Paper 43-27

Avoiding Entanglements: Migrating Applications to the Web

Eric Brinsfield, Meridian Software, Inc.®, Raleigh NC

ABSTRACT

After many years of developing SAS/AF® applications, most of us find ourselves facing or pondering conversions to Web-based SAS® applications. Before beginning the process, we need to ask ourselves a series of important questions. For example, is the end result worth the cost of migration? Or, should we convert the entire application or just parts of it or should we just start over? And, what tools should we use in the new Web-based software?

In this paper, I will discuss many of the issues that we should take into consideration when preparing for and planning a migration from a fat-client application to a thin-client browser application. I will present specific case studies that illustrate successful migration paths.

INTRODUCTION

For many years, SAS/AF and SAS/EIS® were the primary tools used for SAS user interface development. SAS/AF and SAS/EIS were, and still are, highly customizable and powerful. The functionality and "look and feel" of these custom applications varied widely based on the experience and training of the designers and programmers building them.

Now, as the use of Web browsers has become a household activity, most people are reasonably experienced and comfortable using a Web browser. Although every Web site application is different, they all have some common threads of behavior, based on the browser used to access the site. In other words, different Web browser applications seem to behave similarly just because they are using the same Web browser.

So, with widespread availability and familiarity, Web-based applications offer shorter learning curves and less resistance to change. In many cases, companies have set the Web browser as the standard interface for any new application installed on their networks.

But, how do we move existing applications to the Web browser interface without rebuilding the entire application. To address this issue, this paper will present a case study that illustrates one way to deal with this very common dilemma.

In this paper, I will use a case study to illustrate the issues involved in migrating an application to the Web.

For that case study, I will:

- Provide a general description of the case study
- Document the objectives of the project
- Identify the high-level set of options available for achieving the objectives
- Note the client requirements and constraints on our solution
- Discuss the option selection process
- Explain our rationale for the final solution

Case Study

In 1998, Meridian Software completed a pilot project and delivered a Quality Analysis system to a manufacturing client that provided the following features:

- Real-time data feeds from a production control system
- Master quality database stored in Microsoft SQL Server
- Data-entry subsystem built using SAS/AF Frame technology
- Real-time quality monitoring functions using SAS development tools including SAS/AF, SAS/Graph®, SAS/Stat®, and SAS/QC®.
- Some historical analysis using the master quality database and SAS with the interface built in SAS/AF

In reality, our project was more than a pilot. In this case, pilot meant that we would develop a complete system, but only provide services for one department rather than all. We also deferred some large-scale decisions, such as data warehousing, until after we proved the value of the application. For clarification, note that department refers to discrete steps in the manufacturing process. The pilot focused on one piece of the puzzle with its own set of engineers and technicians.

Our projects with this client have been broken into phases. Phase 1 was the pilot project. Phase 2 involved adding more capability and statistical analysis and only expanded the scope to include one other department. The focus of this discussion is on our proposal process for Phase 3, which added significant capability and scalability and the potential for conversion into a Web application.

OBJECTIVES

Observations after 6 months of production usage

After using the pilot version of the Quality Analysis system for 6 months in a production environment, the client gained incredible insight into their manufacturing process as well as revealing facts about their suppliers' quality control. The system gave the quality analysts access to data that was never available before and reduced historical analysis time from six weeks to hours or even minutes.

After a second phase, that delivered even more statistical power, we found ourselves in the following situation:

- The client wanted to add more features and expand the system to cover additional departments (steps in the manufacturing process)
- Only a few users had access to the system, because they could not justify licensing SAS on the workstations of infrequent users.
- More technicians, engineers, and managers wanted to have access to the analysis and monitoring features
- SAS/Intrnet® software had matured
- Our client had recently installed a set of Web servers and a corporate intranet. They set a new standard that favored Web browser interfaces for all new software. Any application that was not using a Web interface would eventually be phased out. Our application was the only SAS application at the site, which meant that SAS could have been phased out as well.
- Performance was unsatisfactory for users with low-powered workstations. With insufficient memory and CPU speeds, the SAS-based reporting tool seemed sluggish, although it

worked fine on reasonable machines.

Phase 3 Objectives

Upon hearing the issues, detecting some of the future trends, and receiving their request for more features, we saw the need to move to a Web browser-based application. When we provided a proposal for Phase 3 enhancements and expansion, we included plans for a conversion to a Web application. With our plan, we hoped to achieve the following objectives:

- Add new analysis and graphics
- Expand availability by making monitor and analysis features available over the corporate intranet
- Get information and our system in front of engineers and upper management
- Insure future compliance with the corporate direction toward Web interfaces thereby increasing our chances of acceptance and showing how valuable SAS software can be
- Improve performance on low-powered workstations by using thin-client technology that shifts the workload to the server

But, we did not want to propose starting over and discarding all of the work from the past two years. So, we had to consider our options carefully.

OPTIONS

No Change

Doing nothing was not an option because they wanted expansion. We did not consider this option.

Upgrade SAS/AF Application and Spread to More Workstations

Because of cost considerations, we had to evaluate the option of keeping the application completely in SAS/AF and background processes. Although we included this option in our proposal, we pointed out that this option did not reflect the true capabilities of SAS with SAS/IntrNet®. We provided a fair estimate of the upgrade price along with advantages and disadvantages.

If we had to add a few additional reports, staying with SAS/AF would have been a cheaper solution. But, after the pilot version had been in production for a year or so, we noted areas of frustration for the data-entry technicians that we needed to address.

In particular, data entry and reporting were all built into one application. The users did not like closing their data-entry windows in order to go look at a report. We solved this with multiple SAS sessions, but we did not consider this a robust solution for scalability. So, some redesign was necessary whether we converted to the Web or not.

The biggest negatives for a complete SAS/AF solution were cost of additional workstations with hardware upgrades and negative impression in view of the popularity of Web interfaces.

Convert to a Total Web Solution

We considered the possibility of converting all user interfaces to a Web application. This conversion would include all reporting tools, all analysis tools, all administrator tools, and data-entry subsystems. Given the power and complexity of the data-entry applications currently in use, a total conversion seemed like a very expensive option.

In addition to deciding whether to migrate to the Web or not, we also had to consider which Web tools were appropriate. Specifically, what combination of the following tools should we use:

- SAS/IntrNet Application Dispatcher and Load Manager
- CGI
- JavaScript
- Active X
- JAVA
- AppDev Studio
- Or an additional Web development tool

Build a Hybrid Solution (partial conversion to Web)

As the final option, we also evaluated implementing a partial conversion, which would leave more complex functions that were used by fewer people, in SAS/AF, while converting queries and analytical functions to the Web. Very few users were entering new data, while many people wanted to access the informative graphs and reports available in the reporting and analysis subsystems. The hybrid solution looked appealing and offered some obvious payoffs.

REQUIREMENTS AND CONSTRAINTS

To evaluate the options fairly, we had to consider all of the requirements carefully and see which options withstood the customer's constraints. Specifically, we considered the following issues:

Usability

If more users were going to have access to the system, interface usability was critical for success. We could not ask high-level managers to take time to attend training classes on using our system. Because almost everyone is comfortable with a Web browser now, the Web conversion seemed like a very logical and important step for success.

Accessibility

Obviously, the Web application would be more accessible. The client was not willing to license SAS for every possible, casual user, who might want to view a graph. So, SAS/AF was definitely limiting accessibility throughout the company. The Web version would make the application available to everyone in the factory as well as employees at other plants on the same corporate intranet.

Maintainability

One of the reasons IT folks love the thin-client model is because thin-client applications are easier to maintain. All of the code resides in a central location on a server or servers rather than dispersed on individual user workstations. In addition, by using standard Web tools, the client would not be as dependent on contracted SAS expertise for all of their support, if that is an issue.

Performance

By forcing the processing back to the server with a thin-client Web application, we could reduce the demands on underpowered workstations. This benefit moved the scale heavily toward the Web conversion, but we also had to consider the increased workload on the server. By moving all processing to the same server along with increasing the number of users, we had to consider the possibility of overloading the existing server.

Security

Within the company, everyone was permitted to see the data and review the reports, but only selected employees were authorized to update the database. So, data entry posed the biggest security challenge.

Concurrent Access

Concurrency was built into the application from the original design and was still critical as we expanded the availability. With automated process updates every 5 minutes for two different data sources and manual data entry on a frequent but irregular basis, concurrent access to the data had to be studied again in view of the addition of more users coming in from the Web.

"Real Time" data with historical analysis capability

With potentially more people running historical analysis, concurrent access to the same database could become a performance problem. So, with the expansion to more users and conversion to the Web, we had to review the database design and server processing as well. The data-entry requirements were in conflict with the analysis requirements, so how do you spell relief?

D - A - T - A M – A – R - T

We could not justify using data warehousing techniques in the pilot project, but before scaling up, we had to implement part of that strategy. Future expansion will involve even more formal data warehousing steps.

"Most Bang for the Buck"

As with any consulting project, our clients want to get the most "bang for the buck", in both the short term and long term.

We could achieve this only by:

- Reusing as much existing code as possible
- Evaluating cost of development versus value of end result (ROI)
- Delivering the new version in as short a time as possible so new capabilities were available sooner
- Keeping maintenance costs low (maintenance cost had been negligible over the past 2 years already)
- Decreasing demand on workstations so our application and SAS were not the cause for an increase in hardware requests
- Getting the most out of their SAS license, opting to add new products and capability rather than additional licenses, which would be underutilized
- Increasing the expected life span of the application thereby decreasing near-term costs of replacement and increasing long-term ROI

OPTION SELECTION PROCESS

In our proposal to the client, we provided cost figures for each major option along with the description of the pros and cons of each. As you may have guessed from my comments earlier, we recommended a hybrid solution, which left data- entry and administrative tools in SAS/AF and moved all monitoring and analysis solutions to the Web.

Basically, we could not recommend building new features with SAS/AF for all of the reasons listed above. Although SAS/AF is a powerful tool, the Web is too popular and widely accepted to ignore (even after the "dot com bust").

We also considered moving everything to a Web application, but to achieve the same level of flexibility in the data-entry features, we would need more time and more JAVA development.

At the time of the proposal, SAS Software's AppDev Studio™ did not support JAVA Server Pages (JSP) and we (and many of our clients) were not satisfied with the performance of JAVA applets. So, although a JAVA approach might make sense today, we felt that increased price and decreased performance were significant risks with a total Web conversion at that time.

The hybrid solution could be completed quickly, providing the greatest functionality and accessibility at the least cost.

FINAL SOLUTION

In our final solution, which went into production in December of 2000, we implemented a hybrid solution that kept data-entry functions in SAS/AF and moved all reporting and analysis functions to the Web browser. To support our new design, we also recommended some hardware and database changes that improved the overall performance and function of the Web application.

Each of the major changes is described below, starting with the system architecture changes.

Split the server

In anticipation of increased demand on the server when more thin-client users started executing SAS on the server, we recommended splitting the single Windows NT Server into two servers, thereby creating a three-tier architecture. One server housed the Microsoft SQL Server database, while the other became the Web server and SAS Application Server. By splitting these functions, multiple SAS sessions on the application server were not competing with SQL Server for CPU and I/O time.

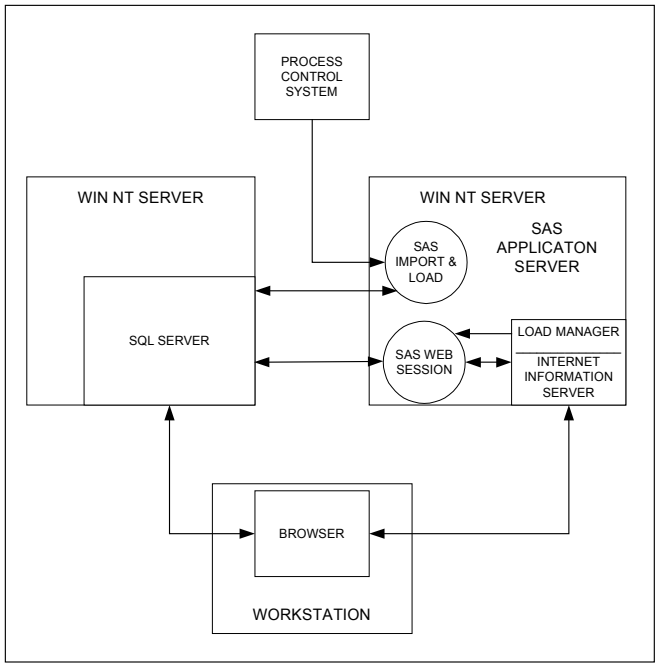

Figure 1. Three-Tier System Architecture

After implementing this split and upgrading the old machine at the same time, we saw significant performance improvements in the SAS/AF-based data-entry applications and the import processes even while supporting thin-client analytical users. To achieve this split, we did have to modify the design of our import process slightly to take advantage of the parallel processors. In addition, we optimized all queries to split the workload appropriately between the two servers. Large queries would reduce the result

sets within SQL Server before passing them back to SAS for further processing.

Split the database

The original database was designed to support real-time updates and data entry. Consequently, historical analysis activity could sometimes be slow or go into wait state during large imports. These conflicting objectives provide part of the motivation behind today's data warehouse technology. As part of Phase 3, we recommended implementing a more typical data warehouse approach before expanding to any other departments.

So, we took two steps to provide the fastest performance for both types of users (transactional and analytical). First, we partitioned the database into current data (less than three months old), historical data (between 3 and 12 months old), and archived data, which was moved offline.

All transactional processes, such as data entry and automatic import, accessed only the current partition. In addition, monitor programs that took a quick pulse of the system in real time, used only the current data.

Secondly, we designed and built a data mart that is refreshed twice a day automatically, but can be refreshed on demand if desired. The data mart was designed to optimize analytical processing and included data from the current and historical databases.

By implementing this strategy, we improved performance for data entry and imports, while providing impressive speeds in the Web application. At any given time, the data mart may not contain the most recent process data, but for historical analysis or defect trend analysis, new data points are not critical. In addition, this strategy increases the scalability of the application as we incorporate more departments into the application.

Upgraded Data Entry

Although we did not convert the data-entry application to the Web, the hybrid approach provided enough leeway in the budget to allow us to streamline the existing SAS/AF application. By removing functions from the SAS/AF application that had been shifted to the Web, we could give users an application that was more focused on their specific data-entry tasks. They could use the Web for monitoring, while using SAS/AF for the data entry on the same machine.

We also took advantage of the split server and optimized the data-entry inserts and queries in consideration of that design. Consequently, the technicians were much happier, because data entry was faster and easier and they could pop over to the Web application to see their results instantly without closing their data-entry application or running two SAS sessions.

As a standard client-server application, the data-entry subsystem was unaffected by any workload on the SAS application server, because SAS and the SAS/AF application ran on the data-entry workstations and communicated directly with SQL Server. The data-entry application did not need to request any services from the SAS application server.

Converting Monitoring and Analysis to the Web

As part of phase 3, we evaluated the current application and categorized the reports and graphs as either process monitors or quality analyzers. We used this dichotomy in the Web application. In addition, the new version also added a yield analysis feature that produced reports on yield rates and production levels rather than on quality and defect levels.

Each category has the following attributes:

Process Monitors:

- Provide snapshots of current processes in real time
- Utilize the "current" database, so historical data was not available
- Are customized for a specific function
- Execute with a default set of options and filters, so they could be run with a single click of the mouse
- Run extremely fast
- Offer very few customizable options, if any at all
- Include status reports, event viewers, control charts, trend charts, density plots, and Pareto charts

Quality Analysis and Yield Analysis:

- Provide a historical view
- Utilize the data mart and did not usually touch the real time database
- Are extremely flexible, permitting the user to define data subsets and specify graph or reporting options
- Include predefined and custom reports, control charts, graphs, trend charts, density plots, Pareto charts and advanced analysis
- Enable users to download the data that was used to create the report into SAS data sets or Excel

We utilized SAS/Intrnet and the Application Dispatcher with Load Manager for load balancing. The Load Manager controls how many SAS sessions are made available to the browser sessions and reuses open sessions that are not in use.

We used a combination of HTML, JavaScript, and VB Script with Active Server pages to make the Web site dynamic. SAS was used primarily for reporting and analysis after the user selects options. The Active Server pages communicate directly with SQL Server to create dynamic select lists and to perform non-analytical functions for the browser sessions.

Because the Meridian Software standard for software development encourages SAS/AF developers to isolate SAS programs in a standard directory or macro library rather than embedding them within the SAS/AF application or catalog entry, reuse of existing report programs was very easy.

Instead of collecting and passing user parameters with an AF window, we collected the parameters within a browser and passed them to the same programs. Our only major modifications to the report programs were to add utilization of the Output Delivery System (ODS), because we also upgraded to Version 8 in Phase 3. ODS made the conversion to the Web even easier.

SUMMARY AND FINAL ANALYSIS

Faced with upgrading an existing SAS/AF application, you should seriously consider converting all or part of the application to the Web. SAS/IntrNet and the CGI interface provide a painless path for existing SAS programmers to take into the Web application development world. SAS programmers do not have to jump directly to JAVA.

With proper analysis, you can convert quickly and cost effectively to the Web browser without losing the power of SAS. We achieved a significant cost savings by settling on a hybrid solution rather than a total Web conversion. Our client is very happy with the new interface and the users seem more comfortable working in the Web environment.

If we were starting on the project today, we would probably consider using more JAVA now that Version 2.0 of AppDev Studio is available. The development tool offers more features with more communications capability and support for JAVA Server Pages. It is quite likely that we will be using AppDev Studio in Phase 4, when that happens.

CONTACT INFORMATION

Eric Brinsfield
Meridian Software, Inc.
12204 Old Creedmoor Road
Raleigh, NC 27613
(919) 847-6750
merecb@meridiansoftware.com
www.meridiansoftware.com

TRADEMARKS