Paper 126-31

Moving SAS® Solutions to a Production IT Environment

Mike Miller, Ermas Consulting, Inc., Atlanta, GA
David Simpson, Ermas Consulting, Inc., Atlanta, GA

ABSTRACT

The SAS® System provides a robust, flexible set of tools that allow the user to quickly develop a broad range of analytical solutions, whether they are multi-dimensional reporting, statistical models or ad hoc analyses. Innovation is a direct result of this capability, with many of these solutions becoming indispensable to running the business unit and regular updates being requested by management. When the analytic solution moves from a one-time solution to a recurring information process, it presents a unique set of challenges for both the business and IT units.

At Ermas Consulting, we've found a number of organizations struggling with the challenge of putting analytic solutions in a production environment. IT is often engaged to provide the production support and ongoing maintenance of the analytic solution. IT is viewed as being the best equipped to manage this type of work. However, the not too uncommon answer from IT is that the solution is not easily supportable without significant rework and must be managed on an ongoing basis with an inflexible change control process. This paper explores the reasons why IT finds putting analytic solutions into production difficult and then explores a framework that harnesses the new capabilities within SAS® V9 to bridge the challenges faced by the SAS® analytic and IT functions. A framework that allows the SAS® analyst to remain an innovative force within the business unit, while IT provides the disciplined approach required to institutionalize the innovation within the business.

INTRODUCTION

Teams using the SAS® Software System are often embedded within non-IT units of an organization. As examples, these analytic teams are to be found in areas such as Marketing, executing direct marketing campaigns, in Risk, building loss forecasting models or in Finance, providing product profitability assessments. These groups are often at the heart of new innovation based on a company's information assets and work in close alignment with the business unit. The solutions these teams develop become indispensable to running the business. The catch phrase, "Information as a competitive weapon", is at the heart of what these teams do.

Once a solution is developed, often these teams become responsible for the management, maintenance and operations as well. A SAS® analyst having developed a risk model may find themselves in the unfamiliar position of updating the model scores each month by the 3rd business day and publishing the data for use by other analysts. The motto, "You built it, you own it" is not unfamiliar in analytic teams. Initially, the challenge is small as is the number and complexity of the production processes an analytic team must support. However, production support activities can quickly become a material portion of the work.

Using analytic resources to maintain production support activities is not a good use of skill sets. Innovation based on analytics stops as those resources become more focused on keeping processes running instead of the next big thing. Once a solution is established, organizations often turn to IT in order to provide the production support and ongoing maintenance. The solution has matured to the point where it is viewed as an information technology application instead of an analytic application. IT is viewed as the appropriate choice as they have the right resources and background to manage applications. At this point, the CMO may call the CIO to get IT involved and provide an estimate on managing the analytic solution.

The not to unfamiliar answer from IT on managing the analytic solution may go something like this, "We can certainly manage the application. The application does not conform to our standards and it will take significant recoding to get the application into our managed environment. We estimate that it will take (supply your own number here) months to get the application moved into our production environment." At Ermas Consulting, we've encountered a number of organizations in this situation. Usually no applications are moved over, or if an application is moved over, the cost precludes any further efforts. The analytic group continues to maintain production support on their solutions and as a consequence, is resource constrained and less effective.

What is needed is a framework that allows analytic solutions developed within the business analytic teams to be easily migrated to a production IT environment. That framework must also incorporate a change control process that can support frequent updates to meet today's rapidly changing competitive environment. This paper examines the root causes that have so far made this a slow and expensive process. A framework is then developed to address those root causes which is based on the new capabilities delivered through SAS® V9. With SAS® V9 it is possible to

both speed the migration of analytic solutions and make it easier for analytic teams to develop the initial solution. This allows both groups to take full advantage of their respective talents. Analytics can continue to push the pace of innovation, while IT can ensure that the innovation is institutionalized through a reliable, repeatable process.

ROOT CAUSES

As a starting point, let's try to identify some of the root causes that are barriers to the movement of analytic solutions to an IT production environment. This will begin to define the solution space for our framework and provide a reference against which it can later be evaluated. Also, the solution space is limited to address batch-type solutions and not real time or interactive solutions. Figure 1 Root Causes, presents four primary areas where we may find barriers: the computing environment, the solution method, organization and people, and procedures.

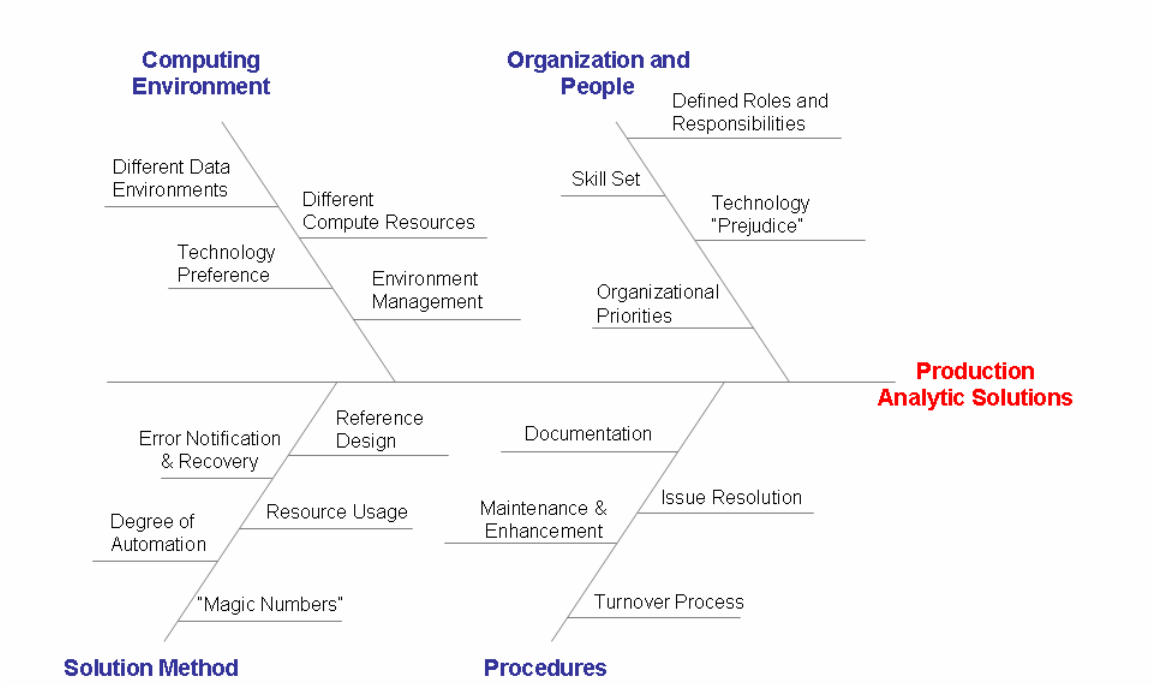

Figure 1 Root Causes

COMPUTING ENVIRONMENT

The computing environment is a natural place to start. Differences in the computing environment are going to present obvious barriers to moving a SAS® solution between analytic and IT units. It is not uncommon for the analytic teams to be working off of data sources that are not available to the IT production environment. Departmental data marts, external files from vendors, and user defined data are just a few examples. If the analytic and IT environments do not share a common set of data stores then moving a solution over to IT is problematic. The missing data stores themselves must first go through an ETL process, be put into production, and moved into the IT production environment before the application can be migrated.

The data environments must then remain in-sync as there is an on-going dependency between the analytic and IT environments. Ideally, this barrier is addressed by having a common data environment across both analytic and IT environments. In any event, our framework must recognize this dependency and address the management of analytic and IT data environments. Differing compute resources such as file systems, database schemas, server names, etc. present a similar situation. The underlying resources themselves may be different, and often will be, but if they are referenced differently then those references will need to be converted. This problem is not as significant as having differing data environments, but still needs to be addressed.

SAS® is not a primary choice of IT units for application development. This paper is not going to delve into those reasons, let it suffice that most IT units are staffed with developers who have a traditional background in Java, C/C++ or similar technologies. IT is not resourced to enhance or even maintain a SAS® based solution. In addition, IT organizations are looking to reduce the number of technologies they support in the current environment of cost cutting and out-sourcing. We believe that it is misguided for IT to take the approach of converting the SAS® solution to a more familiar, Java/C++ solution. There are the obvious upfront cost and time concerns. However, more importantly, this leads to an on-going inefficient use of resources as IT tries to stay in-sync with the analytic unit which will

continue to evolve the SAS® based solution in order to meet changing business requirements. In essence, there is shadow system being maintained.

SOLUTION METHOD

Another natural place to look for barriers is in the solution method. Analytic solutions are usually just that – solutions developed by individuals with a background in the statistical sciences and not computer science. Our own background is in computer science which required us to take a Bataan like death march through mathematics. We can build a reasonably performing response model to a direct marketing program. But, any statistician would find our methods and approach lacking the rigor that comes with a more formal background in statistics. A similar comment can be made about many information processes that are built by analysts.

Most IT groups have a reference architecture and a set of design solutions or patterns that are repeated throughout their applications. These design solutions or patterns represent best practices to problems that are seen on a recurring basis. There are obvious benefits in this approach. Something similar is needed for analytic solutions, a lightweight framework that the analyst can use to build a reasonably well structured information process. This framework should address the recurring, non-analytic challenges faced by the analyst. One of those challenges is how data is acquired as the initial step in the analytic process. This often presents a significant coding effort on the part of the analyst that is also prone to being a static, hard-coded process. Additionally, the analyst must often transform the data from a relational to a time-series format in order to apply statistical techniques. Instead, is there an approach that allows the analyst to specify the data and format required without also having to specify the code? Such an approach will be outlined as part of our solution framework.

There are other recurring problems and common pitfalls to be found: error recovery and notification, the degree of automation, the use of “magic numbers” and resource usage. Let’s address error recovery and notification first, as every SAS® programmer probably does it differently. It is okay for the analyst to scan their log files for error and warning messages either manually or with scripts. And, the analyst who built the solution certainly knows the recovery procedure. However, in a production environment the scheduling software must be notified that an error has occurred, ideally the type of error and a notification sent out to the person on call. There must also be documented recovery procedures in place for common errors unless the analyst wants a call at 3 AM.

The degree of automation and the use of “magic numbers” are part of a general requirement to parameterize the entire solution. It is okay for the analyst to make slight changes each month in order to get their process to run. However, in a production environment, there will be a handful of individuals responsible for hundreds of processes. Therefore, each process must be “lights-out” automated. This automation must be to the point that all inputs/outputs are passed as parameters so that each month (week, day, etc.) can be run without modification. In addition, minor “tweaks” that an analyst might make, for example, formatting a report in Excel, must be fully automated in the production environment. Constants should not be hard-coded as magic numbers, but represented as macro variables in an include file. Values that are really not constants, but slowly changing dimensions, such as the prime interest rate, should be stored in a lookup table.

ORGANIZATION AND PEOPLE

The organizational priorities between the analytics and IT units are not necessarily always aligned. For example, analytics may have an emphasis on speed to market over quality while technology is primarily focused on cost containment. It is natural to view these priorities as competing, but depending on how the different responsibilities are assigned in our framework, they can be complementary. For example, if code maintenance and enhancement of existing analytic solutions remains a responsibility of Analytics, then they can appropriately balance the risks of a speedy new deployment. If maintaining the overall environment and production schedule is an IT responsibility, then having a focus on cost containment and meeting service level agreements may be good objectives.

Different skill sets and technology “prejudices” also between IT and the analytic units embedded in staff functions such as Marketing, Finance or Risk. As mentioned earlier, SAS® is not a primary choice of IT units for application development and can be viewed with some suspicion. Most IT units are staffed with developers who have a traditional background in Java, C/C++ or similar technologies. IT is not resourced to enhance or even maintain a SAS® based solution, and keeping that SAS® staff in IT can be problematic. One IT shop that tried to source SAS® programmers found they were constantly having them recruited away to analytic units. Also, the technical background to understand analytic applications is not a common skill set found within IT. Conversely, IT concepts such as version control, unit testing, code re-factoring, etc. are rarely found in analytic units.

With V9, the SAS® Software System moved from departmental software to an enterprise class software solution. The multi-tier framework of client machines, web and application servers are familiar to IT, whereas SAS® V8 was departmental software that could be managed by analysts. The complexity and scope of SAS® V9 requires that it be managed through a unit with professional IT skills. Let IT manage the environment and production schedule with

which they are very familiar. However, having IT maintain or worse, convert the solutions to a different code base does not take advantage of the existing skill sets. Provide the generalized design solutions and development guidelines that will allow the analysts to perform the development and maintenance activities in SAS®. Let IT partner with the analysts to ensure rigor around version control and testing. These ideas will be developed further as part of team roles and responsibilities around each of the activities within the framework.

THE ERMAS ENVIRONMENTAL IMPACT

The Ermas framework can be thought of as having an 'environmental' impact in an organization. This impact can best be described in seven environments: Managed, Production, Data, Output, Patterns, Computing and People.

MANAGED ENVIRONMENT

Using the SAS® Management Console to create and manage computing environments that are Analytic and IT 'agnostic'

Central to the success of the Ermas framework is the SAS® Management Console (SMC), which assists in managing the SAS® environment. Through the use of the SMC, metadata for resources that are used in the framework can be created, updated and managed. Additional SAS® V9 applications can the make use of this metadata to access these resources, examples being: ETL Studio, Enterprise Guide, and the SAS® add-in for Microsoft Office.

METADATA (DATA ABOUT DATA)

SAS®' implementation of metadata allows for an isolated repository to be created for a project. All metadata specific to that project can be stored in this metadata repository and made available to the project developers (analysts) to use. The developers can create/update/delete metadata, without affecting the other repositories. Once the project is at a stage for the IT department to get involved they can begin the process of promoting the project repository to the appropriate production metadata repository. The developers maintain their freedom and the IT group can comfortably isolate analytic development from production systems.

WHAT RUNS WHERE

Since there can be many different computing resources available in an enterprise, the SMC provides a central location to manage them, including the various SAS® servers installed across the enterprise. Very often the infrastructure includes Development, Test and Production environments. SAS® V9 caters for this and enables the analysts to be given resources for development on the development and test systems. Once the project is ready for hand over to the IT department they can further test the system in their existing development/test environments before moving them onto the production environment. Examples of resources that can be 'moved' around include Libraries, Stored Procedures, Databases and Jobs.

SECURITY & ACCESS

Underlying the resources is a comprehensive system to manage access to Systems, Data and Stored Processes, thus enabling the IT department to put controls in place, where needed. Control to access resources is brought together by the use of Users, Groups and Authentication Domains, enabling a single sign-on facility for analysts and developers. With resources centrally defined and managed, analysts and developers can use tools such as SAS® Enterprise Guide and SAS® ETL Studio to access resources, without always knowing where they are or what server they are on.

PRODUCTION ENVIRONMENT

Using the LSF Scheduler and Stored Process Server to create a production environment

Once a solution has been developed there are two SAS® features that can assist in leveraging the work. The first is to create Stored Processes so that the functionality can be accessed in other programs or environments. The second is the creation of production jobs, which can then be scheduled to run as needed and hence tracked by administrators.

STORED PROCESSES

Making use of Stored Processes is of great importance in the framework. Though this strategy, common functionality can be made available for reuse time and time again. These stored processes can even become available to standard office applications such as Microsoft Office (through the use of the SAS® Add-In for Microsoft Office) and to SAS® Enterprise Guide, allowing business users and analyst/developers to access standard data and to build upon standard components.

JOB SCHEDULING

The SMC allows for jobs to be managed. Jobs can be created from SAS® ETL Studio and SAS® Enterprise Guide and these jobs are run under the control of the LSF Scheduler (or other enterprise job scheduling system). As a result the IT department can manage control of the jobs enabling them to efficiently use computing resources. Jobs can be run when the resources are available with errors tracked and reported.

DATA ENVIRONMENT

Using a data mart strategy with a flexible, template driven approach to solving the data acquisition problem

The goal of this environment is to create an environment where the data needed by analysts and developers is easily accessible without having to write a program to extract data. Through a template-driven approach, analysts are able to specify the data needed for their routines without having to write a single-line of code. Their analytic routines can then be attached at the end of the template. This completely eliminates the need for IT to recode applications developed by analysts. In addition, through the SMC it is much easier to manage the analytic and IT environments so that they remain in-sync. This code gives an example of a generalized function that can be used to extract data from a data source (RDBMS or SAS® Data set). Those using this template will receive consistent SAS® data sets each time they need to extract data from data sources. The dataset created is based on an input dataset defining the population to be extracted. That population can be defined at a household, customer or account level. The extracted variables are put into a monthly time-series format by appending each variable with the suffix YYYYMM or an integer, based on January 2000 as the starting point.

```
options nocenter nodate nonumber ls=72;
/*-------------------------------------------------------------------------+
 | Program       : ts_xtrct.SAS                                            |
 |                                                                         |
 | This program is designed to provide an easy interface to extracting     |
 | time-series information from the SAS Data Mart.  To specify the         |
 | variables, time periods, and dataset information for retrieving data,   |
 | the user should modify/add the following parameters to the USER         |
 | PARAMETER SECTION:                                                      |
 |                                                                         |
 |    libname   - modify the libname declaration to specify a directory    |
 |                for the output dataset.                                  |
 |    out_lib   - libname to use for the output user dataset.              |
 |    in_lib    - libname to use for the input user dataset.               |
 |    out_ds    - output user dataset name.                                |
 |    in_ds     - input dataset that identifies the parties or households  |
 |                for which to extract data.                               |
 |                                                                         |
 |    beg_dt<n> - Beginning time period for which to extract time series   |
 |                variables.                                               |
 |    end_dt<n> - Ending time period for which to extract time series      |
 |                variables.                                               |
 |    vars<n>   - Variables to extract from from the SAS Data Mart.        |
 |                The program will automatically identify which dataset    |
 |                each variables comes from.                               |
 |    level     - Specify either HH or PARTY.  Identifies whether to       |
 |                extract household or party-level data.                   |
 |                                                                         |
 | Variables from the olb, olbhh, and hh SAS Data Mart datasets are merged |
 | with the input dataset to produce the output dataset.                   |
 |                                                                         |
 | The user may extract discontinuous date ranges by specifying multiple   |
 | beg_dt<n>, end_dt<n> combinations. Each vars<n> variable corresponds    |
 | to a matching date range.                                               |
 |                                                                         |
 | Date ranges must be in pairs of beg_dt<n> and end_dt<n> in the form     |
 | <'01MONYY'D> where <n> is a number starting with one.                   |
 |                                                                         |
 | The variables in vars<n> can vary for each date range. Any variables    |
 | present in a particular date range which are not found in subsequent or |
 | previous date ranges will simply not be listed for those date ranges.   |
 |                                                                         |
 | This applies to all statment, TRW, MCIF and MISC variables. The         |
 | exception to this will be Account variables. Any account variable       |
 | listed in any of the vars<n> will be extracted for every observation.   |
 +-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------+
```

```
 |                      USER PARAMETER SECTION                              |
 +--------------------------------------------------------------------------*/

libname indd  '/lcm/pymts/source';
libname outdd '/lcm/pymts/cmpg';

%let out_lib = outdd;                     * Output SAS libname;
%let in_lib  = indd;                     * Input SAS libname;

%let out_ds  = trigger200301_activity;     * Output SAS dataset;
%let in_ds   = trigger200301;              * Input SAS dataset;

%let level = HH;                 * HH for Household, C for Customer, A for account;
%let timeFormat = YYYYMM;        * YYYYMM or INTEGER;

%let beg_dt1 = '01AUG2002'D;               * Time-Series begin date;
%let end_dt1 = '01AUG2002'D;               * Time-Series end date;
%let vars1 = payee_num new_payee_num ebill_num new_ebill_num;

%let beg_dt2 = '01SEP2002'D;               * Time-Series begin date;
%let end_dt2 = '01JAN2003'D;               * Time-Series end date;
%let vars2 = svc_cd bills_paid_num bills_paid_amt web_login_num ;

/*--------------------------------------------------------------------------+
 |                    END USER PARAMETER SECTION                            |
 +--------------------------------------------------------------------------*/

/*--------------------------------------------------------------------------+
 |              MACROS: DO NOT MODIFY THIS PORTION OF THE CODE              |
 +--------------------------------------------------------------------------*/
%include '~nbk719h/code/prod/include/ts_xtrct_def_global.inc';
%include '~nbk719h/code/prod/include/SAS®mart_lib.inc';
%include '~nbk719h/code/prod/include/rem_dups.inc';
%include '~nbk719h/code/prod/include/interval_to_yyyymm.inc';
%include '~nbk719h/code/prod/include/map_vars_tofile.inc';
%include '~nbk719h/code/prod/include/vars_ts_acnt.inc';
data _null_;

/*--------------------------------------------------------------------------+
 |                           END MACRO SECTION                              |
 +--------------------------------------------------------------------------*/

data &out_lib..&out_ds (compress=yes);
   set &in_lib..&in_ds;

   %get_vars;   * Get the time-series information;

/*--------------------------------------------------------------------------+
 |                          USER CODE SECTION                               |
 +--------------------------------------------------------------------------*/
proc sort data=&out_lib..&out_ds (compress=yes) noequals; by cell;
proc contents data=&out_lib..&out_ds;
```

OUTPUT ENVIRONMENT

Final output deployment of files and reports through a portal strategy

Each organization has its preferred vehicle for the presentation of results; in fact different divisions might have different preferences, even within the same organization. Examples being Microsoft Excel (or other Office applications), PDF, etc.. With SAS® V9 there are more options than ever as to the output format of results, it is relatively straight forward to output data in a number of formats; HTML for the Web, XML, PDF or Microsoft Excel.

SAS® PORTAL

By using the SAS® Portal as the central location for locating reports, the output can be easily located and different formats catered for. Users will be able to: search for the information that they require, go to one location to get the latest information, rather than searching through a list of emails.

SAS® PUBLISH AND SUBSCRIBE

Use of the SAS® Publish and Subscribe features would allow the consumer of information to manage what they receive and possibly in what format. A developer would publish the information, and it is up to users to browse and subscribe to the information that they want to. To make different formats available the developer would have to add support for them, which might not be a goal for their particular project. An alternate method is for the IT department to complete the available output formats by adding them when they take over the project.

PATTERNS ENVIRONMENT

Creating design patterns to solve the recurring problems of error recovery and notification, environment references, hard-coded values and process automation

Patterns are often described in theoretical terms, and there are many good resources for you to read up on patterns, but we though it better to show how this approach might begin to see itself impact SAS® projects. Below are a number of small SAS® macros that are used to deal with the recurring need to report on the results of the running of a SAS® program. The macros below are:

- emailMsg - You've all got one of these already, haven't you !
- numRows - Get the number of rows in the table provided
- audit - Program to check the log file for errors or warnings

```
/*-------------------------------------------------------------------+
  Macro: emailMsg
  Parms: msgSubject - subject line of the email.
         msg - body of the email msg.
         type - TEXT, FILE, or PIPE.  TEXT the body of the email msg
                can be handled as a text string.  FILE the body of the
                email message is in a file that needs to be
                concatenated to the email message. PIPE the body of
                the email message will come from a UNIX command that
                needs to be executed and the output redirected to the
                body of the email.

  This macro uses the sendmail command to email a message from within
  a SAS program.  The message body can be a text string, the contents
  of a file or the output of a  UNIX command.  The macro expects to
  find a file named email.header in the working directory.  This file
  contains the From: and To: portions of the email in the format:
     From: "YourName" <va2pwap010@xyz.com>
     To: mike.j.miller@xyz.com, john.doe@xyz.com

  The "YourName" portion of the email.header will be displayed as who
  the email is coming from and does need to be in quotes.  The To: portion
  of the email can contain one or more comma separated email addresses.
 +-------------------------------------------------------------------*/
%macro emailMsg(msgSubject, msg, type);
   %local quote;
   %let quote = %str(%');

   %if ( %scan(&msg,1,%str( )) eq ) %then %do;
      %sysexec echo &quote.Subject:&msgSubject.\n\n&quote |
               cat email.header - |
               sendmail -f va2pwap010@xyz.com -t;
   %end;
   %else %if ( %scan(&type,1,%str( )) eq ) or
            ( %scan(&type,1,%str( )) eq TEXT) %then %do;
      %sysexec echo &quote.Subject:&msgSubject.\n\n&quote&quote.&msg.&quote |
               cat email.header - |
               sendmail -f va2pwap010@xyz.com -t;
   %end;
```

```
   %else %if ( %scan(&type,1,%str( )) eq FILE ) %then %do;
      %sysexec echo &quote.Subject:&msgSubject.\n\n&quote&quote.&msg.&quote |
               cat email.header &msg - |
               sendmail -f va2pwap010@xzy.com -t;
   %end;
   %else %if ( %scan(&type,1,%str( )) eq PIPE ) %then %do;
      %sysexec %str(&msg > .temp; )
        echo &quote.Subject:&msgSubject.\n\n&quote&quote.Program:&msg.&quote |
        cat email.header - .temp |
        sendmail -f va2pwap010@xzy.com -t;
   %end;
%mend emailMsg;

/*-------------------------------------------------------------------------+
  Macro: numRows
  Calls: calls the isTable macro.
  Parms: objType - Type of object: SAS, TERADATA. SAS is a SAS®
                   dataset and TERADATA is a teradata view or table.
         object  - For a SAS object, the libname and dataset
                   name can be passed: <libname>.<dataset> OR just
                   the dataset name and the libname will default to work.
                   For a TERADATA object, both the database name and
                   the table name must be passed: <dbase>.<table>
         retVar  - Name of the macro variable to return the number of
                   rows.
  Returns: 0 in <retVar> if the the table does not exist or is empty.
           n the number of rows in the table.

  This macros checks for the existence of the object and returns 0 if the
  object does not exist.  If the object exists, the object is queried and
  the number of rows in the object are returned through the macro variable
  specified by <retVar>.
 +-------------------------------------------------------------------------*/
%macro numRows(objType, object, retVar);
   %global &retVar;
   %let &retVar = 0;
   %local quote dsid;
   %let quote = %str(%');
   %let objType = %upcase(&objType);
   %let object = %upcase(&object);

   %if &objType=TERADATA %then %do;
      %if %index(&object,.) = 0 %then %do;
         data _null_;
            error "%upcase(error) numRows: invalid TERADATA object "
                  "passed: &object";
            abort return 101;
      %end;

      %isTable(&object);
      %if &isTable %then %do;
         %connect2(teradata);
         options nomacrogen nosymbolgen notes nomprint;
         proc sql noprint;
         %include connect;
         select num_rows into :&retVar from connection to teradata
         (SELECT count(*) as num_rows FROM &object);
         disconnect from teradata;
         quit;

         %let &retVar = &&&retVar;
```

```
      %end;
      %else %let &retVar = 0;
   %end;
   %else %if &objType=SAS %then %do;
      %if ( %sysfunc(exist(&object)) ) %then %do;
         %let dsid=%sysfunc(open(&object,i));
         %let &retVar = %sysfunc(attrn(&dsid,nobs));
         %let dsid = %sysfunc(close(&dsid));
      %end;
      %else %do;
         %let &retVar = 0; /* SAS object does not exist. */
      %end;
   %end;
   %else %do;
      data _null_;
         error "%upcase(error) numRows: invalid OBJECT TYPE passed: &objType";
         abort return 101;
   %end;
%mend numRows;

options nocenter threads=yes cpucount=8 errorabend;
/*-------------------------------------------------------------------------+
  Program: audit.SAS
  Parms:   sysparm - the log file(s) to be audited should be specified
                     on the SAS command line as -sysparm logfile.  The
                     UNIX wildcard characters like * can be used to
                     specify more than one log file.
  Email Notification: The program sends e-mail notifications as part
                      of each run. To be on the distribution, edit the
                      email.header file and add your address.  Addresses
                      should be common separated.
  Description
  -----------
  This program is called after a scheduled run of a SAS program to check
  the log or lst files for error or warning messages.  The log or lst
  files to be checked should be specified on the command line with
  -sysparm logfiles.  The program calls <audit.ksh> to scan the file(s)
  for error or warning messages and e-mails the messages as
  specified in email.header.  For completeness, the audit.ksh routine
  is shown below:
          #!/bin/ksh
          tempcnt=`grep -E 'ERROR|WARNING' $1 | wc -l`
          tempcnt=`expr $tempcnt \* 1`

          if [ $tempcnt -gt 0 ]; then
             # Error or Warning
             echo "$tempcnt error(s) found in log files in $PWD"
             echo
             echo "Listing of errors follows"
             echo
             grep -En 'ERROR|WARNING' $1
          fi
          exit 0
 +-------------------------------------------------------------------------*/
filename audit pipe "audit.ksh &sysparm";
%include 'emailMsg.inc';

* Execute audit.ksh and read the number of errors found;
%let numErrors = 0;
data _null_;
   infile audit pad end=last;
```

```
   input errorLine $255.;
   if last then call symput('numErrors',left(_n_));

data _null_;
/* Helper routine to execute audit.ksh a second time if errors are
   found and return any error or warning messages. */
%macro sendMsg;
   %if &numErrors NE 0 %then %do;
      %emailMsg(%upcase(urgent:) Error or Warning found in SAS File(s),
                audit.ksh &sysparm, PIPE);
   %end;
   %else %do;
      %emailMsg(Audit of &sysparm Ok, No errors or warnings found in SAS files);
   %end;
%mend sendMsg;
%sendMsg;
```

COMPUTING ENVIRONMENT

Alleviating resources usage through the parallel processing and multi-threading capabilities of SAS® V9

With many enterprise servers consisting of multiple processors, making good use of these resources would be advantageous. For example the template discussed above <ts_xtract.SAS®> for extracting time-series data makes use of MPCONNECT to run multiple tasks in parallel. As a result extracting data for analytic applications can happen in parallel for each set of time series variables that is being requested. In addition, the load can be spread across multiple workspace servers.

PEOPLE ENVIRONMENT

Team roles, turnover procedures, version control, maintenance and enhancement procedures, and issue resolution

Without the buy-in from those using the system; success will be limited. A change in the People Environment is required to enhance the likelihood of success. First of the people environments is training. With the skill sets quite different between the analysis and the IT developer it is important to accept this situation and put procedures in place to set realistic expectations for areas such as turnover and version control. Not only will adequate training be required, but training designed to show the specific benefits of using elements of the framework. There should be a system whereby analysts and developers get incentives for re-using code and data. Allow time to re-factor and update code AFTER the project is complete and delivered.

CONCLUSION

In this paper we have set out a number of issues that have been around for quite some time in most organizations using SAS®. With SAS® V9 we have a software solution that will allow us to begin to bridge the gap between the analysts and the IT department. Done correctly the analyst will get more time to do what they want to do and the IT department will be able to better manage the resources under their watchful eyes. If any one environment is more critical than all the rest; it would be PEOPLE. Getting those involved skilled in using the appropriate SAS® V9 tools and the development and use of re-useable resources is critical.

CONTACT INFORMATION

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

Name	Mike Miller	Name	David Simpson
Enterprise	Ermas Consulting Inc.	Enterprise	Ermas Consulting Inc.
Address	4080 McGinnis Ferry Road, Suite 101	Address	4080 McGinnis Ferry Road, Suite 101
City, State ZIP	Alpharetta, GA 30005	City, State ZIP	Alpharetta, GA 30005
Work Phone:	770 390-0650	Work Phone:	770 390-0650
Fax:	770 390-9094	Fax:	770 390-9094
E-mail:	mmiller@ermas.com	E-mail:	dsimpson@ermas.com
Web:	http://www.ermas.com/	Web:	http://www.ermas.com/