy but smart programmer? Then you will like my paper!!

We have seen that SAS® programs over the last decade have started becoming more and more standardized. We have standard headers, standard init files, standard macros and standard ways to output reports and each client has their own standards. In turn, we are required to remember lot of standards and their documentation. What if we could talk to SAS and these standard structures would automatically appear in your code? This is now possible with Windows Speech Recognition Macros.

Introduction

Speech recognition has improved tremendously over the last decade. Take it from a person with an accent. It supports communication between humans and computers in a natural way in your voice. We have lots of commercially available voice recognition softwares that can interpret human speech and convert it into a text, or listen for a keyword and perform a bunch of tasks. Even though the technology seems sophisticated and expensive, Microsoft has made speech recognition easily available to us. In this paper, we are going to discuss about this freely available software in all the windows PC called Windows Speech Recognition Macro (WSRM) and how we could take advantage of it in SAS programming.
Mundane tasks like writing standard headers, standard macros and comments can be easily done using windows speech recognition.

- This can be very useful for people with disabilities or people suffering from Spondylosis or Carpel Tunnel syndrome.
- This encourages programmers to follow standards so that they don't have to read through PDF files to understand these standards.
- Coding efficiencies and standards can be easily enforced.
- Using Voice Command helps you to concentrate more on your logic and less on the syntax.

However, we do understand that this cannot replace the traditional 'keyboard –mouse' usage for programming but, when used in combination with Voice Commands; it can really speed up the coding process.

Where to find it and how to install it?

- Download the WSRMacros.msi file.
- Control Panel >Speech Recognition>Set up your microphone (can take the speech recognition tutorial)

The program will install itself in the C:\Program Files\WSRMacros\folder and the executable file is called, you guessed it right, WSRMacros.exe. The WSRMacro application gives you the ability to write XML program that allows windows operating system to listen for keywords and then process the tasks as defined in the XML program.

How it works

Just as how SAS macros help us to do our job quickly and efficiently, WSRM can also help us in achieving higher levels of efficiency.
Every day when I fire up my computer to start working on a study I have to open up lot of documents stored at various locations. WSRM can help us to open all these documents with just one single command.
The most powerful feature of WSRM is that it uses XML language to create macros. XML is a standard set of rules that specify ways to organize or "mark up" text files. These rules are called a *schema* and have been developed by

Microsoft specifically for processing speech macro-related commands. When we type in a macro and save the file, the Windows Speech Recognition Macro facility compares what's in our file to the schema. If it matches, the macro facility will then compile the macro in a low-level computer language that the computer can execute. If it doesn't match, we'll get an error message telling there was a problem. We can also use VB script within a speech macro to add more intelligence to our speech macro.

As explained before, XML needs to be well formed as per the schema, otherwise; windows won't know what we want it to do for us. You also need to be careful about the case, since XML is case sensitive. The figure 1 on the left shows the correct way of opening, closing and embedding tags in WSRM macro, while figure on right is the incorrect way.

Figure 1:

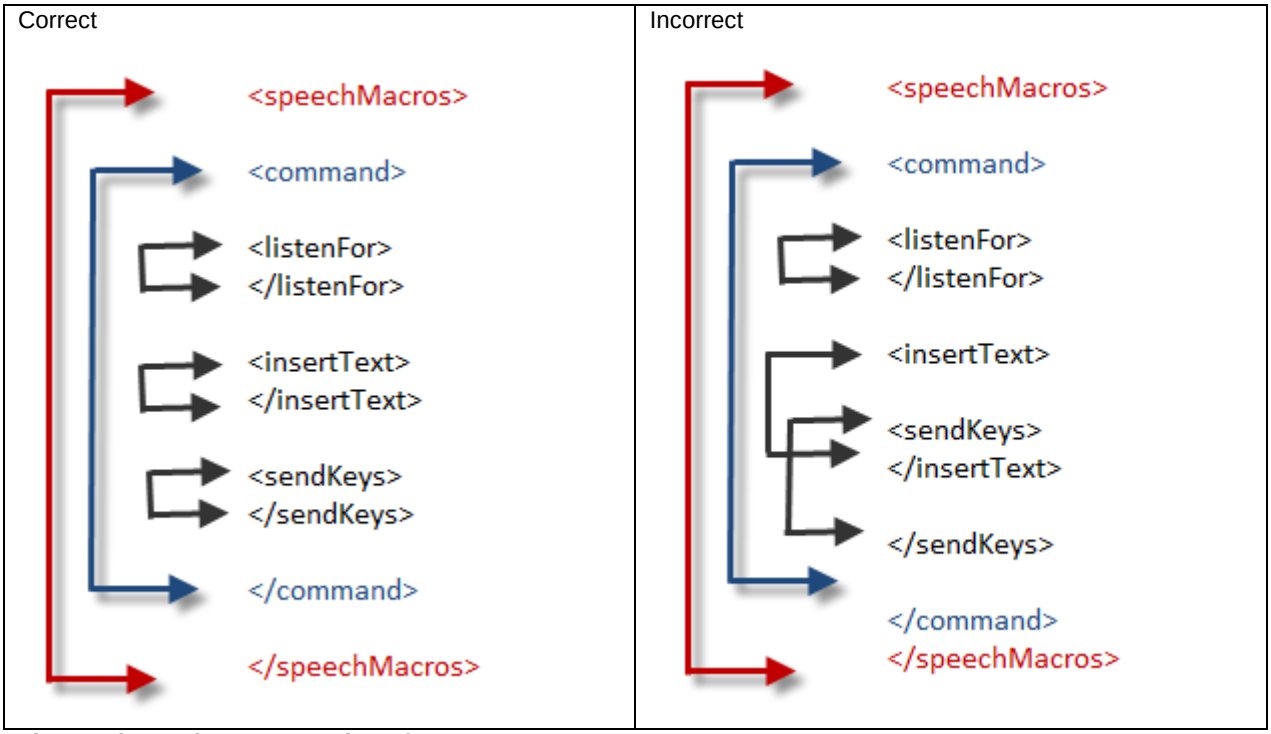

Figure 1 is a Schema Formation of XML Tags

- All WSR Macro files start and end with a "speechMacros" tag.
- Nested within it is the opening and closing "command" tag. A series of tasks can be provided within it as a single command.
- The next command as the name suggests is what acts as the trigger. The computer will listen to the text enclosed within the "listenFor" tags. You insert the action that the computer should execute between the insert text.
- If you want to include raw character data (VB or Java script program inside the macro) within the insertText tags, you use the CDATA tags.
- The "sendKeys" tag is used to enclose the computer keys for a specific command like for instance, {{Cntr}} s for saving a file.
- One thing to remember is you need to save the XML file with the same name as the text you insert within the "listenFor" tag and save it with an extension of WSRMac.
- Do not forget to to add the signature to the document.

Your First Windows Speech Recognition Macro

Here' a simple example of the usage of WSR macro to save a file using the "sendKeys" tag.

Example:

```
<?xml version="1.0" encoding="UTF-16"?>          All WSR Macros have this tag with the XML version and encoding details.
<speechMacros>                                   Starting tag
```

```
  <command>                                  A series of tasks can be provided
                                             within the command tag.
    <listenFor>save file</listenFor>         Action to be taken is provided in this
                                             tag. It acts as the trigger.
    <sendKeys>{{CTRL}}s</sendKeys>           Provide the shortcut keys for the
                                             action to be taken.
  </command>                                 Closing command tag
</speechMacros>                              Ending tag
```

Example for Using the Shortcut Key to Save a File

More examples of the usage of WSR, is provided in the Appendix section.

Drawbacks

1. You need to train the computer to understand your voice and maintain a backup of the voice profile, so that it can be used on another computer.
2. Voice recognition in Cell phone technology has improved significantly over last few years but same progress is not made in computer based/windows speech recognition.
3. It requires a quiet work environment and a microphone as an added hardware.
4. To develop and maintain the XML macro's one needs to be comfortable with XML, Java Script, VB Script or Perl Script.
5. There is a learning curve involved in it if the user wishes to fully utilize this feature.

Even though Voice recognition technology has been in existence for a long time now, it's potential was not exploited enough till recently. It has a long way to traverse in order to attain popularity among the masses and to be used on a frequent basis. Google now and Apple Siri are examples of this technology already in use.

I think with the development in technology in the recent decade, voice recognition and the ability of computers to translate that into executable commands has an immense potential in the future. In fact, Microsoft had demonstrated a program which has the ability to translate spoken English to spoken Chinese in real time in 2012. Google and IBM Watson group (already famous for the quiz show Jeopardy) are investing heavily on research to promote this technology. A recent article in Forbes, "Future of the web is audible" quotes - "As more web developers adopt speech based interfaces, the way we interact with the internet may quickly involve into something that looks more like the computer on Star Trek: The Next Generation."

REFERENCE :

WSRMacros - The user's guide.pdf

CONTACT INFORMATION:

Your comments and questions are valued and encouraged. Contact the author at:

Name:	Vikas Gaddu, Smitha Madhu
Enterprise:	Anova Groups
Address:	8801 Fast Park Dr #301
City, State ZIP:	Raleigh, NC
Work Phone:	1 800 253-1960
Fax:	1 919 386-0285
E-mail:	Contact@anovagroups.com
Web:	www.anovagroups.com

Appendix

Here are a few more examples of application of WSR in SAS which can save you some time and effort.

1. **Inserting Text**

In this example, I've incorporated the trigger "adverse macro" in the " listenFor" tags to insert a macro into my SAS code. This helps me to call the whole macro along with default value and don't have to remember all the parameters.

```
<speechMacros>
<command>
<listenFor>adverse macro</listenFor>
<insertText>

 %adverse(dlm=# /*Delimiter*/,
          data=MAX_AE_SEV02/*Input Dataset*/,
          spanlabl= /*span columns label*/
          ..............................
          ..............................
          ..............................);

</insertText>
<sendKeys>{ENTER}</sendKeys>
</command>
</speechMacros>
```

2. **Inserting Comments**

Having comments in your code is quiet frequent and a good programming practice. The WSR macro has the ability to insert custom based comments in your code. Within the "insertText" tags you need to use the /* {[…]} */. The forward slash and astrix are more specific to SAS comments.

```
<speechMacros>
<command>
<listenFor>Comments [...]</listenFor>
<insertText>/* {[...]} */</insertText>
</command>
</speechMacros>
```

3. **Standard Header for Programs**

All of us are aware of the standard practices we need to follow as a clinical SAS programmer. The headers we use on a regular basis in SDTM and ADaM programming can be easily loaded into the SAS editor window using the WSRMacro facility.
Here is an example of a header that generates the standard header template and dynamically generates the date. The VB script tag and the CDATA tags are nested within the "listenFor" tags.

```
<speechMacros>
<!-- ***************************** -->
<!-- ***** insert the date ********** -->
<!-- ***************************** -->
<command priority="1">
<listenFor>new header</listenFor>
<script language='VBScript'>
<![CDATA[
Dim strDateTime
Dim strDate
strDateTime = Now()

strDate = Month(strDateTime) & "/" & Day(strDateTime) & "/" & Year(strDateTime)

NamedStates.SetNamedStateValue "strDate", strDate
]]>
</script>
<setTextFeedback>{[strDate]}</setTextFeedback>
<insertText>
```

```
/*****************************************
***  Program Name:
***
***  Purpose:
***
***  Programmer name:
***
***  Date: {[strDate]}
***
***  Modification History:
***
***
*****************************************/


</insertText>
</command>
      </speechMacros>
```

4. Opening Programs /Documents within a Folder

Every day when we fire up our computer we have to open bunch of documents to start our day. When working on a specific study we have to open documents related to that study that itself takes up bunch of time. This task can be delegated to voice macro which will open documents for the specific study when that study number is spoken.

```
<?xml version="1.0" encoding="UTF-16"?>
<speechMacros>
  <command>
  <listenFor>Open ?Study [StudyName] </listenFor>
  <setTextFeedback>Opening Documents for
{[StudyName.Alternate]}</setTextFeedback>

 <!-- set a bookmark -->
 <script language='VBScript'>
 <![CDATA[
 On Error Resume Next
 Dim wSAP

 Set wSAP  = CreateObject("Word.Application")
 Set wSAP2 = CreateObject("Excel.Application")

 studyNameVal = "{[StudyName]}"

If studyNameVal = "101" Then
                          wSAP.Documents.Open("C:\G101\Check List.doc")
             wSAP.Visible = True
             wSAP2.Workbooks.Open("C:\G101\SDTM_VS.xls")
             wSAP2.Application.Visible = True

             set wshshell=createobject("wscript.shell")
             wshshell.run """C:\Program Files (x86)\Adobe\Acrobat
8.0\Acrobat.exe""

""C:\G101\GZGD00304_aCRF_20111018.pdf"""

       set wshshell=nothing
       set wshshell=createobject("wscript.shell")
       wshshell.run """C:\Program Files\SAS\SASFoundation\9.2(32-bit)\sas.exe""
                                                    ""C:\G101\vs.sas"""

       set wshshell=nothing
End If

If studyNameVal = "102" Then
```

```
           wSAP.Documents.Open("C:\G102\Check List.doc")
           wSAP.Visible = True
           wSAP2.Workbooks.Open("C:\G102\SDTM_SU.xls")
           wSAP2.Application.Visible = True
           set wshshell=createobject("wscript.shell")
           wshshell.run """C:\Program Files (x86)\Adobe\Acrobat 8.0\Acrobat.exe
""
                        ""C:\G102 \GZGD00304_aCRF_20111018.pdf"""
           set wshshell=nothing
           set wshshell=createobject("wscript.shell")
           wshshell.run  """C:\Program Files\SAS\SASFoundation\9.2(32-
bit)\sas.exe ""
                         ""C:\G102\su.sas"""
            set wshshell=nothing
       End If
]]>
</script>
</command>
       <listenForList name="StudyName" propname = "Alternate">
              <item propval="Case Study">101</item>
              <item propval="Case Study">102</item>
       </listenForList>
</speechMacros>
```