Paper 33-26

V6 to V8 Applications: To Web or Not To Web?

Sharon Muha, SAS Institute, Cary, NC
Elizabeth Malcom, SAS Institute, Cary, NC

ABSTRACT

When upgrading a V6 SAS/AF® application to V8, what are the possibilities? Should the application be web-enabled? If not, why not and what are the other options? Should it make use of Java™ technology? What are the platform restrictions? What are the interface issues? Are there data-integrity concerns? These questions are just some of the issues that must be addressed when retooling applications. This paper steps you through the decision-making process and highlights decisions made by developers who were tasked with upgrading a SAS Publications Division V6 SAS/AF FRAME application to a web-enabled V8 application.

INTRODUCTION

When an application must be upgraded to use a new SAS software release, the process occasionally involves only a simple change to point to the executable file of the new release rather than the executable file of the old release. However, application upgrades are unfortunately rarely that simple. Over the life of the application, user requirements frequently change, as do the business problems that the application was originally created to solve. In such cases, upgrading to a new software release often provides an opportunity to re-examine the application to determine what enhancements can or should be made.

As developers in the Business Applications Department of the Publications Division at SAS, we were recently tasked with upgrading a Release 6.09 application to Version 8. When upgrading an application from Version 6 of SAS software to Version 8 of SAS software, one of the key decisions that we faced and that you will face is whether to web-enable the application. In the current business atmosphere, where all things web and wireless are hot, it is tempting to assume that your application should automatically go to the web. However, there may be compelling reasons not to make that jump.

To determine whether web-enabling your application is appropriate, we recommend first stepping back and examining the scope and requirements of your application. What are the user requirements? What constraints do the user requirements put on your system? What are the system requirements? Are there new applications that will need to interface with your existing application? If so, what platforms are these new applications on and should your application be on the same platform? Only after examining such issues can you make an educated decision about which implementation (stand-alone application, Java applet, JavaServer Pages™ technology, and so on) to pursue.

DEFINING USER REQUIREMENTS

Knowing who your users are and what your users need to accomplish by using your application is critical to successfully upgrading the application. Especially if your system has not been upgraded or enhanced recently, your user base may have shifted, and the problems that your users are trying to solve may have changed.

IDENTIFY THE USER BASE

Do you know who your users are? It may sound like a ridiculous question, but it can be easy to overlook segments of your user population. For example, sales staff may use your application to record sales data, but do executives, middle management, or project managers need reports from that data? If so, those people are also indirect users of your application, and you must be certain that your application allows for gathering the data that are needed for their reports. In short, as you prepare to gather user requirements, be sure that you have identified at least one representative from each significant group in your user population.

IDENTIFY THE BUSINESS PROBLEMS

Do you know what your problem is, or more accurately in this case, do you know what business problems your application addresses? Are those problems, in fact, the problems that still need to be addressed? As business models change, your user base may shift and the problems that your application was designed to address may become obsolete. For example, if your application tracks sales by region, but your sales regions have been redefined since the application was updated, you will need to redefine the regions for which data can be entered.

Ideally, the information that you are looking for when identifying the business problem is what your application can do to help your users do their jobs more effectively and efficiently. Keep in mind that different users may need different things from the same application.

IDENTIFY CURRENT APPLICATION DEFICIENCIES

Although we strive to create intuitive, bug-free, user-friendly applications, the truth is that there is some shortcoming in almost every application. Find out from your users if there are aspects of your current application that are problematic to them. For example, are your users entering data on one platform and then having to port that data to a different platform for reporting or for use in another application? Is there a button whose label makes no sense to them? Is there a required selection buried in a secondary window when it should be in a more obvious location that is easily accessible in the application? Is it obvious how to save data that have been entered and then exit the application or move to a different area in the application? It is obviously impossible to include here an exhaustive list of questions you could pose to your users about your application. However, if you simply ask, "What do you find difficult about this application?" most users will be more than happy to tell you.

DETERMINE THE NECESSARY LEVEL OF ENHANCEMENT

Once you have identified your users, determined what business problems your application is working to solve, and identified any current deficiencies in your application, you will be in a position to determine how much enhancing you need to do to your application as you upgrade to the new version of SAS software. If your application is meeting your users needs and addressing the problems they need it to, congratulations -- you are likely in the rare and fortunate position that you simply need to port your application to the new version with no significant changes. On the other hand, if there are new users or new problems that your application needs to address or if the usability of the application needs to be improved, yours is the far more common situation, and you can now embark on determining further system requirements before selecting which implementation option is most appropriate when you convert your application.

USER REQUIREMENTS CASE STUDY

As applications developers in the Business Applications Department of the Publications Division at SAS Institute, we were recently tasked with converting a Release 6.09 SAS/AF time-tracking application to Version 8.

Following our recommended user-requirements definition strategy, we examined our user population and determined that we had several distinct groups of users: writers and editors, project managers for individual documentation projects,

department and group managers of the various departments and groups internal to the Publications Division, and the vice president of the Publications Division. Additionally, while in the past only writers and editors were required to track their time using the application, it was decided that after the first of the year, all Publications Division employees would be tracking their time, which meant that we had the needs of additional departments to consider. We selected representatives from the various employee groups and questioned all of the managers to determine how they were using the application and what they needed from it.

As we tried to identify the business problems our application needed to solve, we determined that the writers, editors, and other employees, while they were required to use the application to enter data, rarely needed to report on the information that they entered. Project managers did not need to enter data (at least not in the context of being a project manager); however, they did want to be able to get reports on the data related to their projects. Group and department managers also needed reports related to their employees (overtime reports, time away from work, and so on). Finally, the vice president needed to be able to report on the data for business-reporting purposes (to provide IRS data and to determine cost of books, for example).

Our users had no trouble at all identifying deficiencies in the 6.09 application. The reporting system was sorely lacking to create adequate human-resources management reports and made no allowance for project-management reports. The interface that writers and editors used to enter data was antiquated by Version 8 standards and in many cases was non-intuitive or slowed down what should have been a fairly quick data-entry process. Finally, tracking codes needed to be created to support data entry by employees other than writers and editors.

After gathering our user requirements, it was clear that we needed to overhaul the time-tracking application in the process of upgrading it to Version 8, and we moved on to determining what our system requirements would be for the new version of the application.

DETERMINING SYSTEM REQUIREMENTS

Now that you have gathered your user requirements, your focus can shift to the application itself and to the system in which that application must function. You will likely find that your user requirements constrain how you implement your application. Additionally, you will need to consider issues such as platform availability, user location (anywhere from the same floor of a building to multiple international offices), and data integrity and security.

All the issues that must be addressed and the questions that must be asked to determine system requirements for any given system cannot realistically be included in this paper. However, keeping in mind that our ultimate goal is to convert an application from Version 6 to Version 8 and to determine whether the Version 8 application should be a web implementation, this section poses general questions in some of the major areas that you should address as you plan your conversion.

IDENTIFY PLATFORM ISSUES

Should your application continue to reside on its current platform? Have the user requirements indicated that there may be a need to change platforms? For example, you could move your mainframe application to a PC platform and, leaving the application data on the mainframe, use SAS/CONNECT® software to access the data.

Consider whether your application currently interacts with or will need to interact with other applications. Are the applications on the same platform, and if they are not, should they be? Because SAS software provides numerous tools for cross-platform communication, running related applications on different platforms is a realistic scenario; however, there may be additional considerations that make a different implementation a better choice.

IDENTIFY YOUR USERS

The location of your user population may greatly influence the implementation option you ultimately select for converting your application to Version 8. Are your users centrally located so that you have immediate access to the machines from which they will run your application? Will your application simply be installed on a network so that users throughout the office complex, city, state, country can run it from the network?

IDENTIFY DATA CONCERNS

The form, integrity, and security of the data created and accessed by your application are obviously key issues with respect to application development. As you consider upgrading, note whether your application is to be used for simultaneous data updates. Is there already code in place to ensure data integrity (i.e., record locking to ensure that one update does not incorrectly write over another)? Is access to some or all of your data restricted to a particular user or groups of users? Does your application access or create data that must be kept secure? For example, does your data contain financial information or account numbers that must be kept private?

IDENTIFY YOUR DEVELOPMENT STRENGTHS

As you consider your implementation options, also consider your development strengths. What are your areas of expertise? Can you write SCL code easily but have no experience with Java technology, or conversely, do you have Java experience but no experience creating SAS/AF applications?

If you have more than one implementation option and development time is critical, play to your strengths and use the implementation option with which you have the most experience. If you have several implementation options and you have enough time in your development cycle to absorb a learning curve, consider selecting an implementation strategy that will enable you to develop your skills in a new area.

SYSTEM REQUIREMENTS CASE STUDY

Once the user requirements for the Publications Division's updated time-tracking application had been determined, we turned our attention to defining system requirements. The application we were upgrading was originally a stand-alone application; however, as one of our first requirements, we knew that the new application would need to interface with a new, undeveloped system that would be used to track the life-cycle of Publications Division products. Although the new tracking system was not yet developed, we had to consider ways in which the new system and the time-tracking system would interact and model our data tables accordingly. Additionally, we were told that the application needed to be web-based. Ideally, your users will give you, the developer, the opportunity to determine based on all the available information what platform and implementation is most appropriate. However, in many cases, as in this case, you will simply be told which strategy must be used.

In examining our user base, we determined that our users were primarily local users and users from one regional office. However, we also noted that the users of the larger tracking application into which the time-tracking application would be integrated are located worldwide. Given that we knew that the two tracking applications would potentially share data, we felt that we needed to locate the data where the larger application (and the worldwide users of the larger application) could access that data.

With respect to data concerns, we did not have to concentrate greatly on this area. The applications that we develop run completely behind a firewall and, at least with respect to the time-tracking application that we were immediately concerned with, do not contain sensitive, private data. Our primary concern in this

area was being certain that records were not being updated simultaneously, but those concerns were alleviated by constructing the data tables to avoid this problem.

The strengths and areas of expertise in our development group were largely in the SAS/AF software area. None of us had significant Java experience though all of us were familiar with object-oriented programming principles.

Having determined the user and system requirements, we were now in a position to examine all of our data and determine an implementation strategy for the conversion of our application.

SELECTING AN IMPLEMENTATION OPTION

Now that you have identified your requirements, you can begin to tackle the big question: will your application be a stand-alone application or a web-enabled application?

Because the application we were tasked with converting is an internal application that runs completely behind a firewall and does not contain particularly sensitive data, we had the luxury of not having to address data and server security issues; therefore, because these issues are very much outside the realm of our experience, we do not make recommendations here for which implementation is best suited to deal with data and server security. We recommend that you discuss such concerns with your systems-support staff and if necessary with your SAS representative or a SAS consultant.

STAND-ALONE APPLICATION

Despite the fact that the web is hot and everyone seems to want to use it, there are some advantages to using a stand-alone application instead. If your application depends heavily on users navigating through it in a particular way, you may want to consider a stand-alone application. In a web-based application, there is no guarantee that your user will step through the application as you expect. For example, a user could simply shut down the browser (intentionally or accidentally) halfway through executing an application task. A user could return to a previous application web page without saving data from the current page. There are ways to exert some control over a wandering user; for example, you could use JavaScript to identify and respond to particular events. However, if you find yourself thinking that it would be great if a user could bring up a separate browser completely dedicated to your application, then consider going with a stand-alone application because users typically cannot be counted on to use web browsers that way.

Just as you avoid the wandering-user problem by implementing a stand-alone application, so too do you avoid dealing with data-caching issues. Generally speaking, you can count on knowing the settings of a stand-alone application window because you programmed them to act the way that they do. Knowing the settings, you can program the window to refresh after particular events. You cannot count on all users to have data-caching options set the same way on their browsers, and making sure that users are looking at the most up-to-date data on any given web page can be a challenge.

Another advantage is that it is easier to control simultaneous access to the same record in a stand-alone application. Record locking is difficult when implementing a web application but is relatively straightforward to manage in a stand-alone application.

With respect to user-interface considerations, stand-alone applications have the advantage of allowing you more control over the application window, for example, the size of the window and where it will appear when it comes up. Also, you can more easily create complex graphical user interfaces (GUIs) that change based on selections the user makes as he or she makes them. For example, a checkbox the user selects may change which items are available on a selection list in the same window.

If you are working in a PC networked environment, a considerable disadvantage to a stand-alone application is that either you have to have the application installed on each user's machine and then any time a patch is needed, everyone has to install the patch or upgrade individually, or you have to have an application server that is powerful enough to handle your number of users (which can be a challenge if you have a significant number of users and a complex application). If applications are installed locally, it is difficult to guarantee that all users have installed the most recent version. In the application-server scenario, in the event of an update, you can at least put an icon on the desktop that would point to the latest version of the application.

In a Unix environment, the issues seem to be more ease of maintenance and performance considerations. The farther away your users are from where your application is running, the slower the stand-alone application runs because data related to both the compiled catalog entries and the data sets need to be transmitted so the processing can be done at the client. One way around this problem is to install the application locally, which can require a heavy maintenance load. Another possibility is to surface the application to regional offices on the web via a Hydra application server. In this scenario only small amounts of data are transmitted, and all the real work is being done on an application server at your headquarters. This approach still requires a second installation of the catalogs on the Hydra application server, but maintenance is not quite so heavy as installing the application locally. The advantage to using a web implementation, specifically a JSP solution, in this situation is that theoretically there is only one copy of the application and the data to maintain and performance should be relatively the same regardless of the access location.

WEB IMPLEMENTATION

Hip, hot, here, and now - it's the web and your application can be on it. In addition to the "hipness" factor, though, it is true that web applications are generally viewed by users as being more easily accessible. They also have the distinct advantage that there is no need to install an application on everyone's desktop. Also, using tools provided with SAS software, you can access your application data on virtually any platform. For example, you could use the ROCF classes provided with webAF™ software to access your data, or you could use JDBC calls. So, even if your application data are stored on a mainframe, that in itself should not keep you from putting your application out on the web. An additional advantage to pursuing a web-based implementation is that you can use the functionality already built into the browser for basic searching capabilities and for common tasks such as printing.

A disadvantage to a web implementation is that if you are dealing with a complex GUI, it can be difficult or impossible to reproduce your stand-alone application interface in a web page. For example, if you need to vary the available selections in a list based on information that the user provides elsewhere in the GUI, you may not be able to do that in a single web page. In addition, you will need to address data-caching issues to be sure that pages are generated appropriately (or not generated at all) when a user attempts to revisit a page. Finally, you must determine where your web server is going to reside and make sure that all the necessary servers to run your application can communicate with each other where they are.

There are two obvious choices for a web implementation: Java applets and JavaServer Pages (JSP™) technology. Each option, as you might expect, has advantages and disadvantages.

JAVA APPLET

If you use a Java applet rather than JSP technology, you have more control over the GUI than you do if you use the available HTML entities. The Swing classes, in fact, enable you to create fairly elaborate interfaces.

A major disadvantage to pursuing an applet over a JSP solution is that you have to be sure when using an applet that everyone has the correct plug-in. Alternatively, if users do not install the expected plug-in, the rendering of your application GUI may vary widely relative to the browser configuration. Also, download time for applets can be significant. Will your users be patient enough to wait for the download?

Data processing is another area where there is some advantage to using a JSP solution rather than an applet because data processing in an applet is confined to the client. If you have to write back to the server using an applet, you will need to implement a SAS middleware technology solution, and while that is certainly possible, writing data can be much more straightforward with JSP technology.

JAVASERVER PAGES TECHNOLOGY

Using JSP technology, you can process your data on the server side, which allows you a very thin client. Also, you do not have to deal with download issues in the same way you do with applets. There can be an initial download hit the first time a user runs a JSP solution, but after that, the process speeds up considerably.

With respect to GUI development, because most developers are generally familiar with the existing HTML entities, the learning curve to implement a JSP GUI will likely be lower than the learning curve to create an applet GUI. However, your JSP GUI may not be as snazzy unless you are willing to branch out and implement some JavaScript or pull in a graphic designer to assist with livening things up with graphics (keeping in mind that graphics will affect load time).

IMPLEMENTATION OPTION CASE STUDY

Deciding which implementation to pursue was frankly the most difficult decision we dealt with in converting our application. We had been instructed to pursue a web-enabled solution, so that decision was easy. However, figuring out which web implementation to pursue on what platform, was a challenge.

Our current application data lived on Unix, but we had the option of moving it to an NT environment for the new time-tracking application. The future Publications tracking application that our time-tracking application will interact with will have data on numerous platforms, so the location of the data we might need to access in the future was not really a deciding factor when choosing our data platform. The current development platform in our department is primarily HP-UX reached via an emulator running on an NT workstation. There were indications that the department would move to the NT platform for future development. That being the case, we decided that our data would live on the NT platform.

The server issue, which servers we needed and what platforms they would live on, was another one that dogged us for longer than we would have wished. We determined that based on our number of users (about 150 occasional users and 10-15 constant users), either a Unix or an NT application server would be reasonable to meet our needs. With respect to web servers, we found that we were more familiar with NT web servers and could more easily run and test web servers on our NT workstations than on our Unix network. Additionally, we found that we could get an NT application server cheaper and faster, so we concluded that an all-NT solution would best suit our needs.

We ultimately chose to pursue a JSP implementation over a Java applet. Although in either case we had to learn how to program in Java, a JSP implementation seemed easier to understand, and we had more JSP sample programs available to use as models. We also wanted to avoid plug-in issues and hoped to capitalize on data-processing efficiency by taking advantage of the server-side processing that JSP technology provides.

CONCLUSION

Unfortunately, there simply is not a unique set of steps you can follow to convert your application from Version 6 to Version 8. The response "Well, that depends..." followed closely by yet another question will persist through most of your conversion-implementation analysis. However, patient and detailed examination of your current application and the goals of your new application will ultimately yield a workable implementation strategy that will in turn ultimately yield a V8 application best suited to the needs of your users.

ACKNOWLEDGMENTS

Thanks go to Leah Redfern for her contributions to this paper and to the project on which the paper was based.

CONTACT INFORMATION

If you have any comments or questions, please feel free to contact either author at:

Sharon Muha / Elizabeth Malcom
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Phone: 919-677-8001
Fax: 919-677-4444
Email: sharon.muha@sas.com
liz.malcom@sas.com