CAR Tool using Stored Processes

Bank of Montreal

Risk Capital and Stress Testing

13 / 12 / 2013

Car Tool using Stored Processes

- **In the past Calibration was done using a spreadsheet.**
  - Difficult to maintain
  - Not auditable
  - Not replicable
  - Open to user error (cut and paste)
- **Calibration using SAS Stored processes**
  - Auditable
  - Replicable
  - More secure

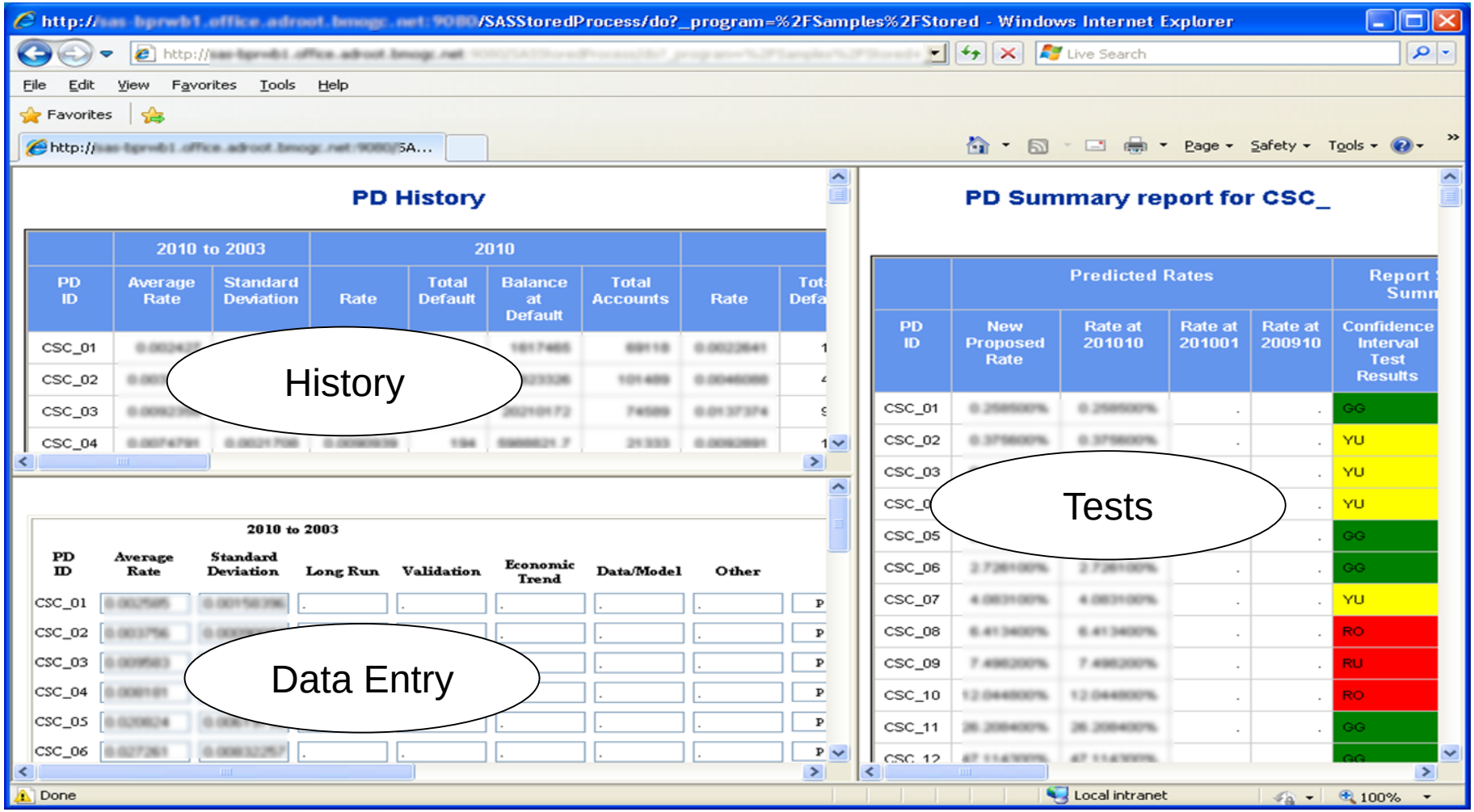
Notes:

- **In the Past Calibration was done using a spreadsheet**

- Some problems with using a spreadsheet for the process were that
  - it was **not auditable** - the data could be changed,
  - it was **not replicable** - there was no way to replicate these manual changes
  - it was **open to user error** - the user could use cut and paste to copy values and formulas in the spreadsheet

- **Calibration using SAS Stored Process**
  - **Auditable** - The new process locks down the data, the user is not able to change the data version control is in place.
  - **Replicable** - All parameters that are input are captured in sas tables and is fully replicable
  - **More secure** – users can only change parameters, data and code is locked down

Car Tool using Stored Processes



Notes:

This is what the tool looks like when launched.

There is one step prior to this that allows the user the ability to select the model they want to calibrate and the effective date of the data they want to use

Car Tool using Stored Processes

- **Development process: 3 areas were identified**
  - Historical reporting
  - Entering parameters
  - Tests – historical data against new parameters

Notes:

- **3 areas were identified**

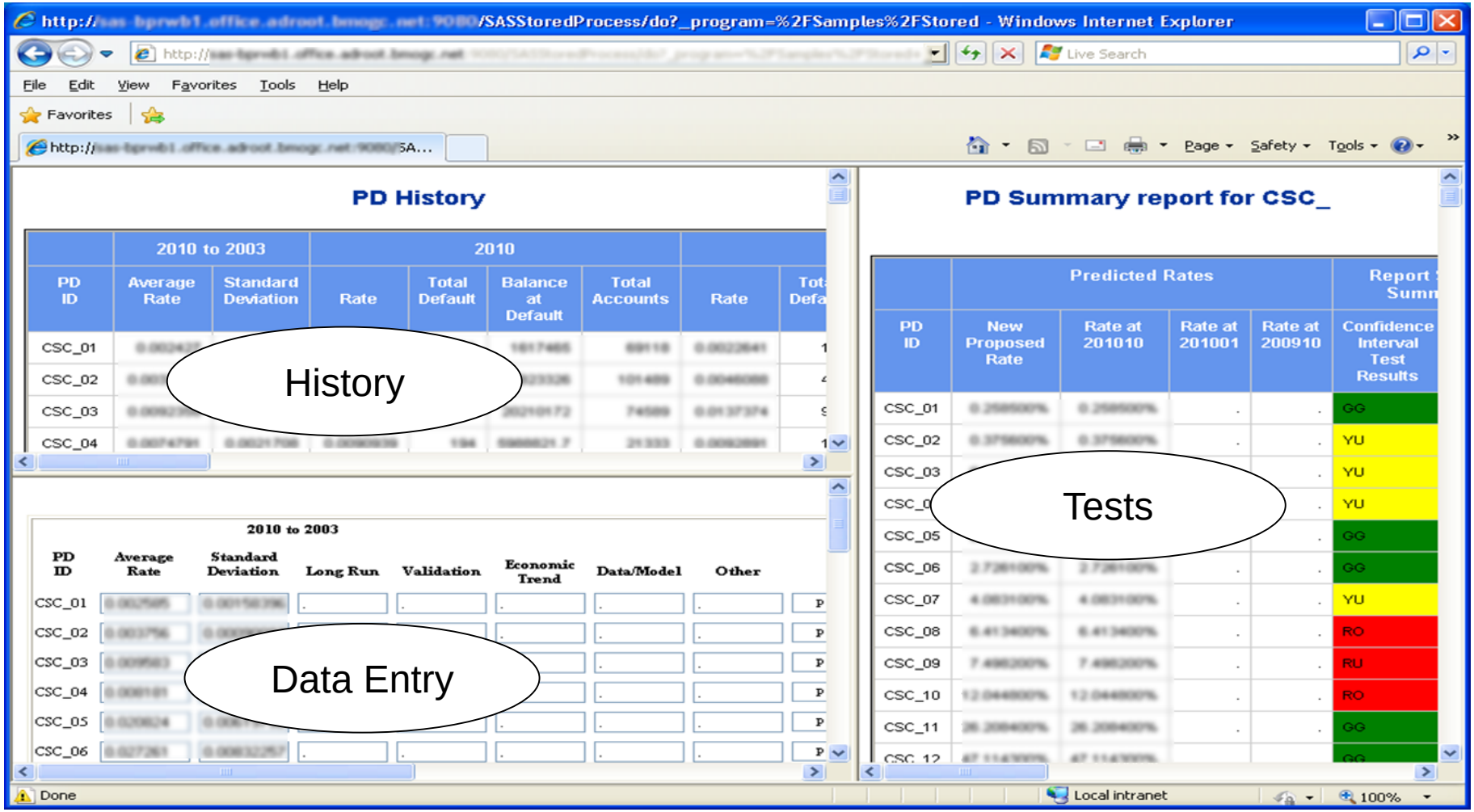
- Through discussions with the user there were 3 main areas were identified:

- **1. Historical reporting** – this contains static information about history.
  - The data is extracted and loaded in a separate process (pre process).
  - This is the most intensive processing portion because it extracts data from the multiple tables on the datamart at account level and various reporting datasets at segment level from both windows and unix. The data is summarized and merged together to create two master reporting datasets one at segment level and one at the full segment intersect level (PD, LGD and EAD).

- **2. Entering parameters** – HTML form that the user inputs the new parameters
  - The user required the averages and standard deviation calculations (from history) pre populated into the form. They had to be able to edit the values where adjustments were necessary as well as add in additional rates to create new parameters.

- **3. Test results** – testing new parameters against historical data
  - Existing tests (that were from the spreadsheet) were incorporated into the process, summary reports and colour coding was added.
  - The tests all use proc report as an output. ODS is used and the output is set to _webout, this allows the stored process to stream the report back via the portal rather than creating static reporting to a specific location.

Car Tool using Stored Processes



Car Tool using Stored Processes

- **Process uses**
  - SAS Stored Processes
  - Base SAS
  - HTML
- **Environment**
  - AIX 64
  - Enterprise Guide 4.3
  - SAS 9.2

Notes:

- **Process uses**
- **SAS Stored Processes -** A stored process is used to launch the application.

  - The user selects the model and effective date of the data that they want.
  - There are hidden macro variables that the user doesn't see that controls which environment is used as well as which data to point to.
    - *Eg. The env macro variable in production is set to "prod", when the process executes code called bmo_param_setup_prod (&bank._param_setup_&env) is executed which assigns the data library to the "prod" data. This allows multiple stored processes to be created using the same code but pointing to separate data repositories.*
  - Macro variables that are created from this are then passed through to the program.
  - The program creates HTML that creates the 3 panel layout and executes a separate stored process for each panel.

- **Base SAS –** is used to create reports and HTML

  - The Tests panel uses base sas to join the historical data with the parameters updated in the data entry panel and then proc report to output the results

- **HTML**
- The form panel uses sas to create the HTML form

  - A button is put at the end of the form that allows the user to update the sas tables with user input rates, this is a stored process that launches the update code then reloads the form with the updated values.

Car Tool using Stored Processes

- **Three most important things to remember…**
  - Know your data
  - Know your data
  - Know your data

Car Tool using Stored Processes

- **Other important things**
  - Understanding the data
  - How to store the data
  - How to get the shortest run time for the tests

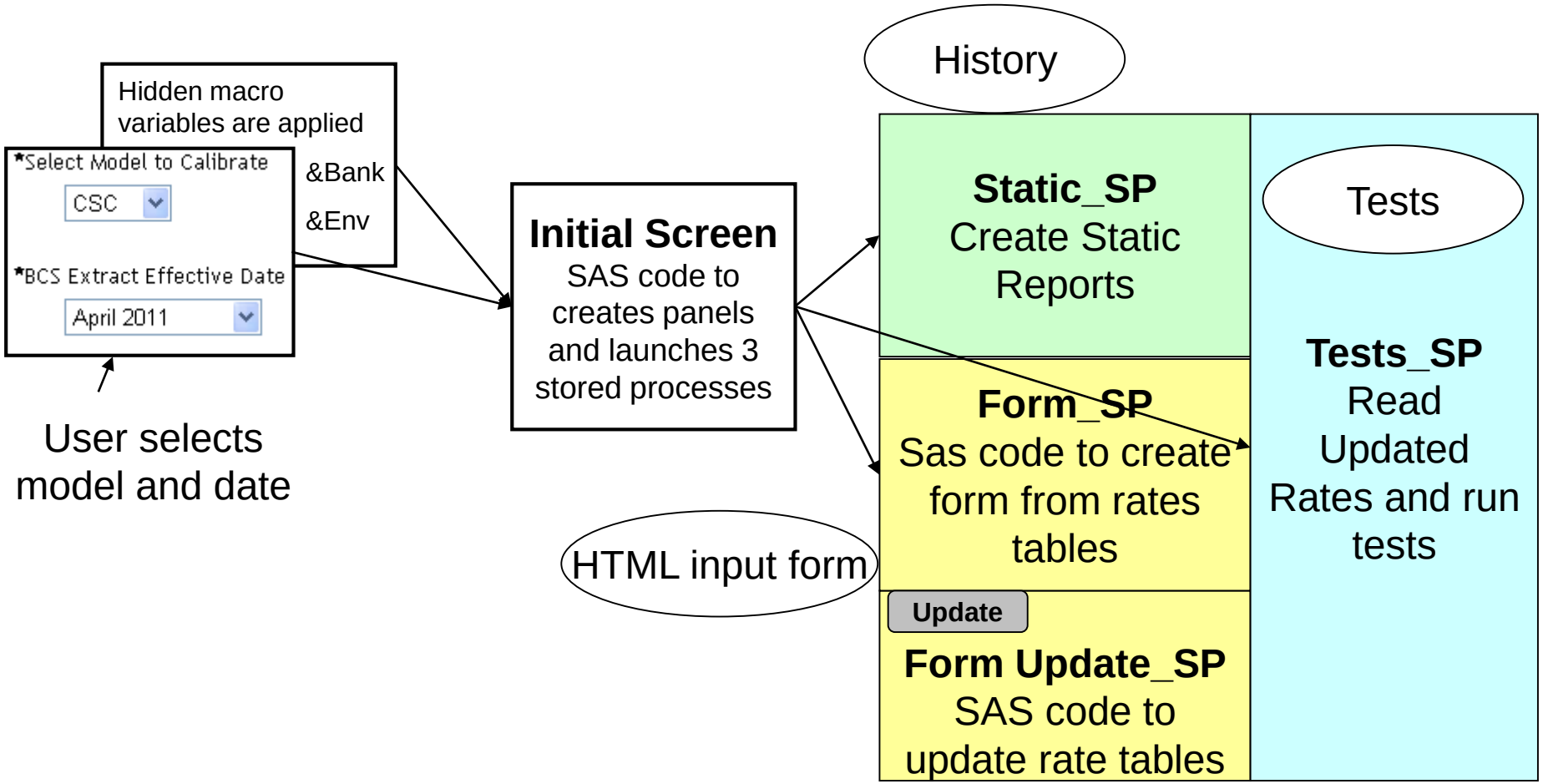
Notes:

- The most important part of the whole process is to understand the data.

- **Data:**
- how the data is used by the user
- where it comes from
- how it fits together
- **Storage:**
- how to get the best performance out of that data
- how to arrange the data to get the best performance
- at what level should the data be summarized
- can it all be stored together or should it be split up
- **Runtime:**

  - the shortest run time comes from a combination of all the above. It may make most sense to store the data at the lowest level required for all reporting but if only 10% of the reporting is required at that level and the other 90% needs data at a more summarized level it may make more sense to have the data a both levels, the larger data at a lower level for the 10% of the reports and the smaller set of data for the other 90%, this is how the data ended up being stored for this process. This way the 90% of reports don't have to spend resources on summarizing each time the report is called. - you have more data but faster response, the compromise here was space

Car Tool using Stored Processes



- This shows the process that runs the application.

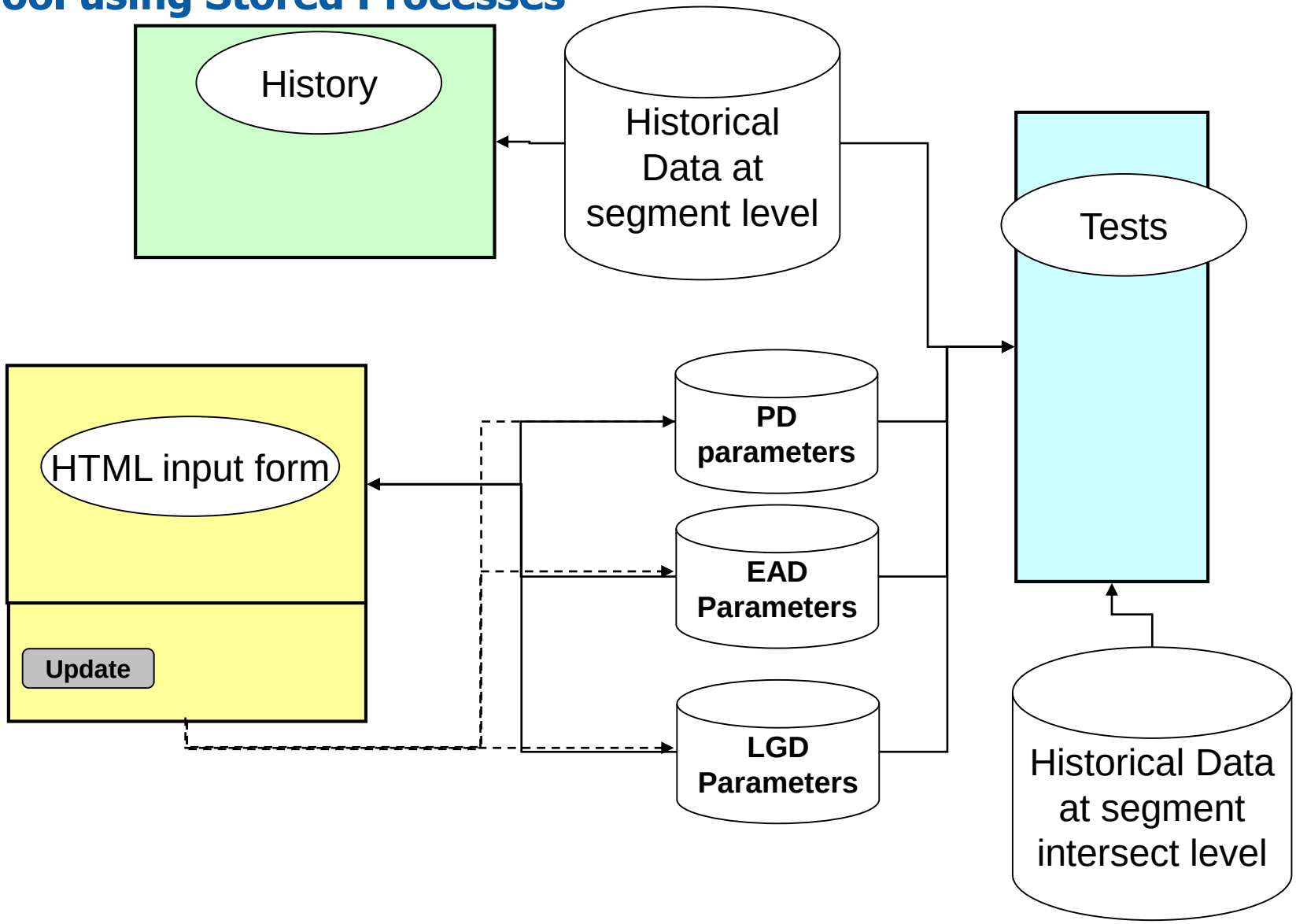
- The application can be launched from:
- the Stored process portal
- the information portal
- a link (url)

- We used the information portal because it allows the user to customize their page and keep all applications in the same place.

- The first stored process allows the user to select the model and the effective date of the data.

- Three stored processes are executed to create the three panels, a fourth is called when the form update button is selected, this fourth stored process is only run at the time the button is pressed and captures the inputs made by the user, and when complete re launches the form with the updated rates ready to cycle through again.

Car Tool using Stored Processes



Notes:

- This shows the data that is used by each process and is prepared ahead of time.

- The History Panel uses the segment level summary data – higher level summary - less data.

- HTML input form uses the parameter data, parameter data could have been saved in one table but kept it separated to make querying simpler.

- Tests uses the segment level summary data for most of the reports, the segment intersect summary data (lowest level data for 10% of reports) and the parameter data

Car Tool using Stored Processes

Initial Screen:

```
data _null_;
file _webout;
put "<html>";
put "<frameset cols='75%,25%'>";
put     "<frameset rows='50%,50%'>";
put       "<frame name='static_rpt' scrolling='yes'  target='static_rpt'";
put         "src='https://your_server_name:port/SASStoredProcess/do?_service=default%nrstr(&_debug)=0
           %nrstr(&_program)=/location_of_stored_process/Static_SP";
put         "%nrstr(&hostnm)=&hostnm%nrstr(&lprot)=&lprot%nrstr(&uprot)=&uprot
           %nrstr(&model)=&model %nrstr(&bank)=&bank %nrstr(&bal_month)=&bal_month  %nrstr(&env)=&env
           %nrstr(&user)=&_metauser'>";
put       "<frame name='dynamic_rpt' scrolling='yes'  target='dynamic_rpt'";
put         "src='https:// your_server_name:port /SASStoredProcess/do?_service=default%nrstr(&_debug)=0
           %nrstr(&_program)=/location_of_stored_process/Form_SP";
put         "%nrstr(&hostnm)=&hostnm%nrstr(&lprot)=&lprot%nrstr(&uprot)=&uprot
           %nrstr(&model)=&model %nrstr(&bank)=&bank %nrstr(&bal_month)=&bal_month  %nrstr(&env)=&env
           %nrstr(&user)=&_metauser'>";
put     "</frameset>";
put  "<frame name='test_rpt' scrolling='yes'  target='teststatic_rpt'";
put    "src='https:// your_server_name:port /SASStoredProcess/do?_service=default%nrstr(&_debug)=0
       %nrstr(&_program)=/location_of_stored_process/Tests_SP";
put    "%nrstr(&hostnm)=&hostnm%nrstr(&lprot)=&lprot%nrstr(&uprot)=&uprot
       %nrstr(&model)=&model %nrstr(&bank)=&bank %nrstr(&bal_month)=&bal_month  %nrstr(&env)=&env
       %nrstr(&user)=&_metauser'>";
put " <noframes>";
put " <body>";
put " <p>This page uses frames, but your browser does not support them.</p>";
put " </body>";
put "</noframes>";
put "</frameset>";
put "</html>";
run;
/* debug = 0 turns off; 131 turns on log if a problem; 128 turns on log full time*/
```

Notes:

- A stored process is created around this code. When creating the stored process there are things that you have to remember:

    i. SAS Code: under Include code for, you must de-select the stored process macros option, and select the Global Macro Variables and Libname references. You will be asked if you want to apply the default stored process macros %stpbegin and %stpend – you do not want them – select NO.

    ii. Execution Options: you must run this on the stored process server and select streaming for your results (_webout streams the data back to your browser).

    iii. This example shows that the screen will be split first vertically 75/25 then horizontally 50/50. Each frame executes a further stored process. If the browser does not support frames then a message will appear.

    iv. Prompts: this is where the model selection options are set up. In this example the model and bal_month are set up as macro variables that the user selects. These macro variables are passed through all subsequent processes. There are two hidden macros created that drive which data is being read (bank) and which code is being run (env).

Car Tool using Stored Processes

Form code was split into 5 parts

1. Form Start – HTML hearder

```
put '<Form Action="/SASStoredProcess/do" method="POST">';
```

2. Form Head - sets up headings for each section of the form

```
%macro form_head(TYPE);
%if &type = %STR(PD) %then
%do;
   put '<br>';
   put '<div>';
   put '<div align="left">';
   put '<table class="Table" cellspacing="2" cellpadding="2" rules="none" frame="box" >';
   put '<thead>';
   put '<tr>';
   put '<th class="c Header" style=" font-size: 8pt; font-weight: bold;" colspan="2" scope="colgroup"> </th>';
   put '<th class="c Header" style=" font-size: 8pt; font-weight: bold;" colspan="2" scope="colgroup">'"&to_year" ' to
&from_year"'</th>';……..
%end;
%else ……
%if &type =%STR(LGD_NIX) %then
  %do; …….
  %end;
   put '</tr>';
   put '</thead>';
%mend form_head;
```

Notes:

- Forms:
- A stored process is wrapped around a process that generates an html input form using Base SAS. The from generation code is split into five parts, this was done so that the form could be dynamically generated from the data.

    i. Form_start - this is just the html header setting up the form. The most important part here is to set the Form as Action and point to the stored process server and the method to POST

    ii. Form_head – this sets up the headings for each section of the form. Separate headings are needed for PD, LGD and EAD and these are passed through as type (pd example below).

Car Tool using Stored Processes

3. Form Body

```
%macro form_body(TYPE);
put '<tbody>';
  %do i = 1 %to &&&type._&model._num;
     put '<tr>';
     %if &type = %STR(PD) %then
     %do;
        put '<td class="c Data" style=" font-size: 8pt; width: 75px;">'''&&&type._id_&model&i''' </td>';
        put '<td class="l Data" format:0.0000 style=" font-size: 6pt;" tabindex=1 false="decimial_value">
        <input type=text size="8" name=' '''' "_&type.&&&type._id_&model&i..R1" '''' 'value='''&&&type._R1_&model&i'' ' style=" font
        size: 8pt;" /></td>';
        put '<td class="l Data" format:0.0000 style=" font-size: 6pt;" tabindex=2 false="decimial_value"> <input type=text size="8“
         name=' '''' "_&type.&&&type._id_&model&i..R2" '''' 'value='''&&&type._R2_&model&i'' ' style=" font-size: pt;" /></td>';
          …….
       %end;
       %else
        %if &type =%STR( LGD) %then
        %do;…….
        %end;
        put '</tr>';
  %end;
  put '</tbody>';
  put '</table>';
  put '</div>';
  put '</div>‘
%Mend form_body;
```

4. Form Button – creates button which launches stored process

```
put '<input type="hidden" name="_program" value="/location_of_stored_process/Form_Update_Rate_SP" &_debug=0 />';
put '<input type="submit" Value="Update Rates"/>‘
```

5. Form End – this closes the HTML Form

Notes

iii. Form_body – this creates the input form for each section, name and value are the most important parts for each cell. Name will be the macro variable name that resolves to the cell value, value is the resolved value of the macro variable that pre-populates the cell (pd example below).

There are two macros that run prior to the form

- **%Cr8var** which creates macro variables used in the process for “name”. The data is read in and the number of segments for the model is determined and a macro variable made up of the type and model is created. This controls the loop that controls the number of rows created in the form. Each segment is given a macro variable with a count variable so that it can be referenced through a loop
- *e.g. _&type.&&&type._id_&model&i..R6 resolves out to pd_id_csc_R6 this then becomes the macro reference for the value that is input by the user and passed back through.*
- **%Recent_rates** creates macro variables used in the process for “value”. The most recent version of the rates are read from the rate table. Each part of the rate is given a macro variable value
- *e.g. &&&type._R6_&model&i resolves out to pd_R6_CSC_ which in turn resolves to the current version of the rate in the rate table for that cell.*

iv. Form Button - this creates the button that when selected lauches the update button which will take the user entered values and update the parameter tables with these values. The hidden values are the additional macro variables that will be passed through to the next process

v. Form_end – this closes the form

Car Tool using Stored Processes

Form Update

- This is launched when the button on the Form is selected then redisplays the form with the updated version

```
%macro updatepd;
%put &&&&pd_&model._num;
data car_pd_updt;
length seg_id $10  r1 r2 r3 r4 r5 r6 r7 8 r10 $255;
model = "&model";
%do i = 1 %to &&&&pd_&model._num;
     seg_id = "&&&&pd_id_&model&i.";
     r1       = &&&&&&_pd&&pd_id_&model&i..R1;
     r2       = &&&&&&_pd&&pd_id_&model&i..R2;……
     RATE  = sum((sum(1,R3)*R1),(sum(R4,R5,R6,R7)*R2));
     /* pod_pct = (1+long_run)* average_rate+(validation+economic trend+data Model+other)*standard deviation*/
     output;
%end;
run;
%mend updatepd;
%updatepd
;

%include src_rpt("form.sas");
```

Notes:

- This is launched when the button on the Form is selected. When the stored process is launched the macro variables in the name portion are resolved to the values in the value portion, these are then converted to variables, the version number is incremented by 1and the data is saved in the respective parameter files. At the end the form is re launched and will show the latest (updated) values.
- Now we have each iteration of calibration stored in SAS datasets so can extract the latest version of parameters or recall older versions if required.
- Though this application is specific to one use, the process has been recycled for stress testing where the parameters are replaced by stress scalars, the process is a calculation engine to calculate EL and RWA. Because our platform is Unix it is not often easy to deliver results to the users who insist on Excel output. One enhancement added was the ability to launch excel and produce results in the spreadsheet and the user is then prompted to save the spreadsheet wherever they want.

Car Tool using Stored Processes

Other:

- Delivering output in Excel

```
%global _ODSDEST _ODSSTYLE _ODSOPTIONS;
%macro output;
  %local OLD_HEADER OUTPUT_FORMAT;
  %let OUTPUT_FORMAT = &_ODSDEST;

  %if (%upcase (&out_typ) eq EXCEL) %then %do;
    %let _ODSDEST = TAGSETS.EXCELXP;
      data _null_;
        rc = stpsrv_header('Content-type','application/vnd.ms-excel');
        rc = stpsrv_header('Content-disposition',"attachment; filename=master_&type_cd..xls");
      run;
  %end;

  %if (%upcase (&out_typ) eq HTML) %then %do;
    %let OUTPUT_FORMAT= Compact HTML;
  %end;
%let _ODSSTYLE=journal;
/* Create output */
%stpbegin;
%xls_rpt;
run;
%stpend;
%mend output;
```

- Getting round metadata restrictions
  - Want to create dynamic selection lists but don't have access to the metadata server?
    - Use SAS code to dynamically create your HTML form

- If you are on a Unix box you know how frustrating it is create output for users in Excel format on Windows.
- In another application I created the ability to create output directly to Excel. Instead of writing the data to the Unix server this piece of code opens up an excel spreadsheet on the users desktop then writes the output to this file that they can then save on windows. In this example I have a variable set in the first screen that allows the user to select where they want the output directed – to the browser or excel.
- In our installation we are extremely limited on what we can do, especially on the metadata server. Creating things like cascading prompts and dynamic selections becomes next to impossible when you don't have access to register metadata.
- Using SAS code to generate HTML allows you to get around all these issues. The only limit to flexibility then becomes your coding ability. You can use your code to dynamically create input forms, list selections or whatever you want.
- You can use the stored process wizard as a shell to launch your code then let the code drive what you see on your screen. You can customize using logos, backgrounds or whatever you want.

Resources

- http://support.sas.com/rnd/itech/doc9/dev_guide/stprocess/datapass.html
- http://support.sas.com/resources/papers/proceedings09/031-2009.pdf
- http://www.w3schools.com/html/html_forms.asp

Thanks to:

Harry Droogendyk for pointing me in the right direction

Contact

Anita.Measey@bmo.com