SCL Rides Again! Porting RESMENU to the Web

Presented at NESUG 2003 by
Michael L. Davis, Bassett Consulting Services
Ralph W. Leighton, The Hartford

Presentation Organization

- Introduction to RESMENU:
  - What it is and how it works;
  - Why it needs to be rewritten.
- Primer on SAS/IntrNet and the Application Dispatcher
- Programming techniques used to rewrite RESMENU

Part 1: RESMENU – What it is; and why it became necessary to rewrite it.

What is RESMENU?

- "Reserving Systems on the DEC".
- Corporate Actuarial Reserving Automation Support – a 100% SAS Shop.
- RESMENU : a character-based SAS/AF Program Application Driver executing RAS SAS programs.
- Consists of PROGRAM entries.

Genesis of RESMENU

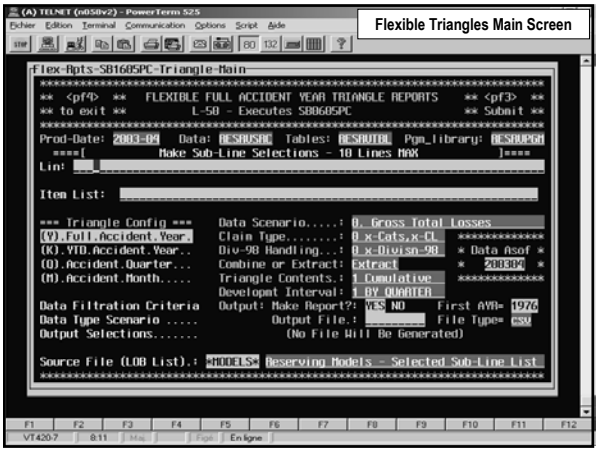
- Created in mid-1990s.
- Part of the wholesale conversion of Reserving function models and reporting programs from FORTRAN to SAS
- Platform was Release 6.12 on DEC Alpha running OpenVMS
- Horsepower of the Mid-tier did not permit GUI SAS/FRAME I-F.

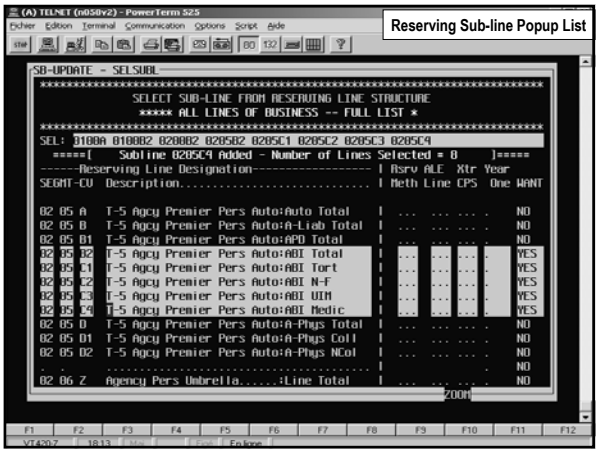
RESMENU Functions

- Data Services to the End-User :
  - User can define and retrieve his/her own reports
  - Data : From raw Claim Detail to Summary Loss Development "Triangles".
- Models and Information :
  - for the Reserving Area customers primarily.
  - Projections of Reserves based on Claim Patterns.

RESMENU Functions *(continued)*

- Production : RAS uses RESMENU to run all of its Production processes.
  - Update of Data Sources and derive summaries
  - Mechanical Model Projections
- Application Management :
  - The function screens document what is underneath them.
  - Special screens for security, program mgmt, data documentation, etc.





RESMENU – Mission and Context

- Loss- (Claim-) Related Reserves : Largest liabilities on a Property/Casualty insurers balance sheet
- Primary Mission: to support actuaries and their staff in the analysis of the adequacy of the reserve amounts posted, based on claim information. Case and IBNR.

RESMENU – Mission and Context

- Secondary Mission: to support Planning Consultants and other clients who need claim data and/or loss development information (Loss Triangles, time curves)
- Functionality: Provides a real-time ability to query loss data in a variety of ways not available in other systems

A Twenty-Cent Insurance Lesson

LOSS RESERVES IN INSURANCE:

- Indemnity: Money owed to or paid to claimants.
- Allocated Expense: Money owed for or paid out for assistance in processing a claim.

A Twenty-Cent Insurance Lesson (con't)

INSURANCE AS A BUSINESS:

B a c k w a r d s ! ! We sell our product first;
Then we incur our costs.

MAJOR PROBLEM: Setting up appropriate Reserves for:

Claims reported and not yet settled - *Case Reserve;*

A N D

Claims from accidents that have occurred,
but which we don't know about yet - - *I . B . N . R*

LOSS DEVELOPMENT TRIANGLES
Claim Reporting Patterns by Date of Loss

75-21-K3 : Oil Storage Tank Multi-Peril
INCURRED LOSS TRIANGLE - as of 12-31-1998
<< Dollars in Millions >>

Devel Year	Accident Year 1989	1990	1991	1992	1993	1994	1995	1996	1997	1998
1	98.5	115.0	117.5	111.9	110.5	105.7	117.8	134.6	130.7	158.6
2	136.4	160.3	168.7	161.9	158.5	147.8	161.4	183.7	182.1	
3	153.0	181.4	190.2	188.9	178.6	170.7	183.7	209.8		
4	158.8	193.8	199.9	197.8	186.5	182.5	190.5			
5	162.2	195.8	204.0	199.7	188.6	180.7				
6	162.7	196.7	204.6	199.1	188.1					
7	163.8	196.6	205.1	199.3						
8	165.0	197.5	204.6							
9	165.1	197.3								
10	164.9									

After some point all losses will be effectively reported

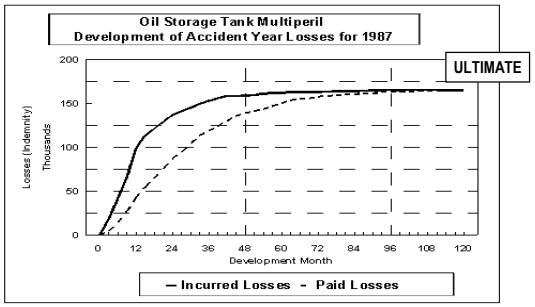
LOSS DEVELOPMENT TRIANGLES
Claim Payment Patterns by Date of Loss

75-21-K3 : Oil Storage Tank Multi-Peril
PAID INDEMNITY LOSS TRIANGLE - as of 12-31-1998
<< Dollars in Millions >>

Devel Year	Accident Year 1989	1990	1991	1992	1993	1994	1995	1996	1997	1998
1	42.8	51.8	53.0	54.2	49.6	45.1	51.6	60.5	64.3	73.3
2	86.9	104.3	107.0	104.4	97.5	90.3	103.8	119.0	118.1	
3	119.0	141.0	142.3	141.9	136.1	130.6	143.8	161.8		
4	139.5	168.5	173.6	169.7	160.4	153.7	167.0			
5	150.0	181.8	190.4	186.3	176.4	165.4				
6	158.1	187.4	196.6	194.2	183.2					
7	160.6	193.3	200.2	197.4						
8	163.6	196.3	202.7							
9	164.5	196.5								
10	164.8									

After some point all losses will be effectively paid out

LOSS DEVELOPMENT TRIANGLES
Eventually Claims are fully Reported and Paid



LOSS DEVELOPMENT TRIANGLES
Importance of Triangle Reports

- How the actuaries use the data: use patterns of development in older data to see where immature accident periods are likely to develop to.
- Key RESMENU Function: Flexible Triangles Reports.
- 55 Users Tied into RESMENU.

Problems Introduced by Version 8

- SAS dropped support for text-based terminal interface.
- X-Window interface was ghastly
- Each SAS window begat separate task
- Users limited to single SAS session

Problems Introduced by Version 8

- Necessity to Rewrite the RESMENU Interface.
- To go SAS/AF Frames -- a major rewrite without any increase utility.
- Decision to Move to Web-based I/F.
  - Wider Availability of System
  - Eliminate Special User ID Set-Ups

Principles for RESMENU Migration

- Use a SAS software solution
- Thin-Client
- Closely couple tables to front-end
- Minimize rewrite of existing programs
- Cross-validate user selections
- Provide concurrent end-user sessions
- Convenient development environment

Initial Selection of webAF and JSP

- Initial approach was to create Java Server Pages using webAF
- Licensed SAS and AppDev Studio (webAF, SAS/IntrNet) for PCs
- Added license for SAS/IntrNet for Alpha mid-tier server
- Implementation proved slow and difficult

An Alternative Solution Emerges

- Solution used existing licenses but changed the approach
- SAS Component Language (SCL), launched by SAS/IntrNet would generate the HTML screens for users and launch reports and other programs
- "proof of concept" application demonstrated feasibility of solution

Part 2: A Primer on SAS/IntrNet

Parameters Sent By HTML FORM

- Location of the Application Broker
- Service used to process the request
- Name of SCL catalog entry to be run
- Parameters required by program
- Value of _debug dispatcher options

SAS Application Dispatcher

- "Submit" button sends parameters to a socket, launch or pool server
- Test with socket service but use pool service, in which load manager starts and closes servers as necessary
- Each server is instantiated by PROC APPSERV

PROC APPSRV

- ALLOCATE FILE statements set up SAS program and external file directories
- ALLOCATE LIBRARY statements set up SAS libraries for catalogs and data sets
- DATALIB statements declare librefs that allocate data sources
- PROGLIB statements declare librefs that allocate program directories, catalogs

PROC APPSRV - Sample

```
PROC APPSRV;
  Allocate  File RESRVPGM '\\CORPACT\\SASPGMS';
            /*  RESMENU Programs    */
  Allocate  Library RESRVTBL   'RESRVTBL'   server=N058V2;
            /* RESMENU Tables.. */
  Allocate  Library RESRVSRC   'RESRVSRC'  server=N058V2;
            /* Source Data for Reports */
  Allocate  Library RALPHSCL  'RESRVCAT'   server=N058V2;
            /* SCL Program Catalog Location. */
  Proglibs  RALPHSCL  RESRVPGM ;
  Datalibs  RESRVSRC  RESRVTBL ;
Run;
```

Broker Runs Non-Visual SCL

- SCL normally associated visual displays
- In SAS/IntrNet, SCL is run in batch
- How to feed parameters? – macro vars
- HTML form elements (text boxes, check boxes, radio stations, drop-down lists) generate name/value pairs
- Use SYMGET, SYMGETC, SYMGETN to get values at execution time

What do the SCL Programs Do?

- Read control tables into SCL lists
- Write HTML and form headers
- Write HTML form elements
- Write choice items (e.g., drop-down lists, radio stations)
- Write closing tags for HTML page and form

OUTPUT to _WEBOUT

- How does a SCL batch program communicate with the user screen?
- _webout is a fileref, a TCP/IP socket connection to the Application Broker
- Sending output to _webout pipes the SCL generated HTML screen back to the browser
- Test SCL by allocating _webout to a file and examining in an HTML editor

Maintaining State in SAS/AF

- In RESMENU in SAS/AF, User passes from on Screen to another.
- Each successive step picks up params defined or passed along from preceding screen.
- Example: table selected in one screen used as input in a second screen to populate a selection list.
- Important to maintain this capability.

Maintaining State via Sessions

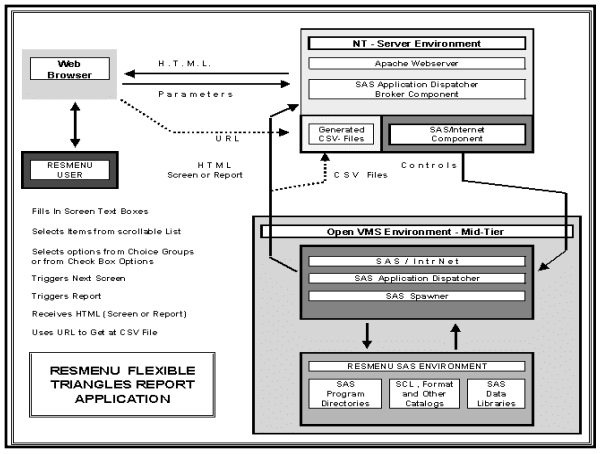
- HTML pages are "stateless"
- How to retain selections across screens?
- APPSRV_SESSION instantiates session
- Macro variables prefixed with "SAVE_"
- Data sets in SAVE libref retained
- Session available until browser closed or session time-out

Exporting Results to Spreadsheets

- RESMENU creates reports as Comma-Separated Value (CSV) files as option
- Use DS2CSV macro or ODS tagset
- ODS RTF or PDF for formatted reports
- GIF or JPG can be pasted into MS Office
- Automatically generate CSV, RTF, PDF files to a random filename referenced by hyperlink

Connecting to Data

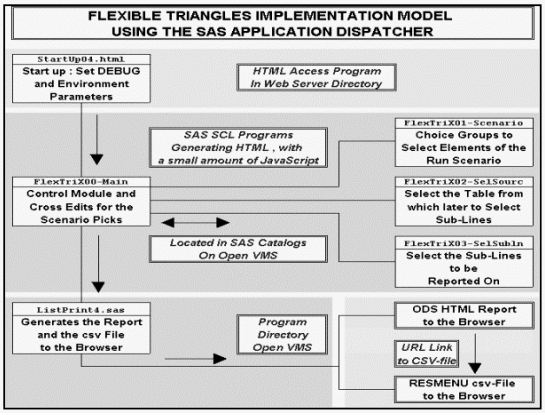
- Data currently hosted in Version 6.12
- SAS/CONNECT used to access OpenVMS RESMENU data libraries from development PCs
- Only format catalogs are cloned to PC because of problem with PROC APPSRV allocating LIBRARY libref



Part 3: Programming Techniques

Part 3: Programming Techniques

- See the diagram on next slide
- Four component screens
  - MAIN
  - SCENARIO
  - SELSOURC
  - SELSUBLN



Program Flow: SCL Generating HTML

- Check or create session
- Retrieve any "save" macro variables
- Open _webout
- Create HTML header, body tags
- Build Form elements via subroutines
- Create "Submit" button via subroutine
- Write "save" variables, closing tags

Taming Code Creation Using SCL Lists

- SCL lists similar to arrays
- See Horwitz paper or documentation
- Each line of HTML loaded into list
- Can be saved to SLIST entry for reuse
- Write to _webout via DO-loop
- Be sure to use CLEARLIST and DELLIST functions to prevent memory leakage

Creating Headers and Titles

Fill Up The List

```
rc = Insertc ( htmlcode,  '<HTML>',  -1);
rc = Insertc ( htmlcode,  '<HEAD>',  -1);
rc = Insertc ( htmlcode,  '<TITLE> Flexible Triangle "Sources" Table </TITLE>',  -1);
rc = Insertc ( htmlcode,  '</HEAD>',  -1);
rc = Insertc ( htmlcode,  '<BODY>',   -1);
rc = Insertc ( htmlcode,  '<H3>FlexTri03-SELSUBL – Select Sublines </H1>',   -1);
rc = Insertc ( htmlcode,  '<H1>RESMENU Flexible Triangles</H1>',   -1);
rc = Insertc ( htmlcode,  '<H1>Sub-Lines Selection</H2>',   -1);
```

Dump it to _webout

```
Do JJ = 1 to Listlen( htmlcode );
   rc = Fput(fweb_out, Getitemc(htmlcode, JJ)
   rc = Fwrite(fweb_out) ;
End;
```

Creating Choice Group and SAVELIST

```
CLMTYPE_list = Makelist() ;
  rc= insertc( CLMTYPE_list,  '0 Losses, x-CL, x-Cats', -1);
  rc= insertc( CLMTYPE_list,  '1 Catastrophe Losses',  -1);
  rc= insertc( CLMTYPE_list,  '2 Continuing Litigation Losses', -1);
  rc= insertc( CLMTYPE_list,  '3 Losses Ex-Continuing Litig',  -1);
  rc= insertc( CLMTYPE_list,  '4 Total all Losses',  -1);

rc= savelist( 'catalog',
          'RALPHLIB.ChoicGrp.CLMTYPE.slist',
           CLMTYPE_list,
           dummy_list,
          'Contents of Ralph List' ) ;
```

Use of Colour

- Sample coding convention

Color	*Use*
Black	SCL code except where noted below
Blue	Comments
Red/Orange	Items that should stand out, such as section labels, RETURN statements, key data set and file names, LINK statements
Green	HTML code and messages "put" to SAS log
Pink	JavaScript

- Color a selected block using function keys assignments

Modules and Subroutines

- Internal modules are "link" subroutines improve understandability and ease of maintenance
- External modules are separate programs that concentrate commonly used code sequences in one place
- External modules run via CALL DISPLAY

Modules and Subroutines (con't)

```
/* Write HTML header  */
   ImageFile = 'http://'||trim(server_nm)||'/Ralph/logo.gif';
   Call Display ( 'RALPHSCL.RESCAT.S01HDR.SCL',
                  fweb_out,
                  "FlexTri01-MAIN – Control Module",
                  "MAIN SCREEN" )
/*Create Dummy Form at top to Display the Debug Parameter  */
   Broker = 'http://'||trim(server_nm)||'/cgi-bin/broker.exe ';
   Link S02_FormStart ;
/* Perform the Cross Edits for the Current Choice Group Selections  */
   Link S03_CossEdits ;
/* Get the  Current Choice Group Selection Descriptions  */
   Link S04_OptionDesc ;
/* Create the Form for Scenario Choice Group Selections and Errors  */
   ProgramName = "Ralphscl.Triangle.FlexTriX01_Scenario_2.SCL";
   Link S05_ChoiceGrps ;
```

Modules and Subroutines (con't)

```
/* Create the Form for Selecting the "Sources" Table  */
   ProgramName = "Ralphscl.Triangle.FlexTriX02_SelSourc_2.SCL" ;
   Link S06_SourceTable ;
/* Create Form to Go and Select the Sublines  */
   ProgramName = "Ralphscl.Triangle.FlexTriX03_SelSubln_2.SCL" ;
   Link S07_SelSubline ;
/* Create the Form for Launchng the Report  */
   ProgramName = "RESRVPGM.SB0615WB.SAS" ;
   Link S08_MakeRept ;
/* Finish off the HTML for the Screen ..... */
   Link S09_CloseOut ;
```

Creating Selection Lists from Data Sets

- With "Close Coupling" to the front-end, whenever a table is changed, the user screens reflect the revision without performing any special maintenance
- In HTML interface, data sets are used to populate drop-down selection lists
- See example on next screen

Creating "Sources" Selection Group

```
/*  Open the SAS Dataset Table */
dsid_SOURCE = open( 'RESRVTBL.SOURCES' , 'i') ;
call set( dsid_SOURCE ) ;

/*  Create the Label Above the Select Group and Start the Group HTML */
rc = insertc( htmlcode, '<P><B>Select Source Table (i.e. Division):</B><BR>', -1);
rc = insertc( htmlcode, '<SELECT Name="SrcList" Size="8" >',                  -1);

/*  Load up the Select List from the SAS Dataset Table */
do while (fetch(dsid_SOURCE) = 0) ;
        FirstChar = substr(SRCENAME,1) ;
        TABLE = 'XB20LIX'||FirstChar;
        rc    = insertc( htmlcode, '<OPTION Value="'||trim(Table)||'" >'||
                trim(SCRCDESC) ||'</OPTION>', -1) ;
end;

/*  Terminate the Select Group and Close the File  */
rc= insertc(htmlcode, '   </SELECT></P>');
dsic_SOURCE = Close( dsid_SOURCE ) ;
```

Use or Not to Use JavaScript

- Required for "within-screen" linking
- See the disabled "Alert" on next slide
- Alternative to JavaScript is code edit-checks with the SCL that generates next screen
- Authors find SCL alternative is usually easier to implement than JavaScript

JavaScript Example

Used to copy the value associated with a Selected Option to a Text Box Display of the value, "Save_Source".

```
function FillSelectedSource(form) {
   var result = ""
   for (var i = 0; i < form.SrcList.length; i++) {
      if (form.SrcList.options[i].selected) {
          result = form.SrcList.options[i].value
          }
      }
   form.save_source.value = result
    // alert("You have selected:" + result)
   }
```

Development Environment

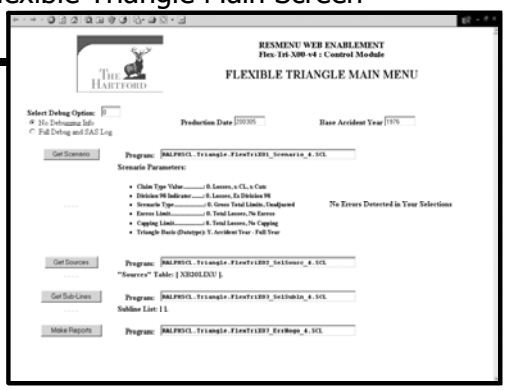
- Triad consisting of desktop SAS, HTML editor, and web server
- Development computer is LOCALHOST
- SCL variable TESTAF is a switch to send output to file instead of _webout
- HTML editor helpful for debugging JavaScript

Run-Time Debugging

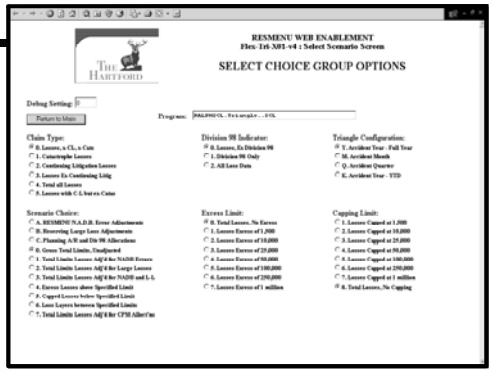
- Use PUT statements to debug SCL
- _debug=131 dumps SAS Log and name/value pairs to browser
- Real version of MAIN has choice group to set value of _debug

Examples of RESMENU Screens and Triangle Reports

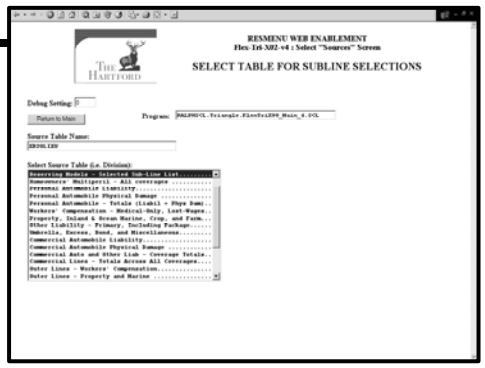
Flexible Triangle Main Screen



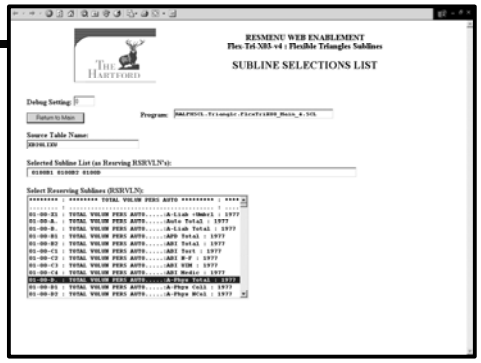
Scenario Screen



Select Table for Sub-Lines



Select Sub-Lines



Incurred Loss Development Report

75-21-K3 : Oil Storage Tank Multi-Peril
INCURRED LOSS TRIANGLE - as of 12-31-1998
<< Dollars in Millions >>

Devel Year	Accident Year 1989	1990	1991	1992	1993	1994	1995	1996	1997	1998
1	98.5	115.0	117.5	111.9	110.5	105.7	117.8	134.6	130.7	158.6
2	136.4	160.3	168.7	161.9	158.5	147.8	161.4	183.7	182.1	
3	153.0	181.4	190.2	188.9	178.6	170.7	183.7	209.8		
4	158.8	193.8	199.9	197.8	186.5	182.5	190.5			
5	162.2	195.8	204.0	199.7	188.6	180.7				
6	162.7	196.7	204.6	199.1	188.1					
7	163.8	196.6	205.1	199.3						
8	165.0	197.5	204.6							
9	165.1	197.3								
10	164.9									

Conclusions

- A good approach for those who want table-driven SAS applications but are not ready to embrace the object-oriented requirements of Java and AppDev Studio
- Authors are still interested in better exploiting AppDev Studio over long-term

Contact Information

Michael L. Davis
Bassett Consulting Services, Inc
10 Pleasant Drive
North Haven CT 06473-3712
Email: michael@bassettconsulting.com
Web: http://www.bassettconsulting.com

Ralph L. Leighton
The Hartford Financial Services Group
c/o Corporate Actuarial Department
1 Hartford Plaza HO-1-02
Hartford CT 06115
Email: rleighton@thehartford.com (business)
rleighton@cox.net (home)