Automated Testing of SAS® System GUI Applications

Brad L. Chisholm, SAS Institute Inc., Cary, NC

ABSTRACT

Graphical User Interface (GUI) applications present a number of challenges to software testers, including test execution time, result verification, and repeatability. This paper will give an overview of some techniques and tools used by SAS Institute's Quality Assurance (QA) department to automate testing of SAS/AF® and other SAS System GUI products, such as SAS/GIS® and SAS/SPECTRAVIEW®. Included will be a discussion of SAS Institute's interactive test capture and replay procedure, techniques for verifying on-screen objects, and screen image captures and comparisons using the Image Data Model Class. Finally, there will be a discussion of industry standard GUI test automation products, such as Mercury Interactive's XRunner and Segue's QA Partner, and their applicability for testing SAS System GUI applications.

INTRODUCTION

For software users, the advent of GUI-based applications has represented a huge leap forward in software friendliness and ease of use. For software testers, however, GUI-based applications generate a whole new class of testing challenges.

First, GUI applications are designed for user driven events, such as mouse clicks and keyboard input. Since those events may be performed on any object at any time, a complex web of interaction is possible. Since a tester must simulate the actions of a user, the time needed to test a GUI application manually is enormous, especially if all possible paths need to be tested. For the Quality Assurance department at SAS Institute, this problem is further magnified by the need to test the application on multiple hardware platforms simultaneously.

Another troublesome facet of interactive testing is the need to verify the underlying task of the application, as well as the interaction of the GUI itself. For example, clicking on a pushbutton may cause a report to be generated (underlying task), as well as cause the pushbutton to change color, and cause a message to be displayed on the screen (GUI changes). Interactively verifying all system responses to each event places a tremendous burden on the software tester, especially as tests are repeated with different data values.

Consistency is also an important issue. As the underlying software undergoes changes during the software development process, it is vital that the process used to test the application remains consistent. Otherwise, it may be impossible to determine if differences are due to software changes, or test irregularities. With manual testing, this is extremely difficult, especially when different people are responsible for performing the tests for later software revisions.

The answer to these problems is finding a way to automate testing of GUI-based applications. The software testing groups at SAS Institute have developed a number of tools and techniques to help achieve this goal, including:

1. PROC FSBATCH, an internal procedure used to capture and replay interactions with SAS full screen applications, and to provide some simple screen verification.
2. The DUMP and DUMPMODE commands, used to list attribute information for SAS/AF objects.
3. Screen captures, using the _SNAPSHOT_ Widget Class method, and image comparison and verification using the Image Extensions to SAS/Graph® software.

THE FSBATCH PROCEDURE

Although many powerful GUI test automation packages are now available, when SAS Institute first faced the challenge of automating the testing of GUI-based SAS System products, choices were limited. One particular obstacle was the need to test on a wide variety of hardware platforms and operating systems, including mainframes and minicomputers. In order to meet these unique needs, developers at SAS Institute created a procedure that allows an interactive user session with the SAS Display Manager System to be recorded, and later played back. This record/playback mechanism allows testers to perform an interactive test once, and have that test played back on almost any later release of the SAS System on any platform. The procedure, PROC FSBATCH, is commonly referred to at SAS Institute as ''the batch driver''.

In record, or capture, mode, PROC FSBATCH sits in the background and generates a file describing the user's interaction with the SAS Display Manager, including mouse clicks and keyboard input. Additional PROC FSBATCH commands can be entered during the capture phase to capture the message line, individual fields, or a textual representation of the entire screen. These commands can be entered on the command line, or assigned to function keys. The table below lists a few of the available commands.

Table 1: Additional PROC FSBATCH Commands

CMD	DESCRIPTION
@DA	dump all fields.
@DF	dump current field(row,col,color,attr,text)
@DL	dump legend window and its fields.
@DM	generate dumpmode type dump on every object in the frame.
@DZ	force an abnormal end by dividing by zero.
@PL	print legend window.
@PM	print only the message line.
@PR	print current screen; dump object info if frame screen.

In playback mode, PROC FSBATCH takes a file that was output during a capture session, and replays the recorded interactions, thus rerunning the test exactly as it was captured. Also during playback, PROC FSBATCH outputs a log file that chronicles the execution of the test. This log file contains essentially the same information as the input capture file, as well as any screen or field dumps that were requested.

PROC FSBATCH has been an essential tool for SAS Institute's GUI testing efforts, but it has a number of limitations as well.

1. Screen locations are based on row and column offsets for cross-system compatibility. As a result, tests must use a lowest common denominator resolution of 24 rows x 80 columns.
2. Although some simple text validation can be performed, selections are typically position based, rather than object based. This means that if an object is moved, the test may not replay properly.
3. GUI validation is limited to simple text-based field and screen dumps. Additional methods, as described in the following sections, are needed for full SAS/AF object validation and graphics validation.

As of this writing, PROC FSBATCH is internal to SAS Institute, and is not currently available to customers of the SAS System.

OBJECT VALIDATION

As mentioned above, it is not simply enough to be able to record a test session and play it back later, you must be able to validate that the test worked properly. In an interactive session, such validation is performed by visual inspection:

- Did the correct list item get selected?
- Was an appropriate message displayed on the message line?
- Did a correct graph get produced?

In an automated session, it is necessary to capture the state of objects of interest, and compare the state information when the test is later played back.

Within SAS/AF applications, the DUMP command can be used to print all the attributes and instance variables of an object to the SAS LOG Window. The DUMP command takes one parameter, the object id of the object to be output. For example, Figure 1 shows a typical listbox, and Listing 1 shows the output from the DUMP command for that listbox.

Figure 1: Sample List Box Object

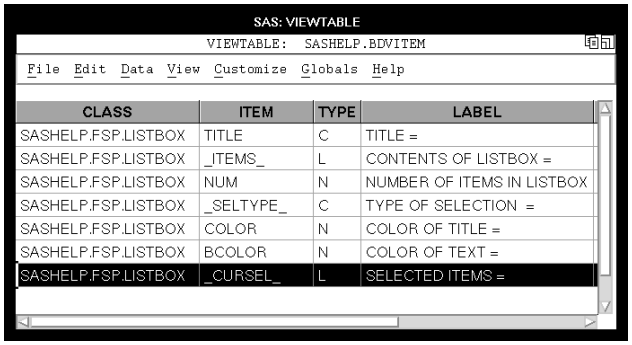
Listing 1: DUMP Command Output for Sample List Box

```
LSTBOX1(
   NAME='LSTBOX1' {P}
   LABEL='LSTBOX1' {P}
   USERATTR=()[31] {L}
   LENGTH=12 {S}
   ROW=10 {S}
   COL=24 {S}
   _UNIQUE_='Y' {P}
   _POPULATE_='DATASET' {P}
   NUM=5 {S}
   NROWS=9 {S}
   LOCATE='SASHELP.PRDSALE PRODUCT' {T}
   TITLE='Products' {T}
   LISTLST=(
      L1=()[35] {L}
      L2=()[37] {L}
      L3=()[39] {L}
      L4=()[41] {L}
      L5=()[43] {L}
   )[33] {L}
   _CLASS_=1013 {I}
   _CLASSNAME_='LISTBOX.LISTBOX' {T}
   _LEN_=0 {I}
   COLOR=28 {K}
   ATTR=0 {K}
   _SELTYPE_='SINGLE' {K}
   _ALLOW_DESEL_='N' {K}
   _SGLCLICK_='Y' {K}
   _DBLCLICK_='Y' {K}
   NSEL=1 {K}
   BCOLOR=28 {K}
   _REGION_=()[1893] {L}
   DESC='List Box' {K}
   CMDPROC=1 {K}
   POPUP=1 {K}
   _CMD_='' {K}
   VBAR=0 {K}
   HBAR=0 {K}
   _FRAME_=45 {I}
   _ITEMS_=(
      'SOFA' {P}
      'BED' {P}
      'TABLE' {P}
      'CHAIR' {P}
      'DESK' {P}
   )[1901] {L}
   _CURSEL_=(
      TEXT=(
         'TABLE' {P}
      )[1897] {L}
      ID=(
         3 {I}
      )[1899] {L}
      ALL=(...)[1901] {L}
   )[1895] {L}
   _VALUE_=(...)[1895] {L}
   _EVENT_=' ' {P}
)[29]
```

While invaluable for validating SAS/AF objects, the DUMP command has a few drawbacks. First, as you can see from the length of Listing 1, it can return more information than you need for your testing purpose. For this listbox example, if you were only interested in validating the item that was selected, the information in the _CURSEL_ list (highlighted) would be sufficient.

It is also inconvenient that you must supply the object id to the dump command. This information is not normally available to a tester, and cannot be queried at run time. In order to use the DUMP command effectively, you need to maintain a reference of the object ids for all objects in your application.

These drawbacks for testing using the DUMP command are addressed by the DUMPMODE command. The DUMPMODE command refers to a data set listing the attributes that should be printed for each object class. There is a master data set with default attributes listed, but the tester can subset the information by creating a custom data set listing only the attributes in which he or she is interested. For example, Figure 2 shows the listbox entries in the master data set. If only the _CURSEL_ information is wanted, a new data set with only the highlighted item can be created.



CLASS	ITEM	TYPE	LABEL
SASHELP.FSP.LISTBOX	TITLE	C	TITLE =
SASHELP.FSP.LISTBOX	_ITEMS_	L	CONTENTS OF LISTBOX =
SASHELP.FSP.LISTBOX	NUM	N	NUMBER OF ITEMS IN LISTBOX
SASHELP.FSP.LISTBOX	_SELTYPE_	C	TYPE OF SELECTION =
SASHELP.FSP.LISTBOX	COLOR	N	COLOR OF TITLE =
SASHELP.FSP.LISTBOX	BCOLOR	N	COLOR OF TEXT =
SASHELP.FSP.LISTBOX	_CURSEL_	L	SELECTED ITEMS =

Figure 2: Listbox Class Entries in Master DUMPMODE Data Set

The DUMPMODE command also allows the tester to select the object to be dumped by clicking on it, rather than passing the underlying object id as a parameter like the DUMP command. This makes DUMPMODE much more convenient when testing a finished application. There are also some extensions in PROC FSBATCH to allow a more seamless interface with the DUMPMODE command, such as the @DM command (see Table 1).

As with PROC FSBATCH, however, the DUMPMODE command is internal to SAS Institute, and is not currently supported for customers of the SAS System.

VALIDATION USING IMAGE CAPTURE

While attribute information may be sufficient for validation of many objects, some (such as graphics) still require visual verification. Automating visual verification requires that bitmaps of screen images be saved and compared. This task is well suited to the Image Extensions to SAS/Graph software.

For SAS/AF-based applications, bitmaps of screen objects can be captured using the _SNAPSHOT_ Widget Class method (documented on pages 55-57 of the Widget Class section of *SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition*). Listing 2 gives an example of using the _SNAPSHOT_ method to capture an image of a displayed object.

Listing 2: Image Capture using _SNAPSHOT_

```
/* Create an instance of the Image Data Model Class*/
/* to hold the captured image.                     */
classid=loadclass('sashelp.fsp.imgdat');
imgid=instance(classid);

/* Call the _SNAPSHOT_ method on the object.       */
call notify('graphobj','_SNAPSHOT_',imgid);

/* Write the captured image out to a catalog entry */
/* for later comparison with a validated image.    */
call send(imgid,'_write_catalog_',
          'work.result.graphobj.image');
```

For this example, the object to be captured was explicitly specified. However, if this code is defined as a method and used to extend the Widget Class, any object can be captured by specifying _SELF_. In practice, this code is typically tied to a pushbutton or function key for ease of use.

Since the _SNAPSHOT_ approach requires SAS/AF Frame applications, other products, such as SAS/GIS and SAS/SPECTRAVIEW, require a slightly different approach. For SAS/GIS, the developers added functionality similar to _SNAPSHOT_ to allow an image of the map display to be captured at any time. A macro variable is used to determine the catalog to which the captured images are written.

SAS/SPECTRAVIEW, on the other hand, already provides the functionality to save any of its views to a TIFF file. Rather than require additional functionality, the QA Graphics Product group wrote a small SCL program to import all of the TIFF files into Image catalog entries after each SAS/SPECTRAVIEW test completed. Once the images are in catalogs, the process is identical for all products.

To validate the images, the test result catalogs are compared to validated baseline catalogs on a picture-by-picture basis using the _DIFF_IMAGE_ Image Data Class method. The _DIFF_IMAGE_ method performs a pixel level comparison of the two images, and outputs a difference image if the images do not match. An example of using _DIFF_IMAGE_ is shown is Listing 3.

Listing 3: Comparing Images Using the _DIFF_IMAGE_ Method

```
/* Create instances of the Image Data Model Class */
/* to hold the test result, baseline, and diff    */
/* images.                                        */
imgclass=loadclass('sashelp.fsp.imgdat');
imgtest=instance(imgclass);
imgbase=instance(imgclass);
imgdiff=instance(imgclass);

/* Read the test result and baseline images.  */
testname='grtest1';
pic='graph1';
call send(imgtest,'_read_catalog_',
          'testlib.'||tstname||'.'||trim(pic));
call send(imgbase,'_read_catalog_',
          'baselib.'||tstname||'.'||trim(pic));

/* Compare the images using _DIFF_IMAGE_.     */
call send (imgdiff,'_diff_image_',imgbase,imgtest);

/* Check return code, and output appropriate message.*/
rc=sysrc();
if rc=0 then
   put 'Images compared identically:' testname pic;
else
   put 'Images had differences:' testname pic;
```

Since image comparison is done on a pixel-by-pixel basis, it is very sensitive to changes in the test environment. In particular, tests must always run at the same resolution, use the same font, use the same color resources, etc. In

practice, the QA Graphics Products group has found that tests run on different platforms using similar display technologies will have images that match,if care is taken to preserve the test environment. For example, all Unix platforms (e.g. HPUX, Solaris, AIX, Digital UNIX) using X Windows will have images that match between them. Similarly, Microsoft Windows hosts (Windows 3.1, Windows 95, Windows NT) will have matching images.

When differences occur, however, it is necessary to visually inspect the images to determine the cause of the difference. The QA Graphics Products group created a Frame application to manage the image results. The application, called ''ImgVer'', uses the Image Class to display the test result, baseline, and difference images side-by-side. The tester can scroll through all the images in a test, and push new baseline ''benchmarks'' from test results if warranted. Figure 3 shows a sample screen shot of the ImgVer application.

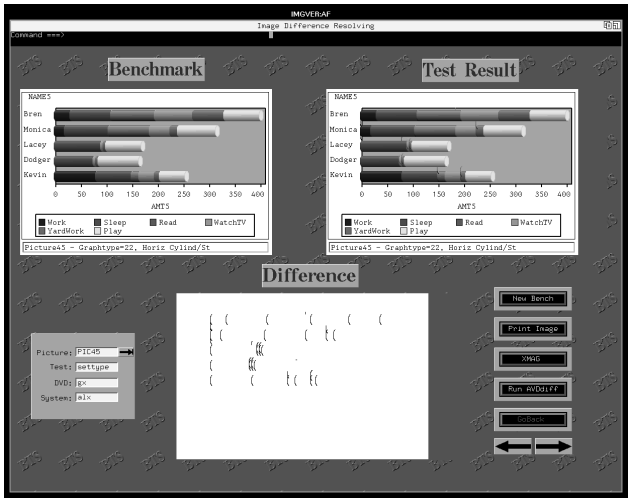


Figure 3: Sample Screen from ImgVer Application

THIRD-PARTY GUI TEST AUTOMATION PRODUCTS

While the Quality Assurance department at SAS Institute primarily uses internal SAS-based tools for GUI testing, there are many powerful GUI test automation tools available from companies that specialize in GUI testing. Some of the more popular packages include Segue's QA Partner, Mercury Interactive's XRunner and WinRunner, SQA's Suite (including SQA Robot), and Rational's Visual Test. Most of these products are based on an object oriented approach to GUI testing, and offer convenient interactive test capturing, as well as powerful test scripting languages.

Since these tools recognize screen objects as objects, the tests can be made fairly robust with respect to incidental differences. For example, you could define a pushbutton named ''QUITBUTN'' that would be recognized if it moved from the top of the screen to the bottom, or even if the caption changed from 'Quit' to 'Exit' or 'Go Back'. Validation of objects is flexible, and is based on attribute information retrieved from the GUI itself. This attribute information is very similar to the information returned by the DUMP and DUMPMODE commands discussed above.

Most of these tools also support bitmap capture, comparison, and difference verification for graphical images. As with the image capture discussion above, however, bitmaps are very sensitive to changes in the testing environment, especially display resolution, video driver, or color depth differences.

Although these testing products offer many benefits, since they are external to the SAS System, there can be problems using them to test SAS GUI-based applications. First, many of the hardware platforms supported by the SAS System are not supported by GUI testing vendors. In general, Microsoft Windows based systems enjoy the widest support, Unix X Windows is next best supported, and only a few vendors support other systems, such as OS/2 or Macintosh. Even if the hardware platform is supported, the products may not work directly with the SAS System. For example, many X Windows testing products require a custom library to be linked with the application to be tested (i.e. the SAS System) in order to use the full power of the object based testing.

Another stumbling block with using these products to test SAS applications is the fact that the SAS System has defined many custom widget and object classes. Since the 3rd party tools rely on object definitions from the underlying GUI tool set (such as Motif, or Microsoft Foundation Classes) to identify screen objects, the custom SAS widgets are often not recognized. Most testing tools do allow limited definition of custom widgets, but it is generally difficult and less effective than the native widget support. In these cases, it may be useful to use a hybrid approach by employing the DUMP command to output the object attributes to the SAS log, then using the GUI testing tool to validate the SAS log file.

CONCLUSION

This paper gave a glimpse at some of the tools and techniques used by the testing community at SAS Institute. Some of those tools, such as PROC FSBATCH and the DUMPMODE command, are not available outside the Institute. Others, such as the DUMP command and the _DIFF_IMAGE_method, are available, but not documented nor officially supported for customers of the SAS System. If you feel some of these tools would be useful in testing your own SAS GUI applications, contact your SAS Institute Technical Support representative about nominating an entry for the SASWare ballot.

Third-party GUI test automation products provide another solution to automating SAS testing. These products offer a rich set of tools for GUI testing, but getting effective results with the SAS System can be difficult due to integration and custom class considerations.

Regardless of the particular techniques employed, however, an automated testing strategy is clearly crucial for effectively testing GUI-based applications.

REFERENCES

SAS Institute Inc. (1995), *SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The author wishes to thank Shearin Bizzell and Brendan R. Bailey for their technical review.

SAS, SAS/AF, SAS/GRAPH, SAS/GIS, and SAS/SPECTRAVIEW are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

OS/2 is a registered trademark or trademark of International Business Machines Corporation.

Other brand and product names are registered trademarks or trademarks of their respective companies.

AUTHOR CONTACT INFORMATION

Brad L. Chisholm
SAS Campus Drive
Cary, NC 27513
Phone: (919) 677-8000 x7061
Email: sasblc@unx.sas.com