Strategic Procurement: The SAS® Solution for Supplier Relationship Management
Fritz Lehman, SAS Institute Inc., Cary, NC

ABSTRACT
The SAS Solution for supplier relationship management (SRM) is a Web-based application designed to help procurement professionals understand how corporate money is being spent and how suppliers are being used. A ranking component enables procurement staff to rank suppliers based on any set of available metrics. An optimization component computes an optimized supplier portfolio. Users can also perform ad hoc queries against the SRM warehouse.

This paper will identify the goals of strategic procurement and illustrate how the SAS Solution for SRM enables procurement professionals to achieve them. The Internet has changed the way people work. More and more SAS solutions and applications are Internet based. Attention will be given in this paper to the technology used in implementing this Web-based solution.

INTRODUCTION
SAS for Supplier Relationship Management (SRM) is a comprehensive solution of software and services that enable a client to collect, analyze and leverage all aspects of their supplier data and purchasing history. SRM provides visibility into the client's supply base, spend history and usage patterns - critical information that optimizes strategic sourcing activities. The intention of this paper is to give a brief overview of a single module within Supplier Relationship Management and also some of the technology used.

SAS SRM SPEND ANALYSIS ARCHITECTURE
SAS SRM Spend Analysis tool, also known as Procurement Vision, is a data driven thin client web based application with all visual components made of Java Server Pages (JSP).

SAS SRM SPEND ANALYSIS
SAS SRM Spend Analysis works with summary data created from the procurement Warehouse. It allows you to view and relate any of the dimensions in the warehouse.

WHAT AM I BUYING AND FROM WHOM
The two most common dimensions in a procurement warehouse are Supplier and Commodity. These two dimensions allow you to answer the questions of what am I buying and from whom. Other dimensions may be defined within the procurement Warehouse to further explore the data. Other typical dimensions are Time, Geography, and Corporate Structure.

The Supplier Dimension is generally made up of one to many levels. If your data has been pre-processed by a vendor such as D&B® then D-U-N-S® numbers may have been assigned to your suppliers. If this is the case then you will have the parent child relationships of all your suppliers. This may show that you have many separate contracts negotiated with different children of a single parent corporation.

The Commodity Dimension is almost always made up of more than a single level. You may have your own commodity groupings or you may use a data providers commodity codes. If you use UNSPSC™ codes then typically you would have four levels in your commodity dimension.

By crossing the two dimensions, you can get a very clear picture of what you buy and from Whom. SAS SRM Spend Analysis gives you the ability to display these dimensions. Figure 2 shows how SAS SRM Spend Analysis module looks with the demo data provided with the software. For the purposes of clarity, the next several screenshots after Figure 2 will focus on the right hand side of the HTML page. This frame is called the Viewer.
WHAT AM I BUYING?

Figure 3 shows a list of UNSPSC2 commodity codes. This is the highest level of commodity code and you can see from the Viewer all your purchases fall into 25 categories. Each row is a link except the total row. When you click on a row you drill down to the next level of commodity code. Drill down on Manufacturing Components and Supplies.

![Figure 3](image)

Figure 4 shows the Viewer frame after the drilldown. It now shows the UNSPSC4 commodity codes that make up the components of Manufacturing Components and Supplies.

![Figure 4](image)

Drill down on Hardware to show the components of the UNSPSC4 code Hardware.

![Figure 5](image)

Figure 6 shows the Viewer frame after the drilldown. It now shows the UNSPSC8 commodity codes that make up the components of Hardware.

![Figure 6](image)
The Current View window is the upper left frame in SAS SRM Spend Analysis. It displays all the dimensions and allows you to see how you’ve drilled through each. Expand the Commodity Hierarchy in the Current View window to see the complete drill path you’ve taken. Figure 7 shows the Current View Frame.

![Current View Window](image)

Figure 7

All values displayed in the Current View frame are links so you can navigate up through the dimension by clicking on them. If you wanted to navigate all the way back to the top of the dimension then you would click on root. This would set the dimension back to it’s original state.

**WHO AM I BUYING IT FROM?**

Using the combo box in the toolbar of the Viewer frame, switch the dimension to the Supplier Hierarchy.

![Viewer Frame](image)

Figure 8

The Viewer frame now contains a list of suppliers but what is most important is that it contains a list of suppliers that you use to supply Hardware. Hardware was the last drilldown done in the Commodity Hierarchy. Figure 9 contains the list of suppliers.

![Supplier List](image)

Figure 9

According to the table in Figure 9, there are 45 suppliers of the various types of Washers. This may be an area for consolidation. Sort the table by descending Current Year Spend by clicking on the Current Year Spend header.

![Supplier List](image)

Figure 10
Some of the suppliers have a very small spend amount. These suppliers may be a target for consolidation or possibly these purchases should be made with a pcard.

**FILTERS**

In addition to the ability to drill into the data, you can also filter the data. Add a filter to see if any supplier of Hardware is also Bankrupt. The Filter Builder is located in the lower left frame of SAS SRM Spend Analysis.

**Check the checkbox next to the supplier LKJ Corp. This adds this supplier to all future where clauses. Using the combo box in the Viewer toolbar, switch the dimension back to the Commodity Hierarchy. This will show you exactly what hardware you are purchasing from LKJ Corp. See Figure 13.**

Comparing the spend in Figure 14 with the spend for Shim Washers in figure 6 you see LKJ Corp supplies you with almost 30 percent of your Shim Washers. It might be a good idea to find an alternate supplier. What else do you purchase from LKJ Corp? Switch to the commodity hierarchy and navigate back up to the top of the Hierarchy using the Drillup button on the toolbar or the Current View frame. The checkmark on the Supplier Hierarchy will still be in effect so all lists will be subset for LKJ Corp. This will show you a list of all commodities being purchased from this bankrupt supplier.

Just one other commodity category is displayed in Figure 14 so the damage this supplier could cause would not be much greater that we saw in figure 13.
HOW IT ALL WORKS

Each time you do a Drilldown, Drillup, switch a dimension, hide or display a column, login or logout, Spend Analysis is generating SQL queries. All code generated is submitted to the SAS System through the IOM interface available in SAS® Integration Technologies. SAS/IT facilitates client/server communication across multiple vendor architectures by supporting multiple client/server interoperability standards.

SUBMITTING QUERIES

There are two types of queries made by SAS SRM Spend Analysis. Those that don’t require SAS System PROCEDURE or DATA Steps and those that do.

JSP IOM EXAMPLE – SUBMITTING SQL THROUGH JDBC

The vast majority of queries submitted in SAS SRM Spend Analysis fall into this category. These JSP’s generally collect information, a query is built and sent to the Back Tier for processing. The results are then used to re-draw the HTML for the next screen. The example in figure 17 is not actual source code for SAS SRM Spend Analysis but example of how to submit a query when PROCEDURE or DATA step code isn’t needed.

```jsp
<%@ page import="java.sql.*, java.util.*, java.text.SimpleDateFormat, com.sas.iom.SAS.IWorkspace, com.sas.iom.WorkspaceFactory, com.sas.rio.*" %>

String SQL=null;
prop.put("host","mktsrm1.pc.sas.com");
prop.put("port","5310");
prop.put("transcodingPreference","server");
prop.put("userName","CARYNT\sasdemo");
prop.put("password","sasdemo1");
prop.put("protocol","bridge");
try {
  ws = wfactory.createWorkspaceByServer(prop);
  Properties jdbcProperties = new Properties();
  jdbcProperties.put("userName","CARYNT\sasdem o");
  jdbcProperties.put("password","sasdemo1");
  jdbcProperties.put("librefs","pvmeta 'e:\\SRM\\allaccess\\webservers\\3.0\\demodat a\\metadata';");
  connection = new MVAConnection(ws.DataService(),jdbcProperties );
  statement = connection.createStatement();
  ResultSet rs = statement.executeQuery("select *
                                        from pvmeta.columns");
  while(rs.next()) {
    out.println(rs.getString("desc") + "<br>");
  }
  rs.close();}
catch(Exception e) {
  System.out.println(e);
or.println("Error"+e);
}
```
JSP IOM EXAMPLE – SUBMITTING SAS PROCEDURE CODE

Some Queries require SAS system PROCEDURES or DATA steps to be executed. The example in figure 18 is not source code from SAS SRM Spend Analysis but is an example of how to submit PROCEDURE or DATA step code.

```java
<%@ page import="java.util.*, java.sql.*, com.sas.iom.SAS.IWorkspace, com.sas.srm.security.*, com.sas.servlet.util.*, com.sas.srm.common.*" %>
<%@ include file="../../includes/CommonJSPDirectives.jsp" %>
<html>
<head></head>
<body>

CustomDataHook PVHook = new CustomDataHook((User) session.getAttribute("PVUser"));
// Get the shared workspace for this user
IWorkspace ws = PVHook.getWorkspace();
String filename="";
String fileref="";
String localhost="";
int port;
SocketListener socket=null;
try {
// Setup our output socket
socket = new SocketListener();
port = socket.setup();
socket.start();
localhost = (java.net.InetAddress.getLocalHost()).getAddress();
fileref = " SOCKET ' + localhost + ":" + port + ";";
ws.LanguageService().Submit("title 'Procurement Vision';");
ws.LanguageService().Submit("title1 'Users of System';");
ws.LanguageService().Submit("filename sock " + fileref);
ws.LanguageService().Submit("ods html body=sock style=sasweb;"
" proc means data=pvtrans.requestlog
noproject;" +
" id datergerequested;" +
" var querytime;" +
" output out=totals sum=totaltime;" +
" proc print data=totals label noobs ;" +
" var username datergerequested totaltime _freq_;" +
" label totaltime 'Total Time for/All Queries Made' " +
" datergerequested = 'Last/Accessed' " +
" _freq_ = 'Number of/Data Queries' " +
" username = 'User';" +
" run;" +
" ods html close;";
socket.write(out);
}
catch (Exception e) {
 e.printStackTrace();
}

*/ you must do this destroy on the CustomDataHook object when you are done. */
*/ This cleans up internal resources and adjusts some counters that enable */
*/ object sharing which allows SRM apps to know when to kill some of the shared */
*/ internals. */
PVHook.destroy();
%
</body></html>
```

Figure 18

CONCLUSION

SAS SRM Spend Analysis is a good tool for you to do simple “OLAP” like analysis of your procurement data. It can help you identify what you spend and with whom. In addition, it is a good example of mixing several different types of technologies to create a sophisticated web-based tool.

TRADEMARK CITATION

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA or other countries. © indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.