Paper 052-2008

Developing a Dashboard to Aid in Effective Project Management

M. Paige Borden, University of Central Florida, Orlando, FL
Maureen Murray, University of Central Florida, Orlando, FL
Ali Yorkos, University of Central Florida, Orlando, FL

ABSTRACT

Accurately tracking programming support activities as a team manager can be a complex task. The overall coordination of the programming resources, testing support and production moves was hampered by the crude reporting functionality available in the change management system. This management challenge was solved by the development of the CMS IR Technical Support Dashboard utilizing SAS® Enterprise Guide, SAS/GRAPH and SAS BI Dashboard 3.1 functionality. Working together with the management team, the dashboard was created to provide quick access in an easy to read graphical display of numerous key performance indicators. There is also functionality providing multiple alerts as defined by the management team (e.g. an application that stays in “user test” for more than 72 hours). The CMS IR Technical Support Dashboard has provided an improved management tool to effectively and efficiently coordinate the multiple projects and constituencies. This presentation will detail the development, testing and release of the dashboard, while providing coding details and end-user requirements.

INSTITUTIONAL RESEARCH

The mission of the Office of Institutional Research (IR) is to provide information of the highest quality and which is both timely and easily accessible, and to facilitate and enhance decision-making, strategic planning, and assessment at the University of Central Florida. One of our primary objectives is the task of developing and implementing a university data warehouse. This enterprise-wide data system facilitates the creation, access, and dissemination, by internal and external stakeholders, of institutional knowledge pertinent to the university. In collaboration with the data warehouse project stakeholder group, which represents a cross-section of key data users, the IR staff has established a data source of twelve years’ worth of student reporting data. This warehouse of information serves as the foundation for the development of a wide variety of reporting applications using SAS Business Intelligence software.

INTRODUCTION

The university developed a change management system (CMS) to coordinate our ERP implementation and maintenance. Within the past year, Institutional Research was granted access to the system to track the implementation and maintenance of our own reporting files, data warehouse development and on-line transactional web systems. Although this access greatly improved project tracking capabilities for our local project manager, the ability to create useful reports was not included in the CMS system functionality. Dashboard functionality was evaluated and implemented to provide a better management tool for the overall project management.

The initial design utilized SAS Enterprise Guide, SAS/GRAPH and PROC GREPLAY functionality and was completed prior to the release of the SAS BI Dashboard 3.1 software. Additional design enhancements were made under the same tool set. Within the past few months, the SAS BI Dashboard 3.1 software was released and continued development has occurred in that environment. Although the final production version was still under development at the time of this paper submission, the authors intend to include the final output and a demonstration of the latest design in SAS BI Dashboard 3.1 at the conference.

FUNCTIONAL REQUIREMENTS

OVERVIEW OF FUNCTIONALITY

Functional team members identified the data tables and data values that would need to be included to support the CMS dashboard. In addition, several key performance indicators (KPIs) were identified as items that needed detailed views to assist with project management. A mix of dials, graphs and/or gauges was desired, along with specific positive and negative alerts to identify specific successes or challenges. Additional requirements included a daily refresh and drill-down options for certain views.

The functional specifications identified 6 project team areas (Institutional Research, Data Warehouse, Reporting Database Service, University Analysis and Planning Support, Operational Excellence and Assessment Support and SACS) to include for the dashboard. These are the core project areas managed by IR personnel. Additionally, rules were specified regarding the amount of data to include on the dashboard using calendar guidelines for each status value (i.e. Development, User Test, Completed). A suggestion was made to the technical staff to limit the data selected for the dashboard rather than use the entire data set from the original tables. The limits were to be based on the project team areas and status fields.

The functional specifications also provided guidelines for field descriptions, such as displaying a 4-letter code versus long description values.

REQUESTED KPI DETAILS

Multiple KPIs were identified as crucial views to manage the project teams. The method of providing the information – whether dial, graph or gauge – was left up to the technical team. KPIs included a) number of jobs by status, b) number of jobs by area, c) number of jobs by area by status, and d) number of jobs by developer. Some of the indicators (a and b) are aggregate graphs; whereas the KPIs included in both c) and d) are multiple graphs, providing details for each status or developer.

REQUESTED ALERTS

Both positive and negative alerts were identified as valuable reviews to quickly identify the successes and challenges of the current projects. Requested alerts include e) completion status exceeds more than 25% of total jobs during the last 7 days, g) jobs in user test older than 7 days, h) jobs waiting for input status for more than 7 days, i) jobs due within the next 7 days, and j) more than 25% of jobs are still in assigned to developer status.

TECHNICAL DEVELOPMENT – SAS/GRAPH AND PROC GREPLAY PROCEDURE DESIGN

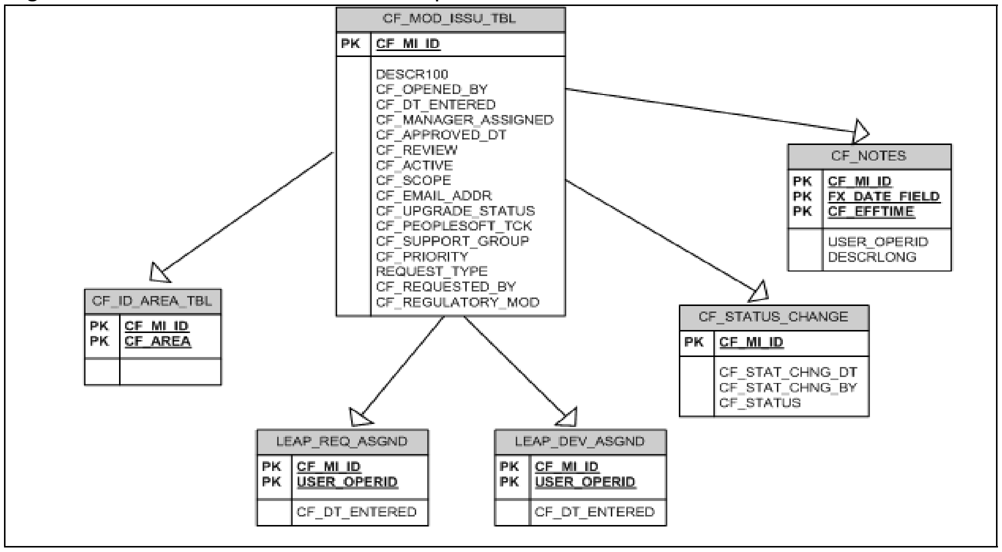
OVERVIEW

After receiving the functional specifications, two programmers began the development process. Although working independently, the steps of development were the same. The first step in the technical development was to identify the data source and develop the underlying database tables that were needed for the dashboard implementation. Once the database tables were determined, it was essential to carefully examine the structure of the table fields that were required to produce the KPI details and the calculated alerts. Upon creating all the intermediate data sets, multiple graphs were produced that displayed KPI details, as well as text-based alerts.

DASHBOARD TABLE DEVELOPMENT AND STRUCTURE

The first step to creating the dashboard table involved the development of a single de-normalized dataset that included all the required fields. There were a minimum of 5 tables that were accessed to create this dashboard. The source table structures and the relationships are shown in Figure 1.

Figure 1: Table structure and relationships



The second step was to create 5 different datasets from the single de-normalized dataset to feed the dashboard graphs (Table 1).

Table 1: Dashboard graphs and its associated datasets

SAS Graph	Chart Type	Dataset
Jobs by Status	Vertical bar	MODS_STATUS
Jobs by Area	Vertical bar	MODS_AREA
Jobs by Developer	Horizontal bar	MODS_DEVELOPER
Jobs by Status by Area	Pie charts	MODS_AREA_STATUS
User Test Status	Pie chart	MODS_USER_TEST

COMPUTED FIELDS FOR ALERTS

The functional specifications called for text-based "Alerts" for specific areas. Table 2 summarizes the computed alerts detailing the associated fields, formulas and format types.

Table 2: Computed fields

Computed Alert	Formula	Type
Completed status of all jobs	CF_STATUS(COMP) / CF_STATUS(TOTAL)	Percentage
Assigned to developer of all jobs	CF_STATUS(ASGN) / CF_STATUS(TOTAL)	Percentage
User test status older than 7 days	CF_STATUS(UTST) and CF_STAT_CHNG_DT < DATETIME()-7	Sum
Wait for input status older than 7 days	CF_STATUS(WAIT) and CF_STAT_CHNG_DT < DATETIME()-7	Sum

The following code was used to create an annotation that displays the "Completed status of all jobs" alert as designed by Programmer 1. If the percentage is less than 25%, the text alert is displayed in red; otherwise the text is displayed in green.

```
data data1_anno3;
   set work.MODS_STATUS;
   where cf_status="COMP";
   length function $8 color $12 style $20 text $50 title $50 styl $20;
   hsys='3'; when='a';
   function='label';
   style='swissb';
   size=5;
   xsys='3'; x=-57;
   ysys='3'; y=95;
   position='6';

      if percentage > .25 then color='cx006633';
      else color='cxcc0033';

      if color='cx006633' then
        text="Completed Status>25% of all jobs";
      else
        text="Completed Status<25% of all jobs";
      output;

      style='swissb';
      size=6;
     xsys='3'; x=-35;
     ysys='3'; y=108;
      position='6';
      text="ALERTS!";
      color='cxcc0033';
      output;
run;
```

SAS/GRAPH DEVELOPMENT

The graphs for this dashboard were created in SAS Enterprise Guide 4. In cases where EG4 was not capable of executing certain functionalities, the graph code was altered in order to take full advantage of the capabilities of the SAS code. Vertical and horizontal bar charts, as well as pie charts, were used to display information for this dashboard. Since the dashboard output file is a .gif file, the programmer needed to change the graph format from ActiveX to .gif during the initial graph development phase. This was done by navigating to Graph menu item under Tools→Options→Results in EG4 (failure to make this alteration will cause alignment issues on the resulting dashboard page since the output styles for ActiveX are different than the .gif output styles). The annotation functionality was used to display text alerts and was integrated with the graph code. An example of a graph code that was used with two annotations (as developed by Programmer 1) is listed below:

```
PROC GCHART DATA=WORK.MODS_STATUS anno=data1_anno3
;
    VBAR     CF_STATUS /
    SUMVAR=COUNT
FRAME       TYPE=SUM
    OUTSIDE=sum
     noframe
    COUTLINE=BLACK
    width=7
    RAXIS=AXIS1
    MAXIS=AXIS2
    GAXIS=AXIS3
    GAXIS=AXIS3
    name="plot1"
PATTERNID=MIDPOINT
   anno=data1_anno2;
RUN; QUIT;
```

PROC GREPLAY PROCEDURE

The last step in the graph development was to display all graphs in one dashboard page by using PROC GREPLAY procedure. The code used by Programmer 1 is detailed below.

```
PROC GREPLAY tc=tempcat nofs igout=work.gseg;
/* Define the areas of a custom dashboard template. */
   tdef ucf des='UCF'

/* First n-chart row (various metrics) */
   0/llx = 0   lly =  0
     ulx = 0   uly =100
     urx =100  ury =100
     lrx =100  lry =  0
   1/llx =30   lly = 60
     ulx =30   uly = 95
     urx =80   ury = 95
     lrx =80   lry = 60
   2/llx =60   lly = 60
     ulx =60   uly = 95
     urx =105   ury = 95
     lrx =105   lry = 60
   3/llx =0   lly = 15
     ulx =0   uly = 75
     urx =60   ury =75
     lrx =60  lry =15
   4/llx =60   lly = 0
     ulx =60   uly = 60
     urx =100   ury =60
```

```
        lrx =100   lry =0
    5/llx =0    lly =0
        ulx =0    uly =40
        urx =45    ury =40
        lrx =45    lry =0
    6/llx =1    lly =60
        ulx =1    uly =85
        urx =31    ury =85
        lrx =31    lry =60
;
   /* Replay the individual indicators into the appropriate areas in the custom
      dashboard template. */
   template = ucf;
   treplay
        0:titles 1:plot1 2:plot2 3:plot3 4:plot4 5:plot5 6:PLOT6;
run;
quit;
```

INITIAL DASHBOARD OUTPUT DESIGNS – SAS/GRAPH AND GREPLAY FUNCTIONALITY
The two competing dashboard designs were very similar in output. Programmer 1 created the output in Figure 2, while Programmer 2 created the Figure 3 output.

Figure 2: Programmer 1 dashboard display

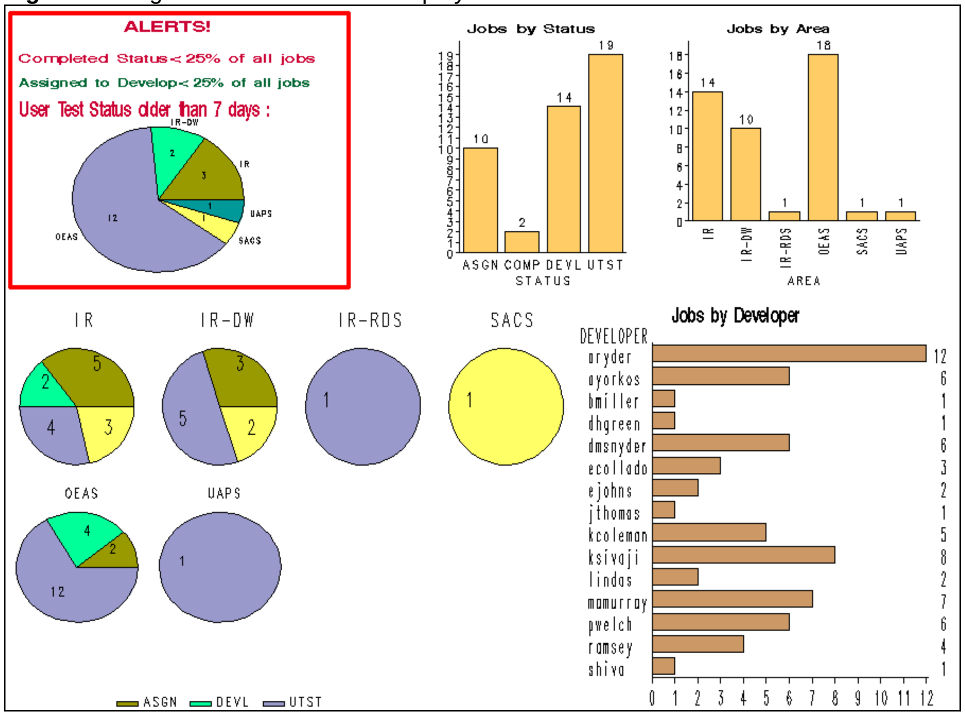
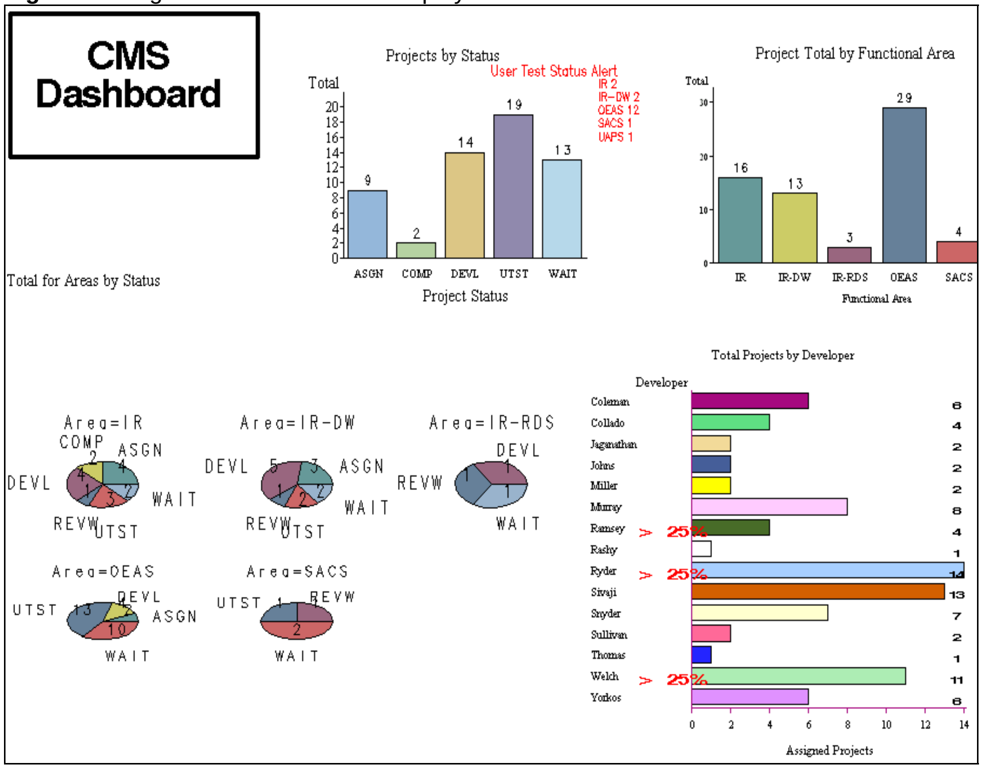


Figure 3: Programmer 2 dashboard display



SECONDARY DASHBOARD OUTPUT DESIGNS – SAS/GRAPH AND GREPLAY FUNCTIONALITY

Following input from the functional users, Programmer 1 altered the display of the dashboard resulting output presented in Figures 4 and 5.

Figure 4: Programmer 1 - Second Attempt – Page 1

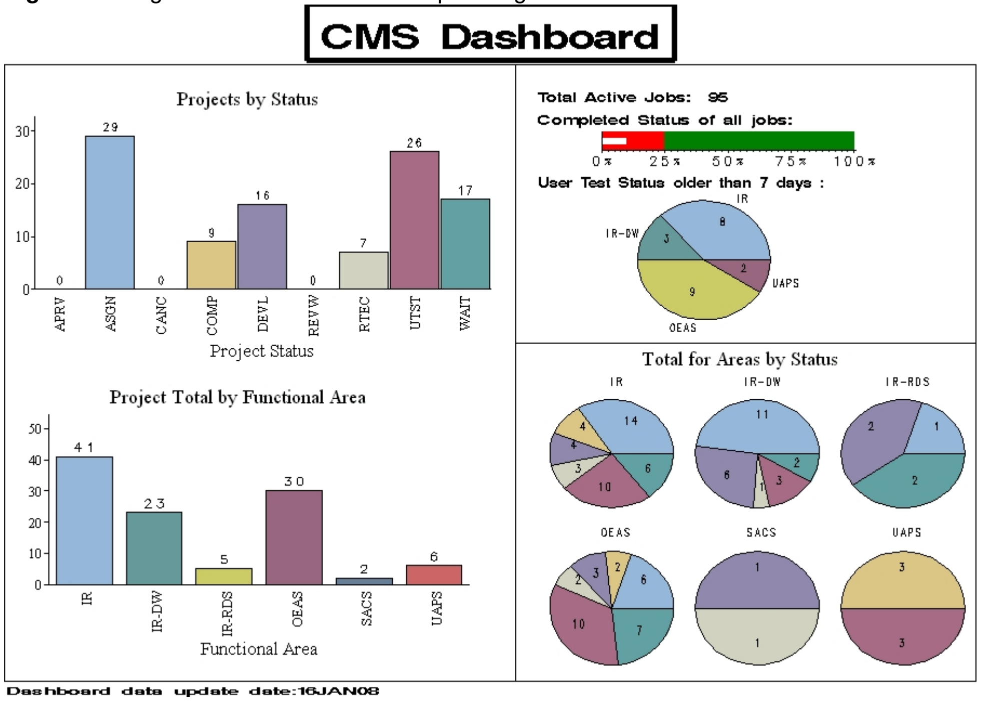
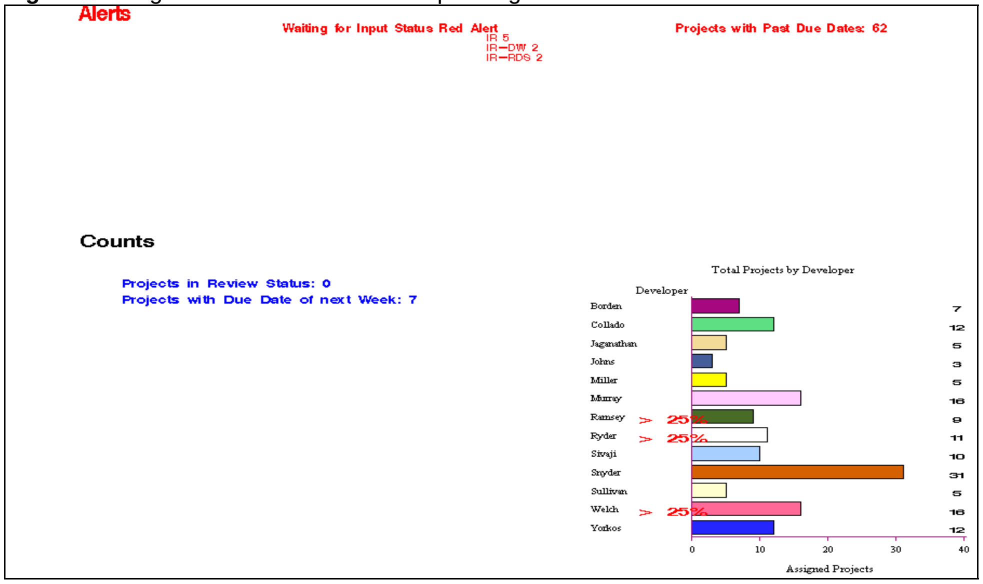


Figure 5: Programmer 2 - Second Attempt - Page 2



Alerts

Waiting for Input Status Red Alert
IR 5
IR—DW 2
IR—RDS 2

Projects with Past Due Dates: 62

Counts

Projects in Review Status: 0
Projects with Due Date of next Week: 7

As the Figure 4 and 5 design indicates, the programmers were having difficulty incorporating the requested designs into a single page of information. Additional development was needed to fine tune the output.

PRODUCTION DASHBOARD DESIGN – SAS BI DASHBOARD 3.1 FUNCTIONALITY

As development began on a third iteration of the SAS GRAPH/PROC GREPLAY version, the SSL-enabled version of SAS BI Dashboard 3.1 was released (Q4 2007). In an effort to explore the new technology, the programmers switched tools and continued their design enhancements in the BI Dashboard environment. Again, the two programmers worked on independent designs while they were both learning and exploring the new tool set. During this design phase, rather than both attempting full designs, the programmers split the indicators and alerts.

The same table development and structure used in the SAS GRAPH/PROC GREPLAY version was also used in the BI Dashboard version. However, both the programmers and the functional team members recognized that the initial specifications would need to be slightly altered to take full advantage of the new tool set. For example, text-based alerts were removed in favor of graphic-based alerts. One KPI measure, number of jobs by area by status, was embedded into the number of jobs by area through drill-down functionality. Other slight design alterations were made to enhance the overall flow and feel of the dashboard.

The delivered functionality for creating indicators using the Manage Data Models and Manage Ranges did not meet expectations. A straight chart of information without the requirement to apply success or failure indicator ranges was desired. After exploring several development alternatives, the programmers used stored process technology to provide the desired indicators. The dashboard was then able to display eight data visualization graphs in a variety of styles.

STORED PROCESS DEVELOPMENT

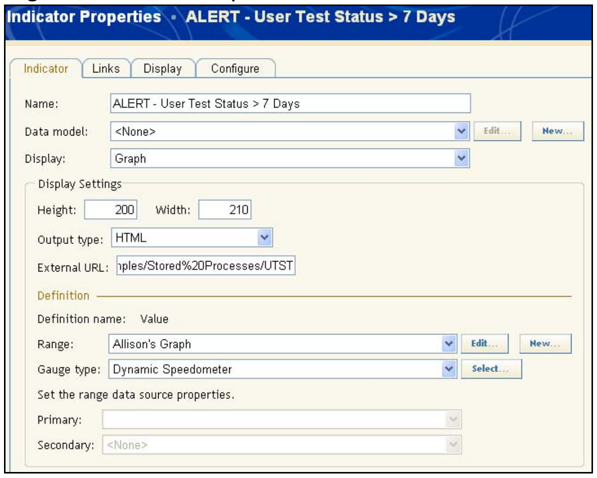
Data Creation

The data creation step was separated from the dashboard code and was run on a daily basis as a stored process to create the raw and summary data used by the indicators. This data creation process is also being evaluated to determine if it can be engineered using DI Studio.

Technical Design

Each of the data visualization graphs required a supporting stored process to be developed. The stored processes were created using SAS Enterprise Guide (EG) Graph Data functionality. The programmers had to modify the SAS code object containing the SAS Graph code to include any annotation requirements. Once the stored process was created, it was incorporated into the delivered technology at the Manage Indicators stage. Figure 6 illustrates the relationship of a stored process to a BI Dashboard Indicator.

Figure 6: Relationship of Stored Process to BI Dashboard


Indicator Properties - ALERT - User Test Status > 7 Days

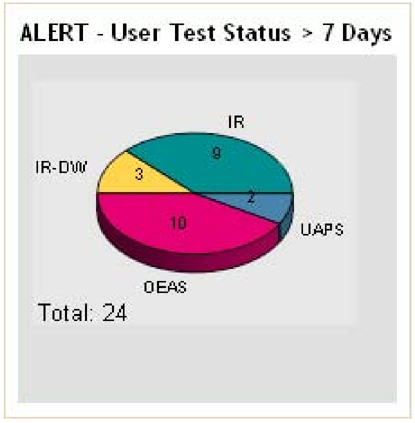
Indicator | Links | Display | Configure

Name: ALERT - User Test Status > 7 Days
Data model: <None> Edit... New...
Display: Graph

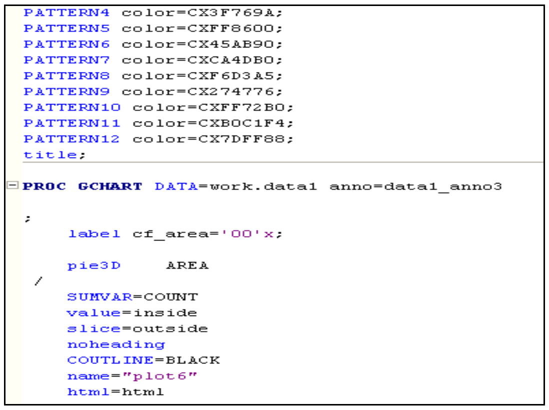
Display Settings
Height: 200 Width: 210
Output type: HTML
External URL: nples/Stored%20Processes/UTST

Definition
Definition name: Value
Range: Allison's Graph Edit... New...
Gauge type: Dynamic Speedometer Select...
Set the range data source properties.
Primary:
Secondary: <None>

Result of SAS Code – pictured as an SAS BI Dashboard Indicator



SAS Code for data visualization Graph
For the Stored Process UTST


```
PATTERN4 color=CX3F769A;
PATTERN5 color=CXFF8600;
PATTERN6 color=CX45AB90;
PATTERN7 color=CXCA4DB0;
PATTERN8 color=CXF6D3A5;
PATTERN9 color=CX274776;
PATTERN10 color=CXFF72B0;
PATTERN11 color=CXB0C1F4;
PATTERN12 color=CX7DFF88;
title;

PROC GCHART DATA=work.data1 anno=data1_anno3

;
    label cf_area='00'x;

    pie3D     AREA
 /
    SUMVAR=COUNT
    value=inside
    slice=outside
    noheading
    COUTLINE=BLACK
    name="plot6"
    html=html
```

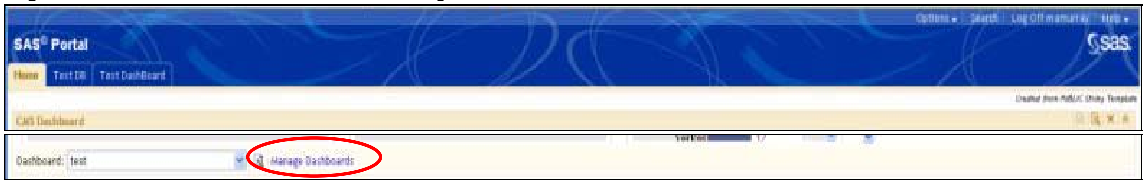
CREATING THE DASHBOARD

SAS BI Dashboard development normally requires the setup and maintenance of 4 components: the Data Model, Range, Indicator, and Dashboard. Due to the use of stored processes for dashboard indicators as described in the previous section, the CMS dashboard, the build process consisted of only 2 steps: defining the Indicators and defining the Dashboard.

Indicator Setup

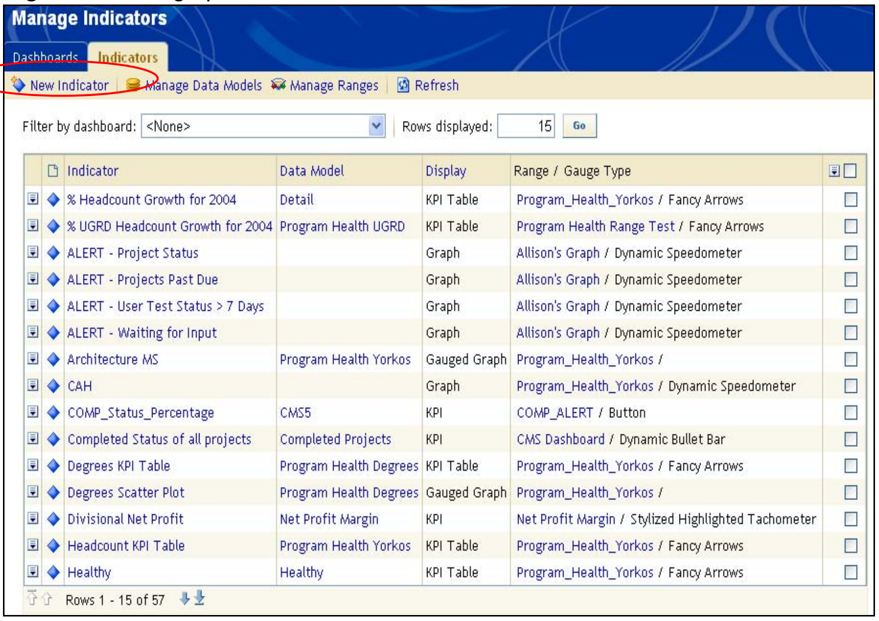
Figure 7 details a condensed Dashboard Portlet awaiting development. The BI Dashboard Portlet on the SAS Information Portal provides access to the SAS BI Dashboard Tool. Select 'Manage Dashboards' to access the BI Dashboard build/maintenance functionality.

Figure 7: SAS BI Dashboard 3.1 - Manage Dashboards



On the main Dashboard portlet page, select "Manage Dashboards"; the "Manage Indicators" page is displayed.

Figure 8: Setting up a Stored Process as an Indicator



Manage Indicators

Filter by dashboard: <None> Rows displayed: 15 Go

Indicator	Data Model	Display	Range / Gauge Type
% Headcount Growth for 2004	Detail	KPI Table	Program_Health_Yorkos / Fancy Arrows
% UGRD Headcount Growth for 2004	Program Health UGRD	KPI Table	Program Health Range Test / Fancy Arrows
ALERT - Project Status		Graph	Allison's Graph / Dynamic Speedometer
ALERT - Projects Past Due		Graph	Allison's Graph / Dynamic Speedometer
ALERT - User Test Status > 7 Days		Graph	Allison's Graph / Dynamic Speedometer
ALERT - Waiting for Input		Graph	Allison's Graph / Dynamic Speedometer
Architecture MS	Program Health Yorkos	Gauged Graph	Program_Health_Yorkos /
CAH		Graph	Program_Health_Yorkos / Dynamic Speedometer
COMP_Status_Percentage	CMS5	KPI	COMP_ALERT / Button
Completed Status of all projects	Completed Projects	KPI	CMS Dashboard / Dynamic Bullet Bar
Degrees KPI Table	Program Health Degrees	KPI Table	Program_Health_Yorkos / Fancy Arrows
Degrees Scatter Plot	Program Health Degrees	Gauged Graph	Program_Health_Yorkos /
Divisional Net Profit	Net Profit Margin	KPI	Net Profit Margin / Stylized Highlighted Tachometer
Headcount KPI Table	Program Health Yorkos	KPI Table	Program_Health_Yorkos / Fancy Arrows
Healthy	Healthy	KPI Table	Program_Health_Yorkos / Fancy Arrows

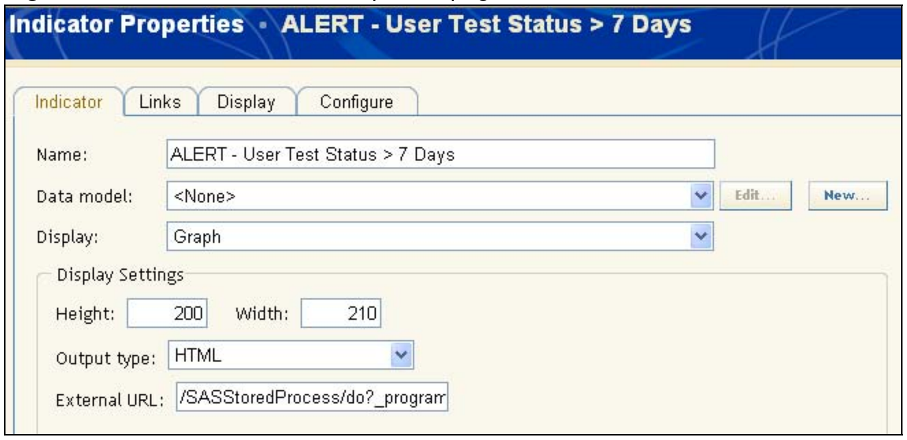
Rows 1 - 15 of 57

A new indicator is created by clicking "New Indicator". The "Indicator Properties" page is displayed (Figure 9).

Defining a SAS BI Dashboard Indicator to use a Stored Process
On the Indicator Properties page:

- Name indicator
- Set the DISPLAY field to the GRAPH option, then the External URL field appears
- Select 'NONE' for the Data Model field
- Set the dimensions of the indicator using the Display Settings
- Select the Output Type which can be "HTML" or "Image"
- Enter the path to the stored process code in the External URL field
    - Example:
      /SASStoredProcess/do?_program=SBIP://Foundation/Samples/Stored%20Processes/UTST
      See Table 3 for explanation of the stored process example

Figure 9: Dashboard Indicator Properties page



Indicator Properties • ALERT - User Test Status > 7 Days

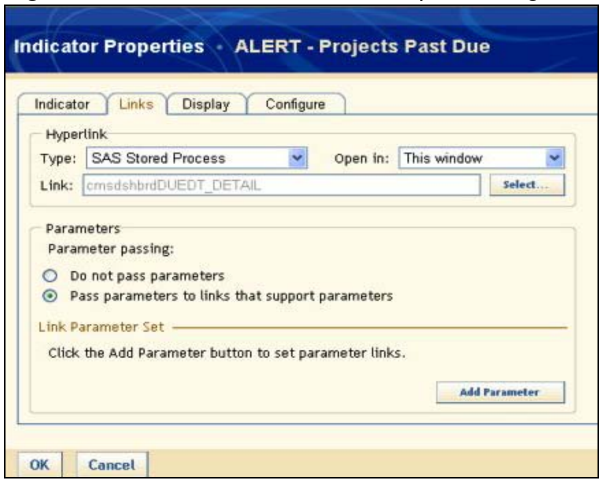
Name: ALERT - User Test Status > 7 Days
Data model: <None>
Display: Graph
Display Settings
Height: 200 Width: 210
Output type: HTML
External URL: /SASStoredProcess/do?_program

Table 3: Detailed Explanation of Stored Process URL

	What it does
/SASStoredProcess/do?_program=	Invokes the stored process server
SBIP://Foundation/Samples/Stored%20Processes/	Specifies the location of the stored process
UTST	The name of the stored process registered

- If the Stored Process contains drill down functionality use the Links Tab on the Indicator Properties Page. The TYPE Field should be set to 'SAS Stored Process'; enter the name of the Stored Process in the LINK field. Select option 'Pass Parameters to links that support parameters' to allow the passing of any parameter values required by the drill down process (Figure10).

Figure 10: LINKS Tab on Indicator Properties Page


Indicator Properties - ALERT - Projects Past Due

Indicator | Links | Display | Configure

Hyperlink

Type: SAS Stored Process   Open in: This window

Link: cmsdshbrdDUEDT_DETAIL   Select...

Parameters

Parameter passing:

Do not pass parameters [✓ Pass parameters to links that support parameters]

Link Parameter Set

Click the Add Parameter button to set parameter links.

Add Parameter

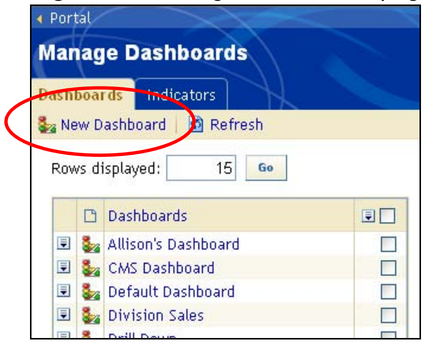
OK   Cancel

Dashboard Setup

After completing the setup for all the indicators, the programmers setup the dashboard output design. The dashboard design has 2 main steps: creating a "New Dashboard", and identifying the indicators to be contained within the dashboard.

From the "Manage Dashboards" page (Figure 11), click on "New Dashboard" option.

Figure 11: Manage Dashboards page


Portal

Manage Dashboards

Dashboards | Indicators

New Dashboard | Refresh

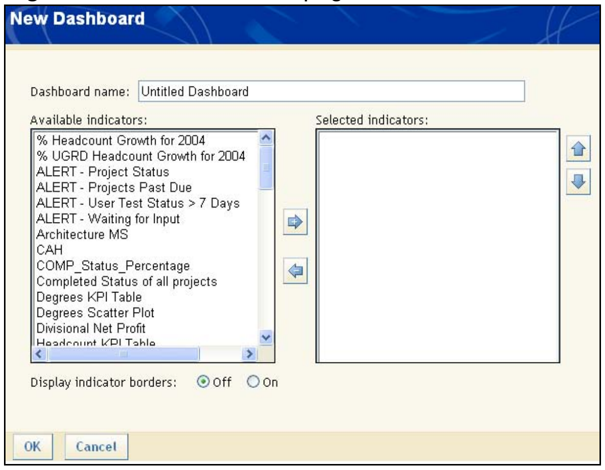
Rows displayed: 15 Go

Dashboards
Allison's Dashboard
CMS Dashboard
Default Dashboard
Division Sales
Drill Down

To create the new dashboard (Figure 12):

- Enter a Dashboard Name
- Move the desired Indicators from the Available indicators box into the Selected indicators box
- Select OK

Figure 12: New Dashboard page



New Dashboard

Dashboard name: Untitled Dashboard

Available indicators:
% Headcount Growth for 2004
% UGRD Headcount Growth for 2004
ALERT - Project Status
ALERT - Projects Past Due
ALERT - User Test Status > 7 Days
ALERT - Waiting for Input
Architecture MS
CAH
COMP_Status_Percentage
Completed Status of all projects
Degrees KPI Table
Degrees Scatter Plot
Divisional Net Profit
Headcount KPI Table

Selected indicators:

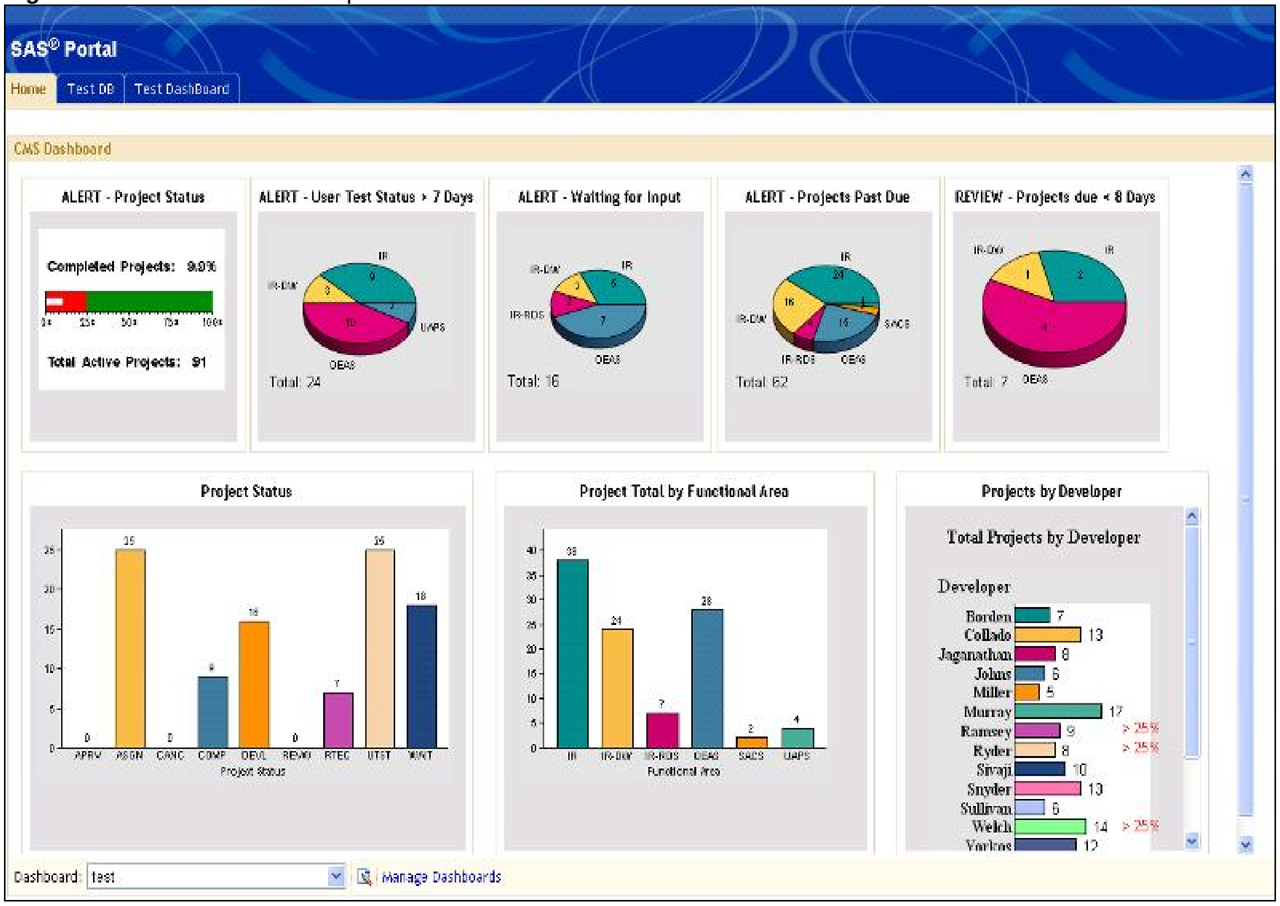
Display indicator borders: [✓ Off] On

OK Cancel

To view the dashboard, select the Dashboard by name from the drop down box on a SAS Dashboard Portlet.

FINAL DASHBOARD OUTPUT
Figure 13 shows the final design of the CMS Dashboard using the SAS BI Dashboard 3.1 tool. The programmers combined all required Alerts and Review data visualization graphs onto one page which provides the desired quick view functionally. Drill-down functionality is available in all but one of the graphs to provide a detailed view of the data.

Figure 13: Final Dashboard Output



COMPARISON OF THE DASHBOARD TOOL SETS

After completing the development and design phases of the CMS Dashboard, the programmers identified pros and cons of both the SAS GRAPH/PROC GREPLAY and the BI Dashboard functionality. Data development and end-user security options were similar in both products. However, the BI Dashboard functionality was preferred in areas such as indicator design, web development and on-going maintenance. Table 4 summarizes the pros and cons of the two different tools.

Table 4: Pros and Cons of the Tool Sets

FUNCTION	SAS GRAPH/PROC GREPLAY	BI DASHBOARD
Data development	Used SAS Enterprise Guide	Used SAS Enterprise Guide
Indicator design - the definition of the functional requirements are the most time consuming	This approach took longer – very detailed coding is required for both the annotation of text data on the charts or slides and then the placement of individual indicators created via SAS code on the page.	Due to the product's ability to allow the output of a SAS stored process to be utilized as an indicator our learning curve entailed how to setup an indicator to invoke a stored process. The SAS Dashboard examples were very helpful in this regard.
Ease of web publishing	The actual placement of individual indicators created via SAS code using PROC GREPLAY was very time consuming.	Straight forward and easy to adjust. Easy to combine the work of several programmers due to independent nature of the Indicators.
End-user security	Same for either tool set	Same for either tool set
On-going maintenance	Entails updating SAS code for both indicator modifications and placement.	Entails updating the stored process code and refreshing the dashboard page.

The SAS BI Dashboard tool eliminates the use of GREPLAY function which requires the programmer to precisely determine the coordinates of each indicator object on the dashboard page. The programmer has the luxury of surfacing multiple stored process results via an easy-to-use web-based interface. All content is displayed in a role-based, secure, and customizable environment. Without any programming knowledge SAS BI Dashboard users can customize how information appears on their personal dashboards. By combining SAS/GRAPH and the stored processes' easy-to-use interface, the SAS programmer can create visualization stored processes that are reusable and dashboard ready. This approach yields stored processes that can be used from almost any type of web application. Most importantly, the visualization stored processes encapsulate complex SAS/GRAPH code and provide an interface so that programmers can use them without any knowledge of SAS programming. It also displays ActiveX output compared to GIF output, which results in clearer graphic images in the web environment.

CONCLUSION

The CMS IR Technical Support Dashboard project joined together functional project managers and technical programmers to provide a solution for more efficient resource allocations. The dueling output options and multiple tool sets allowed for maximum presentation flexibility and a training opportunity for both programmers. The current design of the CMS dashboard provides graphical displays with quick-look views to identify successes, challenges and overburdened technical staff, teams or offices. The drill-down functionality of seven of the graphical displays provides a detailed view of the data when desired. The final product increased the ease of overall project management and specific-project-level oversight.

REFERENCES

SAS/GRAPH Dashboard Samples at http://support.sas.com/rnd/datavisualization/dashboards/index.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Office of Institutional Research
University of Central Florida
P.O. Box 160021
Orlando, FL 32816-0021
Work Phone: (407) 823-5061
Fax: (407) 823-4769
Web: http://www.iroffice.ucf.edu