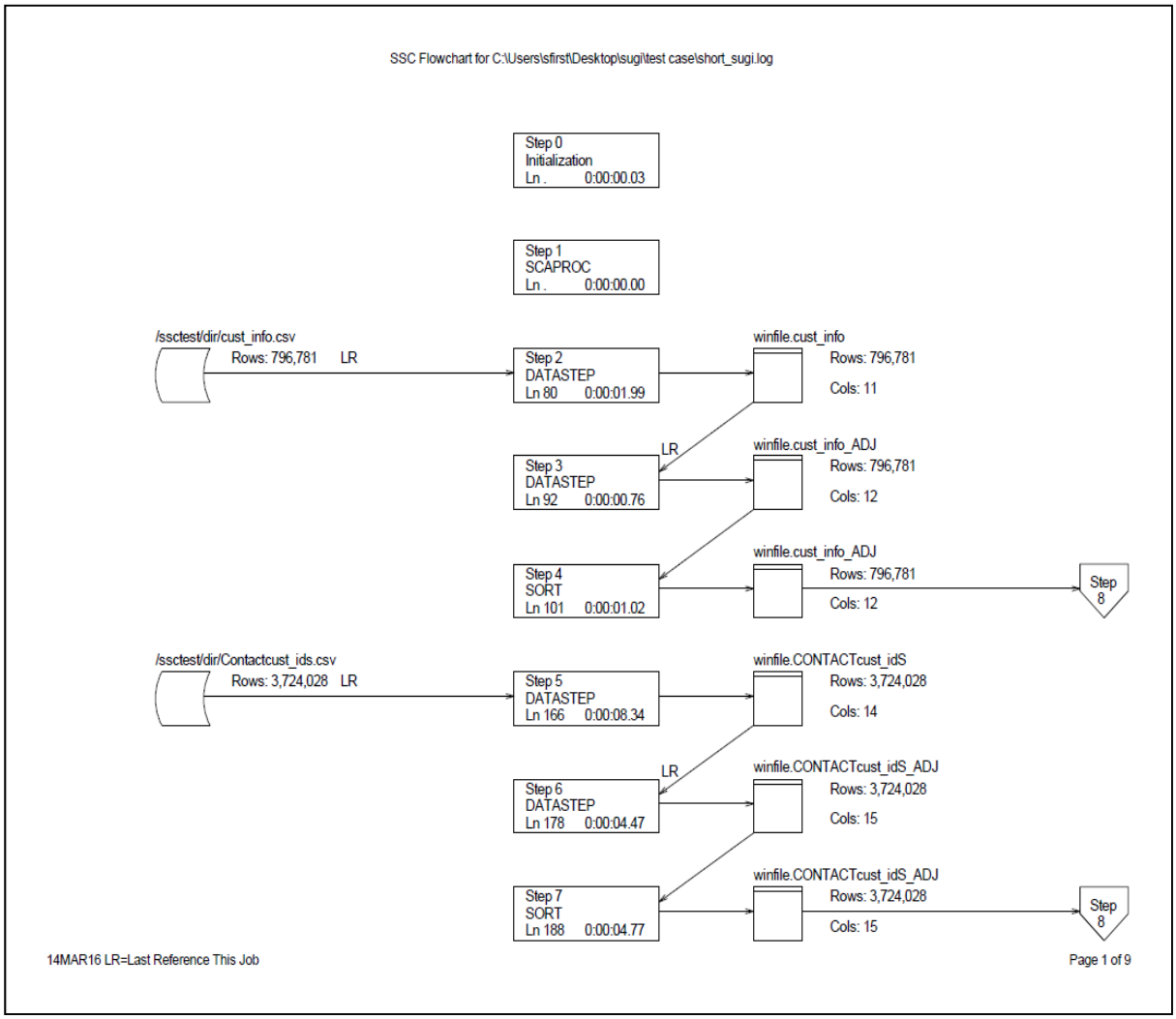
Paper 11580-2016

Innovative Performance Improvements Through Automated Flowcharts In SAS ®

Steven First, Systems Seminar Consultants, Inc.

ABSTRACT

One of the tedious but necessary things that SAS programmers must do is to trace and monitor SAS runs: counting observations and columns, checking performance, drawing flow charts, and diagramming data flows. This is required to be able to audit results, find errors, find long running steps, identify large data files, and to simply understand what a particular job does. Enterprise Guide includes functionality to help with some of this, on some SAS jobs. This paper presents an innovative custom tool that automatically produces flow charts -- showing all SAS steps in a job, file counts in and out, variable counts, recommendations and generated code for performance improvements, and more. The system was written completely in SAS using SAS source programs, SAS logs, PROC SCAPROC output, and more as input. This tool can eliminate much of the manual work needed in job optimization, space issues, debugging, change management, documentation, and job checkout. As has been said many times before: “We have computers, let’s use them”.



Output 1

INTRODUCTION

How did this all begin?

A very common task is to look at a SAS job and try to make sense of what it is doing. Many questions come up:

1. Did it have errors?
2. Are the row counts reasonable?
3. The job ran out of space, where was the problem, and what do I do to fix it?
4. Does my job carry along extra baggage like, files no longer needed, files and columns that are never used?
5. Are columns too wide?
6. Would compression help or hurt?
7. Where is every place that a file is used?
8. What is the data lineage of a column that has passed through multiple steps?
9. A graphical representation may be easier to understand and help with documentation.
10. Or the worst sentence you want to hear yourself or others ask: Just what does the job do?

As I have looked at my own SAS jobs, or other user's jobs, one of the first things that I would do, is to start sketching a rough flow chart with rectangles to represent SAS steps, other rectangles to represent SAS tables and database tables, tape symbols for tape files, disk symbols, and reports. I just used the standard flow chart symbols for each of them, but then I started adding row counts in and out, column counts out, and connected them with lines showing sources and destinations.

The SAS source may be a start, but the SAS log contains much more than just source code. Hopefully enough of the notes and source options are turned on and the log is available for review. If the log is available, I would manually write in row counts for most of the files in and out. If space was an issue, I would estimate used space and perhaps draw a rectangle showing used and available space at each step.

Below is an abbreviated SAS log. Note that lines have been deleted and the line numbers don't necessarily run consecutively. I have bolded some of the great information that the log contains.

```
NOTE: SAS initialization used:
      real time           0.03 seconds
      cpu time            0.02 seconds

27         proc scaproc;
28         record "/ssctest/scaproc.txt" attr opentimes expandmacros;
29         run;

NOTE: PROCEDURE SCAPROC used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

31         libname winfile    '/ssctest/dir';
NOTE: Libref winfile was successfully assigned as follows:
      Engine:        V9
      Physical Name: /ssctest/dir

32         libname cust_id     '/ssctest/dir';
NOTE: Libref cust_id was successfully assigned as follows:
      Engine:        V9
```

```
           Physical Name: /ssctest/dir
33         libname filetemp  '/ssctest/temp';
NOTE: Libref FILETEMP was successfully assigned as follows:
      Engine:        V9
      Physical Name: /ssctest/temp

172       +    data winfile.cust_info    ;
173       +             %let _EFIERR_ = 0; /* set the ERROR detection macro variable
*/
174       +             infile '/ssctest/dir/cust_info.csv' delimiter = ',' MISSOVER
DSD lrecl=32767 firstobs=2 ;
175       +                informat cust_id_ $31. ;
176       +                informat Account_Number 9. ;
177       +                informat Account_Name $30. ;
184       +                format cust_id_ $31. ;
185       +                format Account_Number 9. ;
186       +                format Account_Name $30. ;
194       +             input
195       +                         cust_id_ $
196       +                         Account_Number
197       +                         Account_Name $
206       +             ;
207       +            if _ERROR_ then call symputx('_EFIERR_',1);
208       +             run;

NOTE: The infile '/ssctest/dir/cust_info.csv' is:
      Filename=/ssctest/dir/cust_info.csv,
      Owner Name=bsweinb,Group Name=sasdev,
      Access Permission=rw-r--r--,
      Last Modified=Mon Oct 19 09:13:37 2015,
      File Size (bytes)=118098274

NOTE: 796781 records were read from the infile '/ssctest/dir/cust_info.csv'.
      The minimum record length was 0.
      The maximum record length was 222.
NOTE: The data set winfile.cust_info has 796781 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           1.99 seconds
      cpu time            1.89 seconds

210       +data winfile.cust_info_adj;
211       + length cust_id    $128.;
212       + set winfile.cust_info;
213       + cust_id = lowcase(cust_id_);
214       + run;

NOTE: There were 796781 observations read from the data set winfile.cust_info.
NOTE: The data set winfile.cust_info_ADJ has 796781 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.76 seconds
      cpu time            0.57 seconds
          +proc sort data=winfile.cust_info_adj;
217       + by cust_id;
218       + run;
NOTE: There were 796781 observations read from the data set winfile.cust_info_adj.
NOTE: The data set winfile.cust_info_ADJ has 796781 observations and 4 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           1.02 seconds
      cpu time            1.64 seconds
```

```
220       +  data winfile.CONTACTcust_idS    ;
221       +            %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
222       +            infile '/ssctest/dir/Contactcust_ids.csv' delimiter = ','
MISSOVER DSD lrecl=32767 firstobs=2 ;
223       +               informat First_Name $13. ;
224       +               informat Last_Name $12. ;
225       +               informat cust_id_ $37. ;
235       +               format First_Name $13. ;
236       +               format Last_Name $12. ;
237       +               format cust_id_ $37. ;
238       +               format Contact_Number 10. ;
239       +               format Account_Number 9. ;
240       +               format Account_Name $31. ;
249       +            input
250       +                        First_Name $
251       +                        Last_Name $
252       +                        cust_id_ $
253       +                        Contact_Number
254       +                        Account_Number
255       +                        Account_Name $
256       +                        Parent_Account_Number
257       +                        Parent_Account $
258       +                        Type $
259       +                        Account_Messaging $
260       +                        Service_Team $
261       +                        Mail_Code $
262       +                        createdDate $
263       +                        lastModifiedDate $
264       +            ;
265       +            if _ERROR_ then call symputx('_EFIERR_',1);  266       +
run;

NOTE: The infile '/ssctest/dir/Contactcust_ids.csv' is:
      Filename=/ssctest/dir/Contactcust_ids.csv,
      Owner Name=bsweinb,Group Name=sasdev,
      Access Permission=rw-r--r--,
      Last Modified=Mon Oct 19 09:31:32 2015,
      File Size (bytes)=648408736

NOTE: 3724028 records were read from the infile '/ssctest/dir/Contactcust_ids.csv'.
      The minimum record length was 104.
      The maximum record length was 271.
NOTE: The data set winfile.CONTACTcust_idS has 3724028 observations and 14 variables.
NOTE: DATA statement used (Total process time):
      real time           8.34 seconds
      cpu time            7.83 seconds

268       +data winfile.Contactcust_ids_adj;
269       + length cust_id    $128.;
270       + set winfile.Contactcust_ids ;
271       + cust_id = lowcase(cust_id_);
272       + run;

NOTE: There were 3724028 observations read from the data set winfile.CONTACTcust_idS.
NOTE: The data set winfile.CONTACTcust_idS_ADJ has 3724028 observations and 15
variables.
NOTE: DATA statement used (Total process time):
      real time           4.47 seconds
      cpu time            3.63 seconds

274       +proc sort data=winfile.Contactcust_ids_adj;
275       + by cust_id;
276       + run;
```

WHY A FLOWCHART?

One of the best training exercises for intermediate students is to display an actual SAS log and then as a group describe what it does and perhaps document it. As students work it out, I would draw on a board flows and libraries with input and output files. I always thought that it would be useful to automatically generate those charts, but the task of connecting all the boxes and overlapping lines seemed like too difficult a problem.

One day it occurred to me, that if I limited the lines connecting steps and files to a single page and if the data ran off the page to use off page connectors, the problem became much simpler. More of this will be discussed later.

SAS Enterprise Guide does a very nice job of showing process flows and if a SAS job is imported, and run, it can show very useful and interactive diagrams showing connections and data flows. That process is much more advanced than what I did with the flow charts and I would love to have a process like that in our system. It does require something more than SAS such as JAVA, and we are basically SAS folks. We do have the underlying data though and could conceivably add a process flow sometime.

One disadvantage of using the EG process flow, is that the job would have to be rerun, and also items like row counts are not included. Our process relies at minimum on the SAS log and doesn't require much user change if any. We have had surprising good results with the flow chart program in some cases creating charts for hundreds of pages in length.

It is almost automatic documentation to run the logs through the flow charter and include the charts in documentation.

PARSING THE LOGS

An advantage to using the SAS log, is in most cases it is already there and doesn't require any user program changes. In addition row counts are available and are available nowhere else. PROC SCAPROC does add some additional capabilities that will be addressed later.

There is probably no better tool to parse the SAS log than SAS itself. As part of the SSC SAS Job Performance Tool, we have written SAS jobs that do parse the files and extract job, step, and file information. At first glance it would seem to be a relatively easy program to write, but there are lots of interesting nuances and questions. Even if reading a single job's log, there are considerable challenges.

1. The log was not intended to be program input, but rather to be read by a human.
2. Things like upper/lower case usage can be inconsistent, different releases alter what is written to the log.
3. Each operating system has differences that have to be dealt with.
4. If SAS/CONNECT remotely submits to another machine, there is effectively a new log in the middle of the old one.
5. If a SAS/GRID is in place, again there are multiple logs to deal with. All of this can complicate even a single job log.

There is also a very extensive logging capture feature, where the SAS installer keeps a copy of all SAS logs without the users doing anything or even knowing about the capture. This is a wonderful source for SAS administrators to review job history, trends, and extract performance information. These logs are much like the normal SAS logs, but also have date stamping and other information preceding the normal log lines. There are also provisions to break long running job's logs into separate files at midnight or other times, or when a certain file size is reached. The effect of this, is that now there are multiple logs for the same job that have to be related to each other. It actually is a quite complex bit of logic to do this correctly, but we were able to do it. After a few years of using our parsing programs we are getting quite dependable results.

As an aside, a SAS administrator that I know recently had a user who lost three months worth of programming when her SAS enterprise guide project became unreadable. The administrator was able to

go the system captured log, and with a small amount of editing recover her programs. This is not a recommended method of backup, but to quote the user: "I cried when I lost it, and I cried again when you recovered it".

ADVANTAGES OF PARSING LOGS

One of the most obvious reasons to parse a log, is that there is a tremendous amount of information that may be needed to read, but the row counts, timing and other good timing information is buried within lots of clutter. A recent log that we were examining for performance was over 50 megabytes and 1.4 million lines of text. Parsing this down to the thousands of steps and files was relatively fast and like all summarizing processes makes things easier to handle. It was very easy to spot the trends, locate longest running steps, and look for trouble spots from the input jobs.

Even though the topic of this paper is graphical, it is the gathering of good underlying data that was the most work and has the most value. The primary tables coming out of the log processing are a job level table, a step level table, and a file level table. The STEP and DATAFILE tables are then merged to create a table that we called JOB_FLOW.

THE STEPS DATA TABLE

STEPS contains a row for each SAS step detected in the job. Each step is numbered, line numbers where the step was detected are recorded, a PROC name, Real (elapsed) time), along with CPU time for the steps. Some operating systems and system options allow user times, system time, and excps (number of blocks of data read/written). Steps can easily be sorted and selected to find the most expensive steps in a job or installation.

P00_flowchart_driver file C:\Users\sfirst\Desktop\sugi\test case\short_sugi.log
steps

Obs	sas_step_no	Actual_log_line	Sas_log_line	mprocname	rsecs	csecs	u_csecs	s_csecs	excp
1	0	8	.	Initialization	0:00:00.03	0:00:00.02	.	.	.
2	1	15	27	SCAPROC	0:00:00.00	0:00:00.00	.	.	.
3	2	80	27	DATASTEP	0:00:01.99	0:00:01.89	.	.	.
4	3	92	210	DATASTEP	0:00:00.76	0:00:00.57	.	.	.
5	4	101	210	SORT	0:00:01.02	0:00:01.64	.	.	.
6	5	166	210	DATASTEP	0:00:08.34	0:00:07.83	.	.	.
7	6	178	268	DATASTEP	0:00:04.47	0:00:03.63	.	.	.
8	7	188	274	SORT	0:00:04.77	0:00:08.01	.	.	.
9	8	200	278	DATASTEP	0:00:04.14	0:00:03.35	.	.	.
10	9	210	283	SORT	0:00:04.80	0:00:07.01	.	.	.
11	10	236	353	DATASTEP	0:00:00.00	0:00:00.00	.	.	.
12	11	246	364	SORT	0:00:00.00	0:00:00.00	.	.	.
13	12	256	368	PRINT	0:00:00.03	0:00:00.01	.	.	.
14	13	312	403	DATASTEP	0:00:00.55	0:00:00.49	.	.	.
15	14	320	403	IMPORT	0:00:03.41	0:00:02.23	.	.	.
16	15	332	403	DATASTEP	0:00:00.41	0:00:00.36	.	.	.
17	16	345	65	SORT	0:00:00.00	0:00:00.01	.	.	.
18	17	359	469	DATASTEP	0:00:00.04	0:00:00.00	.	.	.
19	18	370	475	SORT	0:00:00.03	0:00:00.00	.	.	.

Output 2

THE DATAFILE DATA TABLE

DATAFILE contains a row for each file read or written to in the job. Again, each step is numbered, along with path, type information, file information, but row and column counts if available. Most procs and data steps report rows in and out and variables are reported for output files. For some reason PROC SQL does not display row counts in which is a big drawback. PROC SCAPROC can address some of those issues, but does require source changes.

Datafile

Obs	path	pathtype	engine	sas_step_no	ddn	member	dobs	memtype	i_o	dvars
1	/ssctest/dir/cust_info.csv			2			796781	RAW	In	.
2		SAS LIB		2	winfile	cust_info	796781	SAS DS	Out	11
3		SAS LIB		3	winfile	cust_info	796781	SAS DS	In	.
4		SAS LIB		3	winfile	cust_info_ADJ	796781	SAS DS	Out	12
5		SAS LIB		4	winfile	cust_info_adj	796781	SAS DS	In	.
6		SAS LIB		4	winfile	cust_info_ADJ	796781	SAS DS	Out	12
7	/ssctest/dir/Contactcust_ids.csv			5			3724028	RAW	In	.
8		SAS LIB		5	winfile	CONTACTcust_idS	3724028	SAS DS	Out	14
9		SAS LIB		6	winfile	CONTACTcust_idS	3724028	SAS DS	In	.
10		SAS LIB		6	winfile	CONTACTcust_idS_ADJ	3724028	SAS DS	Out	15
11		SAS LIB		7	winfile	CONTACTcust_idS_ADJ	3724028	SAS DS	In	.
12		SAS LIB		7	winfile	CONTACTcust_idS_ADJ	3724028	SAS DS	Out	15
13		SAS LIB		8	winfile	CONTACTcust_idS_ADJ	3724028	SAS DS	In	.
14		SAS LIB		8	winfile	cust_info_ADJ	796781	SAS DS	In	.
15		SAS LIB		8	winfile	cust_idS_ALL_SF	3724028	SAS DS	Out	15
16		SAS LIB		9	winfile	cust_idS_ALL_SF	3724028	SAS DS	In	.
17		SAS LIB		9	winfile	cust_idS_ALL_SF	3724028	SAS DS	Out	15
18	/ssctest/dir/exclude.txt			10			2	RAW	In	.
19				10	WORK	SPECIAL_EXCLUDES	2	SAS DS	Out	2
20				11	WORK	SPECIAL_EXCLUDES	2	SAS DS	In	.
21				11	WORK	SPECIAL_EXCLUDES	2	SAS DS	Out	2
22				12	WORK	SPECIAL_EXCLUDES	2	SAS DS	In	.

Output 3

THE JOBFLOW DATA TABLE

JOBFLOW is a simple join of the two earlier files. At this point, we have rough information for each step and file along with whether a file is input or output to the step. This file is adequate for a lot of the performance reports and can be used as a text report to tie out for audit purposes. The beginnings of files going out and into another step are beginning to be apparent, but in order to draw a flowchart, more values need to be computed.

P00_flowchart_driver file C:\Users\sfirst\Desktop\sugi\test case\short_sugi.log
job_flow

sas_step_no	Actual_log_line	Sas_log_line	mprocname	path	engine	ddn	member	memtype	i_o	dobs	dvars
0	8	.	Initialization							.	.

sas_step_no	Actual_log_line	Sas_log_line	mprocname	path	engine	ddn	member	memtype	i_o	dobs	dvars
1	15	27	SCAPROC							.	.

sas_step_no	Actual_log_line	Sas_log_line	mprocname	path	engine	ddn	member	memtype	i_o	dobs	dvars
2	80	27	DATASTEP	/ssctest/dir/cust_info.csv				RAW	In	796,781	.
	80	27	DATASTEP			winfile	cust_info	SAS DS	Out	796,781	11

sas_step_no	Actual_log_line	Sas_log_line	mprocname	path	engine	ddn	member	memtype	i_o	dobs	dvars
3	92	210	DATASTEP			winfile	cust_info	SAS DS	In	796,781	.
	92	210	DATASTEP			winfile	cust_info_ADJ	SAS DS	Out	796,781	12

sas_step_no	Actual_log_line	Sas_log_line	mprocname	path	engine	ddn	member	memtype	i_o	dobs	dvars
4	101	210	SORT			winfile	cust_info_adj	SAS DS	In	796,781	.
	101	210	SORT			winfile	cust_info_ADJ	SAS DS	Out	796,781	12

sas_step_no	Actual_log_line	Sas_log_line	mprocname	path	engine	ddn	member	memtype	i_o	dobs	dvars
5	166	210	DATASTEP	/ssctest/dir/Contactcust_ids.csv				RAW	In	3,724,028	.
	166	210	DATASTEP			winfile	CONTACTcust_idS	SAS DS	Out	3,724,028	14

sas_step_no	Actual_log_line	Sas_log_line	mprocname	path	engine	ddn	member	memtype	i_o	dobs	dvars
6	178	268	DATASTEP			winfile	CONTACTcust_idS	SAS DS	In	3,724,028	.
	178	268	DATASTEP			winfile	CONTACTcust_idS_ADJ	SAS DS	Out	3,724,028	15

sas_step_no	Actual_log_line	Sas_log_line	mprocname	path	engine	ddn	member	memtype	i_o	dobs	dvars

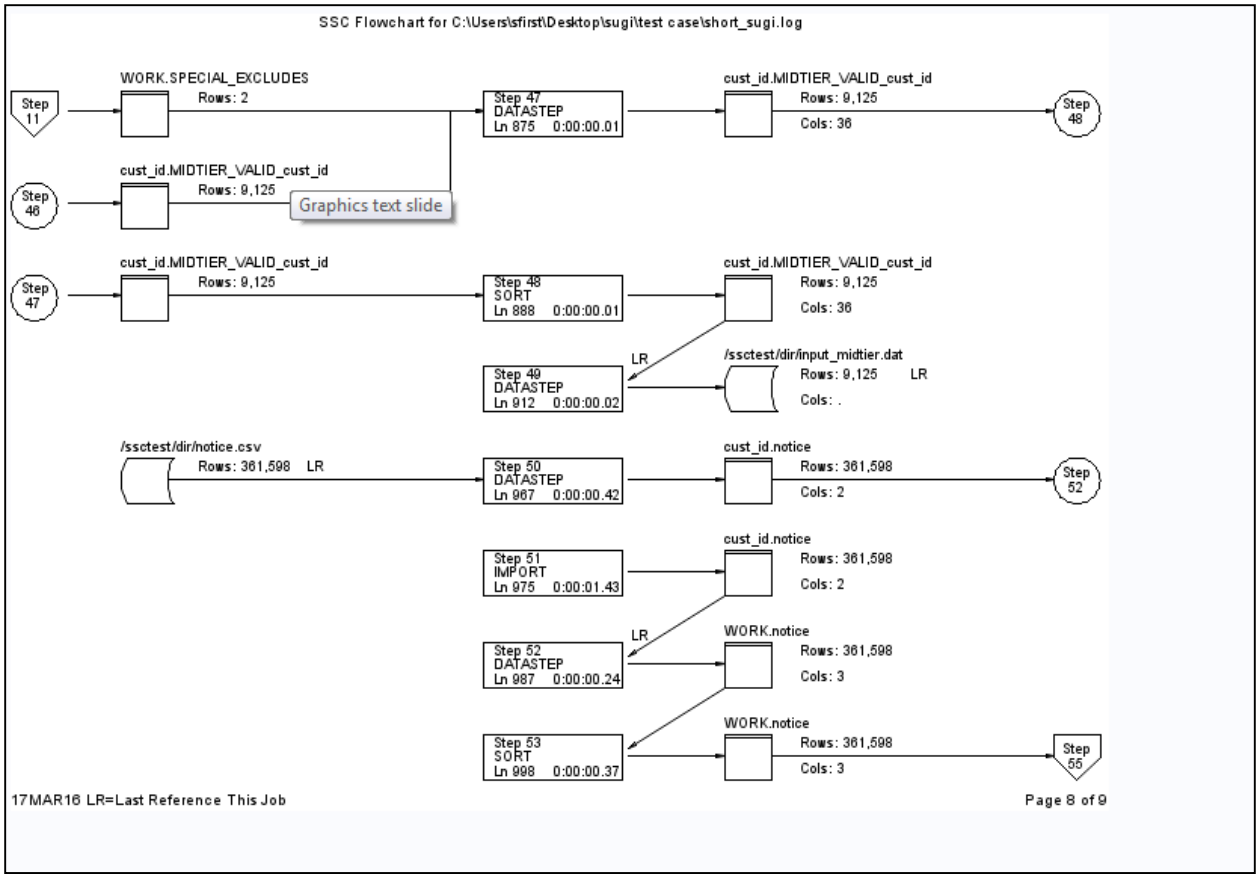
Output 4

DESIGNING THE FLOW CHART

To simplify the first iteration of the flow chart, I did the following:

1. The layout of the flowchart page would have three column and eight rows.
2. All charts are created as Landscape PDF files.
3. All steps are shown whether they have any input or output files.
4. Input files are normally on the left, output files on the right.
5. If the last output file from one step is used in the very next step, a diagonal arrow is drawn to the next step if possible. This logic was the most time consuming.
6. If a file comes from or goes to a different step on the same page, a “On page” connector is used.
7. If a file comes from or goes to a step on a different page, an “Off Page” Connector is used.
8. Pages are constructed and printed separately, any number of pages is allowed.
9. Row and column counts, timings, line numbers, and the last reference to a file are annotated.

Page 9 of the sample flow chart is shown below in Output 5.



Output 5

PROCESSING THE JOBFLOW DATA TABLE AND DESIGNING THE FLOW CHART.

Further refining of the JOBFLOW data to create effectively three columns and 8 rows to be drawn on the page is necessary. The file contains the data needed for drawing and annotating the chart. A series of steps basically created various three by eight element arrays to fill each page and then store those results in a file. The logic at times got intense, especially with diagonal line processing.

Along with the names, steps etc. the file contains the row where the symbols will be drawn (TROW), along with INSYMBOL and OUTSYBOL values set as follows for files:

INSYMBOL: 'None' if never output earlier,
'Conn' if referenced earlier on same page
'Diag' if referenced one step back by Outfile set to Diag
'Coff' if referenced by earlier page.
OUTSYMBOL: 'None' if never output later.
'Conn' if referenced later on same page
'Diag' if referenced one step later in next input file.
'Coff' if referenced by later page.

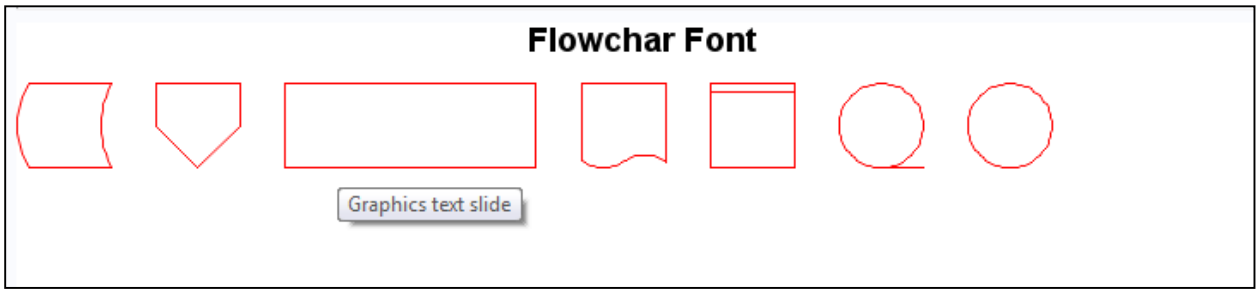
Below is the underlying data to draw page eight.

pageno	sas_step_no	i_last_ref	o_last_ref	trow	first_trow_this_step	insymbol	i_dsname	o_dsname	o_memtype	outsymbol	pred_step	succ_step
8	47			1	1	Coff	WORK.SPECIAL_EXCLUDES	cust_id.MIDTIER_VALID_cust_id	SAS DS	Conn	11	48
	47			2	1	Conn	cust_id.MIDTIER_VALID_cust_id			None	46	.
	48			3	3	Conn	cust_id.MIDTIER_VALID_cust_id	cust_id.MIDTIER_VALID_cust_id	SAS DS	Diag	47	49
	49	LR	LR	4	4	Diag	cust_id.MIDTIER_VALID_cust_id	/ssctest/dir/input_midtier.dat	RAW	None	48	.
	50	LR		5	5	None	/ssctest/dir/notice.csv	cust_id.notice	SAS DS	Conn	.	52
	51			6	6	None		cust_id.notice	SAS DS	Diag	.	52
	52	LR		7	7	Diag	cust_id.notice	WORK.notice	SAS DS	Diag	51	53
	53			8	8	Diag	WORK.notice	WORK.notice	SAS DS	Coff	52	55

Output 6

A CUSTOM FONT

So that I could easily draw flowchart symbols in a variety of sizes, I decided to build a SAS/GRAPH font. I generated all the points need to draw the following: Disk file, off page connector, step box, printed report, SAS table, tape reel, and on page connector. Other symbols can be added as needed.



Output 7

THE LAST DATASET

The final dataset is a standard SAS/GRAPH annotation dataset that contains the instructions to draw the lines, use the font for symbols, and place all the text on each page. The code is tedious but not especially difficult. The final chart was drawn with PROC GSLIDE and the annotation data set.

annods

STYLE	FUNCTION	COLOR	XSYS	YSYS	HSYS	WHEN	POSITION	X	Y	savelx	savely	saverx	savery	text
flowchar	label	Black	5	5	5	B	C	0.0	87.5	40	3.4	59	3.4	O
ARIAL	label	Black	5	5	5	B	C	0.9	91.0	40	3.4	59	3.4	Step
ARIAL	label	Black	5	5	5	B	B	1.0	89.0	40	3.4	59	3.4	11
ARIAL	move	Black	5	5	5	B	B	5.2	90.9	40	3.4	59	3.4	11
ARIAL	arrow	Black	5	5	5	B	B	10.0	90.9	40	3.4	59	3.4	11
flowchar	label	Black	5	5	5	B	C	10.0	87.5	40	3.4	59	3.4	S
ARIAL	label	Black	5	5	5	B	C	10.0	94.5	40	3.4	59	3.4	WORK.SPECIAL_EXCLUDES
ARIAL	label	Black	5	5	5	B	C	17.0	92.0	40	3.4	59	3.4	Rows: 2
ARIAL	move	Black	5	5	5	B	C	14.5	90.9	40	3.4	59	3.4	Rows: 2
ARIAL	draw	Black	5	5	5	B	C	40.0	90.9	40	3.4	59	3.4	Rows: 2
ARIAL	arrow	Black	5	5	5	B	C	43.0	90.9	40	90.9	59	3.4	Rows: 2
flowchar	label	Black	5	5	5	B	C	43.0	87.5	40	90.9	59	3.4	P
ARIAL	label	Black	5	5	5	B	C	44.0	92.0	40	90.9	59	3.4	Step 47
ARIAL	label	Black	5	5	5	B	C	44.0	90.0	40	90.9	59	3.4	DATASTEP
ARIAL	label	Black	5	5	5	B	C	44.0	88.0	40	90.9	59	3.4	Ln 875
ARIAL	label	Black	5	5	5	B	C	49.5	88.0	40	90.9	59	3.4	0:00:00.01
ARIAL	move	Black	5	5	5	B	C	56.3	90.9	40	90.9	59	3.4	0:00:00.01
ARIAL	draw	Black	5	5	5	B	C	59.0	90.9	40	90.9	59	3.4	0:00:00.01
ARIAL	arrow	Black	5	5	5	B	C	65.0	90.9	40	90.9	59	90.9	0:00:00.01
flowchar	label	Black	5	5	5	B	C	65.0	87.5	40	90.9	59	90.9	S
ARIAL	label	Black	5	5	5	B	C	65.0	94.5	40	90.9	59	90.9	cust_id.MIDTIER_VALID_cust_id
ARIAL	label	Black	5	5	5	B	C	72.0	92.0	40	90.9	59	90.9	Rows: 9,125
ARIAL	label	Black	5	5	5	B	C	72.0	88.5	40	90.9	59	90.9	Cols: 36

Output 8

FLOWCHART WRAP UP

Everything produced so far was captured from the SAS log and drawn with SAS/GRAPH. It is not especially elegant or advanced in this day of interactivity and web output, but still the underlying data can in future be sent to a more robust graphical output.

PROC SCAPROC

PROC SCAPROC can be inserted at the beginning of a SAS job and create a text file containing step information and timing, library and file information, macro information, but also column information.

A SAS statement such as SET; reads the last dataset built and all of the columns within that dataset. Because PROC SCAPROC has access to the internals of SAS, there is no other easy way to gain column information from the open file which of course might vary widely.

An example of PROC SCAPROC output for the first couple of steps follows.

```
/* JOBSPLIT: JOBSTARTTIME 21OCT2015:13:10:08.58 */
/* JOBSPLIT: TASKSTARTTIME 21OCT2015:13:10:08.59 */
/* JOBSPLIT: CATALOG OUTPUT WORK.SASMACR.M000_FILE_INFORMATION.MACRO */
/* JOBSPLIT: LIBNAME WORK V9 '/saswrk/SAS_work73AF000063F8_leussa01' */
/* JOBSPLIT: CATALOG INPUT WORK.SASMACR.M000_FILE_INFORMATION.MACRO */
/* JOBSPLIT: LIBNAME WORK V9 '/saswrk/SAS_work73AF000063F8_leussa01' */
/* JOBSPLIT: CATALOG OUTPUT WORK.SASMACR.SYSTEMP.MACRO */
```

```
/* JOBSPLIT: LIBNAME WORK V9 '/saswrk/SAS_work73AF000063F8_leussa01' */
/* JOBSPLIT: CATALOG UPDATE WORK.SASMACR.SYSTEMP.MACRO */
/* JOBSPLIT: LIBNAME WORK V9 '/saswrk/SAS_work73AF000063F8_leussa01' */
/* JOBSPLIT: CATALOG OUTPUT WORK.SASMACR.M000_APPEND_REPORT.MACRO */
/* JOBSPLIT: LIBNAME WORK V9 '/saswrk/SAS_work73AF000063F8_leussa01' */
/* JOBSPLIT: CATALOG INPUT WORK.SASMACR.M000_APPEND_REPORT.MACRO */
/* JOBSPLIT: LIBNAME WORK V9 '/saswrk/SAS_work73AF000063F8_leussa01' */
/* JOBSPLIT: DATASET OUTPUT SEQ WINFILE.CUST_INFO     */
/* JOBSPLIT: LIBNAME PCFILE '/ssctest/dir' */
/* JOBSPLIT: FILE INPUT /ssctest/dir/AccountEmails.csv */
/* JOBSPLIT: ATTR WINFILE.CUST_INFO.DATA OUTPUT VARIABLE:cust_id_ TYPE:CHARACTER
LENGTH:31 LABEL: FORMAT:$31. INFORMAT:$31. */
/* JOBSPLIT: ATTR WINFILE.CUST_INFO.DATA OUTPUT VARIABLE:Account_Number TYPE:NUMERIC
LENGTH:8 LABEL: FORMAT:9. INFORMAT:9. */
/* JOBSPLIT: ATTR WINFILE.CUST_INFO.DATA OUTPUT VARIABLE:Account_Name TYPE:CHARACTER
LENGTH:30 LABEL: FORMAT:$30. INFORMAT:$30. */
/* JOBSPLIT: SYMBOL GET M_JOB_DIR */
/* JOBSPLIT: SYMBOL SET _EFIERR_ */
/* JOBSPLIT: SYMBOL GET SYS_IOUSEEE */
/* JOBSPLIT: ELAPSED 1994  */
/* JOBSPLIT: SYSSCP LIN X64 */
/* JOBSPLIT: PROCNAME DATASTEP */
/* JOBSPLIT: STEP SOURCE FOLLOWS */
libname filetemp  '/ssctest/temp';
/* JOBSPLIT: TASKSTARTTIME 21OCT2015:13:10:10.58 */
/* JOBSPLIT: DATASET INPUT SEQ PCFILE.ACCOUNTEMAILS.DATA */
/* JOBSPLIT: LIBNAME WINFILE '/ssctest/dir' */
/* JOBSPLIT: OPENTIME WINFILE.CUST_INFO DATE:21OCT2015:13:10:10.58
PHYS:/ssctest/dir/accountemails.sas7bdat SIZE:161177600 */
/* JOBSPLIT: DATASET OUTPUT SEQ WINFILE.CUST_INFO_ADJ.DATA */
/* JOBSPLIT: ATTR WINFILE.CUST_INFO.DATA INPUT VARIABLE:cust_id_ TYPE:CHARACTER
LENGTH:31 LABEL: FORMAT:$31. INFORMAT:$31. */
/* JOBSPLIT: ATTR WINFILE.CUST_INFO.DATA INPUT VARIABLE:Account_Number TYPE:NUMERIC
LENGTH:8 LABEL: FORMAT:9. INFORMAT:9. */
/* JOBSPLIT: ATTR WINFILE.CUST_INFO.DATA INPUT VARIABLE:Account_Name TYPE:CHARACTER
LENGTH:30 LABEL: FORMAT:$30. INFORMAT:$30. */
/* JOBSPLIT: ATTR WINFILE.CUST_INFO_ADJ.DATA OUTPUT VARIABLE:cust_id_ TYPE:CHARACTER
LENGTH:31 LABEL: FORMAT:$31. INFORMAT:$31. */
/* JOBSPLIT: ATTR WINFILE.CUST_INFO_ADJ.DATA OUTPUT VARIABLE:Account_Number
TYPE:NUMERIC LENGTH:8 LABEL: FORMAT:9. INFORMAT:9. */
/* JOBSPLIT: ATTR WINFILE.CUST_INFO_ADJ.DATA OUTPUT VARIABLE:Account_Name
TYPE:CHARACTER LENGTH:30 LABEL: FORMAT:$30. INFORMAT:$30. */
/* JOBSPLIT: ATTR WINFILE.CUST_INFO_ADJ.DATA OUT VARIABLE:cust_id TYPE:CHARACTER
LENGTH:31 LABEL: FORMAT:$31. INFORMAT:$31. */
/* JOBSPLIT: ELAPSED 766  */
/* JOBSPLIT: PROCNAME DATASTEP */
```

A fairly easy program to parse the SCAPROC txt file can create tables at the step, file, and now column levels. There is some overlapping information from what was captured from the log and some unique information.

sca_steps

Obs	Sas_step_no	Sca_log_line	Sca_mprocname	Sca_step_starttime	sca_elapsed
1	2	26	DATASTEP	21OCT2015:13:10:08.590	0:00:01.99
2	3	43	DATASTEP	21OCT2015:13:10:10.580	0:00:00.77

sca_datafile

Obs	Sas_step_no	Sca_log_line	sca_memtype	sca_i_o	sca_ddn	sca_member
1	2	15	SAS DS	Out	WINFILE	CUST_INFO
2	3	30	SAS DS	IN	PCFILE	ACCOUNTEMAILS
3	3	33	SAS DS	Out	WINFILE	CUST_INFO_ADJ

sca_columns

Obs	Sas_step_no	Sca_log_line	sca_ddn	sca_member	sca_vn	sca_vt	sca_vl	sca_vlb	sca_vf	sca_vi
1	2	18	WINFILE	CUST_INFO	cust_id_	C	31		$31.	$31.
2	2	19	WINFILE	CUST_INFO	Account_Number	N	8		9.	9.
3	2	20	WINFILE	CUST_INFO	Account_Name	C	30		$30.	$30.
4	3	34	WINFILE	CUST_INFO	cust_id_	C	31		$31.	$31.
5	3	35	WINFILE	CUST_INFO	Account_Number	N	8		9.	9.
6	3	36	WINFILE	CUST_INFO	Account_Name	C	30		$30.	$30.
7	3	37	WINFILE	CUST_INFO_ADJ	cust_id_	C	31		$31.	$31.
8	3	38	WINFILE	CUST_INFO_ADJ	Account_Number	N	8		9.	9.
9	3	39	WINFILE	CUST_INFO_ADJ	Account_Name	C	30		$30.	$30.
10	3	40	WINFILE	CUST_INFO_ADJ	cust_id	C	31		$31.	$31.

Output 9

ADVANTAGES TO USING SCAPROC

1. More accurate information in some cases.
2. Files read via PROC SQL are reported.
3. File and column attributes are reported. These values can be used to trace back column lineage.

DISADVANTAGES TO USING SCAPROC

1. Source code must be altered and rerun.
2. Row counts are not reported

COMBINING SCAPROC WITH LOG DATA

It would seem like joining log data with SCAPROC data would give the most information available. We are attempting to do just that, but it is not a trivial join.
The most difficult issue is the SCAPROC sometimes reports steps in a different order than the log does. The whole question of which step elements belong to becomes very tedious. The most common occurrence of this is when a PROC IMPORT is executed, sometimes SAS creates a “Scrubber” data step

to clear input files. The PROC step is reported as well as the generated DATA step, but in different orders in the log and SCAPROC.

CODE IMPROVEMENTS

Since the underlying data contains indicators that a step is the “Last Reference” of a file, there is the possibility that the file is not needed in the original job. Of course, the file may be stored for archival purposes, or could be referenced by other jobs. Therefore, the program can’t prudently delete code or delete files. It can however generate Proc datasets statements to delete possible unneeded files that the user can manually verify as unneeded, and then manually insert the cleanup code. Other possibilities would be to determine minimum lengths needed for columns, perhaps identify unneeded columns, or even unneeded steps.

It is relatively easy to identify long running steps and point them out to the user, but we have found that quite often a macro may generate a series of individually small steps, that together are repeated thousands of times and overall represent a significant performance degradation that perhaps some pattern detection might point out.

There are virtually unlimited opportunities for programs to examine run time statistics and make or at least suggest improvements automatically.

CONCLUSION

- There is a lot of great run time information in the SAS log and PROC SCAPROC output files.
- A graphical display with annotation can build a pdf of a flow chart easily.
- Other data flow diagrams can be built from the underlying data.
- There is much to be gained for documentation, performance, and data lineage.
- Improvement code can be generated by programs.

RECOMMENDED READING

Previous SAS Global Forum papers by Steven First

The Missing Semicolon: http://www.sys-seminar.com/newsletter

CONTACT INFORMATION

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

Steven First
President
Systems Seminar Consultants
608 278-9964
sfirst@sys-seminar.com