Comparing macro development for personal productivity tools: an experience in validating accessibility of Talking Books

Gabriella Dodero[1], Katia Lupi[1], and Erika Piffero[1]
1 DISI, Università di Genova, Via Dodecaneso 35, 16146 Genova, Italy
dodero@disi.unige.it, {katia.lupi, erika.piffero}@gmail.com
WWW home page: http://sealab.disi.unige.it/Krakatoa/DisiAbles

Abstract. We describe an experience in developing macros for both Power Point and Impress, to be used in accessibility validation for educational multimedia (Talking Books) designed for visually impaired people. Minor disadvantages in the use of Impress are outlined, which however do not constitute a serious obstacle to adoption of Open Source tools for our purposes.

1 Introduction

There is a number of experiences and studies on how personal productivity tools are being used, and the issues in migrating from one proprietary environment, like MS Office, to an open source one, like OpenOffice.org, have extensively been dealt with (see for example [1]). However the issue of macro development in either environments has not yet received comparable attention, and most of available studies about macros are related to their use in spreadsheets or word processors [2].

This paper describes an experience in validating accessibility of Talking Books, i.e. multimedia training materials, developed with the two most popular personal productivity tools (Impress and PowerPoint) and validated by means of macros. It is the natural follow-up of a previous experience [3, 4], where we described how Cultural Heritage professionals without technical expertise may produce a Talking Book, a computer based teaching aid both for normal and for visually impaired people. The first Talking Book was developed with PowerPoint, following the detailed instructions in the manual [5].

The guidelines to be followed in order to make an accessible Talking Book are partly suggested in such a manual, partly derived from Italian legislation about accessibility [6, 7], as well as from the expertise of therapists employing computer based aids for visually impaired people. Once the content of the Talking Book has been developed, a tedious manual task is started, by enforcing compliance to the accessibility rules, in order to make it truly accessible. Automation of compliance checks to accessibility rules avoids such a task, and it is made possible by a suitable set of macros.

Two implementations of such a validation procedure have been undertaken [8, 9], by developing macros for both PowerPoint and Impress (respectively using Visual

Please use the following format when citing this chapter:

Dodero, G., Lupi, K., Piffero, E., 2006, in IFIP International Federation for Information Processing, Volume 203, Open Source Systems, eds. Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G., (Boston: Springer), pp. 247-252

Basic for Applications and Basic). Such macros have been used both to test the existing Talking Book for accessibility, and for developing new ones.

This paper describes our experiences and compares the two implementations.

2 Development of a Talking Book

Talking Books are usually created by people with minimal computer literacy, having expertise or interest in cultural or entertainment activities of visually impaired people. So, creators of Talking Books may be schoolteachers, parents of disabled children, CH university students or museum personnel, all of them not being professional software developers.

Talking Books creators are interested in making certain contents accessible, and the availability of open source applications saves them licence costs, both for creation and for redistribution of the Talking Book to other visually impaired people (of course costs due to reproduction of copyrighted contents, if any, cannot be avoided). To this aim, a new manual was prepared [10], which details the various operations to be done, illustrating how to use OpenOffice.org Impress to create a Talking Book, on a PC equipped with Windows XP.

Then, we developed macros, that should be applied by the Talking Book creator when he/she decides to validate his product for accessibility, either during the development, slide by slide, or when the Talking Book is completed. Compliance with accessibility guidelines requires the following checks:

- Font size greater or equal to 20;
- Font must be one out of : Arial, Tahoma, Verdana, Times New Roman;
- Italic modifier not allowed;
- Double spacing between words;
- Check of brightness for text and background with the following formula (Red, Green and Blue are the RGB components of text or background colors): ((Red * 299) + (Green * 587) * (Blue * 114)) / 1000 >= 125
- Check of contrast between text and background colors with the following formula (considering Color1 the text color and Color2 the background color):
  [Max (Red1, Red2) – Min (Red1, Red2)] +
  [Max (Green1, Green2) – Min (Green1, Green2)] +
  [Max (Blue1, Blue2) – Min (Blue1, Blue2)] >= 500.

When the check is performed, the macro user (Talking Book creator) is prompted with a list of possible incompatibilities. Then he/she may decide whether to manually correct them, or let the macro automatically perform the suggested modifications.

In this way such a macro may be used as a pure validator, or even, it may be used to automatically transform a non accessible file into an accessible one. In fact, as a useful side result, these macros may be applied to presentations for lectures or conferences (PowerPoint or Impress files without audio components), so that visually impaired people in the audience are not discriminated.

3 Validation macros

Macro development within Microsoft PowerPoint and OpenOffice.org Impress can be done by means of two very similar object oriented programming languages, respectively Visual Basic for Applications and Basic. Our macros must access objects, and possibly change their properties in order to implement the above described checks. The two tools use different objects and properties in order to define a presentation, and the example provided in the Appendix (the function removing the Italic modifier) gives a flavour of such differences, most of which are just syntactical ones. The only significant difference in internal object structure and properties is described hereafter.

A Power Point presentation consists on a set of slides, each one containing various shapes. Inside shapes we may find text frames, that is where macros must operate. Shapes describe an area inside the slide, having properties like HasTextFrame (true if there is text inside).

An Impress presentation is composed by a set of draw pages, each one made by a set of typed elements called shapes. Text is contained only inside shapes having certain types, so if we wish to identify where text can be found, we have to check if the current shape has one of the following types: TitleTextShape, SubTitleShape, TextShape, OutlinerShape.

For both tools, it is possible to customize the toolbar by adding a new button in order to activate the accessibility validation macro on a new presentation.

On the other hand, we found a minor but sometimes annoying difference in macros behaviour. OOo does not apply macros to currently selected text elements, while Power Point makes no difference in treatment between selected and non selected texts.

During macro development, we carefully searched websites devoted to macro developers, like for example www.bettersolutions.com, www.ooomacros.org and others. We realized that the Web provides many more details, useful examples, and explanations on how to manipulate Power Point objects with respect to what is available about Impress objects.

Specifically, we were unable to find the object names and properties of background colors, so the check on contrast between background and text colors has not yet been implemented in the Impress macro. The documentation describing such objects and their properties for OOo appears more difficult to be searched than it is for Power Point, and the effort required to find out the names and properties we need, by actually inspecting the source code, is possible in principles, but appears too big. However the frequent updates to OOo related sites make us confident that information about background color properties will soon be available as well.

This would complete our experience, so that our macros will finally be made available to the public.

4. Conclusions

We have described our experience in developing macros for both Power Point and Impress, to be used in accessibility validation for educational multimedia (Talking Books) designed for visually impaired people. We experienced minor disadvantages in the use of Impress macros, which however do not constitute a serious obstacle to adoption of Open Source tools for our purposes.

Use of macros for improving accessibility inside personal productivity applications is a technique which has proven successful for Microsoft Word (see for example [11, 12]), yet it has not received so far a widespread diffusion as one might expect. Furthermore, the application of macros inside validation tools for Talking Books, as those we have developed, is the only one we are aware of.

It should be remarked that there are two types of stakeholders for accessibility validating tools: creators of Talking Books (or just creators of PowerPoint and Impress presentations), and visually impaired people, who in the end shall be the users of such products (or the audience of such a presentation). The first experiences collected with the creators (a group of Cultural Heritage university students, developing Talking Books to illustrate the contents of various Museum rooms to visually impaired visitors) showed the ease of use of the validation tools, especially appreciating the possibility of automatic corrections. Almost no one in the creators group was aware of the existence of the OOo toolset, while most of them had some familiarity with the MS Office suite. They all worked with Impress without difficulties, following the detailed instructions in [10].

The resulting Talking Books are being experienced with a real audience including both normal and visually impaired people, inside the Museum. Meantime, conference presentations with accessible slides have already been given (at a national Computers and Disabilities conference, HandyTED 2005) with both normal and visually impaired attendees.

5 Acknowledgements

The authors are grateful to Silvia Dini from Istituto Chiossone, who gave precious suggestions about accessibility guidelines for visually impaired people, and to the CH students from the Museology Course of Dr.A.Traverso, who developed the Talking Books.

6 References

[1] COSPA Consortium for Open Source in the Public Administration. Website: www.cospa-project.org

[2] I.C. Laurenson, Introduction to OOo macro development, OOCON 2005, Koper, September 2005. Website: http://marketing.openoffice.org/ooocon2005/.

[3] P. Signorini, Multimedia products for visually impaired people in archaeological museums, Graduation Thesis (in Italian), University of Genova, Laurea in Conservazione dei Beni Culturali, July 2005.

[4] G.Dodero, P.Garibaldi, P.Signorini, and A.Traverso, Visually impaired people and archaeology: a Talking book to know the “Principe delle Arene Candide”, Proc. HandyTED 2005 (in Italian), ITD-CNR, Genova, November 2005. Website: www.itd.cnr.it/handyted2005.

[5] R. Walter, How to create talking books in Power Point 97 and 2000, ACE Centre 2002. Website www.auxilia.it.

[6] Dispositions to ease access of disabled individuals to computer based systems, Italian Law no. 4/2004, appeared on GU n. 13 on 17 Jan 2004. Website: http://www.innovazione.gov.it/ita/news/2003/cartellastampa/doc_leggestanca.shtml.

[7] Requirements for compliance with Law 4/2004, Act of the Italian Ministry of the Innovation and Technologies, appeared on GU n.183 on 8 July 2005.

[8] K.Lupi, Talking Books for Visually Impaired People: user interfacing features, Final Report (in Italian), University of Genova, Laurea in Informatica, Oct. 2005.

[9] E.Piffero, Access to heritage related information for visually impaired users, Final Report (in Italian) University of Genova, Laurea in Informatica, Oct. 2005.

[10] L.De Lucia, How to create a Talking Book with OpenOffice.org 2.0, Final Report (in Italian), University of Genova, Laurea in Informatica, 2006.

[11] A.Cantor, Enhancing the accessibility and usability of Microsoft Office applications using Visual Basic, Technology and Persons with Disabilities Conference, California State University at Northridge, 2004. Website: http://www.csun.edu/cod/conf/2004/proceedings/csun04.htm .

[12] A.Cantor, Macros FAQ, version2.0.(2005). Website: www.cantoraccess.com/macro-docs/macrosfaq.htm

Appendix: Two functions for removing the Italic font modifier

```
Public Function correctItalic()
  For i = 1 To ActivePresentation.Slides.Count
    With ActivePresentation.Slides(i)
     For k = 1 To .Shapes.Count
      If .Shapes(k).HasTextFrame Then
        With .Shapes(k).TextFrame.TextRange.font
          If .Italic = msoTriStateMixed Or .Italic = msoCTrue Or .Italic = msoTrue Then
            .Italic = False
          End If
        End With
      End If
     Next k
    End With
  Next i
End Function
```

```
Function correctItalic (slides)
for i = 0 to slides.getCount()-1
          slide = slides.getByIndex(i)
          if slide.hasElements()then
                    for k = 0 to slide.getCount()-1
                              shape = slide.getByIndex(k)
                              tipo = shape.getShapetype()
                              if tipo = "com.sun.star.presentation.TitleTextShape" or
                              tipo ="com.sun.star.presentation.TextShape" or
                              tipo = "com.sun.star.presentation.SubtitleShape" or
                              tipo = "com.sun.star.presentation.OutlinerShape" then
                                        fPosture = shape.getText().CharPosture
                                        if fPosture = com.sun.star.awt.FontSlant.ITALIC then
                                                  testo = shape.Text
                                                  cursor = shape.createTextCursor
                                                  cursor.CharPosture = com.sun.star.awt.FontSlant.NONE
                                                  testo.CharPosture = com.sun.star.awt.FontSlant.NONE
                                                  testo.InsertString(cursor, "", false)
                                        End If
     End If
   Next k
 End if
Next i
End Function
```

The function in the top box is written for Power Point, the one in the bottom box is written for Impress.