Enhancing user feedback to iteratively improve MIPAV

Sara Shen, Justin Senseney, Matthew J. McAuliffe, Benes Trus
Center for Information Technology, National Institutes of Health, Bethesda MD

Introduction

MIPAV – Medical Image Processing, Analysis, and Visualization

MIPAV is a free image processing program created by the Division of Computational Bioscience at the NIH[3].

Users can contact the program developers through email to request assistance. However, because MIPAV is used predominantly by people who are not programmers, details needed in order to make changes are often lost in translation.

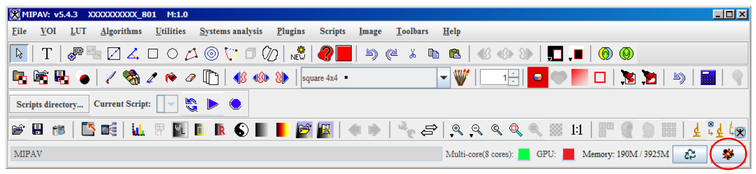


Figure 1. The MIPAV toolbar with the Bug Report button circled in red.

Goals

• To create an integrated bug report system for MIPAV that compiles user inputted information and sends it as an email

• To thoroughly relay information about bugs in MIPAV by replicating a user's experience with the program

Materials

• Java application programming interface for glassPanes[1]

•MIPAV class intended to capture and save images[3]

•MIPAV functionality intended for logging exceptions[3]

Methods

• Identify key elements that comprise the user experience including file types, methodology, and anticipated results

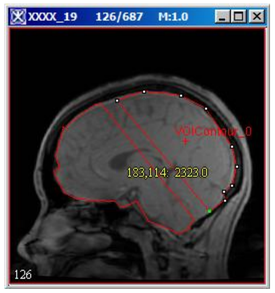
• Find ways to replicate theses elements in a way that can be processed for communication through emails

• To understand limitations, both in the system and code, and the extent to which information can be shared

Privacy

Many users of MIPAV deal with confidential information that should not be shared with developers. Patient identifiable data in particular has to be handled with caution because of how much personal information can be linked with patient images, as shown in Figure 2.

(A)



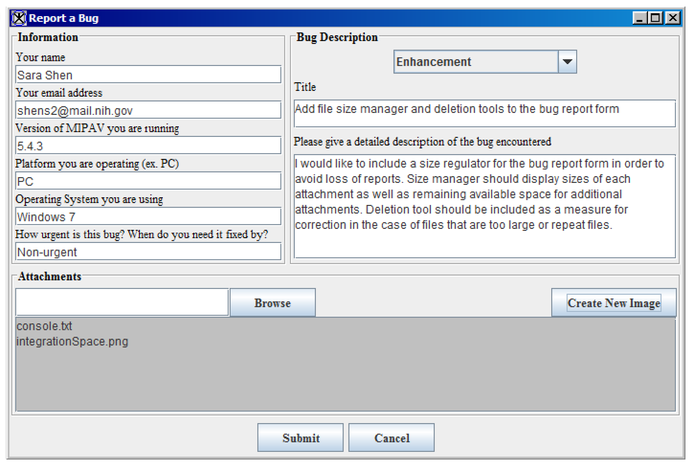
(B)

	Tag	Name	
☐	(0008,1060)	Name of Physician(s) Reading Study	
☐	(0008,1070)	Operator's Name	
☐	(0008,1090)	Manufacturer's Model Name	SIGNA
☐	(0008,1110)	Sequence element	
☐	(0008,1110)	0008,1150: ReferencedSOPClassUID	1.2.840
☐	(0008,1110)	0008,1155: ReferencedSOPInstanceUID	XXXXX
☐	(0008,1111)	Sequence element	
☐	(0008,1111)	0008,1150: ReferencedSOPClassUID	1.2.840
☐	(0008,1111)	0008,1155: ReferencedSOPInstanceUID	XXXXX
☐	(0010,0000)	Patient Group Length	84
☐	(0010,0010)	Patient's Name	XXXXX
☐	(0010,0020)	Patient's ID	
☐	(0010,0030)	Patient's Birth Date	XXXXX
☐	(0010,0040)	Patient's Sex	
☐	(0010,1010)	Patient's Age	XXXX
☐	(0010,1030)	Patient's Weight	XXXXX
☐	(0010,21B0)	Additional Patient History	XX

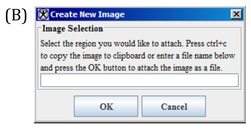
Figure 2. (A) A sample DICOM image with volume of interest overlays. (B) The anonymized patient data that is imbedded in the above image.

The window shown in Figure 3(A) takes standard user inputted information and file attachments. The program formats it into an email[2], as shown in Figure 4.

(A)



(B)



(C)

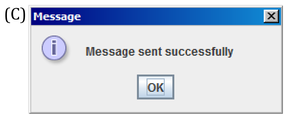


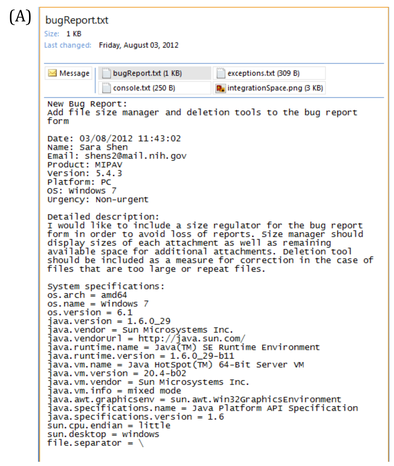
Figure 3. (A) A sample report form that has been filled out. (B) A new window appears when the "Create New Image" button has been pressed, providing instructions and a place for the user to name their image. (C) A "Message sent successfully" popup appears when the email has been sent.

Results

Figure 4. Emails generated through the bug report form contain all manually attached files as well as two standard files, bugReport.txt and exceptions.txt.

When developers receive the email, they can open the bugReport.txt and exceptions.txt files shown in Figure 5, which contain comprehensive details about the bug being reported, including the user inputted information, system specifications, and the program's stack trace.

(A)



(B)

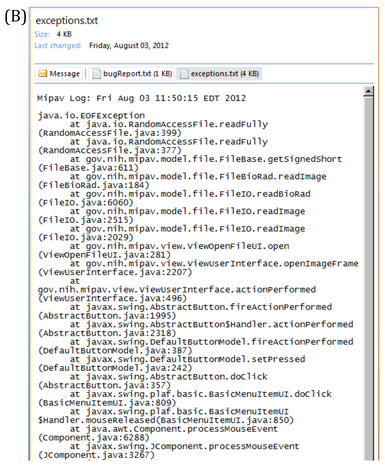


Figure 5. (A) The bugReport.txt file contains all of the user inputted information, as well as the system specifications. (B) The exceptions.txt file contains all of the exceptions normally written to the MIPAV console, including full stack traces and line number in which exceptions occurred.

Conclusion

There are many aspects of the user experience that can be replicated, but challenges still remain. Operating system compatibilities and email server size regulations are areas of Java that this project has identified for future improvement.

Future

• Direct inclusion of images through keyboard shortcuts

• Email size manager with attachment deletion function

• Changing the screen capture function to use translucent windows

• Auto-launching the bug report form in the event of an error or crash

• Logging Java errors as well as exceptions

References

[1]"How to Use Root Panes (The Java™ Tutorials > Creating a GUI With JFC/Swing > Using Swing Components)." *Oracle Documentation*. Oracle, n.d. Web. <http://docs.oracle.com/javase/tutorial/uiswing/components/rootpane.html>.
[2] "Oracle Mail Java API" *Oracle Documentation*. Oracle, 2006. Web. <http://docs.oracle.com/cd/B25553_01/mail.1012/b25459/ad_email_java.htm >.
[3]"Medical Image Processing, Analysis and Visualization." *Medical Image Processing, Analysis and Visualization*. CIT, 14 Mar. 2007. Web. <http://mipav.cit.nih.gov/>.

Email: shens2@mail.nih.gov
sshen12@terpmail.umd.edu