Multiple language support on IBM ViaVoice

Daisuke Tomoda, Yoshinori Tahara
Software Development Laboratory - Yamato (YSL)
IBM Japan, Ltd.

Abstract

In language support of conventional application, it is usually only considered what operating system language the application runs on. Sometimes, language support itself can be just the message translation. If speech application, such as speech recognition, supports multiple languages, it is important to consider the speech language separately from the message language. In this paper, the authors propose a “language pack” solution to support multiple languages on IBM ViaVoice.

1. Introduction

Generally, a user selects one language for an application, considering the user’s language and operating system language. In speech application, such as speech recognition and text to speech, the user may install multiple language versions because he/she may want to switch the language independently to the operating system language. In order to support multiple languages in speech application, data language (speech language) has to be considered separately from the operating system language. In this paper, the authors propose “language pack” approach, which provides a method to support to switch language setting of the installed speech product, IBM ViaVoice.

In Chapter 2, the authors introduce categories of selectable language in speech application environment. In Chapter 3, the authors describe issues of the current implementation of multiple language installation in IBM ViaVoice product. In Chapter 4, “language pack” solution to solve the issues. Samples of ViaVoice language switching are shown in Chapter 5, and conclusion is provided in Chapter 6.

2. Language categories

The authors introduce three language categories: message language, spoken language, and product language. To categorize the language is a necessary step to consider multiple language support in speech application.

2.1 Message language

Message language is a language used for displaying messages to the user. It is the language for title of dialog, text on buttons, menus, and so on. Message language is usually identical to operating system language. The operating system usually, such as Windows, has a capability of language selection. The user can set operating system language in language options, such as “region setting” in Windows.

2.2 Spoken language

Spoken language is a language used for speech recognition and text to speech. Spoken language can be considered as data language in speech application. In conventional application, it is not important what language can be handled by the application. In speech application, the user is always aware of the current language. Even application behavior may change according to the selected spoken language. It is essentially different from message language.

2.3 Product language

The authors also define product language in order to distinguish message language from product package. ViaVoice package contains speech data and message resource data. Because operating system language may not match message resource language which is included in ViaVoice package, message may not be displayed in selected language. To identify this issue and solve it, product language is defined as resource data language packed in the released ViaVoice product.

3. Design issues of the conventional approach in ViaVoice

Currently, ViaVoice product is released for each language. To enable multiple languages, the user has to install the ViaVoice products as many languages as the user wants. Message language is determined statically, by specifying a parameter of executable modules. If the user wants to change the message and/or spoken language, he/she has to stop ViaVoice, and re-select another program icon which has another language parameter. Though the language parameter of the program means message language, spoken language is also switched, because message language and spoken language are always the same. Therefore, if spoken language does not match operating system language, the product may not work. For conventional applications, it is surely easy for the user to understand why the product does not work. In speech application, however, the user may want to speak another language which does not match operating system.

Because product is released for each language, common modules are overridden if the user installs more than one product. Sometimes common module version is different by language. The product behavior may change unexpectedly by installing more than one language ViaVoice. When uninstalling the product, common modules should not be deleted unless all the language version is being uninstalled. Generally, multiple installation of the different versions of the same product is very complicated. To control install and uninstall the language common modules multiply may cause unexpected problems.

In speech application, function behavior may have to vary based on the selected spoken language. For example, if application displays dictated text in a windows, space should not be put between words in Japanese and Chinese. Spoken language essentially has to be considered separately from message language.

Issues described in this Chapter can be summarized as follows:
1. Usability issue: When the user change language, the user needs to switch whole ViaVoice product. The user can not select message language and spoken language separately.
2. Quality issue: Function may not work on deferent language operating system. Function behavior varies by product language, which should be based on spoken language.
3. Development issue: Implementation of installation and uninstallation is more complicated.

4. ViaVoice language pack

As a solution for issues described in Chapter 3, the authors propose "language pack" installation for multiple language support in ViaVoice product.

The authors define conceptually two types of module groups, common modules and language dependent modules. These types are already well defined in the current ViaVoice products. Program code, messages, non-speech data, and speech data are separated to language dependent or independent. Each language dependent module contains language identifier in the file name. Based on the module definitions, the authors define the following packages:

![Figure 1. ViaVoice package](image)
• Language package = language dependent modules and data
• Base package = language independent modules + US English language package

Figure 1 shows the proposed package concept. It is an example of Japanese language product package which contains base package and Japanese language pack.

4.1 Installed image

As usual applications, ViaVoice program folder and VoiceCenter icon is created on the desktop. Regardless the number of installed languages, there is only one ViaVoice folder is created. The program name and folder name are written in English. Folders and icons for documentation such as help files contain product language information. Only the base package installation creates program folder and icons, and language pack installation only creates language dependent documentation folder and icons.

4.2 Language selection

In the current ViaVoice implementation, the user can select spoken language in ViaVoice option [1]. As described in Chapter 3, if the user changes the spoken language, the message language is also changed. In the authors’ proposal, it is allowed to provide a new option for message language selection. These options are independent, so that the user can select spoken language and message language separately. Selectable spoken languages are determined by installed engine data, and selectable message languages are gathered from the ViaVoice message resource data installed in the system. The default spoken language will be determined in installation. The default message language will be the same as the current operating system language if it matches one of the message resource languages. If operating system language is not found in the list, the default language can be base package language, that is, US English.

4.3 Language service module (vtLocale.dll)

As an implementation of language pack solution, language service module vtLocale.dll is defined as a ViaVoice component. vtLocale has mainly the following functions:
• Determines spoken language by Engine language
• Determines message language by operating system language and installed product languages
• Determines function behavior based on the current spoken language

Each ViaVoice component calls vtLocale in initialization, to set and get the current spoken language and message language. Using the message language, the ViaVoice component loads language resource modules to display language dependent messages. Application also can call vtLocale to determine the user interface behavior dynamically. For example, vtLocale returns a switch value, e.g., true of false, if the dictated text should be separated by a space, or not. Figure 2 shows the role of vtLocale.dll in IBM ViaVoice.

Figure 2. vtLocale.dll language service module

5. Examples of language combination

5.1 Case 1

Consider the following situation:
• Operating system : Japanese
• Installed language pack : Japanese

Because the default language is Japanese for both of message and spoken languages, Japanese message is displayed and the user can speak or hear Japanese voice. The user can change the message language and spoken language independently to English or Japanese. If English is selected for spoken language and Japanese is selected for message language, messages like dialog text is displayed in Japanese and dictated text is English. User interface behavior, such like text correction, is determined by spoken language, e.g. English. Navigation function is disabled because operating system is not English.

5.2 Case 2

Consider the following situation:
• Operating system : German
• Installed language pack : French

In this case, message language does not match operating system language. vtLocale returns the default product language, US English, as message language. French is also selectable as message language. If French is selected, French resource modules are loaded and used to display message. In this case, the message may not be correct if the German font on the operating system does not support French unique characters.

If multibyte language product such as Japanese ViaVoice is installed on single byte operating system like US English Windows, the user may not be able to select multibyte language if the operating system does not support multiple language environment. If multibyte font can be displayed on any language setting of operating system, like Windows 2000, this combination can also be accepted.

6. Conclusion

The authors showed issues of conventional approach of language support in speech application, IBM ViaVoice. In order to solve the issues, the authors provided a language pack solution which enables to select spoken language and message language independently. The authors showed how the issues described in Chapter 3 are resolved by language pack provided in Chapter 4. It can be summarized as follows:
1. Usability issue: The user can change both of spoken and message languages independently via ViaVoice options. The user does not need to be aware what and how many language product is installed on the system.
2. Quality issue: Using vtLocale, Function can be enabled/disabled by the spoken language selection. Using vtLocale language service entries, function behavior can be determined dynamically based on the selected spoken language.
3. Development issue: In installer implementation, language pack installation is much more simpler than the multiple product installation.

Although some conventional applications apply additional language package, it usually only copies language dependent resource. The proposed approach, combination of vtLocale and language package, in IBM ViaVoice can be unique and effective way to provide multiple language support as a speech application.

As described in Chapter 5, font mismatch problem, which is caused by mismatch of message language and operating system language, will be solved by operating systems which support multiple languages. The author will verify that proposed approach can be applied on every language on future operating systems.

Acknowledgments

The authors thank Mr. Mario De Armas for his valuable comments and discussions on architecture of ViaVoice product and on designing vtLocale.

<references>