PharmaSUG China 2022 - Paper CC-113

Use GUI with SAS Programs: Overview and New Solutions

Wei Chi, and Hongfang Zhou, Xuanzhu Biopharm Inc.

ABSTRACT

In some scenarios, we want to interactive with our SAS programs through a graphical user interface (GUI). This can avoid to write unwieldy used-once codes and make the job done elegantly. This article explains how to create GUI using Java and Python and how to make SAS programs interact with these interfaces using SAS built-in Java environment and its support for Python. It also briefly summarizes various other solutions and compare about their advantages and limits.

INTRODUCTION

Imagine you need connect to a database to extract data using SAS, it is dangerous to write your username and password in SAS program. The more flexible and safer method is to let users enter their credential information interactively every time they run this program.

Another scenario is you have to run a program frequently but with different parameter values. The process of creating define.xml in clinical submission is an example this kind of case. Usually, we need update CRF page number, add NCI number for code list, reduce variable length to minimum, then we generate define.xml and validate it for standard compliance. For any detected issues, we need fix them and repeat the previous process. This is a create-validate-modify loop until the final result is perfect to submit. In this process we need repeatedly run our program with slightly different configuration since not all steps are necessary in each iteration.

In the scenarios above, a graphical user interface (GUI) is an elegant solution. Users can provide their inputs via GUI, these inputs are passed to SAS, then a SAS program decides how to react per user inputs.

GUI PROGRAMMING ELEMENTS

Though different programming languages and libraries have different terminologies, GUI programming basically includes three elements:

- Control/Widget
- Layout
- Event Handling

Control defines what are displayed in the interface. A control can accept input, display data, or provide visual guides. Common controls are buttons, labels, checkboxes and radio buttons etc. Another kind of control is container which used to contain other controls.

Layout defines how controls are arranged and placed in a container. Some languages and libraries have visual designer, developer can position the controls using mouse and generate the codes automatically. But for others, manual coding is inevitable.

Event Handling defines how to react to user inputs. Developers need write codes to correspond specific events, for example clicking a button or hovering over an item in a list box.

SAS programs are traditionally scripts. Next, let us check what solutions we have to use GUI with SAS.

WINDOW AND DISPLAY STATEMENTS

The easiest way to create a GUI in SAS/Base is using the Window and Display statements in data steps or their counterparts %Window and %Display statements in macros. We can use Window or %Window

statement to define a window and Display or %Display statement to show it. User inputs can be saved in data-step variables or macro variables.

The windows created in this way can only display text label and accept text input, cannot meet the needs of complex interactivity. Though PROC PMENU provides radiobox and checkbox in a menu dialog, usually these functions need be used along with other SAS modules.

SAS/AF

Another solution is SAS/AF which is a module used to help create customized GUI application. In addition to the built-in components, it also provides a language named SAS Component Language (SCL) to extend them. But the problem is that using this module need extra license. Moreover, this tool is largely outdated.

POWERSHELL CORE

Pradhan (2021) gave a solution to create GUI using PowerShell. PowerShell is a script language initially for Windows. With the introduction of PowerShell Core, it is currently open-source and cross-platform. PowerShell Core is built on .NET Core so it can use a lot of .NET libraries including Windows Forms, which is also open-sourced since 2018. This is why we can create GUI using PowerShell. The only drawback of this solution is SAS has no built-in support for PowerShell.

JAVA SOLUTION

SAS has a built-in Java environment. This makes easy to create GUI using Java then run it in SAS. The built-in Java version can be checked by running PROC JAVAINFO. In SAS 9.4, the version is OpenJDK 8.

Java has three libraries for GUI programming. The oldest one is AWT (Abstract Window Toolkit). Swing is built on AWT. The latest one is JavaFX. Since OpenJDK does not support JavaFX very well, we choose Swing to verify our thoughts.

Firstly, we need create a GUI class in Java. The Figure 1 is a sample GUI created by Swing.

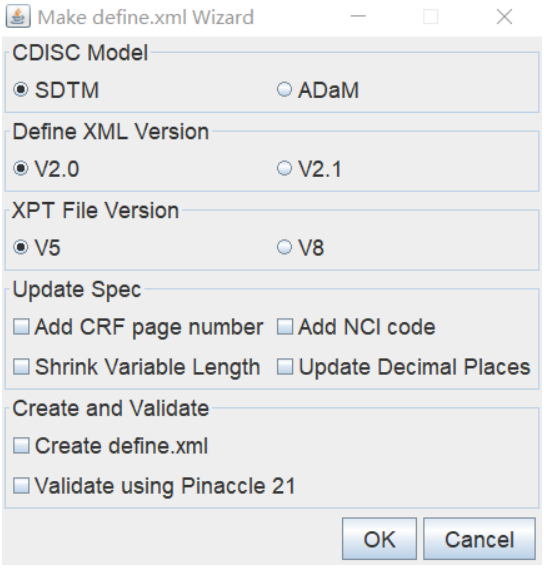

Figure 1. A sample GUI created by Java Swing

Once we created GUI using Java, we can export the compiled classes as a JAR package. Then we need set the CLASSPATH environment variable to let SAS know where it can find these classes. There are 3 kinds of methods to set CLASSPATH. You can set CLASSPATH in your OS directly.

Or in SAS configuration file or SAS command line,

```
-set classpath path\to\sasgui.jar
```

Or in SAS program,

```
options set=classpath path\to\sasgui.jar;
```

Next in SAS, we need create an instance of this GUI class, collect any user inputs from GUI. When user click the OK or Cancel the button, this GUI is set to invisible. By calling the methods of this Java object, user inputs are transferred to SAS as data-step variables. Finally, this object is destroyed in memory. SAS will make the corresponding processing per user inputs.

This process can be implemented using the codes below:

```
data inputs;
  length model defineVer xptVer $20;
  dcl javaobj j("com/xuanzhubio/sasgui/MakeDefineGui");
  j.callvoidmethod("init");
  do while (1);
    j.callBooleanMethod("isDone", done);
    j.callBooleanMethod("isCancel", cancel);
    if done then do;
      j.callStringMethod("getModel", model);
      j.callStringMethod("getDefineVersion", defineVer);
      j.callStringMethod("getXptVersion", xptVer);
      j.callBooleanMethod("isNeedAddPage", addPageNumber);
      j.callBooleanMethod("isNeedAddNciCode", addNciCode);
      j.callBooleanMethod("isNeedShrinkVarLen", shrinkVarLen);
      j.callBooleanMethod("isNeedUpdateDecimal", updateDecimal);
      j.callBooleanMethod("isNeedCreateDefine", createDefine);
      j.callBooleanMethod("isNeedValidateDefine", validateDefine);
      j.callvoidmethod("close");
      output;
      leave;
    end;
    if cancel then do;
      j.callvoidmethod("close");
      leave;
    end;
  end;
  drop done cancel;
run;
```

One thing to note is that usually we organize Java programs in packages. If so, when create a Java object in data step, we need specify the package path. We must use forward slashes (/) and not periods (.) in the path.

PYTHON SOLUTION

Since SAS 9.4 M6, PROC FCMP supports to run Python functions. This opens the door to let SAS program interactive with GUI created by Python. In order to let SAS can run Python function, Python need be installed and two environment variables MAS_M2PATH and MAS_PYPATH need be set. For more details, please refer to SAS document.

Python has multiple GUI libraries. Here we use Tkinter to verify our thoughts. Though Tkinter is a bit old-fashioned, it's a Python standard library. No more IT overhead are needed once Python is installed.

In the Python program, we define a GUI class and create an instance of this class, then create a dictionary to save user inputs and two global functions to be called by SAS. Python functions called by PROC FCMP return values in a Python tuple. The items in this Python tuple are then stored as separate elements in a special dictionary named RESULTS. The keys of this dictionary must be specified as a string in the first line of the Python function body, the values are the tuple items corresponding to keys. The Figure 2 is a sample GUI created by Tkinter.

```
class MakeDefineForm(ttk.Frame):
  ...

class Application(tk.Tk):
  ...

app = Application()
data = {}

def launch():
    "Output: launch"
    app.mainloop()

def getUserInputs():
    """Output: Model, DefineVersion, XptVersion, AddCrfPageNumber,
AddNciCode, ShrinkVarLen, UpdateDecimals, CreateDefine, ValidateDefine"""
    global data
    return data["Model"], \
           data["DefineVersion"], \
           data["XptVersion"], \
           data["AddCrfPageNumber"], \
           data["AddNciCode"], \
           data["ShrinkVarLen"], \
           data["UpdateDecimals"], \
           data["CreateDefine"], \
           data["ValidateDefine"]
```

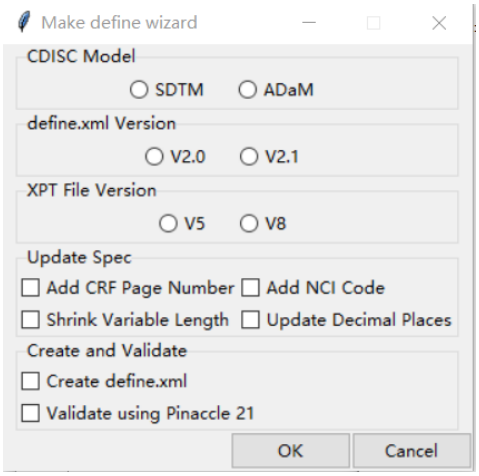

Figure 2. A sample GUI created by Tkinter

In SAS, we create a python interpreter object, submit our python script to the interpreter object. Then launch the GUI, interact with it and save all inputs into a dictionary. Finally, we save all inputs as macro variables since PROC FCMP cannot output a SAS dataset.

```
proc fcmp;
    declare object py(python);
```

```
    rc = py.infile("tkinterGUI.py");
    rc = py.publish();
    rc = py.call("launch");
    rc = py.call("getUserInputs");
    length model DefineVersion XptVersion $20;
    model = py.results["Model"];
    DefineVersion = py.results["DefineVersion"];
    XptVersion = py.results["XptVersion"];
    call symputx("model",model);
    call symputx("DefineVersion",DefineVersion);
    call symputx("XptVersion",XptVersion);
    call symputx("AddCrfPageNumber",py.results["AddCrfPageNumber"]);
    call symputx("AddNciCode",py.results["AddNciCode"]);
    call symputx("ShrinkVarLen",py.results["ShrinkVarLen"]);
    call symputx("UpdateDecimals",py.results["UpdateDecimals"]);
    call symputx("CreateDefine",py.results["CreateDefine"]);
    call symputx("ValidateDefine",py.results["ValidateDefine"]);
    call py.clear();
    run;
quit;
```

LIMITS

All above methods have a serious limit that they cannot be used in SAS EG. Since SAS EG is a C/S architecture application, all these GUI are created in the server side. In order to remedy this situation, Pradhan (2021) proposed to save all user inputs from GUI created by PowerShell into an intermediate text file, then use SAS to read this file. This is also suitable for our Java or Python solution. If so, we need not create Java or Python objects in SAS any more. We just run the binary GUI, and use SAS programs to read the text file via SAS EG.

COMPARISON

Finally, let us compare all these GUI solutions in 5 dimensions in Table 1. The Comparison of all SolutionsTable 1.

Dimension	Window/Display Statement	SAS/AF	PowerShell	Java	Python
Extra License	N	Y	N	N	N
Cross Platform	Y	Y	Y	Y	Y
IT Overhead	N	N	Y	N	Y
Built-In Support	Y	Y	N	Y	Y
Highly Customizable	N	Y	Y	Y	Y

Table 1. The Comparison of all Solutions

CONCLUSION

By comparing all above methods, I think the most productive method is using Java Swing. Since the existence of built-in Java environment, user needn't config anything but CLASSPATH environment variable. Moreover, this solution need not any extra license or latest SAS release, is cross-platform and highly customizable.

REFERENCES

SAS® 9.4 Component Objects: Reference, Third Edition. https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/lecompobjref/titlepage.htm.

Sumit Pratap Pradhan. 2021. “Alternate solution to %WINDOW statement in SAS® Enterprise Guide®: PowerShell” *SAS Global Forum 2021,* 1088-2021.

CONTACT INFORMATION

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

Name: Wei Chi
Enterprise: Xuanzhu Biopharm
Phone: 86-13816832597
E-mail: chiwei@xuanzhubio.com

Name: Hongfang Zhou
Enterprise: Xuanzhu Biopharm
Phone: 86-18221033130
E-mail: zhouhongfang@xuanzhubio.com