Paper AD-03

The Revised NTP TDMS System

Gabriel Cano, Kevin McGowan, Jean Orelien, Constella Group, Durham, NC

ABSTRACT

The Toxicology Data Management System (TDMS) for the National Toxicology Program (NTP) is sponsored by National Institute of Environmental Health Sciences (NIEHS). This is a reporting system that is used to analyze the toxicity of chemical agents by performing long-term and short-term studies of their effects on rats and mice. The current system has been in existence for approximately 20 years. Originally, this was a batch reporting system. In the past 2 years the system has developed into a web reporting system. Formerly this system was a multi-generational development of various programming languages and applications, including SAS. Its revision has long been needed. This system has been revised to use current technologies. The statistical reporting compunent for this system was implemented on a Linux operating system using SAS. An Oracle database was used to access data and store the final report from SAS. This paper will cite various challenges encountered during implementation and the resolutions. Emphasis will be given to aspects of the SQL, ODS, and IntrNet products and system interface

KEYWORDS

SAS/IntrNet, SAS SQL, ODS, Oracle, Linux

INTRODUCTION

The purpose of this presentation is to illustrate the various phases of software development for the statistical reporting component of the NTP TDMS. First, a brief history of the current system will be presented. Next, the revised system will be described with some logistical details. Finally, the more interesting and challenging experiences of the system implementation will be discussed. The majority of this presentation will detail the steps taken to put this system into place and how the team dealt with various obstacles.

A BRIEF HISTORY

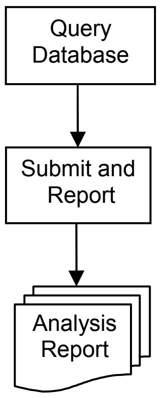
The TDMS database system for the NTP started as an IBM mainframe system at NCTR (National Center for Toxicology Research) in Arkansas in 1980. When the NTP project headquarters moved to NIEHS the TMDS database system moved with it. In 1985 TDMS moved from the IBM system to a VAX/VMS system. One of the reasons it moved to a VAX system was the availability on VAX of the ADABAS software that was used as the database. The TDMS system did not go through any major changes after 1985 until 1998 when it was moved to Open VMS on a VAX Alpha server machine from a larger VAX system. Also in 1998 a web-based interface was added to allow users to request reports from any web browser. In 2002 the NTP decided to completely revamp TDMS converting it to run on more current technologies.

THE OLD SYSTEM AND THE DATABASE

The NTP database consists of animal data such as pathology results, animal weights, food and water consumption and other measurements that are collected from several labs scattered across the United States along with one lab in Italy. The data is collected using personal computers at each lab on a daily basis. The data is transferred from the labs into the TDMS database electronically via modems. The communication to the labs is two-way in that the project directors can also send data back to the labs if it needs correction or updating. At the start of a project study managers download all the data needed to run the study into the lab computers. The software on the computers in the labs is written by the same programmers who maintain the TDMS database.

This is a general logistical picture of the system:

Figure 1. The System



The conversion of the TDMS system from a VAX/ADABAS system to one utilizing current technologies required that all of the database queries be written over from scratch. This ended up being a good idea because the queries in the old system have the following problems:

- Hard to maintain – the queries are written in many different languages such as C, Pascal, Cobol and Natural (Natural is the database query language for ADABAS.) Some of the languages used have no current employee that is an expert in the language.

- Limited use – most of the current queries were designed to be used on only 1 study at a time. For projects where data are needed to be analyzed across multiple studies additional programming has to be done to combine the data from multiple studies.

- Not a relational system – the ADABAS system uses flat files rather than a relational system. This leads to more programming than is needed with a relational system. Flat file systems also typically use up more disk space and run slower than relational systems.

- Availability - Though SAS is already used for statistical reporting more recent improvements to that product could be utilized to make reports more available to scientists/users.

For these reasons and others the report system as a whole needs a reinvigoration of succintness and accessibility.

A REVISED SYSTEM

A system revision has long been needed. Years of multi-generational development have made the system awkward for both user and maintainer. The utilization of current technologies is much desired and have made the fruition of the new web-based system very exciting. These technologies include HTML, Java, an Oracle database running under the Linux operating system and SAS.

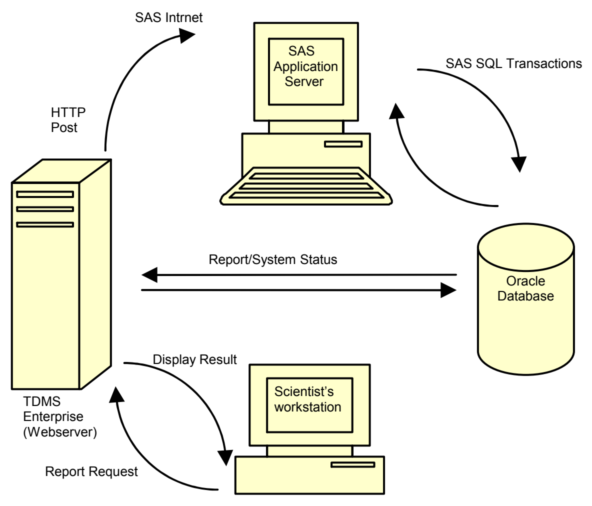
The new Oracle/Linux TDMS is better than the current system in several ways:

- More user friendly - the new system is 100% web based rather than using an old-fashioned character based system.
- Better database design - the new system uses a relational design for better data storage and faster access. The new database is designed to be much easier to compile data across multiple studies.
- Better reports - the upgraded system will output reports in modern formats such as PDF and XML rather than plain ASCII text files. The modern report formats allow the usage of color, graphics and text combined, and better ability to distribute the reports to users on different computing platforms such as Windows, Macintosh, and various forms of UNIX.
- More consistent software development strategy - the new Oracle/Linux system uses Oracle, Java, HTML, and SAS for 95% of the software modules.

Despite the fact that newer tools have been used there were some situations where some of the old technology could not be immediately eliminated. Those portions will eventually be phased out. On the other hand there were points in the system where the newer technology was still unable to accomplish the streamlining that was sought after with this revision. This will be elaborated in the programming development phases section.

This a high level graphic of the revised system, TDMS Enterprise (TDMSE):

Figure 2. Revised System



A request for a statistical report proceeds as follows:

- A scientist/user logs into the system and issues a request for a particular report(s).
- The Application Server (Appserver) then starts a SAS program via SAS Intrnet. This program is now in control.
- A report process is initiated in Oracle database by the program.
- The Oracle database is queried using the information passed from the HTTP post.
- The report entry in the Oracle database is then updated to finished.
- The TDMS then retrieves the report and makes it available to the scientist/user.

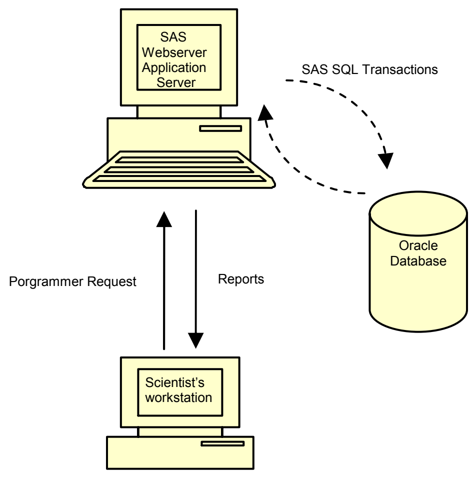
DEVELOPMENT PHASES

Implementation and development of this system brought an array of never ending challenges. Most notably it would take about 2 months for the system web server and the SAS Application server to be configured and ready. Issues regarding firewalls prevented direct system development until all the proper personnell could be contacted. Availability of the Oracle database [and data] was always a concern since that deveopment was outside and apart from the statistical reporting component of the system.

Due to the circumstances at hand a massive effort of parallel development had to happen. The statistical reporting programs had to absolutely be implemented and ready to run by the time all the interface pieces of the system were in place. Development had to begin ASAP or the statistical reporting programs would not be ready for the TDMS beta test.

Figure 3 is a graphic of the most available and stable testing system:

Figure 3. Testing Environment



With this configuration the reliability expectation was most favorable and one that could be controlled. This was also the best contigency plan since the expectation should be that the availability of interfaces, firewalls and the database will always be precarious.

The following developmental phases cite major episodes encountered during development and steps taken to deal with them and the resolution. These are not independent process and in almost all cases development crossed over several phases.

DEVEOPLMENT PHASE 1 – PORTING AND RUNNING

The old system code and data were taken directly from old system work directories and ported to a new testing environment. Initially this was a PC environment since the firewall precluded the developers from entering. Eventually this issue was resolved and the SAS code was ported to Linux. Programs were implemented to fit the new file system. Looking at Figure 3 the initial programmer requests were from SAS batch. The old data was used until the queries could be developed.

Once programs were running code was developed to simulate an HTTP post. Basically an HTTP post would be sent by the TDMSE system and received by SAS/IntrNet as a group of macro variables. This file resmbles what was used for the hypothetical post:

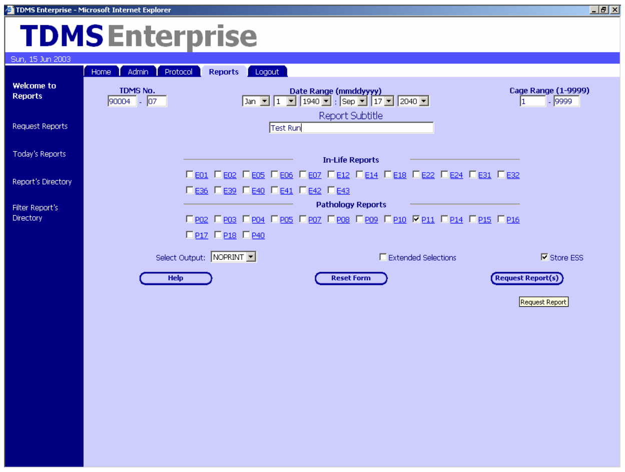
```
%let studynum     = &textStudyNum;
%let testnum      = &textTestNum;
%let repstartdate = &textStartDate;
%let rependdate   = &textEndDate;
%let startcage    = &textStartCage;
%let endcage      = &textEndCage;
%let sendtoprt    = &textPrinter;
%let ntpreport    = &textArrayReport;

many other parameters were also passed along
```

Once the Appserver was online programs were then run with a web browser. Running programs with a web browser would allow the Appserver to carry out a programming request rather than a batch request. The state of the programming environment during this programming invokation is closer to the environment that would exist once the system was up and running.   This is what launching a program via the application dispatcher might look like:

```
webserver/directory/broker?
_program=lib.pgm.sas&_service=default
```

Once HTML test pages were made available prgram request could then be made via those pages and a real HTTP post would be sent. This is what a test page looked like:

Figure 4. HTML Test Page



Sun, 15 Jun 2003

TDMS No. 90004 - 07

Date Range (mmddyyyy) Jan 1 1940 : Sep 17 2040

Cage Range (1-9999) 1 - 9999

Report Subtitle: Test Run

In-Life Reports

☐ E01 ☐ E02 ☐ E05 ☐ E06 ☐ E07 ☐ E12 ☐ E14 ☐ E18 ☐ E22 ☐ E24 ☐ E31 ☐ E32
☐ E36 ☐ E39 ☐ E40 ☐ E41 ☐ E42 ☐ E43

Pathology Reports

☐ P02 ☐ P03 ☐ P04 ☐ P05 ☐ P07 ☐ P08 ☐ P09 ☐ P10 ☑ P11 ☐ P14 ☐ P15 ☐ P16
☐ P17 ☐ P18 ☐ P40

Select Output: NOPRINT   ☐ Extended Selections   ☑ Store ESS

This is what a typical HTTP post looks like:

```
Symbols passed to SAS

#symbols: ???
 "_SRVNAME" = "webserver"
 "_SRVPORT" = "ZZZZ"
 "_REQMETH" = "POST"
 "_RMTHOST" = "XX.XXX.XXX.X"
 "_RMTADDR" = " XX.XXX.XXX.X"
 "_RMTUSER" = ""
 "_HTCOOK"  = ""
 "_HTUA"    = "Mozilla/4.0 (compatible; MSIE
              6.0; Windows NT 5.1)"
 "_mrvimg"  = "/sasweb/IntrNet8/MRV/images"
 "_grfaplt" = "/sasweb/graph/graphapp.jar"
 "_grafloc" = "/sasweb/graph"
 "_SERVICE" = "default"
 "_PROGRAM" = "devenv.test.sas"
 "_debug"   = "131"
 "_VERSION" = "8.2"
 "_URL"     = "/cgi-bin/broker.exe"
 "_ADMIN"   = "[your-name]"
 "_ADMAIL"  = "[your-email]@[your-site]"
 "_SERVER"  = "sashost"
 "_PORT"    = "YYYY"

Using timeout: 60
Content-type: text/html Pragma: no-cache

Content-type: text/html
```

Once the system was fully functional the testing environment scheme changed from the testing system (Figure 3) to the porposed revised system (Figure 2).

As stated before due to the precarious nature of software development system implementation and testing development efforts constantly wavered between the initial batch method to actual system program requests and everything in between.

DEVELOPMENT PHASE 2 – GENERATING OUTPUT

This portion of development and testing dealt mostly with converting the old text report into PDF format. Since development was now occuring on a Linux system the temporary files and data comparisons within SAS were still under the assumptions of a VAX/OpenVMS system. The issue of case sensitivity popped up often. Needless to say this was a tedious task to resolve.

Once the report result file was populated with information (whether correct or not) the text file was converted to PDF format using ODS. The problem here now became inserting page breaks in to a PDF file using ODS under SAS/IntrNet. It turns that SAS/IntrNet has default options that override the capability of inserting pages breaks into a PDF in this situation. This problem was resolved by the following setup before attempting to create the report file:

```
   options nonumber nodate spool;
   ods pdf file="&reportfile";
   title '00'x;
   run;

   create report file here…

   ods pdf close;
   ods listing;
   run; quit;
```

An issue of page orientation also arose. Since development was based on SAS 8.2 reports were relegated to just one orientation for PDF reports. SAS technical support has stated that SAS 9 is due to allow changing orientation for PDF documents.

DEVELOPMENT PHASE 3 – DEVELOPMENT OF QUERIES

Pertaining to code development this was an aspect of report programming that would change the face of the statistical component in a slightly more dramatic fashion. This is due to the introduction of the databse query into the reporting component. Previously this was a separate and independent process. But now that a new database exists those queries a have to be rewritten. The multi-generational development of the old system implies that there is no one expert on the system and those that might be are long gone.

To understand the effort needed to proceed with query development one must understand the elements of the Oracle database.

The database contains approximately 100 tables with the following breakdown by category: 20 core tables, 15 protocol tables, 25 reference tables and 40 intersection tables. There are just over 500 long term studies in the database along with a similar number of short term studies.

The process to develop queries to work with the new Oracle database was broken down into the following steps:

- Documenting existing queries - this was needed to make sure that the new queries would match the existing queries.
- Learning the structure of the new database – this step was needed since the new database design was very different from the old design.
- Development and testing of new queries outside the report system – this included comparing data output from the existing queries to the data output from the new queries.
- Integration of queries into report programs - this step includes testing of the complete report and additional programming to allow the data to be subset during the running of the report.

In addition to the SQL statements that are used to extract the data from the Oracle tables the SAS code that contains the queries also contains additional SAS data steps that are used to manipulate the data after it is retrieved from the Oracle database. The sections of the SAS programs that are not queries are being moved to Linux from the VAX system with very few changes for the first version of the new system. Over time the non-query sections will be updated to make them more efficient by using more built in SAS procs and less cumbersome data step programming.

This is an example of the code used to extract information from the Oracle databse:

```
   proc sql;
   connect to oracle(user=xxx orapw=yyy
   path=&path);
   create table sacrif2  as
   select * from connection to oracle
   (
   select
     animal_num as an_no,
     p_treatments.treat_num as treat,
     r_reference_groups.code as remove
   from
   &ntpdatabase..c_animals,
   &ntpdatabase..p_experiments,
   &ntpdatabase..p_treatments,
   &ntpdatabase..r_reference_groups,
   &ntpdatabase..c_animal_removal_events
   where
   c_animals.pexp_pac_id
     = p_experiments.pac_id and
   c_animals.ptre_pac_id
     =p_treatments.pac_id and
   p_treatments.pexp_pac_id
     =p_experiments.pac_id and
   c_animal_removal_events.RREFG_RAC_ID_ANM_RML_
   RSN
     =r_reference_groups.rac_id and
   c_animal_removal_events.canl_cac_id
     =c_animals.cac_id and
   r_reference_groups.reference_type
     like 'ANIMAL_REMOVAL_REASONS' and
   p_experiments.study_num=&ntpstudy and
   p_experiments.test_num=&ntptest
   );
   disconnect from oracle;
```

DEVELOPMENT PHASE 4 – REPORT INSTANTIATION

Communication with the Oracle database brought on a new set of problems. In order for the TDMSE to see a report an entry had to be defined in the work queue. The process was carried out within SAS as follows: initialize the table with a report entry, enter the report location, and finally signal that the process in finished. Problems arose due to the databse requirements for proper information.

This is a query to define a report as started in the Oracle database:

```
   proc sql;
   connect to oracle(user=xxxxx  orapw=yyyyyy
   path=&ntpsid);
   execute
   ( insert into &ntpdatabase..u_reports
   values (&operatorid,
           &experimentid,
           &sequenceid,
           %str(%'&sysdate.%'),
           NULL,
           NULL,
           &store_ess,
           'application/pdf',
         NULL,
         %scan(&status_rac_id,&estat),
         &reportid,
         %str(%'&txtsubtitle.%')
   )) by oracle;
   disconnect from oracle;
```

This is the SAS call to the Java code to insert the report into the Oracle database:

```
   %let temp_out =
   "/usr/java/j2sdk1.4.1_01/bin/java
   -classpath ""/data1/asi/ntplib/ojdbc14.jar:
   /data1/asi/ntplib"" FileToClob userid
   password database uniqueid reportfile";
   run;
   systask command &temp_out shell="tcsh";
```

The above Java code was developed outside the scope of this presentation. It is only shown for completeness of covering the problems encountered during devleopment. SAS SQL under version 8.2 does not allow an insert of data into an Oracle database. So the process of locating the PDF file had to be done in an external process from SAS. According to SAS techincal support this problem is ameliorated in SAS version 9.

Another noteworthy item is that during program runs with SAS/IntrNet the programming environment variable setting are lost. So even though the path to Java could be defined in the programmer's environment setting the external process called from SAS still does not know the location of the Java program. So the entire path for the location of Java had to be used for the systask call.

Once the program was inserted in to the Oracle database the report request was complete. This is a query to define a report as finished in the Oracle database:

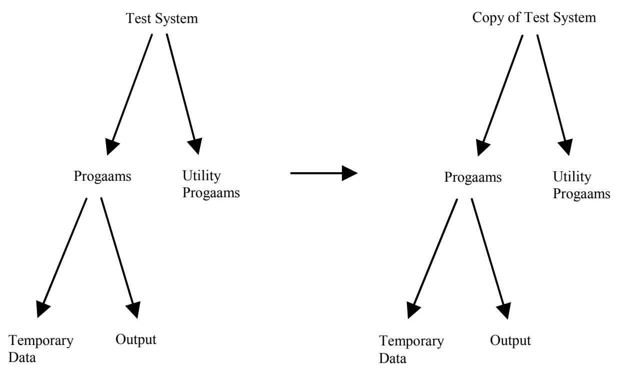
```
   proc sql;
   connect to oracle(user=xxx orapw=yyy
   path=&path);
   execute
   ( update &ntpdatabase..u_reports
   set
   completion_date = %str(%'&sysdate.%'),
   report_error = NULL,
   rrefg_rac_id_status =
   %scan(&status_rac_id,&estat)
   where
   id = &sequenceid
   ) by oracle;
```

Note: it is necessary to have the Oracle database know that a NULL field is being entered for missing or unkown information. This is not a major endeavor but it has to be communicated with the administrators. Otherwise errors and/or false operations will result.

DEVELOPMENT PHASE 5 – TESTING AND DEVELOPMENT

For this effort the system directory structure played a crucial part in how team members carried out testing. Once programs were reasonably functional team members were able make copies of the system and test very different aspects of the programs. But the key was that the programs had to be reaasonably functional.

Figure 5. Sytem Directory Structure



And clearly the ability to test the system heavily had a great impact on how the system was further developed. It did not matter at what point testing needed to occur. Team members could test the system from the batch level or the higher system level.

CONCLUSION

It should be clear from the described development phases that there was a steady level of complexity present throughout system development and testing. The team consisted of anywhere from 4 to 6 personnell working at any one time on the system revision.

It is the hope of the authors that the reader now has an idea of what a possible testing environment might look like for developing a large system which requires SAS/IntrNet and/or interaction with an Oracle database.

Another noteworthy observation that is that to date various projects utilizing SAS/IntrNet require a fair amount of contact with system administrators and a knowledge of the systems at play and the internet capabilities or limitations.

REFERENCES

SAS Web Tools: Static and Dynamic Solutions Using SAS/IntrNet Software Course Notes, 2001 SAS Institute Inc.

SAS Web Tools: Advanced Dynamic Solutions Using SAS/IntrNet Software Course Notes, 2001 SAS Institute Inc.

ACKNOWLEDGMENTS

This paper was made possible by the laborious efforts of individuals employed by Constella Group, Inc. and the National Institute of Environmental Health Sciences.

CONTACT INFORMATION

Contact the author(s) at:

Gabriel Cano
Constella Group
2605 Meridian Parkway
Durham, NC 27713
Phone: (919) 313-7708
Fax: (919) 544-7507
Email: gcano@constellagroup.com
Web: www.constellagroup.com

Kevin McGowan
Constella Group
2605 Meridian Parkway
Durham, NC 27713
Phone: (919) 313-7554
Fax: (919) 544-7507
Email: kmcgowan@constellagroup.com
Web: www.constellagroup.com

Jean Orelien
Constella Group
2605 Meridian Parkway
Durham, NC 27713
Phone: (919) 313-7607
Fax: (919) 544-7507
Email: jorelien@constellagroup.com
Web: www.constellagroup.com

TRADEMARK INFORMATION