

RIPPLE-TRAC proof of concept (POC)



©

Written by
Daniel B. Boucher Sr,
z/OS Solutions Architect,
Logic Online, Inc.

A Separation of Concerns (SoC) Tool

www.logiconlineinc.com

This document contains proprietary information, protected by copyright. No part of this document may be reproduced or transmitted for any purpose other than the reader's personal use without the permission of Logic Online Inc.

WARRANTY

The information contained in this document is subject to change without notice. Logic Online makes no warranty of any kind with respect to this information. Logic Online shall not be liable for any direct, indirect, incidental, consequential, or other damage alleged in connection with the use of this information.

TRADEMARKS

All trademarks and registered trademarks used in this guide are property of their respective owners.

RIPPLE-TRAC is a registered trademark in the state of New Hampshire

Logic Online Inc.
1500A Lafayette Rd #170
Portsmouth, NH 03801
e-mail: info@legacyimpactanalysis.com
www.logiconlineinc.com



www.logiconlineinc.com is a resource for:



<http://www-03.ibm.com/systems/z/destinationz/>

SECTION 1

We have run an analysis on a portion of the code,
this document is an illustration for documenting the following application:

WZWD0V.SRICM.PROD.SORC.BLK3120 and
WZWD0V.SRICM.PROD.INCL.BLK3120

This application has:

110	DB2 Tables
86	INPUT/OUTPUT FILES
68,133	LINES OF COBOL CODE EXCLUDING INCLUDES
6,689	LINES OF INCLUDE CODE
74,822	GRAND TOTAL LINES OF COBOL CODE
4	MISSING COPYBOOK/INCLUDE SPANNING 83 MODULES
3,095	FIELDS WITHIN COPYBOOK/INCLUDE
1,196	DB2 FIELDS WITHIN COPYBOOK/INCLUDE
5,327	FIELDS THAT ARE REFERENCED FROM WITHIN THE SOURCE CODE
5,366	DB2 FIELDS REFERENCED FROM WITHIN THE SOURCE CODE
17	DB2 TABLES REFERENCED OUTSIDE OF THIS APPLICATION

Required components that should be part of documentation:

- Batch job streams (RIPPLE-TRAC can produce a where used on DSN's)
- Order of batch execution via master scheduler
- Physical db2 model with DDL's, if physical model does not exist then the DDL's must be reverse engineered into Erwin.
- DB2 triggers (if any)
- DB2 procedures (if any)
- METADATA (if exists)
- DB2 bind parms
- Data profiling

SECTION 2

For this illustration we asked RIPPLE-TRAC to assess making a modification to 3 components in this application.

This hypothetical illustration involves a what if scenario of changing field SELLG-SCE-NO from 2 bytes to 5 bytes.

THE DB2 TABLE ALC_STOCK
THE DB2 FIELD SELLG_SCE_NO
THE COBOL FIELD SELLG-SCE-NO

```
*****
* DCLGEN TABLE (GMDRRI1.ALC_STOCK) *
* LIBRARY (TGMDR.SRIAT.RAJESH.WORK (ALCSTOCK) ) *
* LANGUAGE (COBOL) *
* STRUCTURE (ALC-STOCK) *
* QUOTE *
* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS *
*****
EXEC SQL DECLARE ALC_STOCK TABLE
( SELLG_SCE_NO CHAR(2) NOT NULL,
  MKTG_DIV_CD CHAR(1) NOT NULL,
  BUSNS_ASCT_CD DECIMAL(11, 0) NOT NULL,
  EVNT_PROCS_DT DATE NOT NULL,
  ALC_PRD_RPT_MDL_CD CHAR(7) NOT NULL,
  MODL_YR_NBR CHAR(4) NOT NULL,
  STOCK_TOT_RETAIL INTEGER NOT NULL,
  STOCK_TOT_FLET INTEGER NOT NULL,
  LAST_UPDT_TIMSTM TIMESTAMP NOT NULL
) END-EXEC.
*****
* COBOL DECLARATION FOR TABLE GMDRRI1.ALC_STOCK *
*****
01 ALC-STOCK.
  10 SELLG-SCE-NO PIC X(2) .
  10 MKTG-DIV-CD PIC X(1) .
  10 BUSNS-ASCT-CD PIC S9(11)V USAGE COMP-3.
  10 EVNT-PROCS-DT PIC X(10) .
  10 ALC-PRD-RPT-MDL-CD PIC X(7) .
  10 MODL-YR-NBR PIC X(4) .
  10 STOCK-TOT-RETAIL PIC S9(9) USAGE COMP.
  10 STOCK-TOT-FLET PIC S9(9) USAGE COMP.
  10 LAST-UPDT-TIMSTM PIC X(26) .
*****
* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 9 *
*****
```

The intent is to locate all references to the 3 items mentioned above including the business rules associated with them.

We will flowchart and perform the complexity metrics on a couple of the business rules.

NOMENCLATURE: (BASED ON RIPPLE-TRAC RESULT SET)

ALC_STOCK table has 23 references 4 of which are in copybooks.
 SELLG_SCE_NO table field has 214 references 102 of which are in copybooks
 spanning 25 modules.
 SELLG-SCE-NO cobol field has 318 references 118 of which are in copybooks
 spanning 26 modules.

We used module SRIPDM03 for this example.

The module below (SRIPDM03) represents RIPPLE-TRAC result set and has 32 of these nomenclature references.

The index column is the relative line pointer where the table is used within this module.
 (~ indicates the relative line pointer inside the include)

..NO COMPONENTS mean that the none of the 3 items were found in the include.

The BACKREF column refers to the include on previous line.

INDEX	MODULE	TYPE	BACKREF	COMPONENT LABEL	COMPONENT NAME	DESCRIPTION
199	SRIPDM03	*		INCLUDE	SRIYSQLC	
~	SRIPDM03		&SRIYSQLC	..NO COMPONENTS		
201	SRIPDM03	*		INCLUDE	SRIYURTC	
214	SRIPDM03	*		COPY	WAAYCDAT	COPY BOOK NOT IN CURRENT COPYLIB
358	SRIPDM03	*		COPY	WAAYCDTR	COPY BOOK NOT IN CURRENT COPYLIB
367	SRIPDM03	*		COPY	WAAYCDTJ	COPY BOOK NOT IN CURRENT COPYLIB
381	SRIPDM03	*		INCLUDE	SRIYAPPL	
~0011	SRIPDM03	F	&SRIYAPPL	..	SELLG_SCE_NO	
~0034	SRIPDM03	F	&SRIYAPPL	..	SELLG-SCE-NO	
388	SRIPDM03	*		INCLUDE	SRIYG011	
~	SRIPDM03		&SRIYG011	..NO COMPONENTS		
395	SRIPDM03	*		INCLUDE	SRIYG041	
~0010	SRIPDM03	F	&SRIYG041	..	SELLG_SCE_NO	
~0032	SRIPDM03	F	&SRIYG041	..	SELLG-SCE-NO	
401	SRIPDM03	*		INCLUDE	SRIYASTK	
~0009	SRIPDM03	T	&SRIYASTK	..	ALC_STOCK	
~0010	SRIPDM03	F	&SRIYASTK	..	SELLG_SCE_NO	
~0024	SRIPDM03	F	&SRIYASTK	..	SELLG-SCE-NO	
408	SRIPDM03	*		INCLUDE	SRIYG261	
~0011	SRIPDM03	F	&SRIYG261	..	SELLG_SCE_NO	
~0026	SRIPDM03	F	&SRIYG261	..	SELLG-SCE-NO	
415	SRIPDM03	*		INCLUDE	SRIYG221	
~0010	SRIPDM03	F	&SRIYG221	..	SELLG_SCE_NO	

~0027	SRIPDM03	F	&SRIYG221	..	SELLG-SCE-NO	
422	SRIPDM03	*		INCLUDE	SRIYG361	
~0010	SRIPDM03	F	&SRIYG361	..	SELLG_SCE_NO	
~0028	SRIPDM03	F	&SRIYG361	..	SELLG-SCE-NO	
429	SRIPDM03	*		INCLUDE	SRIYG271	
~0010	SRIPDM03	F	&SRIYG271	..	SELLG_SCE_NO	
~0029	SRIPDM03	F	&SRIYG271	..	SELLG-SCE-NO	
443	SRIPDM03	*		INCLUDE	SQLCA	COPY BOOK NOT IN CURRENT COPYLIB
589	SRIPDM03	F			SELLG_SCE_NO	
701	SRIPDM03	T			ALC_STOCK	
759	SRIPDM03	T			ALC_STOCK	
1105	SRIPDM03	F			SELLG_SCE_NO	
1109	SRIPDM03	F			SELLG_SCE_NO	
1134	SRIPDM03	T			ALC_STOCK	
1136	SRIPDM03	F			SELLG_SCE_NO	
1140	SRIPDM03	F			SELLG_SCE_NO	
1155	SRIPDM03	T			ALC_STOCK	
1167	SRIPDM03	F			SELLG_SCE_NO	
1171	SRIPDM03	F			SELLG_SCE_NO	
1198	SRIPDM03	F			SELLG_SCE_NO	
1202	SRIPDM03	F			SELLG_SCE_NO	
1229	SRIPDM03	F			SELLG_SCE_NO	
1233	SRIPDM03	F			SELLG_SCE_NO	
1260	SRIPDM03	F			SELLG_SCE_NO	
1264	SRIPDM03	F			SELLG_SCE_NO	
1327	SRIPDM03			DYNAMIC CALL	C-DATE-PROGRAM	
1560	SRIPDM03			STATIC CALL	DSNTIAR	
1601	SRIPDM03			DYNAMIC CALL	USER-ABEND-PGM	

Using the index column from RIPPLE-TRAC result set above find the index line in the code (589) and the associated paragraph that the index is in:

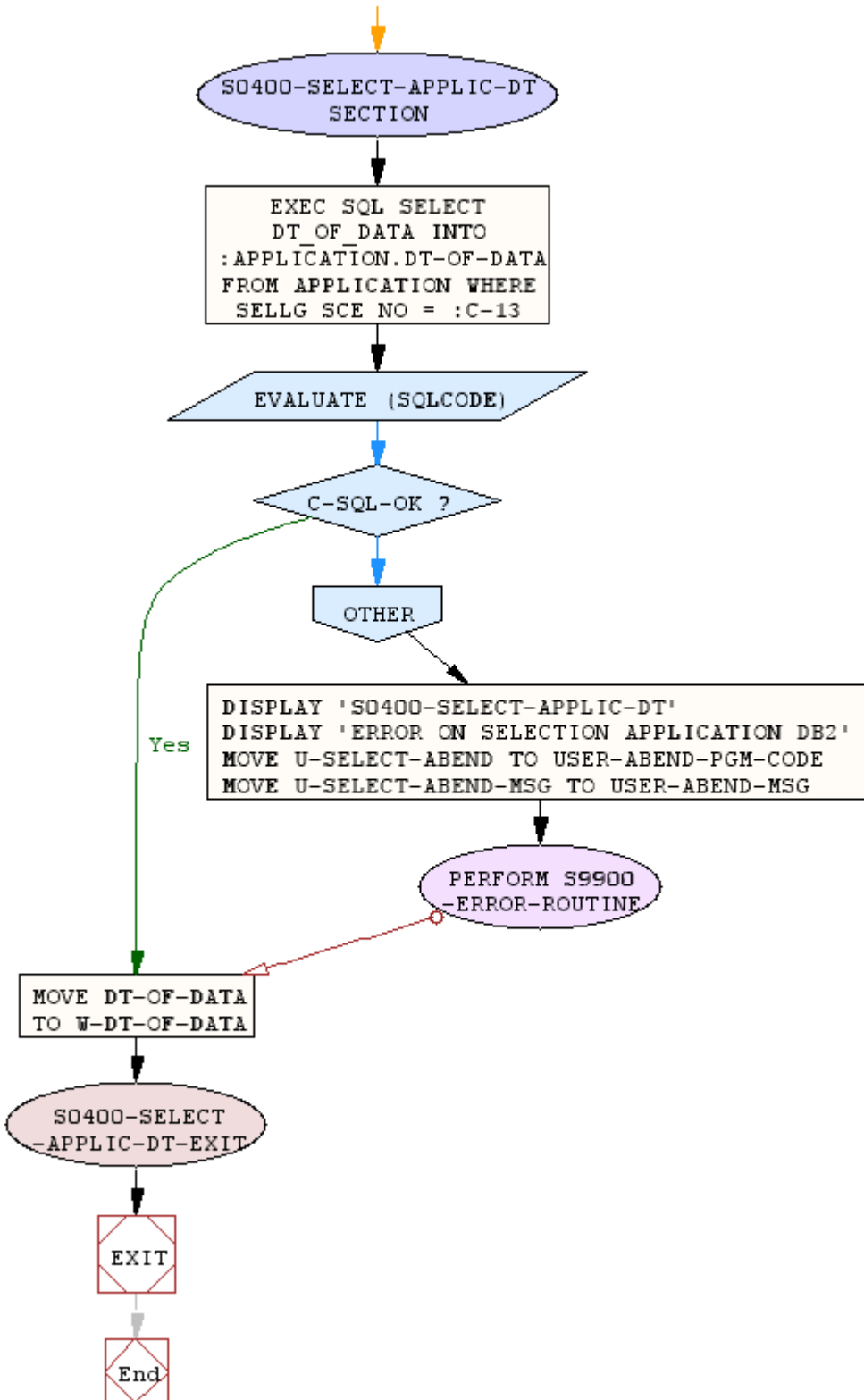
```

S0581 S0400-SELECT-APPLIC-DT SECTION.
S0582
S0583 EXEC SQL
S0584
S0585 SELECT DT_OF_DATA
S0586 INTO :APPLICATION.DT-OF-DATA
S0587 FROM
S0588 APPLICATION
S0589 WHERE SELLG_SCE_NO
S0590 = :C-13
S0591 END-EXEC
S0592
S0593 EVALUATE (SQLCODE)
S0594
S0595 WHEN C-SQL-OK
S0596

```

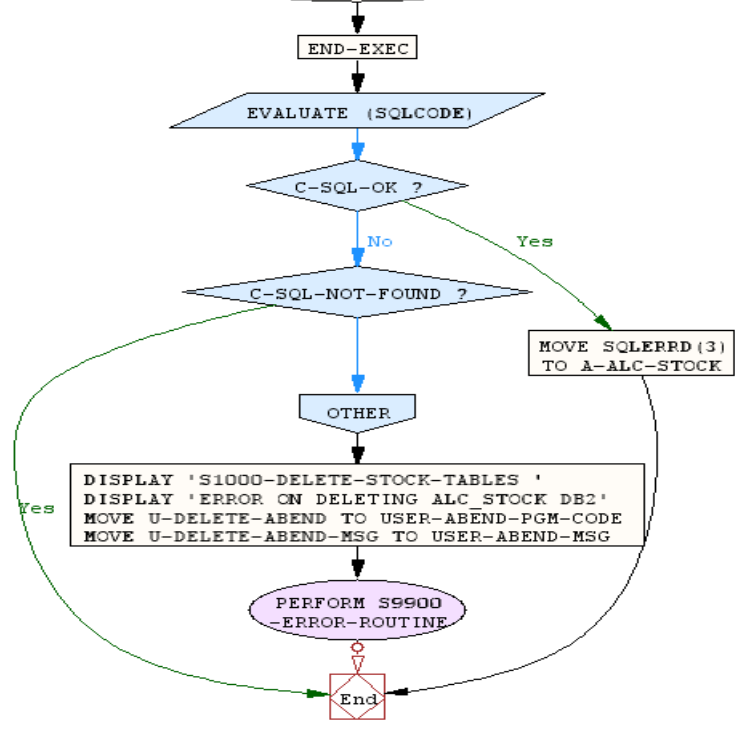
S0597 CONTINUE
S0598
S0599 WHEN OTHER
S0600
S0601 DISPLAY 'S0400-SELECT-APPLIC-DT'
S0602 DISPLAY 'ERROR ON SELECTION APPLICATION DB2'
S0603 MOVE U-SELECT-ABEND TO USER-ABEND-PGM-CODE
S0604 MOVE U-SELECT-ABEND-MSG TO USER-ABEND-MSG
S0605 PERFORM S9900-ERROR-ROUTINE
S0606
S0607 END-EVALUATE
S0608
S0609 MOVE DT-OF-DATA TO W-DT-OF-DATA
S0610
S0611 .
S0612 S0400-SELECT-APPLIC-DT-EXIT.
S0613 EXIT.

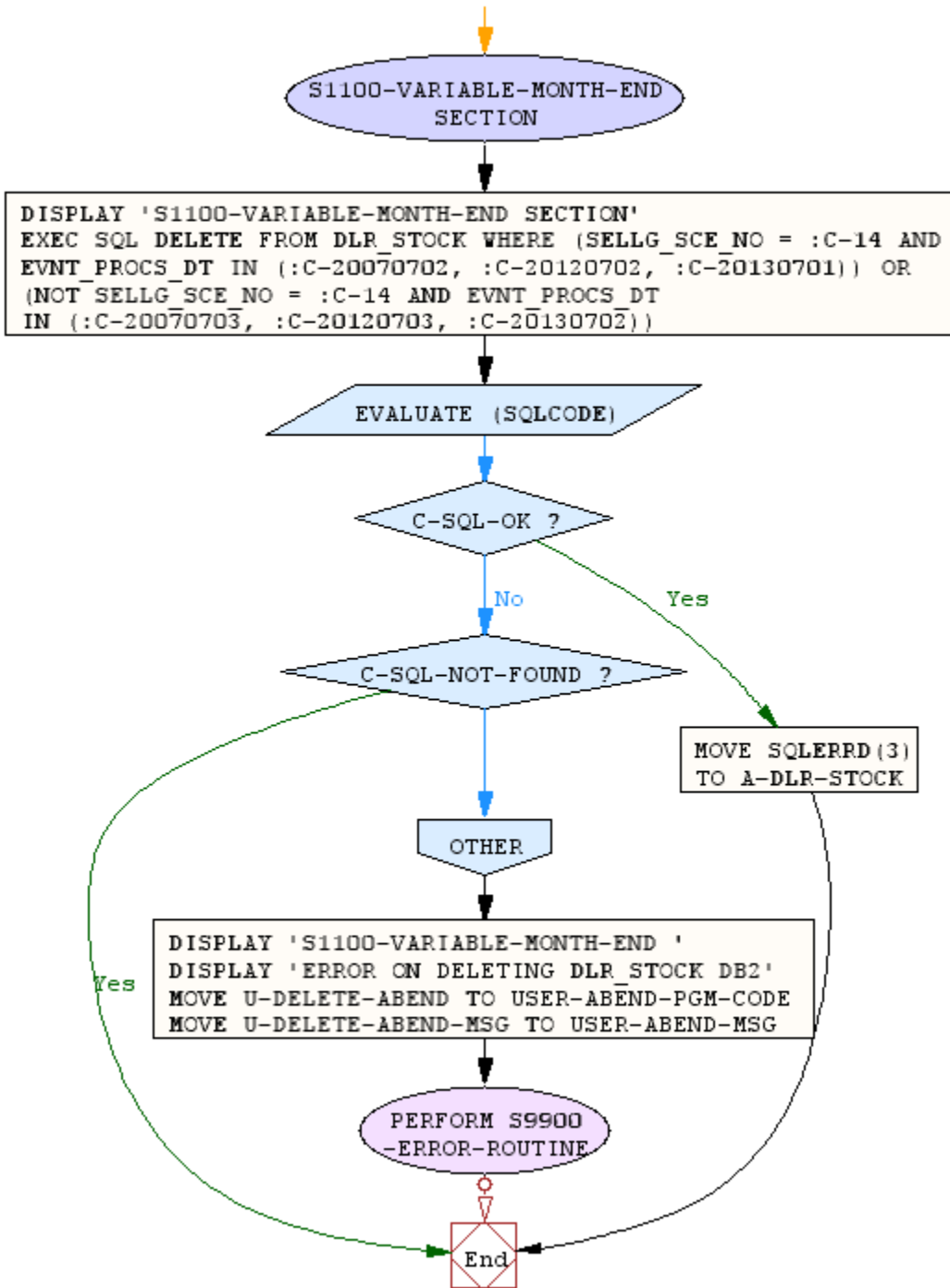
Copy the paragraph into the flow charter and produce charts as follows:

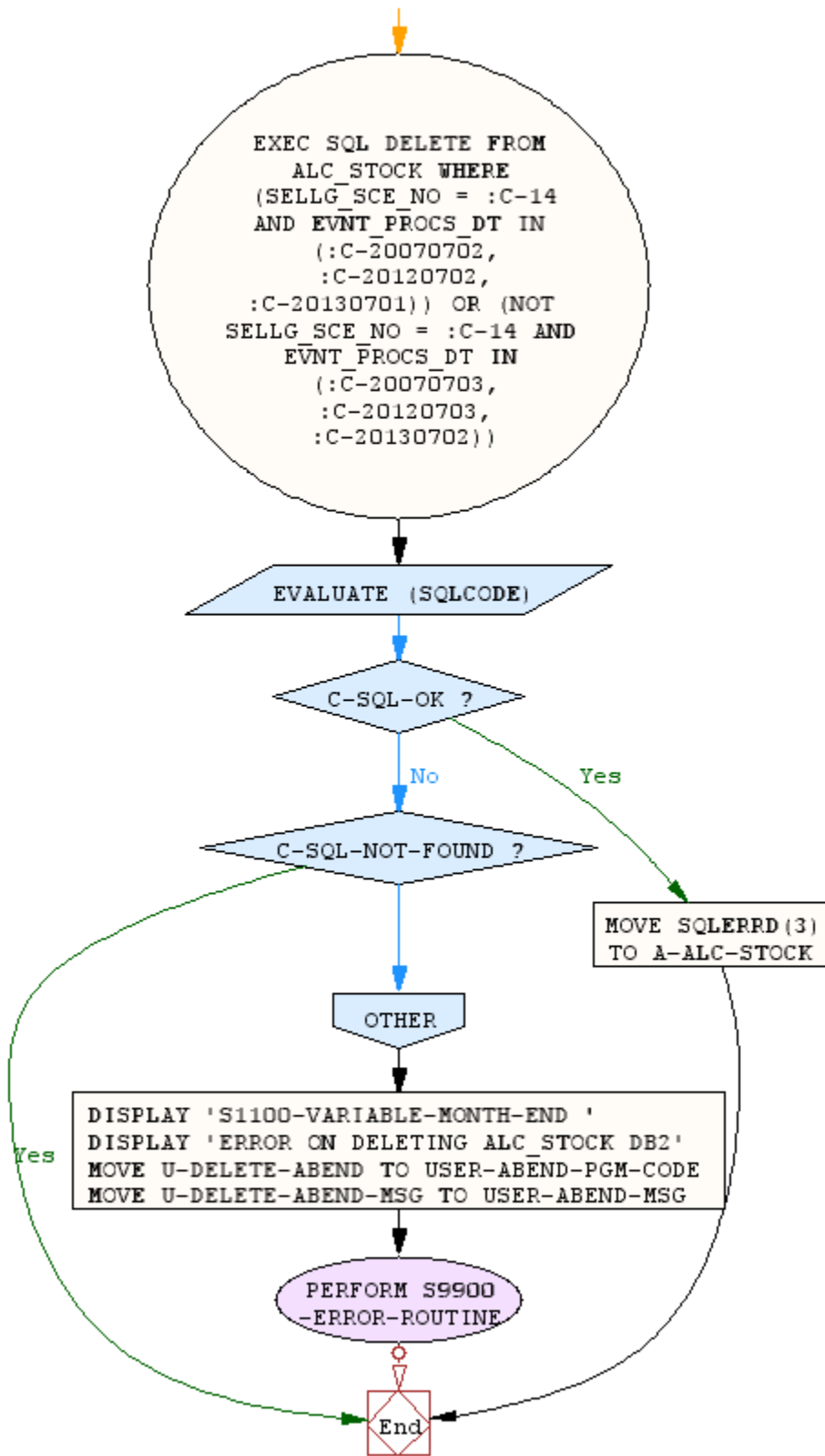


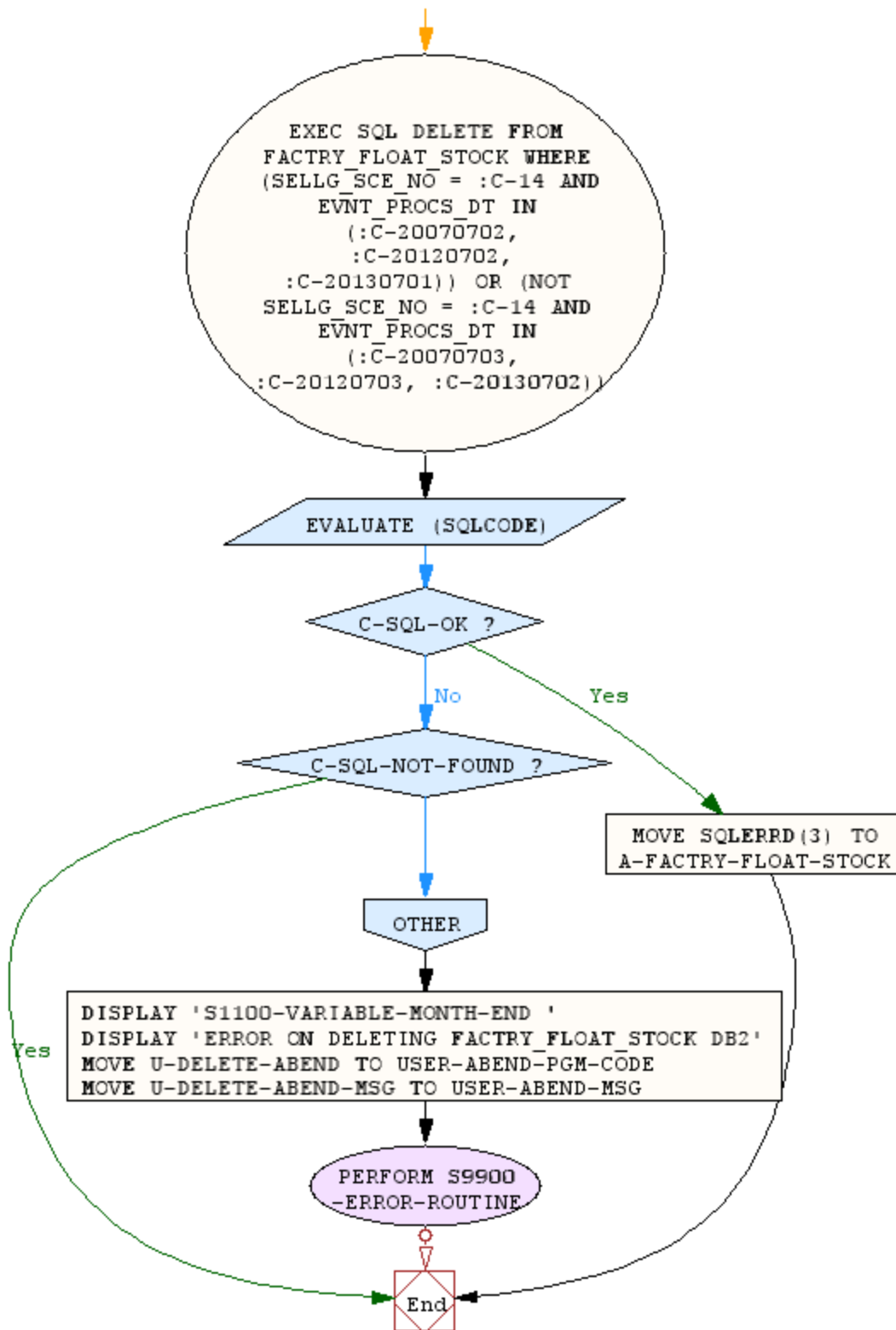
```

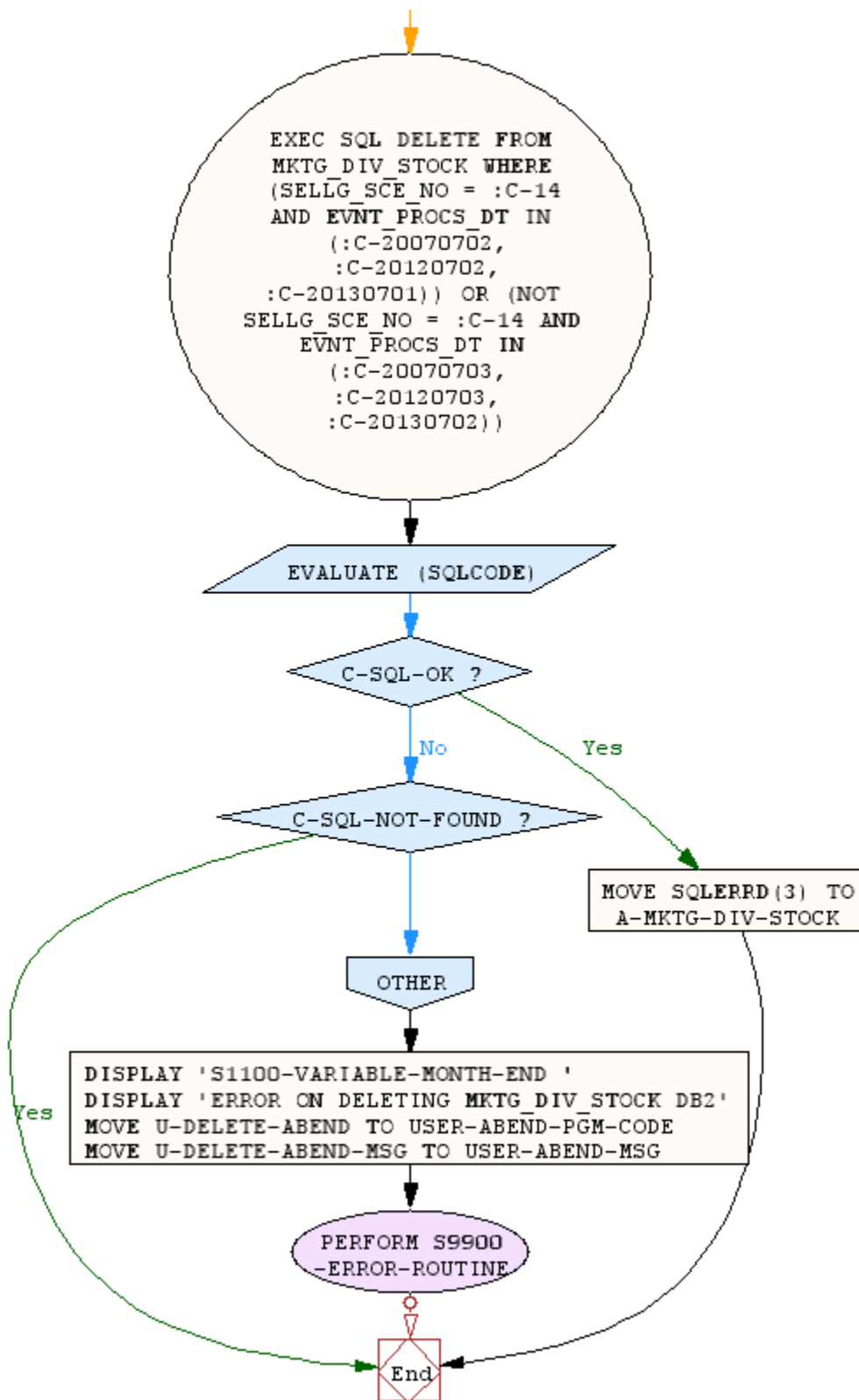
DELETE FROM ALC_STOCK WHERE EVNT_PROCS_DT < :W-DATE-35 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT1 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT2 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT3 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT4 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT5 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT6 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT7 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT8 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT9 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT10 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT11 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT12 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT13 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT14 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT15 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT16 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT17 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT18 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT19 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT20 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT21 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT22 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT23 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT24 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT25 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT26 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT27 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT28 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT29 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT30 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT31 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT32 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT33 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT34 AND NOT EVNT_PROCS_DT =
:W-MONTH-END-DT35 AND NOT EVNT_PROCS_DT = :W-MONTH-END-DT36 AND NOT
EVNT_PROCS_DT = :W-MONTH-END-DT37 AND NOT EVNT_PROCS_DT = :C-20070703
AND NOT EVNT_PROCS_DT = :C-20120703 AND NOT EVNT_PROCS_DT = :C-20130702
    
```

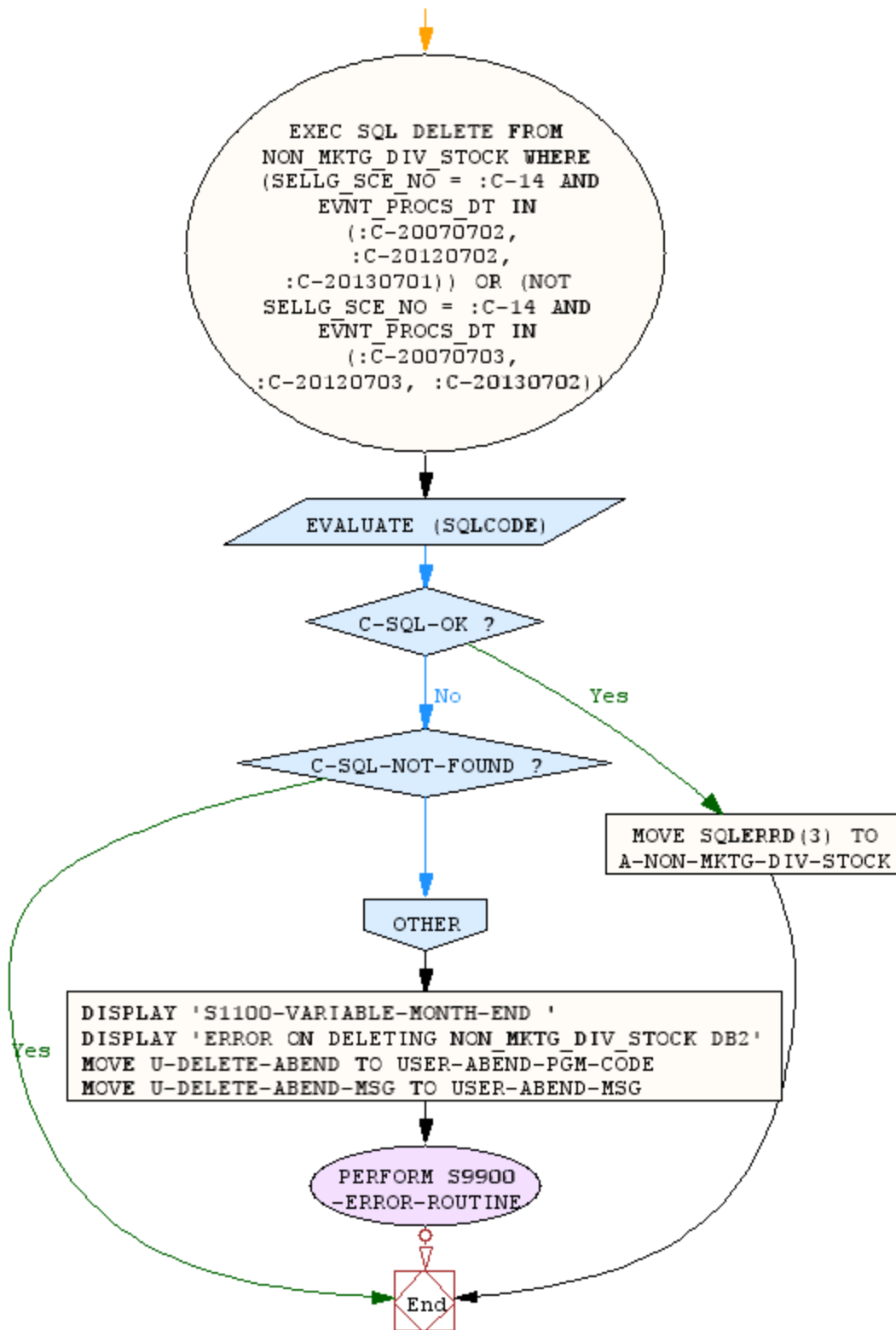


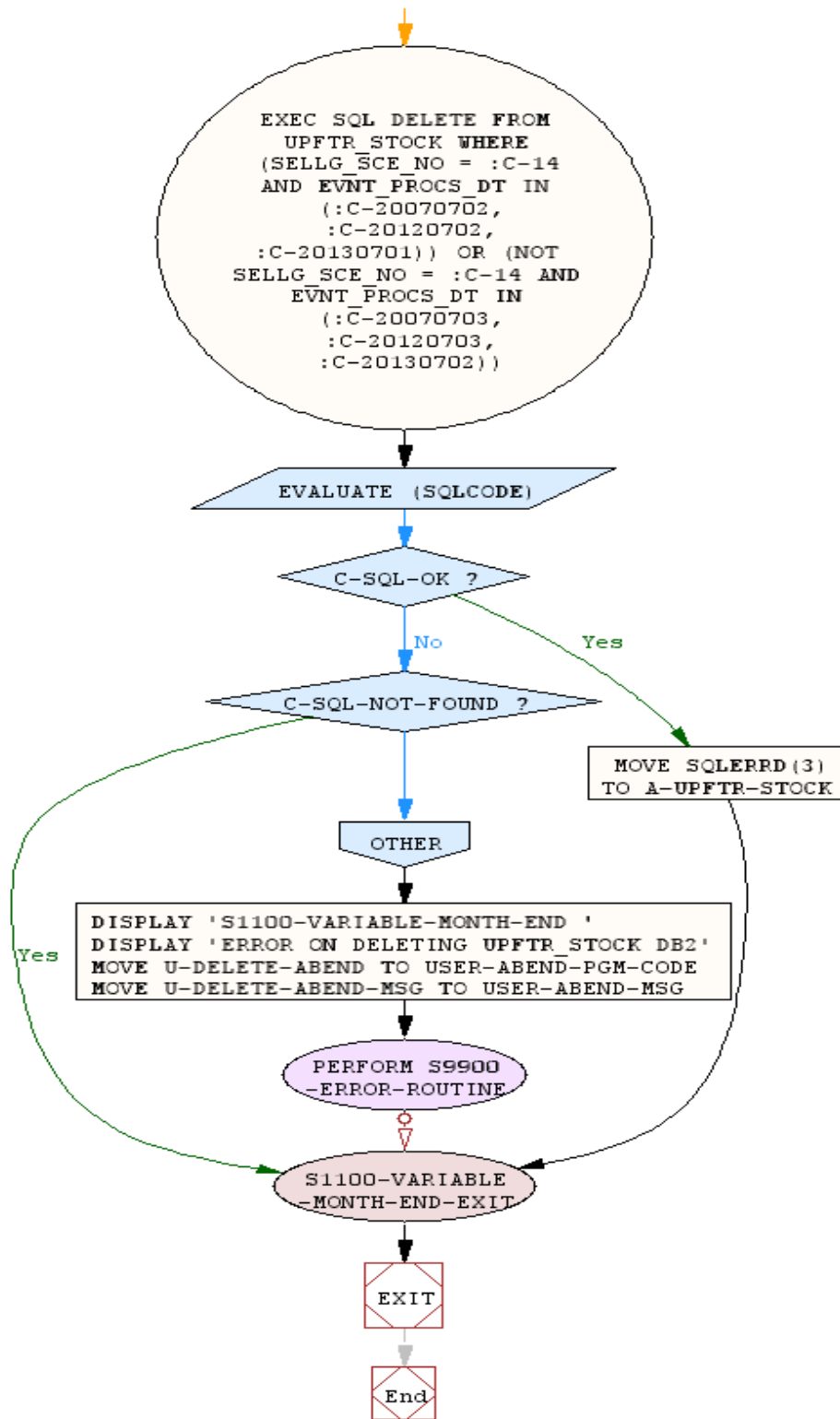




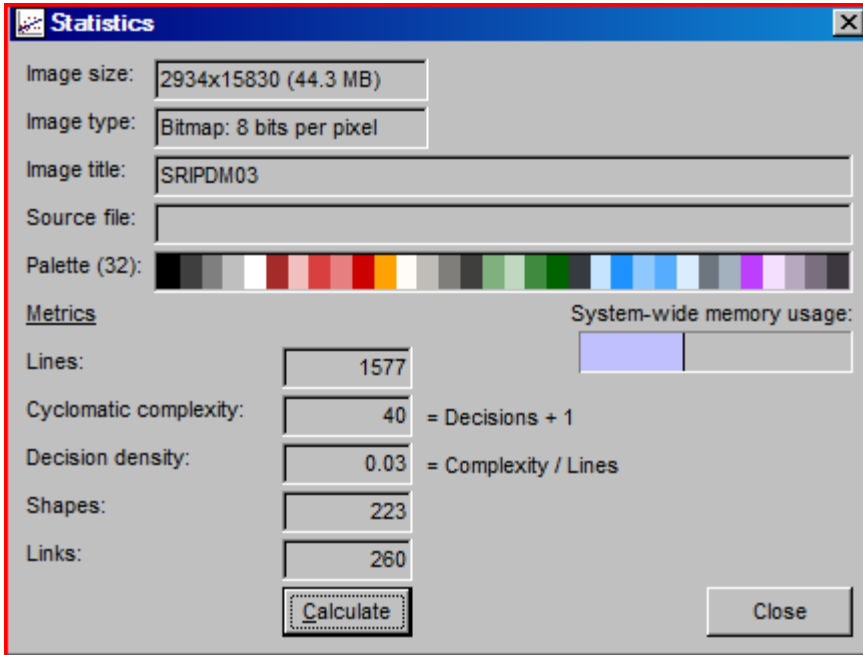








COMPLEXITY METRICS FOR MODULE SRIPDM03



As you can see, this MODULE received a 40 on the complexity scale.

SEE SRIPDM03_COLOR PDF, THIS FLOW CHART DEPICTS ALL THE CODE AND THE RED COLOR BLOCKS REPRESENT THE 3 ITEMS OF INTEREST:

ALC_STOCK
SELLG_SCE_NO
SELLG-SCE-NO

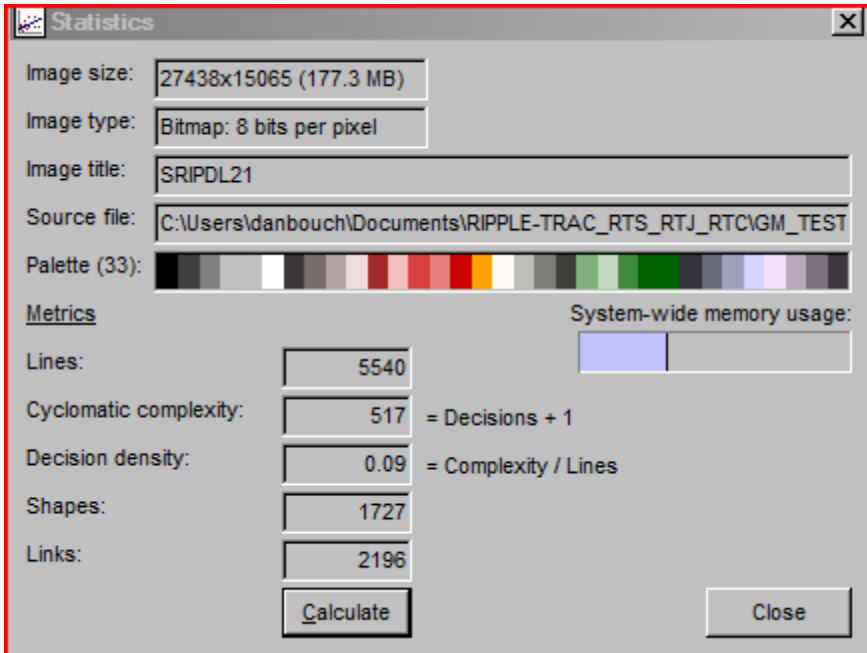
SECTION 3

The largest module in this application (SRIPDL21) was used as input to the complexity metrics analyzer.

As you can see high cyclomatic complexity (517) denotes a complex procedure that's hard to understand, test and maintain. There's a relationship between cyclomatic complexity and the "risk" in a procedure.

In other words, this module needs to be rewritten.

See PDF SRIPDL21_LARGEST_MODULE for flow chart view



SECTION 4

COMPLEXITY METRICS EXPLAINED

The following metrics measure the complexity of executable code within procedures. This includes both the internal complexity of a single procedure and the complexity of the data flow in and out of a procedure.

High complexity may result in bad understandability and more errors. Complex procedures also need more time to develop and test. Therefore, excessive complexity should be avoided. Too complex procedures should be simplified by rewriting or splitting into several procedures.

Complexity is often positively correlated to code size. A big program or function is likely to be complex as well. These are not equal, however. A procedure with relatively few [lines of code](#) might be far more complex than a long one.

VALUES OF CYCLOMATIC COMPLEXITY

A high cyclomatic complexity denotes a complex procedure that's hard to understand, test and maintain. There's a relationship between cyclomatic complexity and the "risk" in a procedure.

CC	Type of procedure	Risk
1-4	A simple procedure	Low
5-10	A well structured and stable procedure	Low
11-20	A more complex procedure	Moderate
21-50	A complex procedure, alarming	High
>50	An error-prone, extremely troublesome, not testable procedure	Very high

The original, usual limit for a maximum acceptable value for cyclomatic complexity is 10. Other values, such as 15 or 20, have also been suggested. Regardless of the exact limit, if cyclomatic complexity exceeds 20, you should consider it alarming. Procedures with a high cyclomatic complexity should be simplified or split into several smaller procedures.

Cyclomatic complexity equals the minimum number of test cases you must execute to cover every possible execution path through your procedure. This is important information for testing. Carefully test procedures with the highest cyclomatic complexity values.

Bad fix probability

There is a frequently quoted table of "bad fix probability" values by cyclomatic complexity. This is the probability of an error accidentally inserted into a program while trying to fix a previous error or enhance the business rule.

CC	Bad fix probability
----	---------------------

1-10	5%
20-30	20%
>50	40%
approaching 100	60%

As the complexity reaches high values, changes in the program are likely to produce new errors.

More on this subject: http://en.wikipedia.org/wiki/Cyclomatic_complexity

NOTE: WE CAN ACTUALLY EDIT THIS FLOWCHART IF WE WISH TO CHANGE