ABSTRACT

Application developers in SAS environments regularly face the question about what tool to use to build the client-based user interface of their client-server applications. These are environments where: (1) the data is in server-based SAS data sets, (2) the primary processing is done with server-based SAS Software applications that cover file management, analysis, and reporting, and (3) the user interface, or front-end, is client-based and is used primarily to direct and control the processing and to present results. Although the server side is all SAS-based, the client side user interface can be, but does not have to be, written with SAS Software. The choice of a front-end tool in this environment has never been simple because of trade-offs involved. But this tool choice has over time become even less clear because (1) non-SAS front-end tools have eclipsed the primary SAS tool (SAS/AF) in popularity, (2) SAS has stopped growing the SAS/AF tool, and (3) SAS has developed a new tool, AppDev Studio. AppDev Studio is primarily used to build the client and server sides of Web-enabled SAS applications, but it can actually build a front-end to any server-based SAS application and data. This paper evaluates SAS/AF, SAS AppDev Studio, and the non-SAS tool class, (with Visual Basic as a prime example), against a series of relevant performance factors. Information is provided to structure and contribute to the decision about which front-end development tool to use.

INTRODUCTION

Background

Several years ago, our client wanted to upgrade their SAS-based environment for clinical trials data collection, management, and analysis. Most important was the addition of a double-key data entry/verify system. Double-key data entry/verify is an approach to keyboard-based data entry that is designed to eliminate virtually all human error from this process. Two different people both key the data from the source document. Each field of data resulting from the two key entry operations is then compared. If the two versions are identical, the entry is considered accurate. If the two versions are discrepant, a human operator re-examines the source and determines which of the two versions is correct. The verification process can be combined online with the second keying of the data.

More recently, the FDA issued new regulations for handling electronic records involved in clinical trials research. These include a requirement to audit the creation and modification of such records. As a result, my firm recently completed a project to add an audit trail feature to the existing entry/verify system.

One aspect of the project in particular seemed interesting, and of value if shared with other SAS-based application developers. The purpose of my paper is to discuss this issue. I introduce it here.

Focus of Paper

Our client wanted to add the original data entry/verify system to a SAS-based clinical data environment used for data management, analysis, and reporting. Their data were stored in SAS data sets, and SAS Software was used for analysis and reporting. It was clear that we would use SAS for the online data entry/verification because the data was in SAS data sets. But the choice of tool to build the system front-end was less clear. By “front-end”, I mean the graphical user interface (GUI) that the user would use to negotiate and control the whole environment, and accomplish tasks.

This same GUI tool question arose for us in subsequent application development projects in SAS environments, and it arose again as we designed the audit trail addition to this data entry system. I believe this question is actually quite common among application developers in SAS environments. So I thought it would be useful to discuss the GUI tool choice in general. By the phrase “SAS environment”, I mean an environment where the data is in server-side SAS data sets, and it is processed using server-side SAS Software applications for data management, analysis, and reporting. In this environment, the client-side GUI (or front-end) to the system does not necessarily have to be written with SAS Software.

Further, I believe that the front-end tool choice in this environment has become increasingly unclear today, as:

- Non-SAS GUI tools have eclipsed the primary SAS tool (SAS/AF) in popularity.
- SAS has stopped growing the SAS/AF tool.
- SAS has developed a new product, AppDev Studio, which is a viable replacement for SAS/AF for building the client-side front-ends to your server-based SAS data and applications, Web-enabled or not.

GUI TOOL CHOICE
Choice Categories

The front-end tool choices for application development in SAS-based environments can usefully be placed in three categories, as follows:

1. SAS/AF – This is the traditional, historical, flexible, and powerful approach to the client side of application development using SAS Software.

2. AppDev Studio – This relatively new SAS product is an integrated suite of development tools to build Web-based applications as well as traditional SAS/AF applications. My discussion today will focus on the web-based aspect via discussion of webAF. webAF is a component of AppDev Studio that can be used to build a client-side GUI for these applications. webAF can build basic Web pages for display in a client-side Web browser. It can also develop Java applets (GUI programs hosted in a web browser), Java applications (GUI programs invoked as a standard client program) and web applications (server-side Java programs that only stream markup such as HTML or XML back to the client).

3. A non-SAS tool, where Visual Basic is probably a prime example. There is another example of this category worth mention, a “3a” if you will. It involves using Java to build the front-end application (it will run in a Window, not a browser). The importance difference between 3 and 3a is that the Java application can use the server-based SCL class library from SAS to fully exploit SAS data and applications on the server. A Visual Basic application will be more limited in this regard. Nonetheless, Visual Basic is probably more representative of the entire non-SAS category, and is used as the example for the discussion below.

Application developers in SAS-environments will probably find it valuable to evaluate their front-end tool choices on the following performance factors. For each factor, I offer a few points for consideration, and comments about how the three tool choice categories perform on the factor.

Basic GUI Functionality and Windows Compliance

Does the tool provide a full set of visual controls (objects) for building the interface? Do the controls operate in a Windows-compliant way, such that users will be familiar and comfortable with their general operation?

SAS/AF: These two issues were probably more important historically for SAS/AF than they are today. This is because early releases of SAS/AF were not the equal of Visual Basic and other non-SAS GUI tools functionally or Windows-compliance wise. In more recent years, AF has improved substantially in these regards. You are likely today to have all the functionality you need from AF regarding visual controls. And, today, AF is largely Windows-compliant in how its visual controls operate. If your project requires the strictest compliance, you will need to look carefully at this issue. However, for most projects, where absolute compliance is not a requirement, you will find the level of Windows compliance in AF to be more than satisfactory.

AppDev Studio: It is my understanding that there are no particular Windows compliance issues in front-ends built with AppDev Studio. The webAF software within the product builds/displays standard Web pages in a standard Web browser, as well as Java applets, Java applications, and web applications that stream standard HTML and XML back to the client. And, with the available Java Swing GUI architecture, you can specifically select a Windows look and feel for the Java applets and applications. Once again, you will only need to look closely at this issue if your project has special requirements in this regard.

Visual Basic: I cannot make any general statement about the Windows compliance of the full set of non-SAS GUI tools available today. I can say, though, that Visual Basic is a Microsoft product, and as such, will build a Windows-compliant user interface.

Client-Server Communication

In many (or most) SAS-environment applications, the PC-based GUI application is used to launch SAS programs and access SAS data on a SAS server in the background, and the results of these background processes are then delivered back to the client PC. There is a network connection between the platforms. Two aspects of this client-server communication are relevant here. First is the ease of providing the networking software to achieve the communication. Second is the directness and completeness with which the client and server applications talk with each other over this network. Sometimes the outright ability to provide a particular functionality can be a function of how well the client GUI software communicates with the SAS software and data on the background SAS server. You should have a full understanding of how the front-end and back-end of the application will communicate before you choose the GUI tool for your application.

When your GUI tool is a SAS tool, (either AF or AppDev Studio), SAS will also provide the software components you will need to make the network connection and communicate between the two applications. With a non-SAS GUI tool, you will have to determine what particular SAS product to license to adequately achieve the connection.

Further, when your GUI tool is a SAS tool, you can achieve full “object-to-object” communication between the client and server applications precisely because both applications are written with SAS tools. More specifically, the client application can directly use the server-based SCL class library from SAS to fully exploit the SAS data and applications on the server. And this type of communication will provide you the most control, the fullest functionality, and the most easily achieved functionality in your application. This can turn
out to be particularly valuable to you, depending upon the specifics of what you are doing in your application.

SAS/AF: Here, you will probably use SAS/CONNECT to communicate between your PC-based SAS session and your background server-based SAS session. And, because both applications are written with SAS tools, you can achieve the aforementioned full “object-to-object” communication between the two applications.

AppDev Studio: In this case, you will use either SAS/IT (Integration Technology) or SAS/IntrNet to achieve the network communication. And, with AppDev Studio, you will benefit from the full object-to-object communication available between your client and server applications. This issue actually highlights one of the valuable aspects of using AppDev Studio to build a Web-enabled (or not Web-enabled) application. Specifically, the bundle of tools in AppDev Studio are designed to fully exploit, and easily exploit, the SAS data and SAS applications on your server, and to provide an easy way to communicate the data and results between the client side and server side.

Visual Basic: If you run a Visual Basic client, I believe that a licensed copy of SAS/IT or SAS/IntrNet will be required in order to access your server-side SAS applications and data. However, the primary issue here is that no matter how the network connectivity is achieved when writing a Visual Basic application, your client application will not be able to talk directly to the pre-packaged SAS server-side components that know how to exploit the server-side SAS software and data. (This is the object-to-object communication I refer to above).

Skills of In-house Staff

Does the organization have programmers with the skills required to develop the GUI using a given GUI tool, or to support it after it is built?

SAS/AF: In a SAS environment, the programmers will be skilled in Base SAS. SAS/AF and Screen Component Language (SCL) skills are a different skill set, but some Base SAS programmers do also have this skill set. And, it would be a reasonable endeavor for a Base SAS programmer to learn the AF-SCL skills if needed.

AppDev Studio: If you use the web/AF product of AppDev Studio to generate your front-end application, you will likely need to understand object-oriented programming, Java, and networking issues. webAF does have a drag and drop facility for building the interface, and it does generate Java code for you. However, you will still need to know the above mentioned subjects to complete a robust user interface. But note that with the tools provided in the webAF software, you can easily call (server-side) SCL code or submit data step code from a Java client. So if your staff does already have SAS/AF skills, AppDev Studio allows these people to gain the web-enablement benefits of Java while still preserving and extending their SAS/AF investment and skills. Finally, I note that for a simpler interface, the webEIS tool of AppDev Studio can also be used. In this case, you will be able to achieve the goal without knowing object-oriented programming and Java.

Visual Basic: These skills are more common in the IS world at large, but not necessarily more common in a SAS-based environment. However, if your SAS environment is within a large organization with a separate IS department, Visual Basic skills will likely exist. Note that Visual Basic programmers will still need to understand enough about SAS to write Visual Basic code to communicate with the SAS system, perform the desired work, and display the results back to the user. Also, knowledge of networking will be essential here.

Thin/Full Client – Software Licenses

Presumably, the application is likely to have several or many clients. What software must be licensed for each client with the choice of GUI tool? What networking software must be licensed for the same tool?

SAS/AF: When you use SAS/AF to build your GUI, you must license Base SAS Software on each client PC in order to run the AF application that is your GUI on the client. You will only need a copy of the SAS/AF software for each developer, but you will need a copy of Base SAS Software for each end-user to run that AF application. There is no such thing as a “run-time only” license for SAS/AF applications. This means you will run a very “Full Client”, and there is more about this to consider in the next two sections below. Also, you will probably need to license SAS/CONNECT software for the networking.

AppDev Studio: You will need to license one copy of AppDev Studio for an entire site to run a client application built with AppDev Studio. This is so regardless of the number of clients that will run the application. You will also need to license a copy of AppDev Studio for each developer. Your application will also most likely involve serving Web pages to a Web browser, so you will also need a Web browser on each client. But this is likely to already be there. And, since your application will serve Web pages and possibly Java applets from a server, as opposed to storing this code on the client, you will run a very “Thin Client”. Also, regarding networking issues, you will need to license server-based SAS/IT or server-based SAS/IntrNet.

Visual Basic: You do not need to license a copy of Visual Basic for each client to run the application. You only need a copy of Visual Basic for each developer. Your client application will be thinner than an AF application, but it will still involve installing an application on the client. Thus, it will be fuller than an AppDev Studio client application. And, I believe that a licensed copy of SAS/IT or SAS/IntrNet will be required for the client-server connection.
A client-side GUI is a software application that must execute on the client PC. The application must be distributed to each PC involved. And, if the software is updated, then updated versions must also be distributed to each PC. The logistics of this should be considered when choosing the GUI development tool. The logistics are different for the three categories of GUI we are discussing here.

SAS/AF: A SAS/AF GUI means you will run a very "Full Client". Base SAS Software and your AF application must be distributed to each client. There are a variety of logistics associated with installing Base SAS software on each client PC, and with installing and running the AF application within the SAS session. The same is true for any enhancements to the GUI application or to the BASE SAS Software, including the annual license update.

AppDev Studio: If you use webAF to build your GUI, the interface will typically use Web pages served to a Web browser. The only client-resident software distribution and maintenance involves the Web browser, which is probably there anyway. If the GUI also includes Java applets running within the Web page, there is only one copy of each applet. It is stored and maintained on the server, and downloaded to the client on-demand. The bottom line is that the logistics of distribution and maintenance of client-side software are minimized in this configuration.

Visual Basic: With a non-SAS GUI tool, such as Visual Basic, you will need to install an application on each client PC. Basically, this application will be an executable file (".exe" file). But regardless of the size and robustness of this application, you will encounter the logistics of distributing and maintaining a copy of the GUI application on each client.

Thin/Full Client - Optimized, Balanced Processing

In most client-server applications, the processing is distributed in some fashion between the server(s) and the clients. The balance of processing between the platforms can be a very important factor in the success of the application. This is too large a subject to discuss thoroughly in a paper of this size. However, I can focus on one aspect of this issue that I feel is useful, and that does vary across the three GUI tools we are discussing. All three GUI tool choices being considered have the ability to do measurable processing on the client-side. But only one choice has the power of SAS Software on the client.

SAS/AF: You will have Base SAS Software installed on each client, and you will have maximal ability to handle major processing on the client, if you want to do so. Processing will be maximal on the client in two ways: (1) SAS Software provides a wide variety of built-in tools for data entry, analysis, and reporting. You are likely to find pre-programmed items for most types of client-based processing you would like to do. (2) SAS Software provides the easiest access to client-based SAS data set, should your application need this.

AppDev Studio: In this case, client-side processing will be handled by Java applets that are served to the client-side Web browser as part of the served Web page. Java applets are executable programs and they can do measurable processing. And, through JDBC (similar to ODBC), Java applets can access SAS data sets on the client. But in order to deliver the same pre-programmed, built-in processing power as a SAS/AF client, a Java client will need to go back to the server side for SAS services.

Visual Basic: The story here is somewhat similar to that for a GUI built with AppDev Studio. Visual Basic is a powerful language that can be used to perform substantial processing on the client. But it will not deliver the same pre-programmed, built-in processing power as will SAS Software on the client. You will be able to access SAS data sets on the client through an ODBC connection. But, you will probably not find it as easy to process the client-side SAS data sets as you would with SAS Software on the client.

Longevity of Tool

It is obviously prudent to consider whether your choice of GUI tool will continue to be supported and enhanced by the tool vendor. Normally, for major tools, including those from SAS, this would be assumed to be so, and I would not include this topic in this tool choice discussion. However, as many in the SAS community already know, SAS has decided that the SAS/AF product is not a part of its strategic future growth direction. They anticipate that Web-enabled applications will continue to grow in popularity, that thin clients will be preferred, and that the Java language will thus be the language of choice for client-side applications.

SAS has demonstrated this position with its new AppDev Studio product. The webAF tool within AppDev Studio is used to develop thin clients for Web-enabled (and non Web-enabled) applications. And webAF generates Java code that is then available to the developer to enhance the client-side application.

SAS-based application developers have thus been wondering if it makes sense to develop new applications using SAS/AF. The obvious concern is that the product might not be supported by SAS in the future. I do not represent the SAS Institute, and obviously cannot provide any official SAS position on this. However, I offer here my understanding of the SAS Institute’s position, from the Institute’s response to this question at several SAS User Group conferences:

- SAS/AF is a mature and powerful product and it will remain a part of the SAS Software, as is. It will
continue to be supported by SAS, but it will not be
grown further.

- SAS is now putting its new energies into new tools,
  with AppDev Studio as the prime example.
- One reason of assurance that SAS/AF is not
  “going away” is that parts of the SAS Software are
  written using SAS/AF. One prime example of this is
  SAS/ASSIST.

Overall, this is an important issue, and it is not easy to
express all that an application developer would want to
know about this in a few bullet points. There are
undoubtedly certain applications for which SAS/AF would
be the best client solution. Thus, if you are in such a
situation, it would be best to contact the SAS Institute
and discuss this concern with them directly.

SAS/AF for the Data Entry System

Having now discussed the user interface tool choice in
general, it seems fitting to mention the choice we made
for the front-end to our double-key data entry application.
We chose SAS/AF for the front-end of our original data
entry/verify system. This choice was driven by the client’s
existing processing configuration, which was a given for
the project and was not a full client-server model.
Specifically, we had multiple PC’s connected on a local
area network, and they received file-serving services
(only) from a network server. SAS Software did not
execute on the server. All SAS processing occurred on
the PC’s. The fact that we already had SAS Software
installed on each client, and knew it would remain there
for other purposes beyond our application, led to a
decision to build the user interface to our system with
SAS/AF.

It is interesting to note that even with this processing
model, which indicated strongly that we should build our
GUI with SAS/AF, we nonetheless felt pressure to
evaluate GUI tool options before our final decision. I think
this was largely because the non-SAS tools had already
(in the mid-1990’s) substantially eclipsed SAS/AF in
popularity. The client thus wanted some consideration of
this. And we faced this issue once again, as we
considered how to add the audit trail functionality to the
existing data entry system with its SAS/AF-based GUI.
We once again decided that AF was the right choice for
the front-end.

ACKNOWLEDGEMENTS

The author would like to thank Rich Main of the SAS
Institute for giving generously of his time to answer my
questions about AppDev Studio and to review this paper.
However, any errors in here remain mine.

CONTACT INFORMATION

Barry R. Cohen, President
Planning Data Systems, Inc.
700 Ardmore Ave. #512
Ardmore, PA 19003

610 649 8701 (voice)
610 649 8408 (fax)
cohenbar@bellatlantic.net
SAS Alliance Consulting Partner

SAS and all other SAS Institute Inc. product or service
names are registered trademarks or trademarks of SAS
Institute Inc. in the USA and other countries. ® indicates
USA registration. Other brand and product names are
trademarks of their respective companies.