

The Future's Bright, The Future's .NET

David Shannon, Amadeus Software Limited.



ABSTRACT

Today is an exciting time to be a user of applications that make use of SAS®.

SAS Institute are steadily using more and more alternatives to SAS/AF, for building interfaces to the SAS System. Such examples are Enterprise Guide and the forthcoming Studio Reporter, written in C++. This sends a clear message from SAS Institute with regards to application development strategies for our front ends that harness the power of the SAS system.

Much has been made of web browsers for easy management and deployment of applications which make life easy for administrators and groups of users who are widely dispersed; however less is understood about the possibilities of rich client applications, that is, traditional desktop applications which provide the user with considerably more interactivity than your browser.

This paper gives an overview of application development strategies with an appraisal on the uses of both web browser and desktop application approaches with Microsoft's .NET technology. Most importantly, this paper takes the perspective of the application user and benefits from real life experiences of developing front ends to SAS for both the browser and desktop.

INTRODUCTION

The face of application development with the SAS System has until relatively recent times been restricted to a framed environment within the SAS System's own display manager.

Towards the latter stages of version 6 came SAS/IntrNet, a whole new approach of using web browsers that can access and drive the functionality of what SAS does best; manipulate and analyse data – but SAS is installed only a server which minimises on deployment and maintenance of the application.

With the introduction of Integration Technologies further possibilities for integrating the SAS System with not only the browser, but also rich desktop applications such as Microsoft Office or bespoke applications written with languages such as C#, C++ or Visual Basic .NET.

These opportunities present designers of applications with potentially many options, each with their own strengths and weaknesses from the perspective of both the users and maintainers of such applications. This paper intends to focus on the desktop applications and set out the options available for application development that harnesses the power of the SAS System, whilst providing users with highly intuitive interfaces.

WHAT'S HAPPENING TO SAS/AF?

When AF was first released in around 1985, it was one of very few development systems that allowed a full colour user interface to be built that could exist of operating systems as varied as MS-DOS, Unix and MVS. SAS became renowned for this strength.

Through the increased computing popularity during the late 1980's and 1990's SAS/AF saw numerous enhancements, notably for the release of Version 8, which allowed applications to be built that closely resembled the Microsoft Windows look-and-feel which computer users have now become accustomed to.

Towards the end of development for SAS Version 6 and through the increased use of SAS Version 8, the focus of demand for application development shifted from fat-client interfaces to browser interfaces and open integrated solutions.

No new features have been added to SAS/AF for SAS System 9, although SAS/AF is fully supporting at the current time. *Unofficially* Cary developers have indicated that SAS/AF will be supported for as long as there is sufficient demand.

Clearly many users of the SAS System will themselves be users of frame applications which reliably perform their intended tasks and therefore quite rightly have no immediate need or intention to migrate their application at the current time. However in the longer term maintainers of such applications should consider the potential and appropriate strategies for migrating the front-end into another format.

It is at this point that organisations and programmers consistently raise two concerns:

- Investment made in SAS applications
- Redundant skills sets

Inevitably both organisations and individuals will have made investments in both time and skills that are specific to applications built and the cost involved of simply moving an application out of SAS/AF into another front-end language is potentially enormous.

It should be made clear at this stage, that *all stop* on the use of AF has not been called, indeed the SCL language has been extended for the latest version of SAS: Version 9. At the same time a method in Integration Technologies allows access to a class in SCL.

This allows for an application development strategy that makes a clear distinction in skills sets, between the developers of the user interface, written in a visual language and the developers of the business logic, written in SAS.

From experience of developing in precisely these scenarios, a conflict has arisen where each language used has the capability to perform a given task, where that task cannot clearly be defined as belonging to one tier or the other. Such cases are optimally implemented in the way that minimises the overhead on performance to the application.

WHY HAVE SAS INSTITUTE TAKEN THIS APPROACH WITH SAS/AF?

Undoubtedly for many years SAS Institute has been devoting large resources to keep AF abreast of the technology advances made by language vendors, in terms of the functionality which AF and SCL gives to programmers and the ultimately to the experience of the user.

As discussed, through the 1990's the emphasis on developing fat-client applications shifted towards thin-client browser based solutions, to which there evolved another demand for SAS to satisfy.

In addition to these factors, recent years have seen increasing demand for open and integrated software solutions.

More recently SAS have provided modules for CGI based web solutions (SAS/IntrNet), and since then also have provided a development environment for SAS centric application development, primarily orientated at Java web development (AppDev Studio). Finally, with the Integration Technologies module, SAS software has been opened up to standard communication protocols such as COM and CORBA for integration with third-party software.

With all of these factors considered it can be clearly seen that maintaining AF in the face of demand for both open and integrated desktop and web based solutions is infeasible.

TECHNOLOGY, TECHNOLOGY EVERYWHERE... BUT NOT A PERFECT ANSWER IN SIGHT

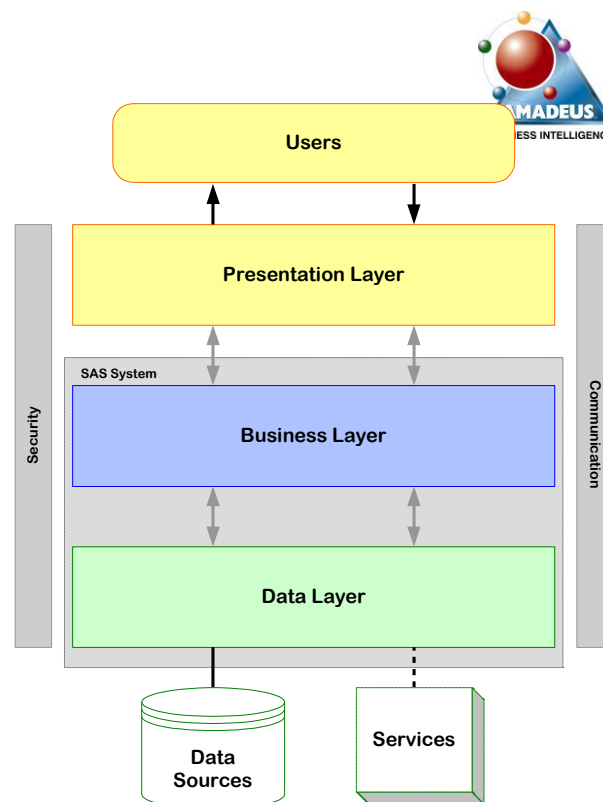
So, the architecture becomes a little clearer. For application development where SAS is at the heart of the business logic, i.e. data manipulation decision making, modelling etc. we can clearly distinguish SAS from the user interface – and even back-end data sources by building our application in tiers.

Throughout our whole application at least two factors should be considered at all times, namely security and the communication between logical layers.

Figure 1 is a simplified model demonstrating this layered approach, and begins to demonstrate how logical areas of an application can be identified.

As we are interested in harnessing the SAS System's strengths, i.e. its data manipulation and modelling; the SAS System is already identified as providing the business logic and data layers.

Figure 1: Logical Application Layers



The remaining choice of software, is that of the choosing the language to write the interface to the application.

Herein lies the potential for dilemma. Amongst several others are the languages of C, C++, C#, Delphi, J#, Java and Visual Basic to name a few of the most common languages in the market today.

Several factors are taken into account when making such a decision including existing hardware infrastructure, communication protocols through to availability of skills sets for implementation and maintenance.

SO WHY MICROSOFT AND WHY .NET?

Firstly I should justify my decision to develop with .NET.

Over 80% of SAS users are now running on the Windows operating system. This follows given that around 98% of personal computers run one flavour of Windows operating system or another.

SAS Institute have opened up the SAS System to interoperate with other packages, primarily in two ways: via Microsoft's component object model (COM) or through Java classes.

At the time of writing (April – May, 2003) I visited SAS Institute's Enterprise Integration Community website; where those who develop SAS post examples of integrating SAS through other languages. I note that of the eleven downloads available, ten of these are Microsoft based. Only one is Java.

Microsoft's languages are the leading rapid application development tools throughout the world. This means that there is a vast resource of skills available, in terms of publications, manuals, training and user communities.

Financial cost is also a factor. To purchase Microsoft's Visual Studio .NET 2002 (enterprise edition) costs around £1000. This includes no less than four programming languages and a host of other software for developing applications. This is extremely cost effective when compared SAS Institute licenses, where its first year fee for any one product is typically in excess of the Microsoft price.

Everything with the “.NET” tag in its name reflects the latest generation of technologies from Microsoft. Anything written for .NET runs on Windows 98 second edition and newer operating systems. The technology that is .NET will be very much the current and future generation of software associated with Microsoft.

Finally, in my justification, I turn my attention to the end users of applications.

The ability to support seamless integration into other Windows desktop software, of which end users are familiar and accustomed to is very much going to be of interest to end users, along with a consistent look and feel to the software.

Most often, it is the opinion about software that its user has, which determines whether or not the software is deemed successful or not. This is the same for SAS centric applications and remarkably studies has shown that the physical presentation of an application accounts for the vast majority of the opinion formed.

Software built using Microsoft Visual Studio languages enable rapid development and aids developers in providing a simplified user experience which minimises the level of required training. Such a design is shown in Figure 2, below.

Figure 2: An Intuitive User Interface



This intuitive design is taken from the Windows XP operating system control panel, where user accounts are modified. The screen uses clear, meaningful statements along with graphics to guide the user through the steps required. Users click on text to make their selection in a similar way to web pages.

PUTTING ON YOUR FACE: WINDOWS, WEB OR BOTH?

Now that the choice to develop with .NET is justified, the format of user interface, or indeed multiple interfaces should be determined.

There are several possibilities, each able to exploit the power of SAS.

Figure 3: .NET Client Side Development Options

.NET Windows Forms

Windows Desktop Application

Offline Client (e.g. Laptop whilst disconnected from LAN)

Smart Client (e.g. PDA)

ASP .NET

Web Browser User Interface

Web Digital Dashboard

Thin Device

Microsoft Office & Other

Office Applications (e.g. Excel workbook data)

Outlook Messaging

SharePoint Server

WEB HAS BEEN THE FLAVOUR OF CHOICE

For a few years customer demand has been to deploy distributable applications to end users through web browsers. This strategy allows for ease of maintenance, deployment and availability. This is the primary strength of web based (or thin client) solutions.

Many companies, however, have found there are constraints with such approaches. Whilst allowing system administrators a versatile approach, web based solutions have some common complaints; such as a high level of latency and poor level of desktop interactivity for the end user.

Whilst the web solution is clearly appropriate for the right application, or in given scenarios; it is not ideal in all cases.

ALL ABOARD THE MERRY-GO-ROUND

Since the emergence of .NET from Microsoft's development labs, there have been clear signs of a shift back towards the richer Windows forms applications that are installed on individuals' workstations, laptops and even so called "smart clients", i.e. handheld computers.

It is not only borne from user frustration that rich client applications are becoming the desired application format, but since .NET forms now facilitate the benefits the browser based solutions provided, i.e. the ease of deployment and maintenance.

The term "no touch deployment" is now frequently associated with Windows form applications built in a .NET language such as Visual Basic or C#. This technology enables developers to build in the functionality for their software to determine for its self whether there is a newer version available and if so install it, without interaction from the user.

Combining all the potential features of a true Windows application, that harness the power of the SAS System whilst minimising the administration – creates a winning situation for all involved.

THE FUTURE'S BRIGHT

I'm keen not to miss the point of this paper, the point is of course the end user. Applications are written to facilitate repetitive tasks whilst being cost as cost effective as possible.

Applications written in the SAS System have traditionally been cumbersome, lots of mouse clicks have been required simply to select an item which drives a task or function. The ability to create meaningful interfaces than induce their users into working through the task at hand have been all but impossible to create.

The next generation has opened up the possibility of remote working with applications. Web based applications allow users to be located over a geographically wide location and yet still use the same software; which is easily maintained by administrators through a web server. Yet again, however the user has been compromised by minimal levels of real time responses to actions. For web users each request for new information has to be submitted to the web server which can take time to respond to; creating the perception of latency.

Now the user can have an application that resides on their computer making use of the local processing power to make real time responses for user requests. Furthermore such applications can include all the rich features of true desktop applications.

If you consider any feature you like in your favourite desktop software – this can be reproduced in any bespoke windows application.

There are several possibilities of course for building applications that interact with SAS. The rich application possibilities are considered in this section.

ENTERPRISE GUIDE V2

Enterprise Guide is the interactive front-end alternative from SAS Institute. Its second version, designed to be released with V9 of SAS supports add-ins. This means you can create your own form based application which then sits inside Enterprise Guide on users desktops.

This would be an approach where the application is simple relative to the level of every day use of Enterprise Guide. It will allow users to have a consistent look and feel to their environment.

Such add-ins can be written in .NET languages; e.g. Visual Basic or C#, as well as the previous versions of COM languages.

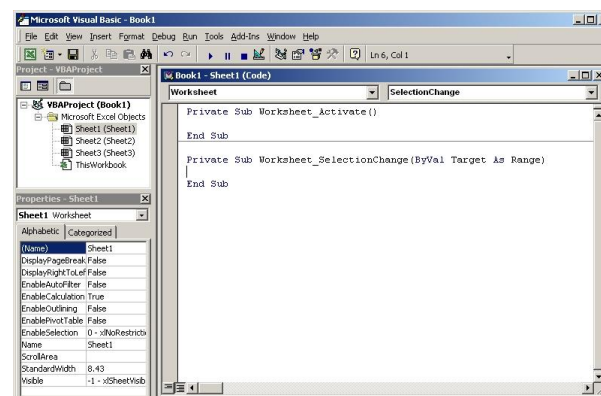
CONTAINER APPLICATIONS

Applications that reside in other software; similarly to the add-ins with Enterprise Guide; however the custom application is coded in Visual Basic for Applications (VBA) which must be supported by the host application.

VBA supporting applications are not only restricted to Microsoft software. Business objects is a popular business intelligence tool, which also supports VBA. Professional software developers can build a link from Business Objects into SAS that drives slice and dice reporting from SAS data.

No additional programming language or environment is purchased as code is written directly into the VBA editor. For example Microsoft Excel 97 and newer accesses the VBA environment through the ALT+F11 keystroke, as shown in Figure 4 below.

Figure 4: The VBA Environment in MS Excel 2000



Such examples of building an application inside a Microsoft Office application are appropriate when, for example, the Excel workbook must send its data to a SAS process and then again receive the results of that SAS job.

WEB APPLICATIONS

The possibilities with .NET web development and interoperating with the SAS System, go beyond anything possible with classic ASP and to some extent SAS Institutes application development environment AppDev Studio which primarily uses JSP's.

For example, the ability to update data row by row in a web page is a feature not supported by SAS Institute's own data components until the release of AppDev Studio that comes with SAS V9.1.

This is a feature that can be dropped into an application today with ASP.NET and a relatively simple OLE/DB connection to the SAS data set containing the data.

As discussed above, I believe that web applications are best suited to scenarios where users are geographically widely dispersed and the number of steps required from one end of the application of another are minimal and the level of use is not constant.

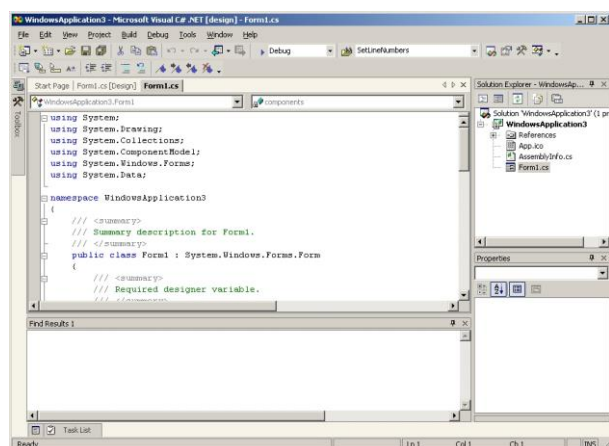
From experience of using both SAS Institute's AppDev Studio and Microsoft's Visual Studio my conclusions are that whilst AppDev Studio is quick to build connections into the SAS System, Microsoft Visual Studio is the more productive environment for building web and form based applications. This is because of several features such as intellisense in the code editor and the availability of components for Visual Studio languages.

WINDOWS FORM APPLICATIONS

Traditional form based applications are the application that give the user the most functionality, interactivity and stability. Furthermore the designs of interface are progressively becoming more intuitive for users – such as the styles seen in Windows XP operating system, Microsoft Money 2000 and newer.

Visual Studio .NET (Figure 5) provides developers with a highly productive environment from which to build such applications.

Figure 5: Microsoft Visual Studio .NET



Such applications can of course connect to remote data sources; even the business logic layer where SAS resides its self can be remote – but the possibility allows for data islands to be held locally such that input data or results are processed locally and feedback to the user appears instantly.

The significant advantage is that now users can be remotely dispersed however with a connection to a central server, administrators can maintain the software by releasing updates such that users are uninterrupted as software updates.

One further possibility not discussed so far in this paper is the employment of web services. Web services can be thought of as an application that resides on a web server however can interact with numerous clients through standard internet communication protocols; e.g. transmitting data in XML, SOAP or WDSL.

THE FUTURE'S .NET

There are several possibilities presented in the discussion here, where the main objective is to ensure the application designer implements the best solution for the end user; rather than the system maintainers.

Microsoft's .NET technology allows developers the best of both rich intuitive user interfaces, coupled with low overhead deployment. These applications can easily interoperate with the SAS System to harness its data manipulation and modelling strengths; whether SAS resides locally with the rich application or remotely on a server.

As applications can be designed in logical layers as discussed above, an application can share common functionality between the various interfaces provided. For example, if a web browser front-end is deployed in addition to a desktop front-end, each interface can connect to the same shared application functionality. As Visual Studio .NET also provides cost free integration to Oracle and SQL Server databases further data integration possibilities may become possibilities to SAS users without the appropriate SAS/ACCESS modules.

Whatever the choice employed users should be presented with an environment which facilitates the task in hand, which is built of consistent features and one which users are keen to work with throughout the lifetime of the application.

CONCLUSION

Those who commission or build applications which harness the SAS System can consider the possibilities of their applications that go beyond the use of SAS/AF.

Consideration should be given to dividing applications into logical layers for the functionality performed by SAS software and the functionality required by a new front-end environment. Further layers could be identified for data access.

The new application does not have to be a compromise for either the end user nor the administrator, as high levels of user interactivity can be gained through Windows forms applications built in .NET that allow no touch deployment for ease of maintenance and connectivity to remote servers through standard network or internet protocols.

Finally the point is the user. Solution architects and developers should never forget that the application will facilitate a business need of the user. There is a lesser need than ever to comprise on functionality for the sake of deployment and interoperating with the SAS System... hence for the users of SAS applications, the future is certainly bright – and the future may well involve .NET.

CONTACT INFORMATION

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

Author Name:	David Shannon
Company:	Amadeus Software Limited
Address:	Orchard Farm, Witney Lane, Leafield, Oxon OX29 9PG
Phone:	01993 878287
Fax:	01993 878042
Email:	david.shannon@amadeus.co.uk
Web:	www.amadeus.co.uk

Version 1.0

© Amadeus Software, 2003.

The Amadeus logo is a trademark of Amadeus Software Ltd, Witney, Oxfordshire, England.

Microsoft products are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

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.