

Resource Analysis with the TIMELOG[®] System

Matthew J. Becker, STATPROBE, Inc., Ann Arbor, MI

ABSTRACT

The efficiency of an organization relies on an awareness of the time required to complete different tasks. In turn, the costs of services and products depend on the amount of resources used. To manage time, STATPROBE, Inc. developed, using SAS/AF[®], a resource analysis software system called TIMELOG. The system provides employees and managers a number of different reports on resource usage. The reports provide statistics on time use and personnel resources. A quantitative analysis is done to survey and manage the effectiveness of the organization. Each of the reports can be run with data filters keeping only data the user desires. The current reports provided by the system are:

- Employee activity reports
- Employee analysis reports
- Project time analysis reports
- Actual versus budgeted time analysis

The TIMELOG reporting module is written in SAS[®] Version 6.10 running under the IBM OS/2[®] Version 3 operating system.

INTRODUCTION

In order to maintain and increase efficiency, organizations must monitor resource times required to complete a project. Statistical analysis on the allocation of resources in previous work can be used to develop benchmarks and assist in the planning process for later projects. Using SAS/AF, STATPROBE, Inc. has developed a time-tracking package, TIMELOG for OS/2, that allows employees to track time spent on a project. Data is stored in a relational database that allows flexible reporting capability. Reports are produced by the time and costs tracking module in TIMELOG.

TIMELOG meets the following objectives for a time and costs tracking system:

- To provide information useful in resourcing a project.
- To provide managers with a quantitative performance analysis of their department and individual employees.
- To provide company management with a quantitative performance analysis confirming efficiency and showing where efficiency is lacking.
- To provide the company with quantitative data to help adjust costing structures.

DEVELOPMENT OF THE TIMELOG FOR OS/2 SYSTEM

STATPROBE, Inc. developed the TIMELOG system to track the amount of time spent on projects. As a contract research organization working with pharmaceutical and medical device companies, each STATPROBE project is associated with a client and each task is associated with a contractually set amount of time. TIMELOG has undergone several generations of development. In the first paper-based tracking system, employees wrote the project work hours on a time sheet. Next, using SAS for DOS[®], the first computer-based TIMELOG was created using data sets and basic screen control language. The latest version of TIMELOG was prototyped using SAS/AF FRAME entries. A sophisticated database was designed in conjunction with the interface and reporting requirements. With object-oriented SAS/AF FRAMES, building a system that had a similar look and feel to the OS/2 graphical user interface was easy.

What proved most challenging for the development staff was determining the optimal database structure for maintainability and system and disk space efficiency. The system we chose uses relational databases with a minimal byte count per observation. For example, the client and project data set structures are:

CLIENT

Variable	Type	Length	Label
cl_num	numeric	4	client number
name	character	\$25	client name

PROJECT

Variable	Type	Length	Label
cl_num	numeric	4	client number
pr_num	numeric	4	project no.
name	character	\$40	project name

It is easy, for example, to access the client name when you select a project. Using the CL_NUM variable in the project data set, we find the corresponding CL_NUM in the client data set and pull off the name.

The report class was another data structure where minimal disk usage was desired. The development team was able to store all of the different report information in named SCL lists in the report class. The report class itself is a named SCL list.

Following is an example of a report entry in the report class:

```
(TITLES=('EMPLOYEE ACTIVITY')
PROFILES=(EMPLOYEE ACTIVITY=
  (ID=1
   WINDOW=4
   START_DT=.
   END_DT=.
   BILLABLE=2
   CL_ACT=1
   PR_ACT=1
   ...
   CL_LIST='STATPROBE'
   DE_LIST='RIS'
   EM_LIST='MATT' 'CARL')
)
```

In the report's SCL list, the information vital for that reports generation is stored. Data such as dates to subset on are stored in the WINDOW, START_DT, and END_DT variables. If you want to run a report on a list of employees, the employee names are stored in the SCL list named EM_LIST. The elements of the report's SCL list are defined when you select the subsetting criteria through the filter frame. The filter frame is discussed in detail later in this paper.

The report class makes it possible to save report definitions to disk using minimal disk space. There are certain to be reports which management would want to run periodically. Redefining the same report every time it is generated is inefficient. The report class utilizing SCL lists can be defined and saved to disk as an SLIST entry.

When a report is saved to disk, you have the option of saving it as a private report or a public report. Private reports are report definitions that only you can access. Public reports are accessible by anyone.

THE REPORT FRAME

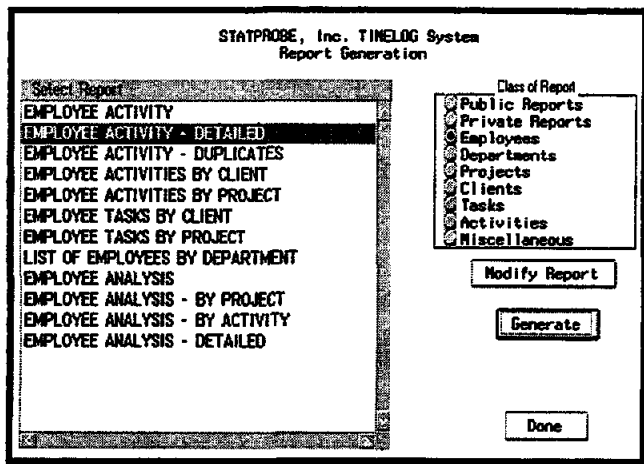


Figure 1. The Report Frame

The initial report frame (Figure 1) consists of a radio box, list box, and three push buttons. This frame's purpose is to select the type of report you want to run. Options are a saved report, defined report, or a new report based on an existing report.

There are a fixed number of report classes to generate. Therefore, we made the radio box values static (non-dynamic).

When you select a specific radio button, its description is used as an index to the report class. With the class selected, we can go to that classes default report definitions and fill the list box with the report titles contained in its SCL list. To do this, we defined the list box's entry as an SCL list.

```
rc=filllist('CATALOG',
            'timelog.reports.public.slist',
            repclass);
```

This function will fill the list box with the report names stored in the public catalog.

Push buttons were used to define more detailed subsetting conditions, generate the default report selected, delete a report (if you have permission), or exit the time and costs tracking module. The MODIFY REPORT, GENERATE, and DELETE push buttons are visible only when a report type and definition are selected. Moreover, the DELETE push button is visible only if you have the right to delete the selected report. Selecting the MODIFY REPORT displays the filter frame detailed later. GENERATE will execute the report program using the default or saved report definition. DELETE deletes the report definition from the report class and the SLIST entry. DONE exits out of the time and costs tracking system module.

THE FILTER FRAME

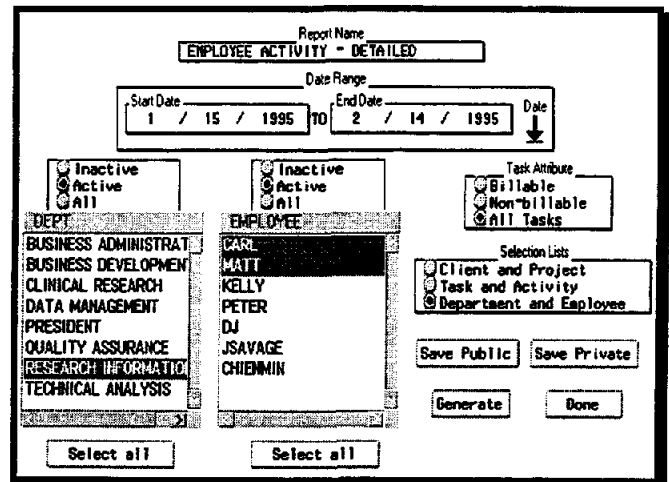


Figure 2. The Filter Frame

The filter frame (Figure 2) is where the majority of a report is defined. Objects contained in the filter frame include text entry boxes, radio boxes, and selection lists.

When you first enter the filter frame, the report name entry box contains the title of the report you selected from the report frame as the default value. If you plan to define and save a new report based on the report you selected, a new title must be entered. The title is stored in the report SCL list under the report class selected from the report frame.

Directly below the report name is the date range object. In this object there are two date fields and a pop-up control. You have the option of entering the start and end dates or selecting a pre-defined window of time. The control arrow allows you to select a window of time (Figure 3). If a window of time is selected, the filter SCL code generates the start and end dates automatically and places them in the date fields. The START_DT and END_DT variables in the report SCL list are missing if a window of time is selected. If the date

variables are missing when a report is generated, the underlying SCL code looks at the WINDOW variable and constructs the START_DT and END_DT variable's numeric values at run-time. Therefore, if you were running a report for a window of time you need not redefine the starting and ending dates each time that report is run. That is, windows of time are defined and stored relative to the date when the report is generated.

Below the date range object are four radio boxes. In two of the radio boxes, you can select whether you want to subset on inactive, active, or all list box entries. Another radio box allows you to select those tasks that are billable, non-billable, or all tasks. The fourth radio box allows you to select client/project, task/activity, or department/employee lists to display.

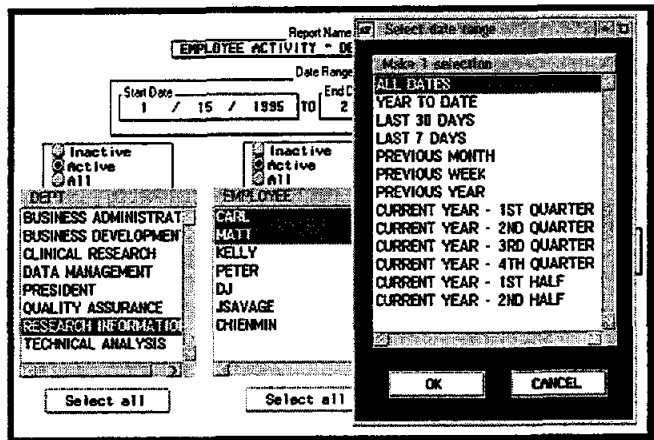


Figure 3. Date Window Selection Box

The list boxes are initialized with a SAS data set variable called NAME when a radio button is selected in the selection list radio box. The left list box appears when one of the radio buttons is selected. The radio button's value is used to index an SCL list of names of SAS data sets. The list box is populated with the NAME variable's text values. The right list box is visible only when a selection is made in the left list box. The values displayed in the right list box depend on the selection in the left list box. For example, if you select 'Client and Project,' the filter frame opens the client and project data sets. The left list box is populated with the text values of the variable NAME in the client data set. If you select one or more of the clients in the left list box, the right list box is populated with the projects associated with those clients. If you unselect a client, its' projects in the right list box are removed. To accomplish this, we wrote SCL code to re-display the values in the list box each time a list box entry was selected or unselected.

REPORTS

The underlying SCL code that generates the reports is object-oriented. Each report is considered an object. When you generate the report, you are running methods defined for that object. Thus we developed methods for each report. Three SCL entry modules were defined:

- REPMETHS.SCL contains methods for each report ID in which another SCL method is called.
- REPOBJS.SCL contains the methods to generate the reports.

- REPUTIL.SCL contains common methods used between reports.

The REPMETHS.SCL entry is very basic in its structure. When you generate a report, a report ID numeric variable contained in the report SCL list is referenced to call a method contained in the REPMETHS.SCL code. For example, if you were running a report with a report ID of 5, when the report was generated it would initially call the method identified as REPORT5. At this point it would branch off to a method in the REPTOBS.SCL code.

```
*** Report methods ***;
REPORT5:
  method title $
  profile 8;

  call method('reptobj.scl','empact',title,profile);
endmethod;
```

This is an example of a report method that branches to a method in the REPTOBS.SCL code.

The bulk of the report programming is in REPTOBS.SCL. While developing some standard reports for STATPROBE, Inc. management, we found that many of the reports produce similar output. Thus, numerous reports call the same method in the REPTOBS.SCL code. Additionally, many of the reports used the same SCL code steps to build the final report data set. Therefore, this code was placed into a method in the REPUTIL.SCL entry. Simply put, the REPTOBS.SCL methods contain a macro report definition defined in a SUBMIT-ENDSUBMIT block and a call to a method that generates the report data set. Here is an example of the structure used in report methods:

```
EMPACT:
  method title $ profile 8;

  *** Defines macro used to generate report ***;
  Submit continue;
  %Macro report(report,datestr,user);
  [PROC REPORT CODE]
  %Mend report;
  Endsubmit;

  *** Call the method which contains the bulk of the SCL ***;
  Call method('reputil.scl','byuser',title,profile);

endmethod;
```

The SCL method contained in the REPUTIL.SCL entry that is called by the above method contains a SUBMIT-ENDSUBMIT block that executes the REPORT macro defined in the above method.

The REPUTIL.SCL entry contains methods that generate the final report data set. Currently, there are two basic types of reports that are run: reports by employee and reports over a list of employees. Therefore, one method loops over the employees defined in the report SCL list and constructs a data set subsetting by any other selections made. Once the data set is created, it executes the %REPORT call to generate the report in the output window. The other method loops over the employees defined in the report SCL list and creates one data set with all of the employee data contained in it. At the

end of the method it executes the %REPORT macro on the entire employee data set created.

Additionally, the REPUTIL.SCL entry contains a method named NUMLIST that converts the report list entries such as the client list to numbers in order to subset the employee's posting data set. For example, if you wished to see how many hours employees MATT and CARL spent on STATPROBE (as the client) work, the report SCL list would contain 'STATPROBE' in the CL_LIST variable and 'MATT' 'CARL' in the EM_LIST variable. However, the posting data sets for employees MATT and CARL contain client numbers and not the textual representation of that client number. Therefore, the SCL code must convert the list of clients, projects, tasks, activities, and departments to their numeric equivalents in order to subset the posting data sets. The NUMLIST method accesses the data sets that contain this information, subsets it on the names in the list variable, and generates a text string of the corresponding numbers.

NUMLIST:

```
method textstr $ listid B dsname $ varname $;
  if listlen(listid) > 0 then do;
    textstr = "";
    dsid = open('timelog.' || dsname);
    do lst_idx = 1 to listlen(listid);
      rc = where(dsid, 'name = "' || getitemc(listid, lst_idx) || '"');
      rc = fetch(dsid);
      textstr = trim(left(textstr)) || ' ' ||
        trim(left(getvarn(dsid, varnum(dsid, varname))));
    end;
    closeid = close(dsid);
  end;
endmethod;
```

CURRENT REPORTS

One of the report types provided by the TIMELOG system produces listings of employee activity. These reports allow you to get a hard copy of your entries. There are three employee activity reports: Employee Activity, Employee Activity - Detailed, and Employee Activity - Duplicates. The Employee Activity report lists client, project, task, activity, task, hours, and a total hours column. The total hours column is the sum of all the hours worked for each date and is useful to see if you worked the required amount of time per day. The Employee Activity - Detailed report produces the same output as the employee activity report except it contains an additional comment column. When you post time, you have the option of entering a descriptive comment to be included in the posting observation. The Employee Activity - Duplicates reports any observation that contains exactly the same information as another observation. When you enter time, an observation may be duplicated in error. This report lists duplicates. All of these reports are run by employee. Therefore, if you ran an employee activity report on all employees in your department, activity for each employee appears on separate pages.

EMPLOYEE ANALYSIS - BY ACTIVITY		
DATE: 01/14/95 - 02/13/95		
USER: MATT		
TASK	ACTIVITY	TIME SPENT
BUSINESS ADMINISTRATION	MISCELLANEOUS	3.00
	STAFF/STAFF MEETINGS	4.25
HUMAN RESOURCES	RECRUITING	1.00
INTERNAL CLERICAL SUPPORT	WORD PROCESSING	2.00
INTERNAL PROJECT MANAGEMENT	REVIEW	0.75
	SUPERVISE/DIRECT	0.50
INTERNAL SOFTWARE DEVELOPMENT	PROGRAMMING	38.75
SYSTEMS ADMINISTRATION	INFORMATION RESEARCH	2.00
	SYSTEM SET-UP	3.50

Figure 4. Employee Analysis - By Activity Report

Another report type provided is the employee analysis reports. The four employee analysis reports are: Employee Analysis, Employee Analysis - By Project, Employee Analysis - By Activity (Figure 4), and Employee Analysis - Detailed. The base employee analysis report contains one column: total time worked. Using the filter frame, you can specify this report to give you the total time worked on a subset of projects. Client and project are added as columns in the Employee Analysis - By Project report. Task and activity are added in the Employee Analysis - By Task report. The Employee Analysis - Detailed report contains client, project, task, activity, and time spent.

PROJECT EMPLOYEE ANALYSIS BY TASK				
DATE: 01/14/95 - 02/13/95				
PROJECT	TASK	TOTAL HOURS	NUMBER OF EMPLOYEES	TIME/EMPLOYEE
ADMINISTRATION	BUSINESS ADMINISTRATION	12.25	1	12.25
	HUMAN RESOURCES	1.00	1	1.00
	INTERNAL PROJECT MANAGEMENT	1.25	1	1.25
	INTERNAL SOFTWARE DEVELOPMENT	3.75	1	3.75
	SYSTEMS ADMINISTRATION	5.50	1	5.50
TIMELOG	INTERNAL CLERICAL SUPPORT	2.00	1	2.00
	INTERNAL SOFTWARE DEVELOPMENT	35.00	1	35.00

Figure 5. Project Employee Analysis By Task

The third type of reports summarize project time. First, the Project Time Summary report contains columns for the project and the total hours worked. Next, the Project Employee Analysis produces a report with the project, total hours, number of employees, and the time per employee. This report gives the you information on how many people are working on a specific project and the average amount of time per employee spent on this project. The Project Employee Analysis By Task (Figure 5) displays a report similar to the Project Employee Analysis, except it breaks time down into tasks in the project. Therefore, it contains an additional column for the task. Finally, the Project Employee Analysis - Detailed report produces output with the project, task, activity, total hours, number of employees, and the average time per employee as columns. Thus, it displays the highest level of detail available. The project time summary reports are run over all users. No user identifying names are contained in the output and all the user data is summarized on the same page.

REPORT DEVELOPMENT USING THE REPORT PROCEDURE

The reason for the use of PROC REPORT was coding efficiency. Producing the reports using DATA, MEANS, and DATA_NULL statements would take a large number of lines of code. On the other hand, PROC REPORT is a combination of the MEANS and DATA steps. Therefore, one PROC REPORT definition produces the same output as using the DATA, MEANS, and DATA_NULL steps.

FUTURE ENHANCEMENTS

Future enhancements to the reporting module will include more managerial reports and graphical displays. Moreover, outputting the resulting reports to an image and displaying the image on the screen makes viewing the output more concise.

CONCLUSION

Supplying management with quantitative data on the use of time and associated costs is extremely important in establishing and maintaining efficiency in an organization. The quantitative data is readily available using the time and costs tracking module of the TIMELOG for OS/2 system. Using the results, employees have increased awareness of how their activities impact the efficiency of the organization and management can identify areas where efficiency can be improved. The information obtained can also be used to plan future projects and document the activities on a project.

REFERENCES

SAS Institute, Inc. (1993), *SAS/AF Software: Frame Entry, Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1990), *SAS Guide to the REPORT Procedure, Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1990), *SAS Screen Control Language: Reference, Version 6, First Edition*, Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1991), *SAS Technical Report P-216 SAS/AF Software, SAS/FSP Software, and SAS Screen Control Language: Changes and Enhancements, Release 6.07*, Cary, NC: SAS Institute, Inc.

ACKNOWLEDGMENTS

SAS, SAS/AF are registered trademarks of SAS Institute Inc. in the USA and other countries. IBM and OS/2 are registered trademarks of International Business Machines Corporation. DOS is a registered trademark of Microsoft Corporation.

* indicates USA registration.

AUTHOR'S ADDRESS

Matthew J. Becker
STATPROBE, Inc.
3885 Research Park Drive
Ann Arbor, MI 48108
(313) 769-5000 ext. 102

E-mail:

- 72700,1075@compuserve.com
- statprob@oeonline.com