SEDS-Lite: Using Open Source Tools (R, BIRT, MySQL) to Report and Analyze Performance Data

Igor A Trubin[1] and **Shadi Ghaith**[2]

[1] IBM GTS ITS Performance Management/Testing/Scalability Services
iatrubin@us.ibm.com
[2] Performance Engineering Laboratory,
School of Computer Science and Informatics, UCD,
Dublin 4, Ireland,
shadi.ghaith@ucdconnect.ie

Statistical Exception Detection System (SEDS) is one of the variations of learning behavior based performance analysis methodology developed, implemented and published by the Authors. This paper takes the main SEDS tools – IT-Control Chart and Exceptions (Anomalies) Detector - and shows how they can be built by Open Source type of Business Intelligence tools, such as R, BIRT and MySQL along with spreadsheets. The paper includes source code, tool screen-shots and report input/output examples which demonstrate building/developing a light version of SEDS.

1. Introduction

Self-learning related techniques as a part of Application Performance Monitoring (APM) is gaining a lot of momentum and is treated as a part of CEP - Complex Event Processing. The Forrester Research [forrester.com]: "Competitive Analysis: Application Performance Management and Business Transaction Monitoring" emphasizes this as shown in the following quote "All products reviewed are using some form of statistical-based analysis to distinguish normal from abnormal behavior"

The Gartner research document [*gartner.com*] titled "*Magic Quadrant for Application Performance Monitoring*" also states that one of the important functionality dimensions and future direction of APM is "Applications Performance Analytics" and one of the future directions is the requirement of the "*behavior learning engine (BLE) capability that can discover the large-scale statistical patterns*".

This paper is an attempt to demystify this BLE capability by showing relatively simple ways to build an Application (SEDS-Lite) with the main elements of BLE by using free open-source tools.

SEDS-Lite, via its generated and analyzed IT Control Charts, will help to reduce severe incidents related to performance and capacity. It will also help to enhance the response time to detect performance anomalies and reduce manual work as well as providing smart alerting based on the system learnt behavior.

The statistical approach used for this is based on known MASF Control Charts technique [BUZ1995] and basically consists of the following major steps:

a. To group and select a representative reference set of time-stamped performance data (historical base-line).
b. To "learn" base-line behavior by calculating mean, Upper Control Limit (UCL) and Lower Control Limit (LCL).
c. To compare actual (e.g. last week) data with base-line data and if any data points exceed UCL or LCL to state the possible anomaly (exception) detection.

This technique was explained in greater detail and published in the numerous CMG papers (e.g. [BUZ1995,TRUB2002,TRUB2010]).

2. SEDS-Lite Architecture

The architecture of SEDS-Lite is shown in Figure 1. and constitutes the following:

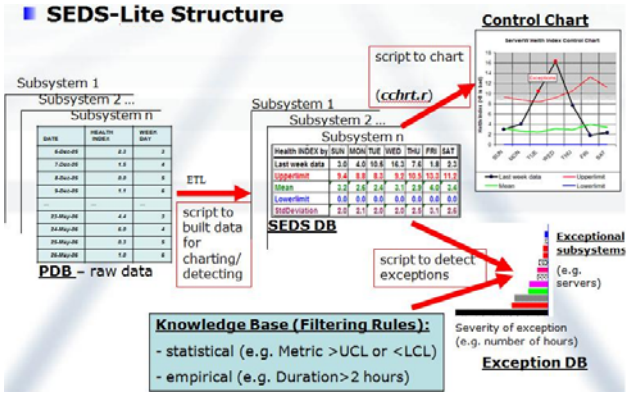

Figure 1 - SEDS-Lite

- **PDB**, Performance Database with time-stamped performance data for some number ("n") of dynamic objects (subsystems and/or metrics), the behavior of which is a subject to learn.
- **"Transformer**", the ETL process (script) to transform the data while calculating the base-line (mean, UCL, LCL) and actual (most recent day or week).
- **SEDS-DB** table(s) with calculated means and control limits that are put together with actual data.
- "**Grapher**" process (script) to plot an IT-Control Chart [TRUB2010]
- **"Detective",** the process to scan SEDS-DB data and to capture any exceptions (anomalies) and record them to **Exception-DB** (table) for reporting/alarming purposes.
- **"Reporter",** the process to report about Exceptional Systems with anomalies captured in bar chart of listing format.
- **Knowledge Base** ideally could be used to add additional non-statistical rules to the detection process for the purpose of reducing false alarms.

Those four data objects (**PDB, SEDS-DB, Exception-DB** and **Knowledge Base**) and four programs (**Transformer, Grapher, Detective and Reporter**) are all that is needed to build a SEDS-lite!

3. Brief Open-Sources Tools Review

Previous reviews of Open Source based Performance tools include the following:

- MeasureIT - Issue 2.05 - *Open Source Capacity & Performance* Management Tools for Windows & Unix Systems (2004 - Brian Johnson)
- Capacity Planning and Performance Monitoring with Free Tools (MSP-CMG'11 - Adrian Cockcroft)

Business Intelligence, Predictive Analytics and Generic Database open source tools are missing from the list above, Some options, shown in the list below, can be used for Performance Analysis and Reporting.

- BIRT - Business Intelligence and Reporting Tools; *http://www.eclipse.org/birt/phoenix/*
- Open Source Databases:
  - MySQL (*http://www.mysql.com/*)
  - NoSQL, PostgreSQL
- R – Statistical analysis programming system; *http://www.r-project.org/*

4. SEDS-Lite Implementation Options

Main SEDS-Lite parts can be built using multiple combinations of the Open Source tools listed above. The following table shows some of them.

SEDS components	Open Source tool		
	Option 1	***Option 2***	***Option 3***
PDB	MySQL	ODBC (MySQL)	ODBC (MySQL)
Transformer	MySQL	BIRT Cube	R/SQL
SEDS-DB	ODBC (MySQL)	BIRT Cube	R (data frame)
Grapher	BIRT	BIRT	R
Detective	BIRT (SQL)	BIRT (SQL)	R/SQL
Reporter	BIRT	BIRT	R

At some point, processing should be done via some ODBC (or JDBC) connectivity, so the source of row data can be in any other relational database with ODBC or JDBC accesses. For manual ad-hoc purposes, all of that SEDS-Lite functionality could be done by spreadsheet functions/formulas [TRUB2012]. In the following sections the sample scripts and reports are shown to demonstrate all three options in detail.

5. OPTION 1: MySQL and BIRT Usage

The raw data for all examples is just a three column table (***CPUutil***) with the CPU utilization metric, date and hour. As seen in the Figure 2, this can be uploaded to the MySQL schema (***ServerMetric***) by using the SQL script (***sqlScriptToUploadCSVforSEDS.sql***).

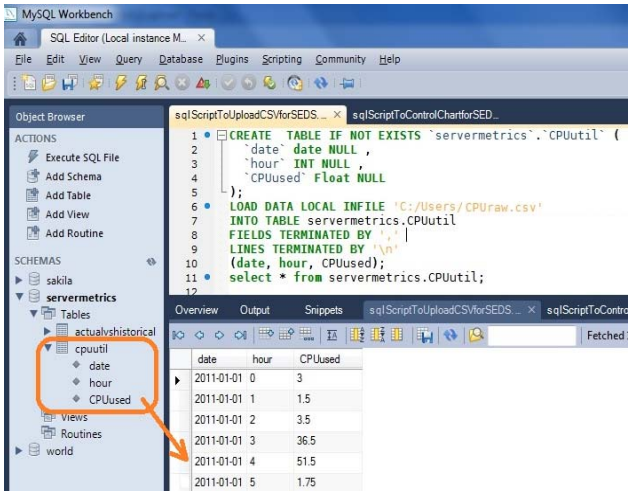

Figure 2 - MySQL as **PDB** with Raw Data

The next step is to transform raw data to "Cubical" format (multi-dimensional extensions of two-dimensional tables). The SQL script is shown in Figure 3 and it has the following parts:

- Create "**Actual**" temporary table which usually consists of the last week data and formed by "where" clause.
- Create "**Historical**" table.
- Create "**AggregateHistorical**" table to group data by the "**HoursInWeek**" field and to calculate control limits (in this case it is Mean +/- 3 Standard Deviation; for not normally distributed data. The 5th and 95th percentiles could be used instead).
- Create the result table named "**ActualVsHistorical**" by joining two previous tables on the "**HoursInWeek**".

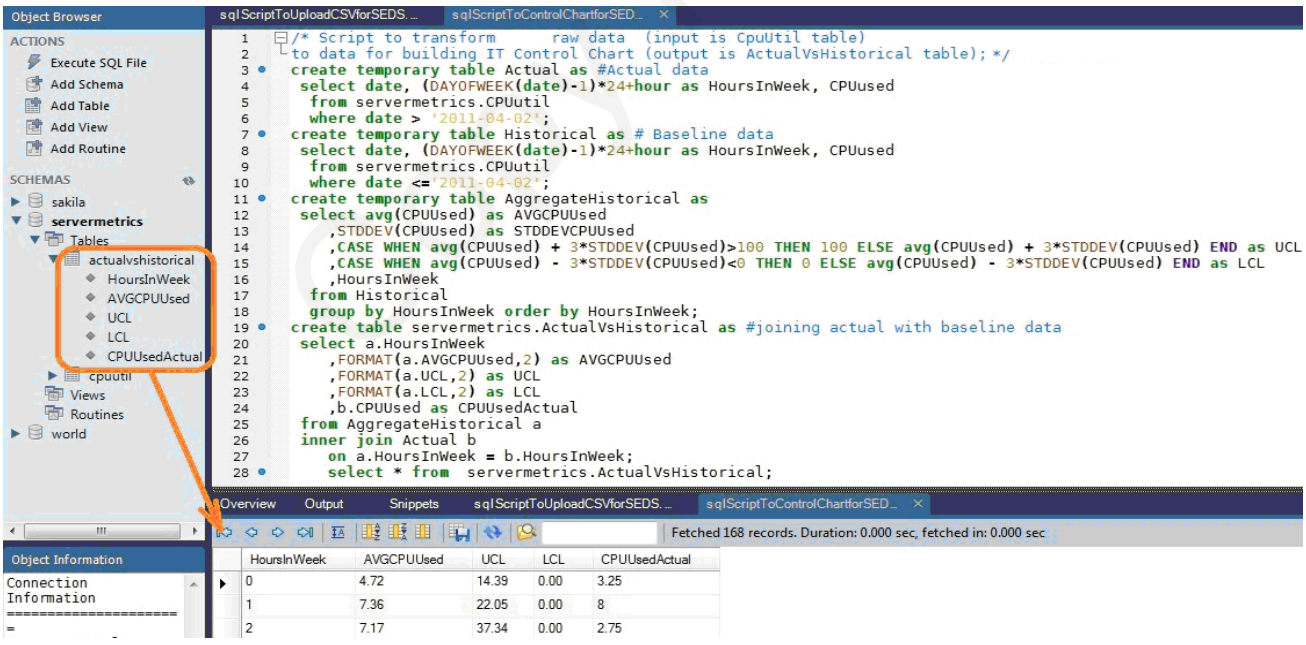

Figure 3 – SEDS-Lite Transformer

The result of running the **Transformer** script (***sqlScriptToControlChartforSEDS.sql)*** is the table shown at the bottom of Figure 3. Now the data is ready to be analyzed and/or plotted as the IT-Control Chart. (Detailed explanation how to build

and read IT-Control Charts is published in one of the CMG.org papers [TRUB2010].)

To plot IT-Control charts, the BIRT tool was used. First the connection to MySQL should be established via JDBC.

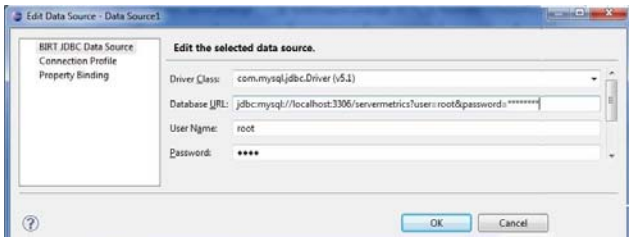

Figure 4 – BIRT->JDBC->MySQL

Figure 4 shows how the connection to MySQL database is established (to "**MySQLti**" with schema "**ServerMetrics**"). A simple select statement is defined to point to the right table to create the following dataset: "**ToControlChartData**" as shown in Figure 5.

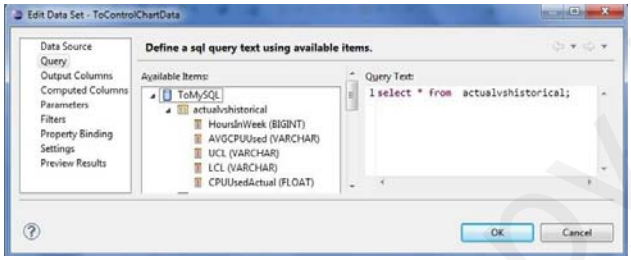

Figure 5 – Select from SEDS-DB

The final step is dropping "***Chart***" object from BIRT "**Palette**", specifying X and Y values and selecting data from "***ToControlChartData***". The screenshot is shown in Figure 6.

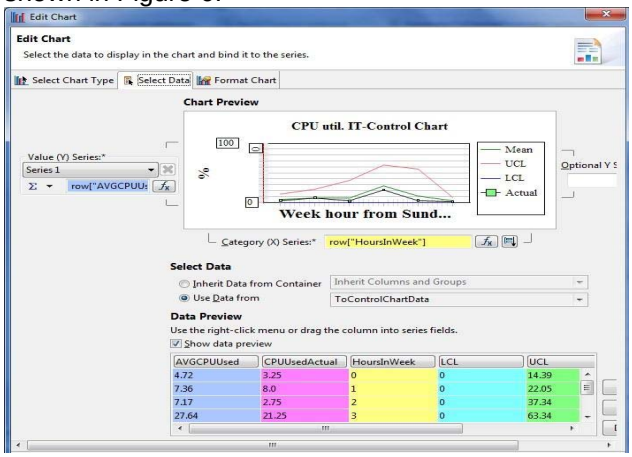

Figure 6 - BIRT "Chart" Object

The result of running this BIRT report can be seen in Figure 7. The simplest way to build MASF data for exception detection is to use 168 weekly hour (7 x 24) averages as a baseline and to show the UCL, LCL as well as the current (actual) week data on the same graph. This IT-Control Chart shows the set of statistically very unusual CPU usage values starting from Thursday morning (Week hour >100) as the actual data curve started exceeding the UCL beginning from that time.

6. EV-Control Chart

Another useful chart could be created from this SEDS-DB data. It is an EV-Control chart. The EV is the Exception Value meta-metric that was introduced at CMG'2001 [TRUB2001] as a measure of the anomaly (exception) severity. It is the difference (integral) between actual data and control limits as showed in Figure 8.

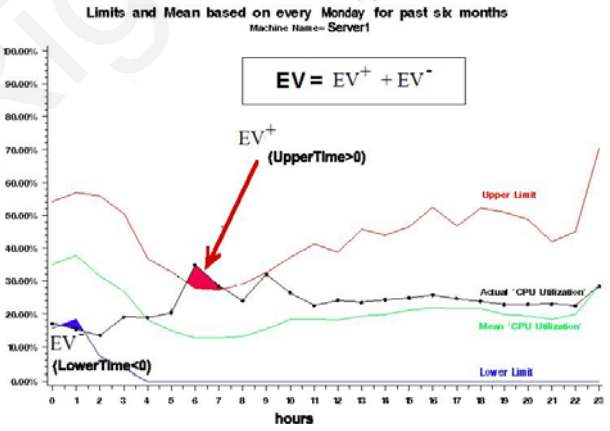

Figure 8 – Exception Value Calculation

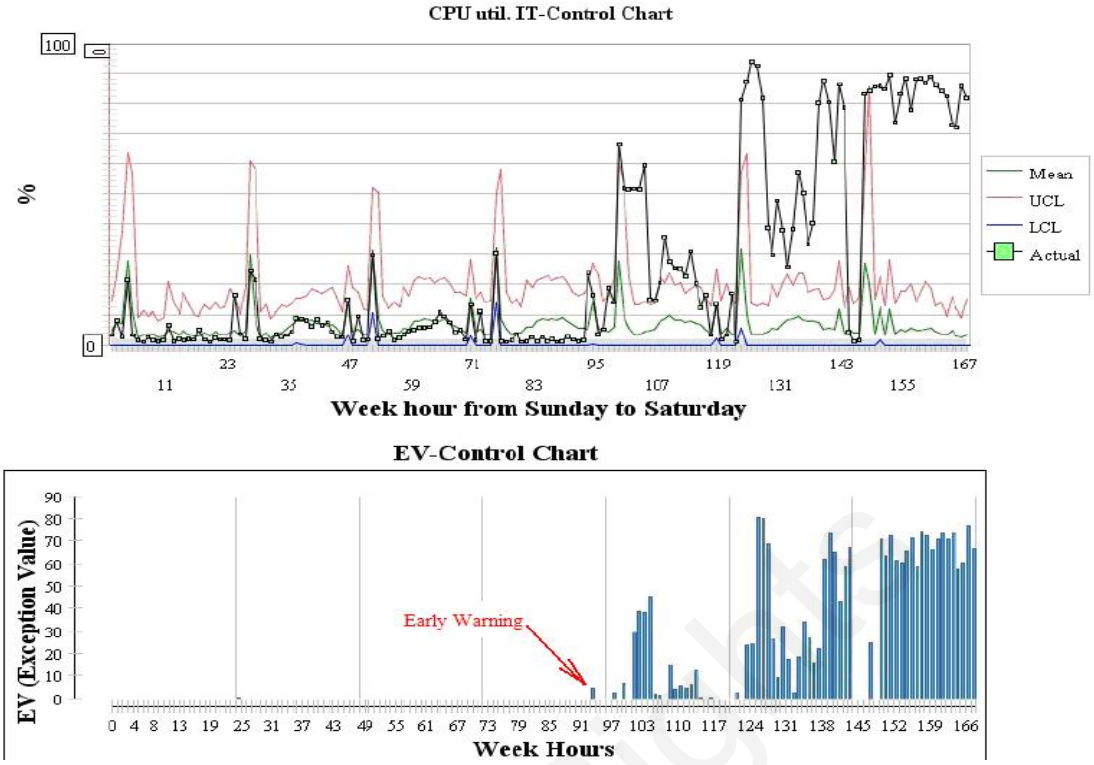

Figure 7 – IT- and EV-Control Charts

In the CMG'08 paper *"Exception Based Modeling and Forecasting"* [TRUB2008] the EV metric was plotted, Figure 9, using a spreadsheet to explain how it could be used for a new trend starting point recognition.

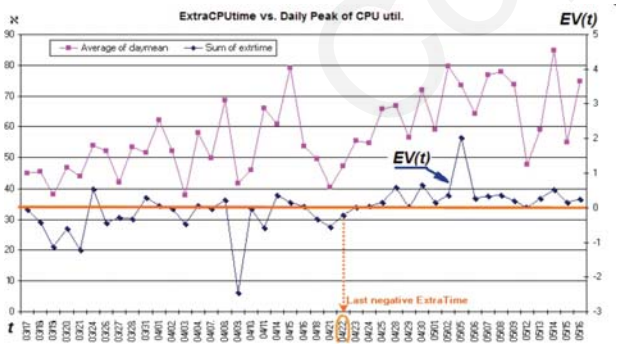

Figure 9 – Recent Trend Automatic Recognition by Using EV.

Plotting that meta-metric and/or its components (EV+ and EV-) over time, gives a valuable picture of system behavior. The next screenshot shows how that metric can be calculated (Figure 10).

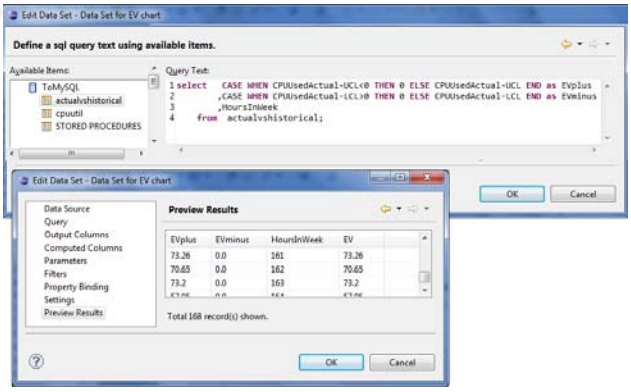

Figure 10 – Exception Value Calculation

The second chart in Figure 7 shows the BIRT example of the EV-chart against the same sample data (Figure 2). The EV-Control Chart clearly shows when the exceptions occurs, because if the system behavior is usual and stable then EV =0. While the higher the EV – the more severe the exception.

In a performance report it is useful to put both IT- and EV– control charts one under another to see a very useful picture of system behavior as shown in Figure 7.

7. OPTION 2: BIRT Data Cubes Usage

As it has been already explained in the CMG paper IT-Control Chart [TRUB2010], the data that describes the IT-Control Chart (or MASF control chart) has at least three dimensions (two time dimensions and one measurement - metric (Figure 11 is example from that paper).

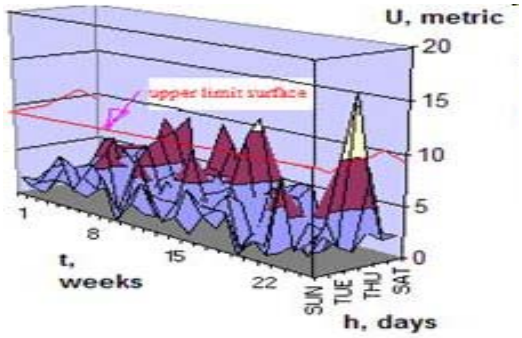

Figure 11 – **SEDS-DB** as a Data Cube

Actually the IT- Control Chart is a just a projection to the 2D cut with actual (current or last) data overlaid. Additional dimensions could be added as object names (e.g. server) and/or metric names (e.g. CPU , Memory utilization, Disk I/Os and so on) So, naturally, the OLAP Cubes data model (Data Cubes) is suitable for grouping and summarizing time stamped data to a *crosstable* for further analysis that includes a control chart generation.

That Data Cubes usage is a somewhat simpler way to implement SEDS-Lite "**Transformer**" and in some cases it does not require programming at all if modern BI tools (such as BIRT) are used.

By the way, in the spreadsheet type of tools (e.g. Excel) the usual method to build Data Cubes is the Pivot table. That could be also used as the SEDS-Lite transformer and the Pivot Chart wizard can be used to build IT-Control Charts [TRUB2012]

The following is an example of using the BIRT Data Cubes to build the SEDS-DB data against the raw data accessed via JDBC from MySQL.

Before building the Cube, the following three data sets are built using BIRT "Data Explorer":

- The Reference set or base-line (just "Data Set" in the Figure 12) is based on the input raw data with some filtering and computed columns (*weekday* and *weekhour*);

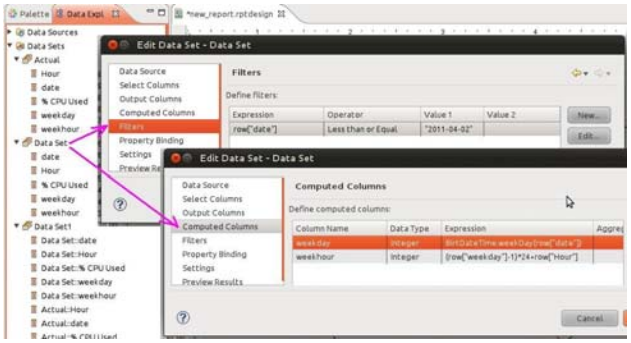

Figure 12 – Defining Base-line as a Data Set for BIRT Cube

- **The Actual data** set which is the same but having a different filter: (*raw{"date"} Greater "2011-04-02"*).
- **The combination of both data sets** for comparing base-line vs. actual; the "Data Set1" (in Figure 13) is built as a "Joint Data Set" by the following BIRT Query builder.

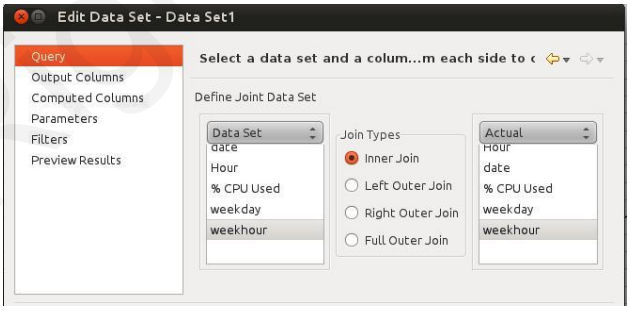

Figure 13 – BIRT Query Builder Usage to Join Base-Line with Actual Data

Then the Data Cube is built in the BIRT Data Cube Builder with the structure shown in Figure 14.

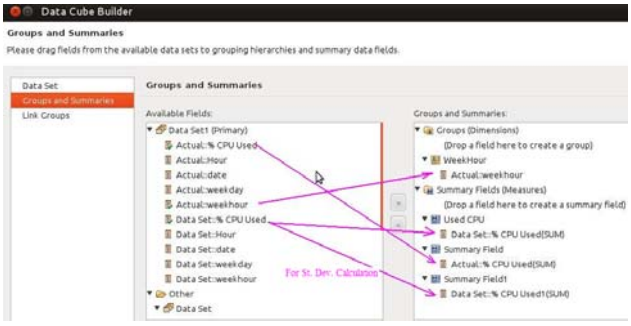

Figure 14 – BIRT Data Cube Builder

Note only one dimension is used here – *weekhour* as that is needed for the Cross-table report below in Figure 15.

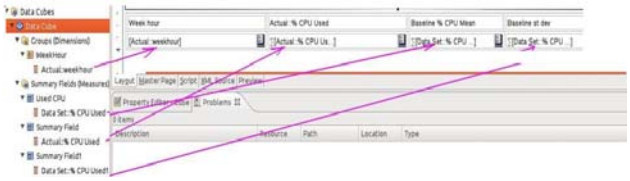

Figure 15 – BIRT Cross-table Object Usage

The final step is placing the “Chart” object in the report from BIRT “Palette” similar to how it is shown in Figure 5 and adding UCL calculations using Expression Builder for additional Value (Y) Series as shown in Figure 16.

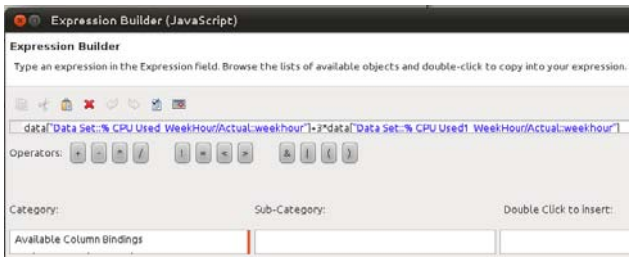

Figure 16 – UCL Calculation During BIRT Chart Building

The result of running “Cube” based report is the same as seen before in Figure 7. The BIRT report package can be exported and submitted for running under any type of web portal.

The report can be designed in BIRT with some parameters. For example, good idea would be to use a server name as the report parameter.

8. OPTION 3: R

There may be different ways to use an R programming system to build SEDS-Lite. The following is a “short cut” example that leverages already developed “**Transformer**” SQL script (Figure 3).

There is an R package named RODBC that can be used to run all types of SQL statements through ODBC.

The R script is shown in Figure 17 and it just sends those SQL statements to MySQL database, one by one, to build **SEDS-DB** data.

```
# R(ODBC)-script to transform raw date-hour data (input is CpuUtil table)
# to data for building IT Control Chart (output is ActualVsHistorical table)

library(RODBC)                      # Start by requesting the RODBC library
myConn <- odbcConnect("SETDStest")  # Now create an ODBC connection object
sqlQuery(myConn,                    # Actual data
 "create temporary table Actual as select date,     (DAYOFWEEK(date)-1)*24+hour as HoursInWeek, CPUused
      from servermetrics.CPUutil where date > '2011-04-02'")
sqlQuery(myConn,                    # Baseline data
 "create temporary table Historical as select date, (DAYOFWEEK(date)-1)*24+hour as HoursInWeek, CPUused
      from servermetrics.CPUutil where date <='2011-04-02'")
sqlQuery(myConn,                    # Control Limits Calculation
 "create temporary table AggregateHistorical as select avg(CPUUsed) as AVGCPUUsed ,STDDEV(CPUUsed) as STDDEVCPUUsed
          ,CASE WHEN avg(CPUUsed) + 3*STDDEV(CPUUsed)>100 THEN 100 ELSE avg(CPUUsed) + 3*STDDEV(CPUUsed) END as UCL
          ,CASE WHEN avg(CPUUsed) - 3*STDDEV(CPUUsed)<0   THEN 0   ELSE avg(CPUUsed) - 3*STDDEV(CPUUsed) END as LCL
          ,HoursInWeek
      from Historical
      group by HoursInWeek order by HoursInWeek")
sqlQuery(myConn,                    # joining actual with baseline data Control Limits and Avarage
 "create table servermetrics.ActualVsHistorical as select a.HoursInWeek ,FORMAT(a.AVGCPUUsed,2) as AVGCPUUsed
          ,FORMAT(a.UCL,2) as UCL ,FORMAT(a.LCL,2) as LCL ,b.CPUUsed as CPUUsedActual
      from AggregateHistorical a
      inner join Actual b on a.HoursInWeek = b.HoursInWeek")
cchrt <-sqlQuery(myConn,"SELECT HoursInWeek, CPUUsedActual, UCL, AVGCPUUsed,LCL FROM actualvshistorical")
```

Figure 17 – RODBC Script to Build SEDS-DB Data within MySQL Database

The result of running the **RODBC** script is identical to the data used for plotting IT-Control Chart by BIRT. The data itself can be seen by just typing the data frame name in the R-Console window as shown below in Figure 18.

```
> cchrt
   HoursInWeek CPUUsedActual   UCL AVGCPUUsed   LCL
1            0          3.25 14.39       4.72  0.00
2            1          8.00 22.05       7.36  0.00
3            2          2.75 37.34       7.17  0.00
4            3         21.25 63.34      27.64  0.00
5            4          3.25 56.77      10.83  0.00
6            5          1.50  9.02       2.67  0.00
7            6          1.00 10.85       3.25  0.00
8            7          2.50  9.40       3.22  0.00
9            8          1.50 10.77       3.50  0.00
10           9          1.25  7.84       2.49  0.00
11          10          1.50  9.12       2.91  0.00
12          11          6.50 20.65       4.79  0.00
13          12          1.25 13.60       4.47  0.00
14          13          2.00 10.39       3.49  0.00
15          14          1.50 17.06       4.95  0.00
16          15          2.00 13.46       3.41  0.00
17          16          1.75  9.89       3.02  0.00
18          17          5.00  9.48       2.76  0.00
19          18          1.75 12.89       4.20  0.00
20          19          1.00 11.56       3.00  0.00
21          20          2.25 13.57       4.58  0.00
22          21          1.75 11.97       3.44  0.00
...
```

Figure 18 – **SEDS-DB** Data Built by RODBC Script

The R "**Grapher**" script will build the same IT-Control Chart as seen in Figure 8; it is presented in Figure 19.

An obvious advantage of using the **RODBC** package is to let the MySQL system do transformation work against the PDB which should be done on a server. To build the IT- Control Chart as well as other analytical work (e.g. "**Detective**" work - exception detection) can be done on a client system that runs R.

```
# R script to plot IT-Control Chart against ODBC data

library(RODBC) # Start by requesting the RODBC library
myConn <- odbcConnect("SETDStest") # Now create an ODBC connection object
sqlQuery(myConn,"SELECT HoursInWeek, CPUUsedActual, UCL, AVGCPUUsed,LCL
      FROM actualvshistorical")
cchrt <-sqlQuery(myConn,"SELECT HoursInWeek, CPUUsedActual, UCL, AVGCPUUsed,LCL
      FROM actualvshistorical")

plot   (cchrt[,1],cchrt[,2],type="l",col="black",ylim=c(0,100.00),lwd=1.6,ann=F)
 points (cchrt[,1],cchrt[,3],type="l",col="red",  ylim=c(0,100.00),lwd=1,ann=F)
 points (cchrt[,1],cchrt[,4],type="l",col="green",ylim=c(0,100.00),lwd=1,ann=F)
 points (cchrt[,1],cchrt[,5],type="l",col="blue", ylim=c(0,100.00),lwd=1,ann=F)

mtext("%",side=2, line=3.0)
 mtext("Week Hours from Sunday to Saturday",   side=1, line=3.0)
 mtext("Utilization IT-Control Chart",        side=3, line=1.0)
legend(10,105,c("Actual","UpperLimit","Mean","LowerLimit"),
         col=c("black","red","green","blue"),lwd=c(2,1,1,1),bty="n")
```

Figure 19 – SED-Lite "**Grapher**" Script Written in R

9. "Detective" Work

How many standard deviations are used for upper (UCL) and lower (LCL) limit calculations on a control charts? 3? 1? What about 0?! Indeed, the simplest way to build MASF data for exception detection is to use 168 weekly hour averages as a baseline, so that would be the case when ZERO standard deviations are used to make UCL=LCL!

For further simplification, the current data could be included in a wider historical baseline (Why not?). The EV meta-metric in this case would be just the difference between the actual metric value and the average over baseline!

Here is an example of the SQL script (Figure 20) to implement that approach. It is based on a real script which was actually developed and successfully tested against real data for the BIRT tool to get exceptional servers list.

The script calculates EV and sorts the output list by that meta-metric to place servers with the most severe exceptions on the top.

```
-- DB2-like SQL script to detect anomalies using EV concept ;

select System_Name, SUM(ExceptionValueOfMetric) as weeklyExceptionValue
from ( select a.System_Name, a.day_of_week, a.hour_of_day, a.CPU_Util_curent, b.Metric_baseline
             ,a.Metric_curent-b.Metric_baseline as ExceptionValueOfMetric -- EV calculation
       from
           ( select System_Name, day_of_week, hour_of_day
                  , MetricValue as Metric_curent -- PDB metric
             From PDB where DATE > CURRENT DATE - 7 DAYS -- actual data to compare with baseline
              group by "System_Name", day_of_week, hour_of_day
              order by System_Name, day_of_week, hour_of_day
            ) a,
           ( select System_Name, day_of_week, hour_of_day
                   , MetricValue as Metric_baseline -- PDB metric
             From PDB where DATE > CURRENT DATE - 180 DAYS  -- baseline
              group by System_Name, day_of_week, hour_of_day
              order by System_Name, day_of_week, hour_of_day
            ) b
       where   a.System_Name=b.System_Name   -- baseline and actual data join
           and a.day_of_week=b.day_of_week and a.hour_of_day=b.hour_of_day
       order by a.System_Name,a.hour_of_day,a.day_of_week
    )
group by System_Name
order by  weeklyExceptionValue DESC -- most exceptional systems will be in the top  of the output
```

Figure 20 – SEDS-Lite "Detective" SQL Script to Detect

10. Summary

In this paper, we presented how to build a light version of SEDS (a BLE approach for managing the performance of applications) with open source tools that include R, BIRT and MySQL. Those tools can be used in various combinations to achieve the required SEDS functionality; three of those combinations were presented and analyzed.

The suggested version (SEDS-Lite) has the following sub-features of SEDS:

A. Generation of IT- and EV- Control charts, those will have the following benefits:
   - Help to reduce system defects and severe incidents.
   - Achieve faster response time to detect performance anomalies.
   - Reduce manual work.

B. Detect Anomalies and report on them, which will have the following benefits:
   - Reduce severe system incidents related to performance and capacity.
   - Provide smart alerting based on system (application) behavior learning and can be used for truly pro-active capacity management.
   - For full features like trend detection and demand forecast, the full SEDS version (described in a number of CMG papers below) should be considered.

11. Acknowledgement

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).

12. References

[BUZ1995] Jeffrey Buzen and Annie Shum: "MASF - Multivariate Adaptive Statistical Filtering", Proceedings of the Computer Measurement Group, 1995, pp. 1-10.

[TRUB2002] Igor Trubin: "Global and Application Levels Exception Detection System, Based on MASF Technique", Proceedings of the Computer Measurement Group, 2002.

[TRUB2010] Igor Trubin: "IT-Control Chart", Proceedings of the Computer Measurement Group, 2010.

[TRUB2001] Kevin McLaughlin and Igor Trubin, "Exception Detection System, Based on the Statistical Process Control Concept", Proceedings of the Computer Measurement Group, 2001

[TRUB2012] Igor Trubin, "How to Build IT-Control Chart - Use the Excel Pivot Table! ", "System Management by Exception" tech. blog www.itrubin.blogspot.com

[TRUB2008] Igor Trubin, "Exception Based Modelling and Forecasting", Proceedings of the Computer Measurement Group, 2008