Paper 1269-2014

The Many Ways of Creating Dashboards Using SAS®

Mark Bodt, Knoware – The Knowledge Warehouse Limited, Wellington, New Zealand

ABSTRACT

For decades, SAS® has been the cornerstone of many organizations for business reporting. In more recent times, the ability to quickly determine the performance of an organization through the use of dashboards has become a requirement.

Different ways of providing dashboard capabilities are discussed in this paper: using out of the box solutions such as SAS® Visual Analytics, SAS® BI Dashboard, through to alternative solutions using SAS® Stored Processes, batch processes and SAS® Integration Technologies. Extending the available indicators is also discussed using Graph Template Language, KPI indicators provided with Base SAS®, as well as alternatives such as Google Charts and Flash objects. Real-world field experience, problem areas, solutions and tips are shared along with live examples of some of the different methods.

INTRODUCTION

Field experience has shown that out of the box dashboard solutions provide an environment to quickly build dashboards, however they may not achieve the specialist needs of a business. Often there are requirements that do not fit standard indictors or dashboards.

The concept of a dashboard is often new to users. So, before embarking on the dashboard journey, it pays to stand back, carefully assess the purpose of the project by asking questions such as: What are they trying to achieve? Is it a dashboard that is required, or instead an exploratory tool? How often will the users look at the dashboard? How many users are there? What are the performance requirements? How often is the data updated? How accurate is the data? How does the data behave and change throughout the year?

The answers to these questions can influence whether an out of the box solution, an alternative solution or a combination best meets the dashboard needs. The outcome of these questions may highlight that more than one solution is required to meet vastly different user needs. For example a small group of users may need to occasionally infinitely explore the data, select filters and drill down many levels. This in effect is more of an exploratory requirement rather than a dashboard. Another large group may need to look at a dashboard on a daily basis and in this case productivity is the key. For the dashboard to be successful for this second group, the performance needs to be snappy and there should be minimal, if any, selections required by the user. Thinking of productivity for the second group: instead of getting the users to take the time to log in to a dashboard on a daily basis, would it be better to batch create dashboard type reports that are emailed to them so they are immediately available to them when they arrive in the morning?

There are quite a range of technology solutions to provide users with “Interpretation at a glance”.

WHAT IS A DASHBOARD?

A dashboard, like the dashboard in your car should give you sufficient information with a momentary look to allow you to make a decision. Car dashboards typically are simple in design, have a carefully selected, limited range of colors that will immediately indicate where all is well, or there is a problem. They have sufficient information, but not too much information. For example there may be an oil light that lights red when there is a problem with the oil pressure. That is all the driver needs to make a decision, they don’t need to know the exact oil pressure, simply whether there is a problem or not.

Some key aspects of a dashboard are:

- The information can be interpreted with a momentary look
- They are typically very visual, because good use of images make it easier to interpret the information
- Often web based, but not necessarily so
- Fits onto a single page that can be viewed without scrolling
- Contains key information required to meet an objective.

TIPS FOR A SUCCESSFUL DASHBOARD PROJECT

The author's experience has uncovered a number of areas that require careful attention when undertaking a dashboard project.

DESIGN

The number one rule is to *keep it simple*. Avoid visual clutter, use clear and concise indicators and be careful when using color. Be consistent with the colors, design and the way that data is displayed. Developers are often tempted to be overly clever, or show too much information simply because the technology allows for many colors, features or detailed data. Avoid pointless animations, visual effects such as drop shadows, glossy skins which visually add to the clutter of an indicator.

Developers may be tempted to show their skills and make the dashboard flexible by adding cascading dropdowns. This turns the dashboard into more of an exploratory tool than a "can be interpreted with a momentary look" dashboard. The user has to make multiple selections, each requiring a server round trip. This can affect the performance of the dashboard and server load.

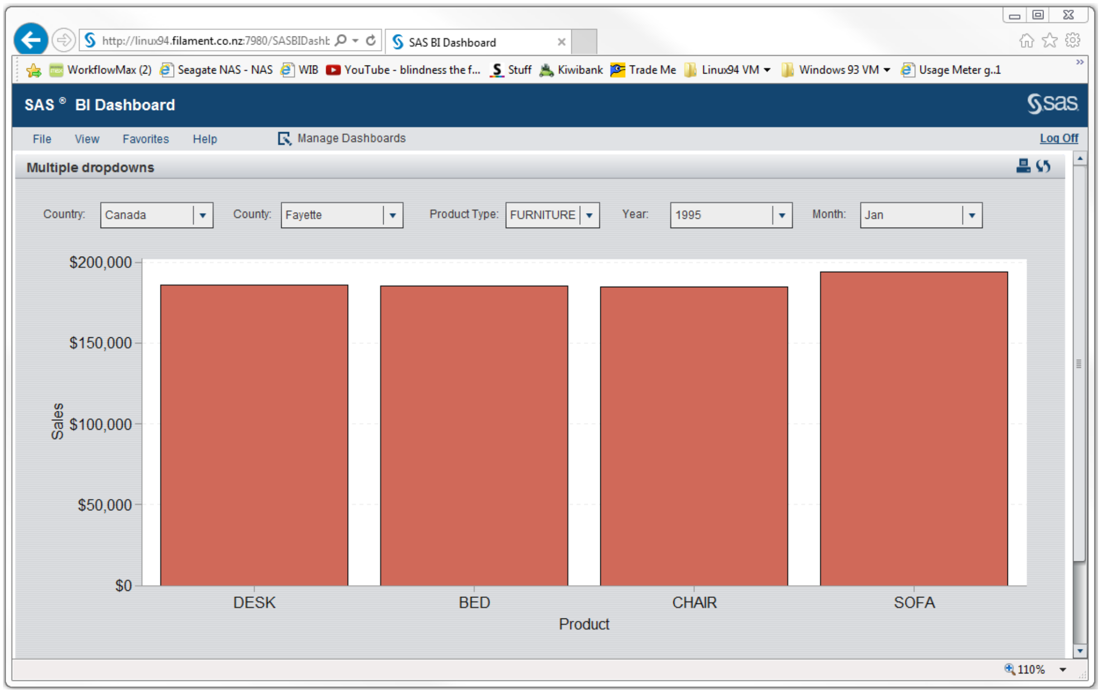


Figure 1. Cascading dropdowns

Two books on the topic of Dashboard Design and Visual Design are "Information Dashboard Design – Displaying data for at-a-glance monitoring" and "Show Me the Numbers – Designing Tables and Graphs to Enlighten". Both are by Stephen Few and excellent books that expose common problems in dashboard design and information visualization.

DATA

Dashboards typically show highly summarized and derived data. The dashboard may be used to make strategic decisions, so it is essential that the data is correct and can be relied upon. Much of the data quality responsibility rests with the data warehouse developers; however it is expected that from time to time, the value of an indicator will be questioned. So it is prudent to have a series of reports – such as SAS® Web Report Studio reports that allow the validation of the data and the tracing of problems. These reports are not just for the user acceptance testing phase of development, but should be available whenever an indicator value is questioned. Quick answers to these queries are often required, so it is important to have these reports ready to go.

The data that is being shown in dashboards may not have been available before. The exact behavior of the data may be quite different from what is expected by the user. It pays to expect the unexpected when live data starts to come through.

Figure 2 below shows what the client was expecting. The sales on the weekend were less as this reflected only internet sales as the physical stores were closed. Figure 3 shows the actual results. The client responded that there was an error in the dashboard as there cannot be negative sales. However the validation reports quickly identified that the problem was business process related and not dashboard related: The internet sales on Sundays were usually quite low; on some Sundays staff would work overtime and process all the credits and policy cancellations for the previous week and as a result the credits outweighed the sales and resulted in a negative value.

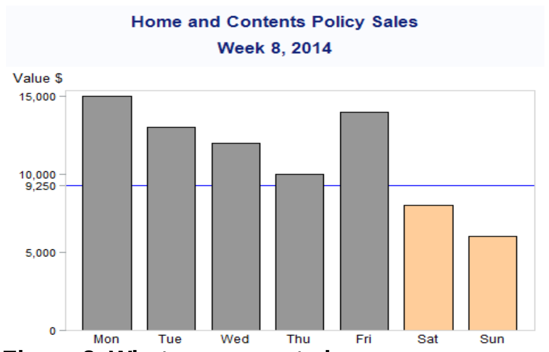


Figure 2. What was expected

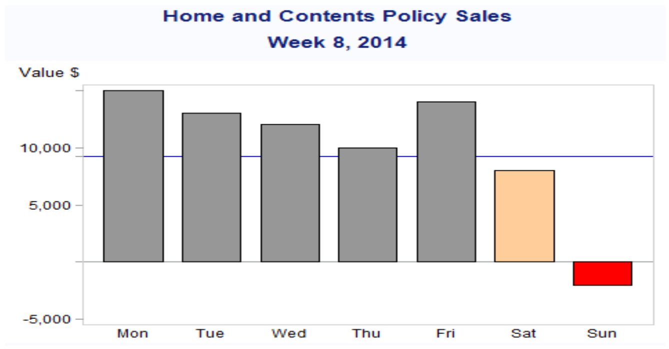


Figure 3. The actual results

These data anomalies can be difficult to predict and may only occur under certain conditions or at a particular time of the year.

The dashboard is often the first port of call for blame, when the real cause is further back in the data chain. A common issue has been that the dashboard data is not updated due to delays in updating the data warehouse. It is a good idea to include an ‘as at’ date on the dashboard e.g. Data as at 22:38 06 Feb 2014. In addition, the dashboard administrator should have a simple report to hand listing when the data was last updated for the various contributing data sources.

PERFORMANCE

A common requirement is that a dashboard is observed frequently, often daily. Whether the dashboard is viewed by executive management or frontline staff, a sluggish or inefficient dashboard will cause frustration and will detract from the project success. Performance is not just the time that it takes for a dashboard to display, but consists of the total time required for the user to get the information they require including:

- Log-in time
- The number of clicks needed to select the dashboard to display
- The dashboard response time
- Interpretation time.

Performance can be improved through the use of good visual design, dashboard design and technology choices:

- Implementing Single Sign on (SSO) can eliminate the need to prompt the user for credentials while retaining the security
- Having data ready to go – pre-summarized and joined can improve indicator rendering performance
- Minimise the number of selections or clicks that the user needs to make to get to the dashboard they require. Avoid dropdowns as these need two clicks to make a selection – one to open the dropdown and the second to make the selection. Where possible take the user directly to the dashboard view they require, rather than forcing them to navigate
- Consider the technical options. Do you need to create the dashboard on the fly for every user, every time they go to a specific page, or can the dashboard pages be created in advance using batch processes? Technology alternatives can significantly improve response time. Rather than creating the indicator images on the server, consider creating them on the client using Adobe® Flash® or other technologies such as Google Charts
- Ensure that the infrastructure is adequate for the expected demand
- Simple, clear no-nonsense indicators will mean that the data can be interpreted quickly.

MANAGE EXPECTATIONS

To users, dashboards are often surfacing new information and technology. Users and business analysts often do not have a background or aptitude in visual design. The dashboard technology capabilities may not be known – even an experienced BI developer may not know all the capabilities or short-comings of a particular indicator. Figure 4 shows an example of this. The user specified that a bar chart indicator was required, but it was not until they were shown the results that it became apparent to them that there was an additional requirement that the Y axis must be consistent every time the indicator displayed, and not change depending on the data values. Some indicators do not allow you to specify Y Axis ranges or increment values.

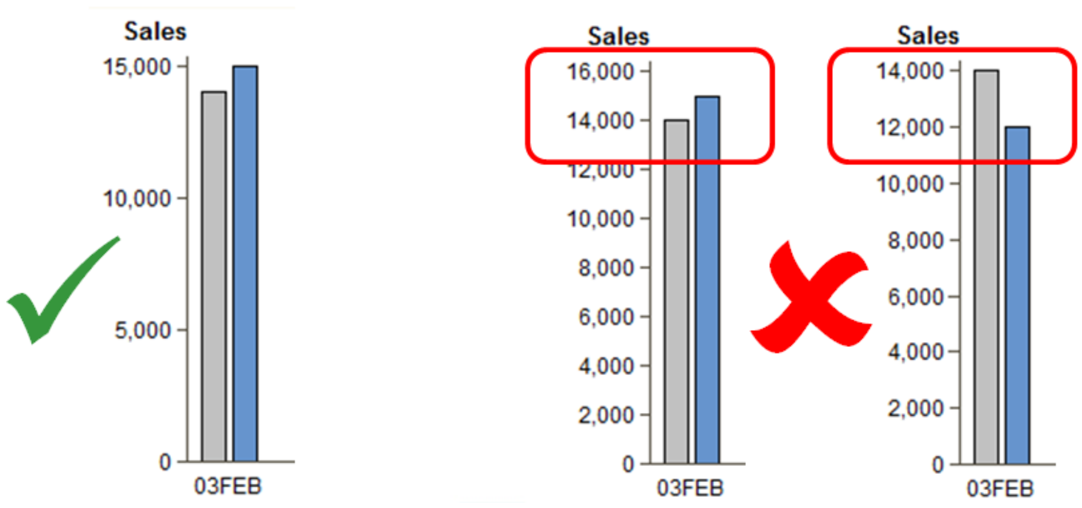


Figure 4. Indicators may not be able to meet users' subtle requirements

Data behavior, quality and quirks are often not known until the live data starts to come through. Each of these factors leads to a degree of uncertainty and risk.

In order to manage expectations and reduce risk, it is recommended to implement a rapid application development approach, whereby the developer builds the dashboard in small bites, testing and demonstrating this to the users, getting feedback, refining the design and applying the changes. This way any data issues, unexpected indicator behaviors and general expectation shortcomings can be rectified early in the project thereby minimizing the risk.

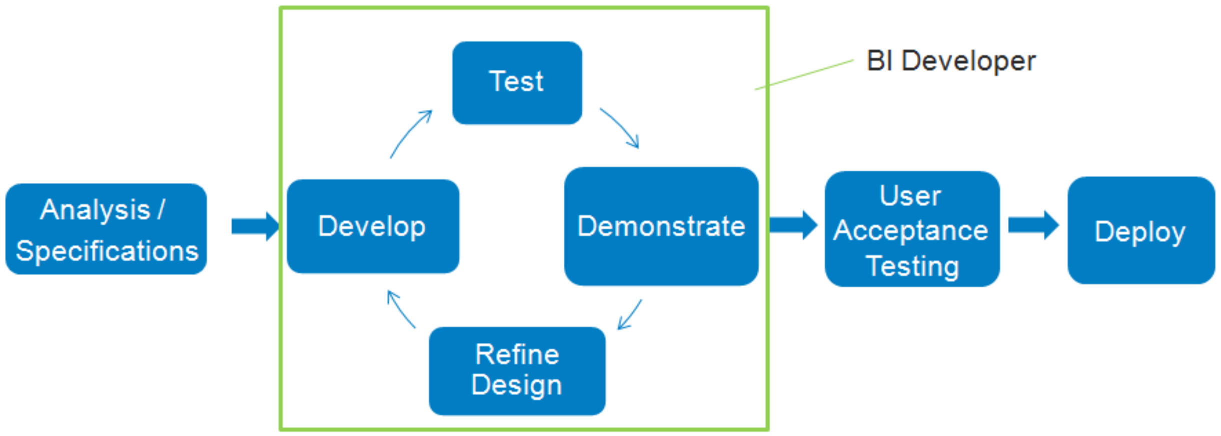


Figure 5. A rapid application development approach is recommended to quickly identify problem areas

TECHNICAL SOLUTIONS

Technical solutions fall into two categories:

1. Out of the box solutions such as SAS® Visual Analytics and SAS® BI Dashboard
2. Build your own solutions.

Each solution has pros and cons and these need to be considered when deciding on which approach to take. There are many papers and presentations at SAS Global Forum covering the out of the box products, so this paper will mainly focus on ‘build your own’ solutions.

OUT OF THE BOX SOLUTIONS

These could be compared to buying flat pack furniture. You assemble the furniture quickly, there is often some degree of customization that you can do, but while the furniture mostly meets your requirements or taste it is not 100% ‘you’. It does provide great value for money and is a quick solution.

SAS® Visual Analytics

SAS® Visual Analytics (VA) has been around for a couple of years. It appears to be the replacement for the SAS® BI Dashboard and SAS® Web Report Studio. VA provides comprehensive data exploration capabilities, reporting and also dashboard functionality. It uses new architecture which is designed to work with very large data volumes and as the name implies has good analytic capabilities. VA is web based. Reports and dashboards can also be viewed in tablet and smartphone apps. VA can be configured to a much greater degree than SAS® BI Dashboard. It is a comprehensive product. In VA 6.3 released in December 2013, new functionality is provided with the SAS® Visual Analytics Graph Builder that allows the building of customized indicators. The indicators are built using a combination of the existing indicators, either layering them – such as overlaying a step plot on top of a bar chart, or tiling charts into a lattice arrangement.

SAS® BI Dashboard

SAS® BI Dashboard has been around for a number of years. It is a web based application that is either stand-alone, or can also be surfaced through the SAS® Portal. A range of indicators are provided as well as tools to configure the data sources, indicators, ranges and the dashboard layout. Simple interactions can be set up between indicators as well as links to other BI content or external web pages.

BUILD YOUR OWN SOLUTIONS

This can be compared to custom furniture. You get exactly what you want, but it takes time and considerable cost for the skilled furniture maker to craft the furniture. Every want and need can be integrated in the final result.

While build your own solutions may release the developer from limitations, it also lands the developer much more responsibility such as having to implement security functionality, integration with SAS and every aspect of design. While there may be mechanisms that could aid with this, it is not built in as it is with the out of the box applications.

Not everyone has the out of box applications, or they have some specific requirements that the out of the box applications do not provide. In the next section various build your own indicator and dashboard options are discussed.

Customized Indicators using Proc GKPI

SAS/Graph® has Proc GKPI, a procedure that generates SAS® BI Dashboard like KPI images.

```
goptions reset=all device=javaimg xpixels=240  ypixels=200;

proc gkpi mode=raised;
     dial actual=.46 bounds=(0 .23 .46 .65 .79 1) /
       target=.9 nolowbound format="percent8.0"
       afont=(f="Albany AMT" height=.5cm)
       bfont=(f="Albany AMT" height=.4cm);
run;
```

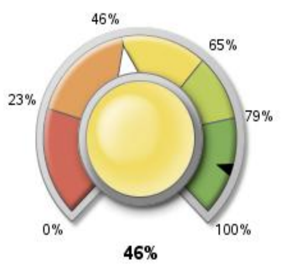


Figure 6. Example of Proc GKPI code and indicator

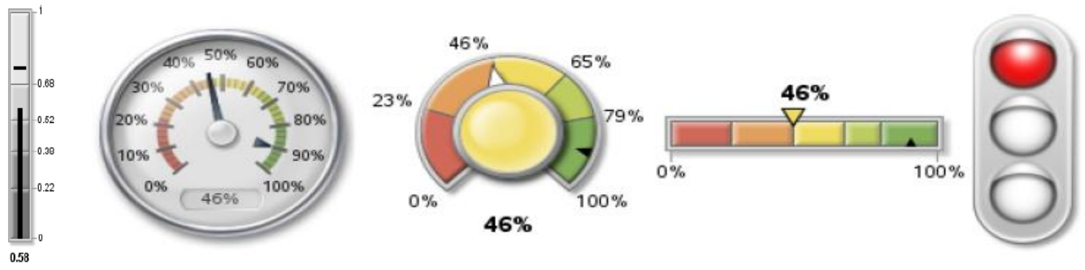


Figure 7. Other Proc GKPI indicator examples

There are a number of code samples on the SAS Support site: http://support.sas.com for building indicators and charts using SAS/Graph® code. Search on *Dashboard Sample*.

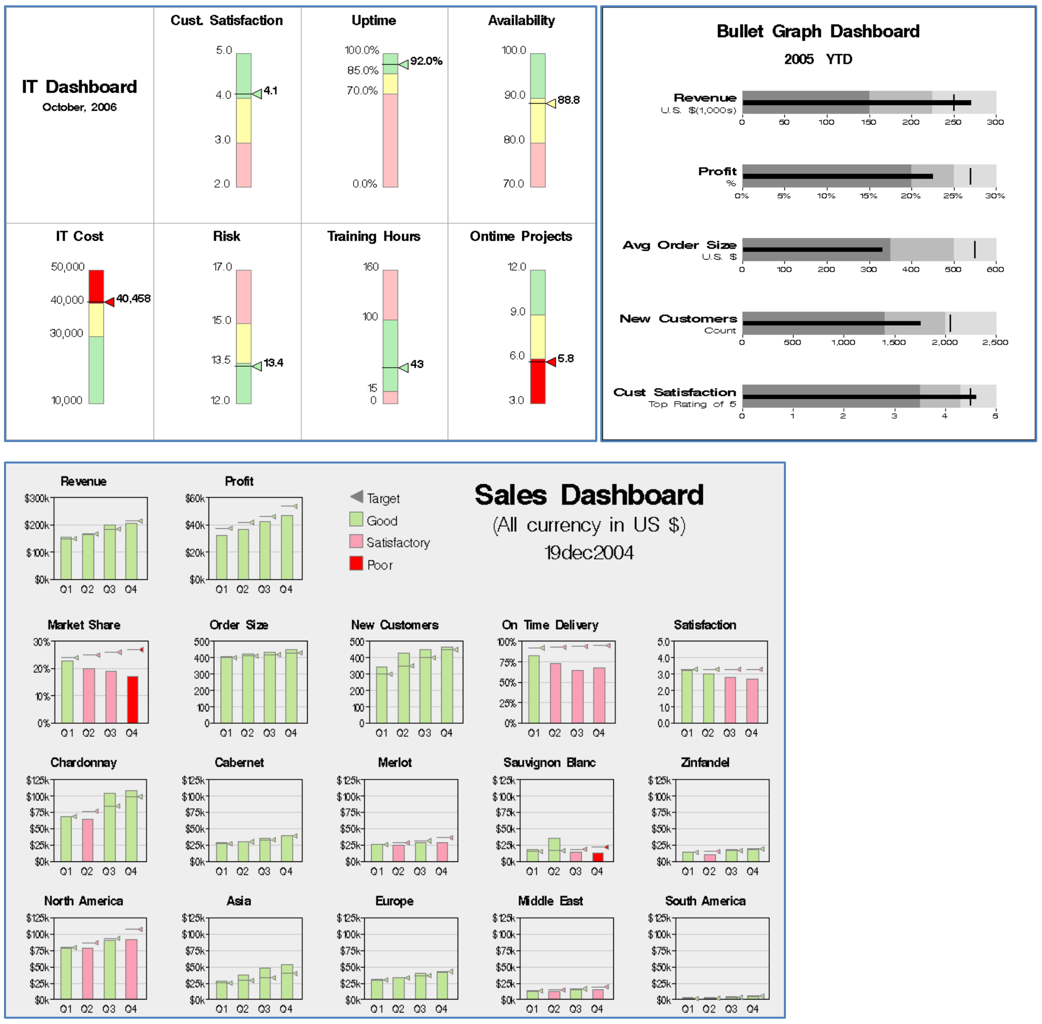


Figure 8. Various SAS Support site samples

It should be noted that the sample code and the other options such as GAnnotate and Data Step Graphics Interface (DSGI) are quite dated and better solutions may be found using SAS® Graph Template Language.

Customized Indicators using SAS® Graph Template Language

The author worked on a dashboard project where the client insisted that the dashboard was to have a Boston Matrix chart as one of the indicators. This chart is not available as one of the indicators in the SAS out of the box solutions and needed to be built. Graph Template Language (GTL) was used for this. GTL is a more recent addition to the SAS inventory of graphing tools. It is a powerful language that provides for the construction of charts either in layers or in a matrix or lattice layout. Creating a chart using GTL is a two step process. In the first step a template defining how the chart is to display is defined and stored. In the second step, the stored template can then be used and re-used to display charts using different data.

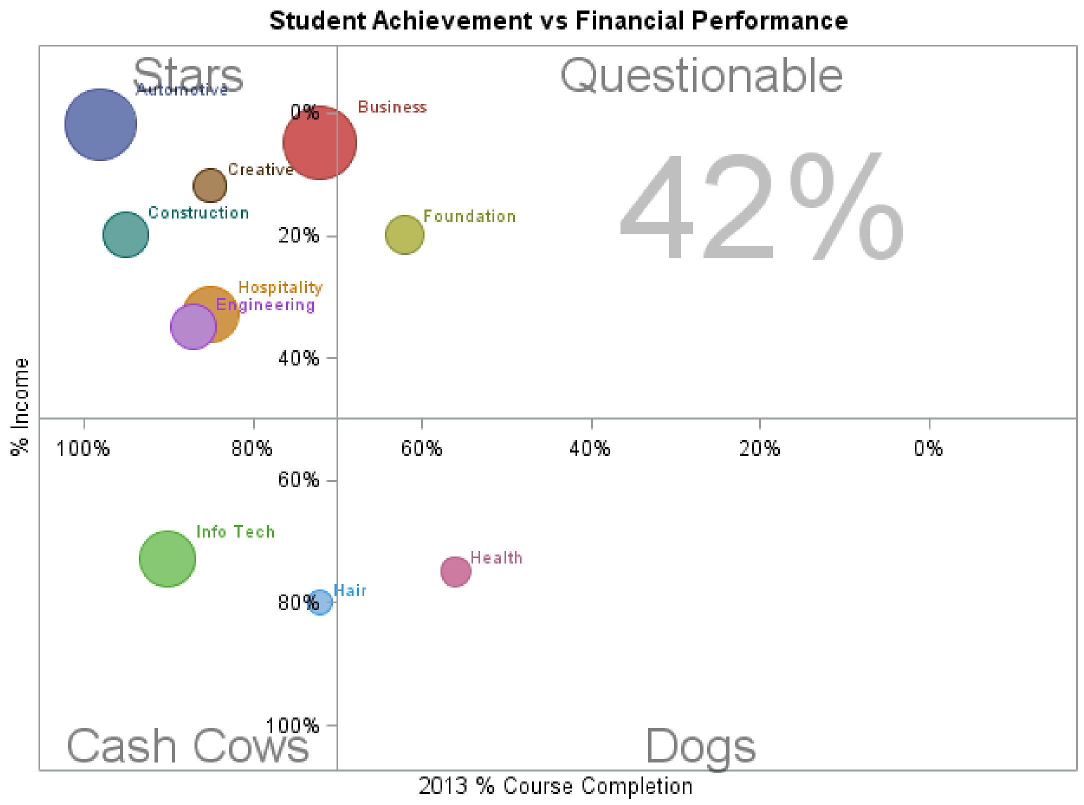


Figure 9. Example of the Boston Matrix chart that the client required (prototype)

In Figure 9, the chart consists of a bubble plot, with offset axes that have positive values listed in the reverse direction to what is standard. Large text labels marking each quadrant are annotated, as is a KPI value (42%).

The chart was to be re-useable with different data and some parameters needed to be passed to the chart – such as the KPI value.

```
* Refer to http://support.sas.com/kb/17/427.html regarding the following ODS
statement;
ODS PATH work.templat(update) sasuser.templat(read)
      sashelp.tmplmst(read);

proc template;
      define statgraph BostonMatrix;
            begingraph;
                  dynamic D_Title D_XAxisLabel;
                  mvar KPI;
                  entrytitle D_title;
                  layout overlay / xaxisopts=(label= D_XAxisLabel   name="xaxis"
                        reverse=True linearopts=(Origin=.7 viewmin=0 viewmax=1) )
                        yaxisopts=(label="% Income" reverse=True
                        linearopts=(Origin=.5 viewmin=0 viewmax=1) );
                        bubbleplot x=CourseCompletionPercent y=IncomePercent
                              size=GrossRevenue /group=school datalabel=School
                             name="BostonMatrix";
                  endlayout;
                  drawtext textattrs=(color=gray size=60pt) KPI / Transparency=.5
                        width=40 widthunit=percent anchor=topright  x=85 y=90;
                  drawtext textattrs=(color=black size=20pt) "Stars" /
```

```
                           Transparency=.5 width=40 widthunit=percent   x=17 y=96;
                    drawtext textattrs=(color=black size=20pt) "Cash Cows" /
                           Transparency=.5 width=40 widthunit=percent   x=17 y=7;
                    drawtext textattrs=(color=black size=20pt) "Questionable" /
                           Transparency=.5 width=40 widthunit=percent  x=65 y=96;
                    drawtext textattrs=(color=black size=20pt) "Dogs" /
                           Transparency=.5 width=40 widthunit=percent  x=65 y=7;
              endgraph;
       end;
run;
```

Figure 10. Code for the Proc Template

This code will only be superficially discussed as there are a number of SAS Global Forum papers that discuss GTL in detail. This prototype code shows examples of receiving both parameters (`dynamic D_Title D_XAxisLabel;`) and a macro variable (`mvar KPI;`). These are received at run time and applied to the template when it renders. The *layout* block defines the bubble plot and the reversed axes. The *drawtext* statements annotate the 'Stars, Questionable, Cash Cows, Dogs and KPI' labels. In this case the template is stored in the work library; however it could be stored in a permanent library for future re-use.

That was part one of the two step process. The second part is creating or rendering the chart by using the template and specifying the data, parameters and macro variable values.

```
%let KPI=42%;

proc sgrender data=BostonMatrixData template=BostonMatrix;
       dynamic D_Title='Student Achievement vs Financial Performance'
                     D_XAxisLabel='2013 % Course Completion';
run;
```

Figure 11. Code for the chart rendering

In figure 11, a macro variable *KPI* is defined and the rendering procedure *proc sgrender* is called. This specifies the data to use, the template to use: the template that was created and stored earlier along with values for the two parameters that were defined in the template (`D_Title` and `D_XAxisLabel`).

```
data BostonMatrixData;


Format GrossRevenue comma8. CourseCompletionPercent IncomePercent percent6.;
input School $16. @20 CourseCompletionPercent 8. @30 GrossRevenue 8.
       @40 IncomePercent 8. ;

/*
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567879
*/
datalines;
Automotive          .98        850000     0.02
Business            .72        900000     0.05
Construction        .95        300000     0.2
Creative            .85        150000     0.12
Engineering         .87        300000     0.35
Foundation          .62        200000     0.2
Hair                .72        50000      0.8
Health              .56        100000     0.75
Hospitality         .85        500000     0.33
Info Tech           .90        500000     0.73
;
run;
```

Figure 12. Code to generate sample data for the Boston Matrix

There are several resources where you can find out more about GTL:

- GTL Manuals available on support.sas.com
- Search the support.sas.com site with keywords *ODS Graphics* or *Graph Template Language*
- SAS Global Forum papers.

Making Customized Indicators Available to SAS Solutions

SAS Solutions such as VA and BI Dashboard can display customized indicators that have been built with SAS code. To do this, the code needs to be wrapped in a SAS stored process. A stored process consists of two parts: the SAS code (with some minor additions) and a metadata registration. The metadata registration contains various pieces of information such as where the code is located, which server the stored process is to execute on, prompts (parameters) to name a few. Once a stored process has been defined, it is available for display in many SAS client products such as SAS® Enterprise Guide, SAS® Add-in for Microsoft Office as well as Web based applications such as the SAS® Portal, SAS® Visual Analytics and SAS® BI Dashboard and the SAS® Stored Process Web Application. It is possible to hyperlink areas of a chart – for example the bubbles in the Boston Matrix, however this functionality may be limited in the BI Dashboard or Visual Analytics.

Creating a Dashboard Application using the SAS® Stored Process Web Application

The SAS® Stored Process Web Application provides a web based method of running stored processes. In the Boston Matrix example above, an image was created, however with SAS code it is possible to create the image as well as write out HTML code. The HTML written out can be fully customized and therefore could include drop downs, radio buttons, drill down links etc., but also embedded script or objects such as JavaScript, JQuery, AJAX, Adobe® Flash® objects or other content.

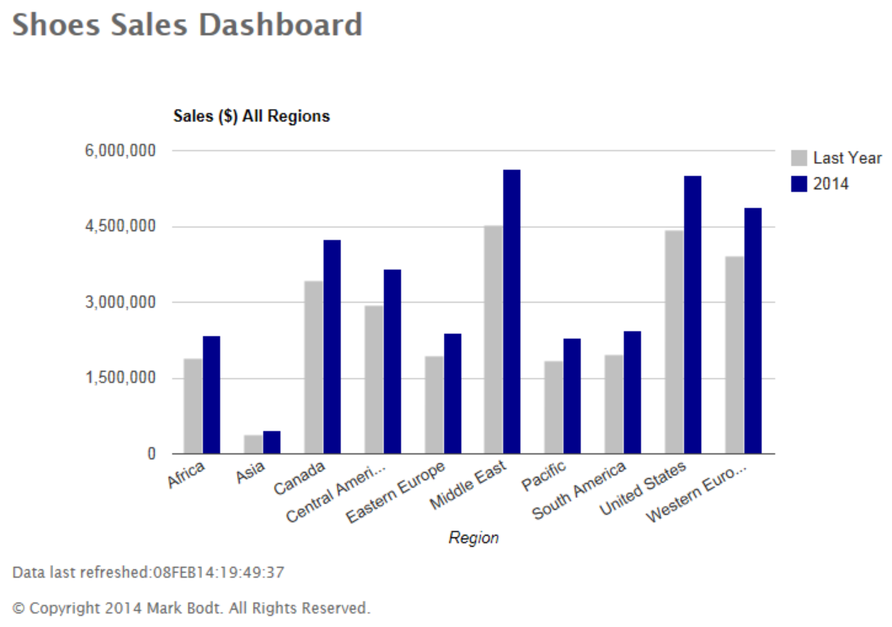
This capability means that complete, complex web applications can be built from scratch using SAS stored processes. The client side scripts and objects – JQuery, Adobe® Flash® etc. provide an enhanced user interface, with SAS providing the power of sourcing the data, summarization and any analytics. Users have built their own complex dashboard applications using the SAS® Stored Process Web Application.

For more information on this topic see:

- SAS Global Forum papers – see references at the end of this paper
- SAS Stored Processes Developers Guide available from support.sas.com
- *The 50 Keys to learning SAS Stored Processes* by Tricia Aanderud and Angela Hall. A great book that starts out with a simple stored process, building up to stored processes for use with web applications.

Customized Indicators using Google Charts or Flash Objects

The previous section discussed how SAS can generate HTML code that can include script or object definitions such as for Adobe® Flash®. This capability means that rather than creating dashboard indicator images in SAS®, code and data values could instead be written out and the indicator created on the client web browser using Adobe® Flash®. There are vendors that sell ready-made Adobe® Flash® based indictor objects that will display slick indicators with a little code and data. There are other free tools such as Google Charts.


Shoes Sales Dashboard

Data last refreshed:08FEB14:19:49:37

© Copyright 2014 Mark Bodt. All Rights Reserved.

Figure 13. Example of the web page showing a Google Chart

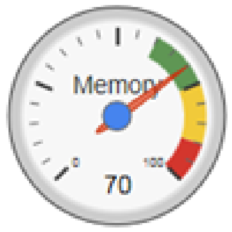
```
<html>
<head>
       <script type="text/javascript" src="https://www.google.com/jsapi"></script>
       <!-- Styles -->
       <link rel="stylesheet" href="/styles/SGF.css" type="text/css" />
       <link rel="stylesheet" href="/styles/cwtelements.css" type="text/css" />
       <!--[        if gte ie 5.5000]>
       <link rel="stylesheet" type="text/css" href="/styles/ie.css" />
       <![endif]-->

       <base href="http://www.markbodt.com" target="_self">
       <script type="text/javascript">
           google.load("visualization", "1", {packages:["corechart"]});
           google.setOnLoadCallback(drawChart);
           function drawChart() {
           var data = google.visualization.arrayToDataTable([
           ['Region', 'Last Year',{ role: 'style' },'2014',{ role: 'style' },'Link'],
           ['Africa ', 1874070 ,'#c0c0c0', 2342588,'#0020c0','/SGF/Region/Africa.html'
],
           ['Asia ',  368185 ,'#c0c0c0',  460231,'#0020c0','/SGF/Region/Asia.html' ],
           ['Canada ', 3404570 ,'#c0c0c0', 4255712,'#0020c0','/SGF/Region/Canada.html'
],
           ['Central America/Caribbean ', 2926202 ,'#c0c0c0',
3657753,'#0020c0','/SGF/Region/Central_America_Caribbean.html' ],
           ['Eastern Europe ', 1915952 ,'#c0c0c0',
2394940,'#0020c0','/SGF/Region/Eastern_Europe.html' ],
           ['Middle East ', 4505423 ,'#c0c0c0',
5631779,'#0020c0','/SGF/Region/Middle_East.html' ],
           ['Pacific ', 1837435 ,'#c0c0c0',
2296794,'#0020c0','/SGF/Region/Pacific.html' ],
           ['South America ', 1947826 ,'#c0c0c0',
2434783,'#0020c0','/SGF/Region/South_America.html' ],
           ['United States ', 4403189 ,'#c0c0c0',
5503986,'#0020c0','/SGF/Region/United_States.html' ],
           ['Western Europe ', 3898400 ,'#c0c0c0',
4873000,'#0020c0','/SGF/Region/Western_Europe.html' ],
           ]);
```

```
          var view = new google.visualization.DataView(data);
          view.setColumns([0,1,2,3]);

          var options = {
          title: 'Sales ($) All Regions',
          hAxis: {title: 'Region', titleTextStyle: {color: 'Black'}},
          series: {0:{color: '#c0c0c0', visibleInLegend: true},         1:{color:
'darkblue', visibleInLegend: true}}
          };

          function selectHandler(e)     {
          window.location = data.getValue(chart.getSelection()[0]['row'], 5 );
          };

          var chart = new
google.visualization.ColumnChart(document.getElementById('chart_div'));
          google.visualization.events.addListener(chart, 'select', selectHandler);
          chart.draw(view, options);
          }
      </script>
</head>

<body>
     <h1>Shoes Sales Dashboard</h1>
     <p></p>
     <div id="chart_div" style="width: 750px; height: 400px;"></div>
     <p>Data last refreshed:08FEB14:19:49:37</p>
     <p>© Copyright 2014 Mark Bodt. All Rights Reserved.</p>
</body>
</html>
```

Figure 14. HTML example of embedding a Google Chart

The prototype HTML in Figure 14 is generated using Base SAS code. Data was sourced from sashelp.shoes and summarized. The data values to use with the Google Chart can be seen in the code – country names, along with values, display colors and drill down hyperlinks.

The advantage of this is that images do not need to be generated on the SAS Server, nor transmitted across the network. This can greatly increase performance and minimize network traffic.



There are some 20 or 30 different chart types and indicators including gauges such as the one shown in Figure 15. These are available for free. URL: https://developers.google.com/chart/

There are many other charting tools using JavaScript libraries. Google search on: *javascript chart*.

Figure 15. Example of a simple Google Chart indicator

Batch Generated Dashboards

A common scenario is that dashboard data is refreshed nightly. The dashboard pages are the same for groups of users, yet every time a user visits a dashboard page, the page is built on the fly. There is always an overhead building on the fly, but provided the hardware is well provisioned and there are not too many users then this will work fine.

An alternative approach is to follow the overnight data refresh with a batch process that creates a series of linked HTML web pages. This can be quite successful, particularly if you incorporate using client side charting methods such as Adobe® Flash® objects or the Google Charts described above. Lightning fast performance can be expected using this method as there is no sever side processing at the time the page is requested.

This method is also suited where there is to be strict security demarcation between the web server and the internal network: for example, where the dashboard may be made available to the general internet public.

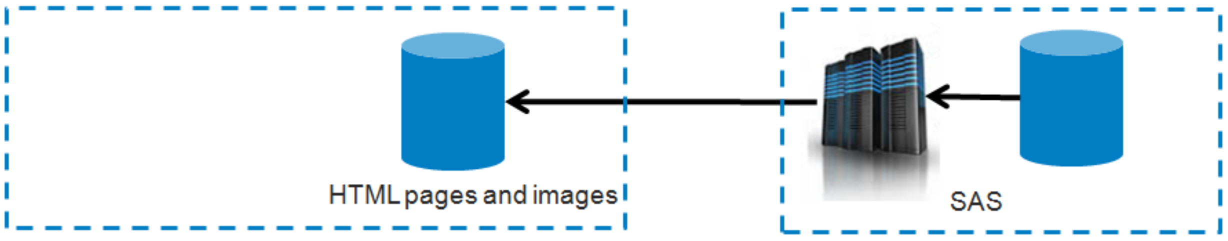


Figure 16. Part one of Batch Generated Dashboards

In Figure 16, as part of a batch process – for example the overnight batch, a Base SAS program generates a series of html web pages. These may be hyperlinked to provide drill-down. The pages are written out to an http server. There are a number of ways of writing html pages out to the http server. This could be achieved by writing to the local server, following up with a call to a shell script that carries out the copy, or one of the many filename engines could be used with a filename statement within SAS.

```
%macro BinaryCopy;
       /*Copies a file one byte at a time */

       data _null_ ;
              /* read / write one byte at a time */
            infile CB_In  recfm = N ;
              file CB_Out recfm = S ; /* S = streaming */
            input byte $char1. @@ ;
            put byte $char1. @@ ;
         run ;
%mend;

* copy page using FTP;
filename CB_In 'salesdashboard_NZ.html';
filename CB_Out ftp 'salesdashboard_NZ.html'cd="/opt/html"
         user='mark' host='delta3.filament.co.nz'
       pass="(sas002)TAABBTEJMYSECRETJHGFFFXMZ==";
         recfm=v ;

%BinaryCopy;

filename CB_In clear;
filename CB_Out clear;
```

Figure 17. Example of code used for FTP copy of a page to an http web server

The above code demonstrates how a file can be copied using FTP from within SAS code. FTP and secure FTP can at times be temperamental, so a more complex solution may be required that creates a check list of files to copy and the files are crossed off the list once they have been copied. Any files that fail to copy can then be re-attempted. With a mechanism such as this, it could also be split into groups of files and have multiple FTP sessions running concurrently to copy the files.

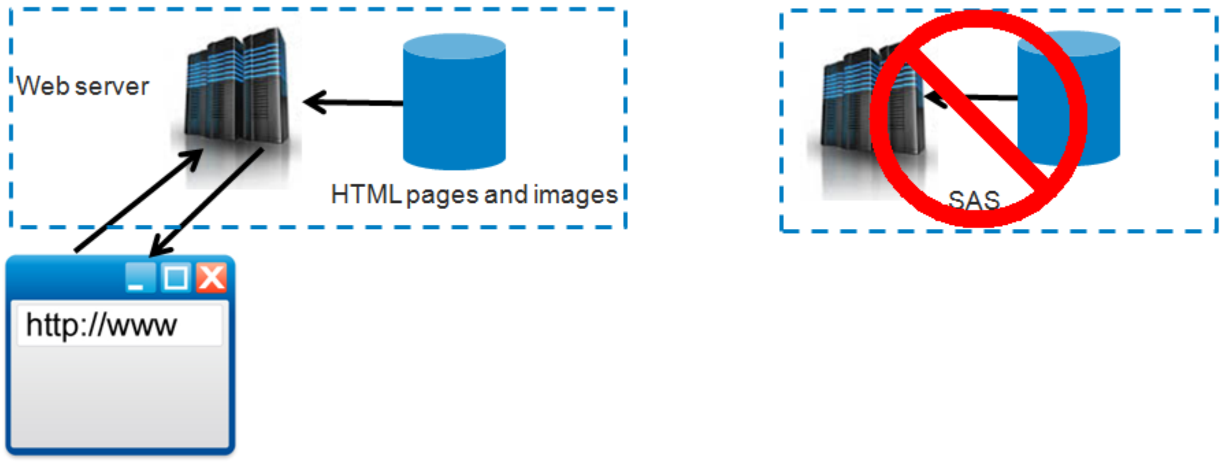


Figure 18. Serving of the dashboard static html pages

Once the batch process has completed, there is no further involvement by the SAS Infrastructure and the link between the http server and the internal network can be severed. The dashboard pages are simply served by the HTTP web server.

This method is good where there are a fixed number of known combinations of pages. The author has built applications that batch generated some 2000 web pages. The end-user performance using this method is excellent.

Creating Dashboards Using a Third Party Language

So far the dashboards created using SAS® have been discussed. Another option is to create the dashboard using a third party language such as Microsoft .net or Java™ and integrating these with SAS.

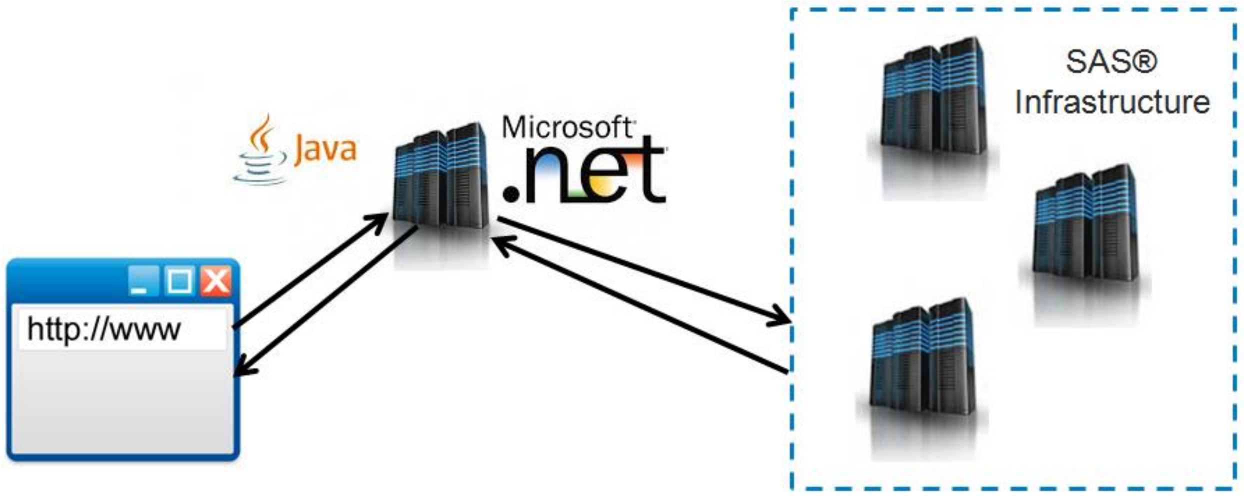


Figure 19. Using a third party language for the web application and integrating it with SAS

Microsoft .net or Java handle the web side of generating the dashboards and when integrated with SAS, SAS handles the data sourcing, summarization and any analytics.

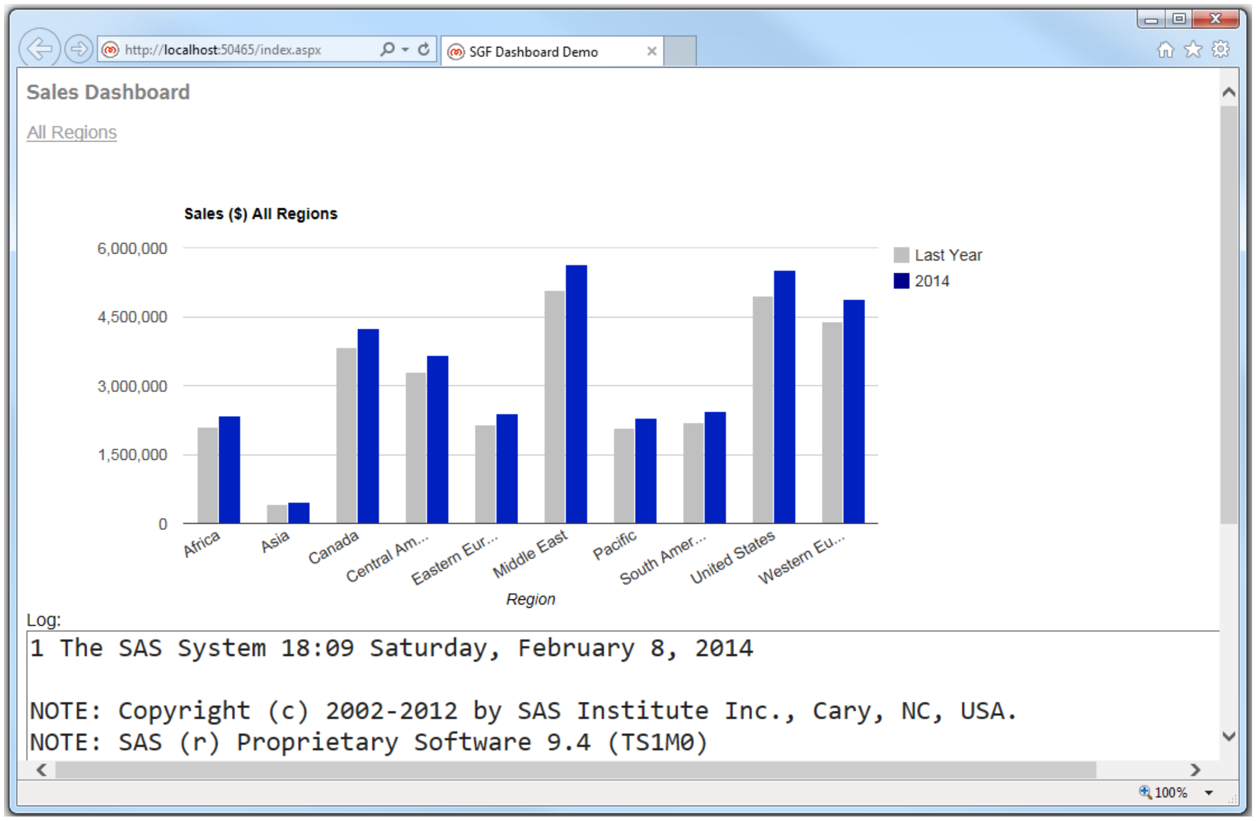
Thinking about this, this is similar to how client applications such as SAS® Enterprise Guide work. In the case of EG, it is a .Net application. The point and click operations in a project generate SAS code. The SAS code is sent to a SAS Server and results come back. This is all achieved using SAS® Integration Technologies which is a SAS provided Application Programming Interface (API) that opens SAS to other languages.

The API comes with some class libraries that need to be added to the project references. The application (.net in this case) creates instances of the classes, such as creating a workspace on the SAS server. SAS code can then be submitted to the server. In Figure 20, a snippet of code asp.net code is shown that sends some code to the SAS server. The SAS code is simple, but it contains an include statement that runs a Proc Means on SASHelp.shoes to summarize the data and SAS code to write out the results to the listing output.

```
            If (obPool Is Nothing) Then

                 ' The pool doesn't exist; create it
                 Dim obServer As New SASWorkspaceManager.ServerDef
                 Dim obLogin As New SASWorkspaceManager.LoginDef
                 obServer.Protocol = Protocols.ProtocolBridge

                 obServer.Port = 8591
                 obServer.MachineDNSName = "linux94.filament.co.nz"
                 obLogin.LoginName = "mark"
                 obLogin.Password = "mysecretpassword"

                 obPool = obWorkspaceManager.WorkspacePools.CreatePoolByServer("Test", _
                              obServer, obLogin)
             End If

             ' Now the pool is created, get a Workspace from it:
             Dim obPooledWorkspace As SASWorkspaceManager.PooledWorkspace
             obPooledWorkspace = obPool.GetPooledWorkspace("", "", 10000)

             ' Note that from this point on, the code is the same in both the COM+
             ' and the Integration Technologies pooling.
             Dim obSAS As New SAS.Workspace
             obSAS = obPooledWorkspace.Workspace


             SASLanguageService = obSAS.LanguageService


             SASLanguageService.Submit("%let Type=" & strSummaryType & ";" & _
                                       "%let Region=" & strRegion & ";" & _
                                       "%let Subsidiary=" & strSubsidiary & ";" & _
                                       "options source2;" & _
                                       "%include
'/opt/sas/SGF_Presentations/ASPDotNet/ASP_JavaScript.sas';")
```

Figure 20. ASP.net snippet of code

The workspace object has string arrays that contain the SAS log and also the SAS list output. In this prototype, the list output is used to generate the JavaScript code to display a Google Chart. For test purposes, the log is also retrieved and displayed in a selection list on the web page. The results are shown in Figure 21. It should be noted that in this example, the ASP.Net application is hosted locally on a Windows machine, but the SAS Server is running on a SUSE Linux machine.



```
1 The SAS System 18:09 Saturday, February 8, 2014

NOTE: Copyright (c) 2002-2012 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.4 (TS1M0)
```

Figure 21. Example of ASP.net generated page

The .aspx page generated is such that clicking on a bar sends the selection back to the server and the page is refreshed with the drill-down page. Even though this requires a server round trip and the associated SAS program processing, the prototype runs very fast.

SAS is open in other ways too. The Open Metadata Interface (OMI) gives programmatic access to SAS Metadata which means that an external application can query the SAS Metadata to determine server connections, get user information or any other item stored in metadata. Furthermore, SAS JDBC and ODBC drivers allow third party products to read SAS Data.

Resources:

- SAS ODBC and JDBC drivers can be downloaded from the SAS Support site support site http://support.sas.com/downloads/index.htm
- SAS 9.4 Integration Technologies – Windows Client Developer's Guide
- SAS 9.4 Integration Technologies – Java Client Developer's Guide
- SAS 9.4 Language Interfaces to Metadata Manual
- SAS Global Forum papers.

Hybrid Solution

While individual solutions have been discussed, there is no reason why the dashboard solution cannot be a hybrid of two or more solutions. For example if 80%of the dashboard is viewed by 80% of the users and performance is important, then batch generated may be the answer for this part of the dashboard. , Where more flexibility is required, dynamically generated dashboard and or indicators could be used.

CONCLUSION

A successful dashboard depends on many factors. With it being new to many users, careful planning and guidance along with an iterative development approach will help identify and iron out any issues up front. It is easy and tempting to incorporate more features than are required, to the extent that the application is no longer a dashboard or does not address the actual business needs.

Dashboards can be implemented in many ways and which option is used will depend on what is available, budget, expertise and business requirements. The final solution could use several technologies – employing the best suited option for each requirement.

REFERENCES

- Few, Stephen. 2013, Information Dashboard Design – Displaying data for at-a glance monitoring. Burlingame California: Analytics Press
- Few, Stephen. 2012, Show Me the Numbers – Designing Tables and Graphs to Enlighten. Burlingame California: Analytics Press
- Paper 257-2011 Using SAS® Stored Processes with JavaScript and Flash to Build Web- Based Applications, Philip Mason, Wood Street Consultants Limited, Wallingford, Oxfordshire, UK
- Paper 016-2011 Unleashing the power behind Stored Processes Ian Amaranayake, Amadeus Software Ltd., Witney, U.K.
- Paper 014-2009 Using AJAX and SAS® Stored Processes to Create Dynamic Search-Suggest Functionality Jeffery A. Fallon, Cardinal Health, Dublin, OH
- Paper 003-2013 Create Your Own Client Apps Using SAS® Integration Technologies Chris Hemedinger, SAS® Institute Inc., Cary, NC
- Google Charts https://developers.google.com/chart/

RECOMMENDED READING

- *Few, Stephen. 2013, Information Dashboard Design – Displaying data for at-a glance monitoring. Burlingame California: Analytics Press*
- *Few, Stephen. 2012, Show Me the Numbers – Designing Tables and Graphs to Enlighten. Burlingame California: Analytics Press*
- *SAS® 9.4 Graph Template Language Reference*
- *SAS® 9.4 Integration Technologies – Windows Client Developer's Guide*
- *SAS® 9.4 Integration Technologies – Java Client Developer's Guide*
- *SAS® 9.4 Language Interfaces to Metadata*
- *The 50 Keys to learning SAS Stored Processes by Tricia Aanderud and Angela Hall*
- *Building Business Intelligence Using SAS: Content Development Examples by Tricia Aanderud and Angela Hall*

CONTACT INFORMATION

Contact the author at:

Mark Bodt
Knoware
PO Box 10 541
The Terrace
Wellington 6143
New Zealand
mark.bodt@knoware.co.nz
www.knoware.co.nz