PhUSE 2008

Paper CS09

Automating Elements of Validation in the Context of an ISE/ISS

P. Kearney, Online Success Ltd., Preston, England

ABSTRACT

The purpose of this paper is to show how we ensured that the derived datasets we created for an Integrated Summary for Efficacy/Safety (ISE/ISS) adhered to the formal specification which we had been given, with the minimum of effort and the maximum of certainty.

The standard for these datasets was provided by a set of empty SAS® datasets. We should produce derived datasets which were identical in all details to the structure of those datasets.

We decided to create a macro (%STRUCTURE) which would enforce those standards.

Program specifications describing the derivation rules for each field were produced and used to govern the writing of the derivation programs.

%STRUCTURE would also create an audit trail highlighting errors in the program specifications or in the derivation programs.

As the Client's requirements evolved we were able to make a precise assessment of the impact of the required changes.

We will describe how we guaranteed that the derived datasets which we created for that ISE/ISS were consistent with the formal specification for those datasets.

INTRODUCTION

The Client specified for each derived dataset the

Variables
Field length
Required formats
Labels
Sort Order
That variables which could not be created (because data had not been captured on a contributing trial) should be present but set to missing.

The %STRUCTURE macro would enforce all of the client's requirement. In addition it was possible to request that %STRUCTURE should check and report on duplicated records.

This strategy meant that during the creation of the work file which would be converted into a final derived dataset it was necessary only to ensure that the basic format (numeric or character) of each particular field needed to be defined. It was not necessary to define formats or labels or to sort that work file into any particular order.
If an error had been made which meant that the %STRUCTURE macro could not produce an output dataset which adhered to the required standard, then an error message was produced which highlighted the problem. This

covered both the cases where a programming error had been made, and those cases where the specification of the derivation program itself deviated from the required standard. In practices both of these situations did arise (all be it infrequently), and the error messages produced by %STRUCTURE indicated what problems needed to be rectified.

THE DATASET SPECIFICATION

A Word document was produced for each of the required datasets, which gave the source and where appropriate the derivation rules for each of the fields needed for that dataset. If a field could not be created, it was specified here as being missing.

THE ENFORCEMENT OF THE CLIENT'S STANDARDS

For each of the required datasets a derivation program was written which adhered to the specification described above.

The final step in each derivation program involved the invocation of the %STRUCTURE macro, which would be given a work file, and the identity of the required output dataset. After checking that the work file was consistent with the client's standard for that dataset, %STRUCTURE would impose field lengths, formats and labels upon the output dataset, which would also be sorted into the required order. In every industry clerical tasks are delegated to computers. In this instance we had succeeded in delegating to the machine the task of imposing the client's standards upon the derived datasets which we were producing.

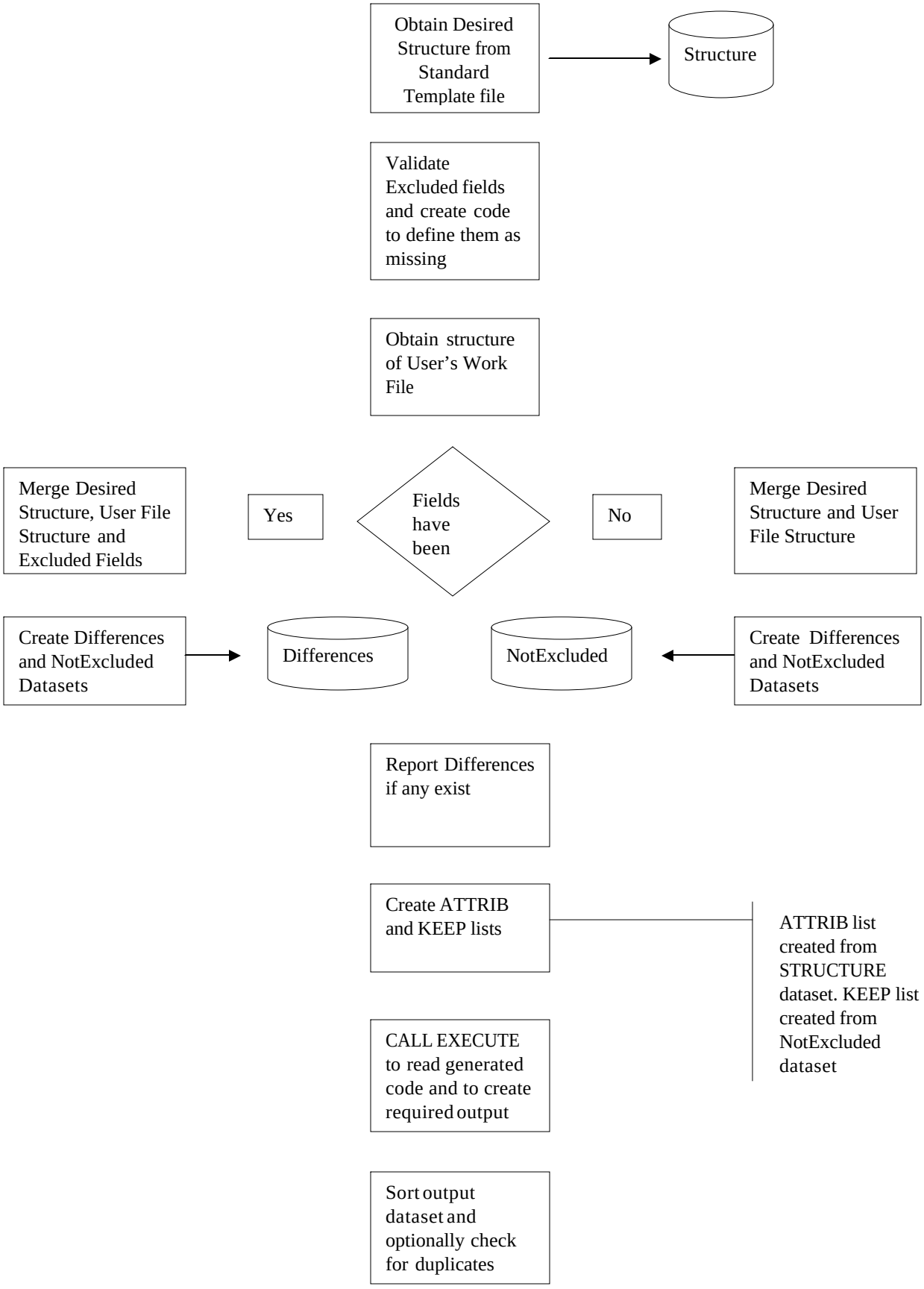
ERRORS AND THE RESOLUTION OF PROBLEMS

If the derivation program had not created a required field then a structural error report would be produced identifying the missing field. On occasion a required field would have been omitted from the dataset specification. This would cause the same error message to be produced. A manual check between the program and the specification would identify such an error. The automated production of an error report which served to identify precisely the error which had been made, and which would address sins of omission in either the derived program specification, or in the derived program itself, was a novel characteristic of this manner of working. In this instance also we had succeeded in delegating to the machine a large part of the task of validating both program specifications and programs themselves.

THE IMPACT OF CHANGE

Initially the client's specifications for the derived datasets were themselves described as being fixed and unchangeable. Over the lifetime of the project however, reality forced some changes to be made in those specifications. This would simply entail our receiving an updated version of the empty datasets from the client, and our using these to drive a rerun of the derivation programs. If the change entailed simply a change in a label or the lengthening of a character field, then %STRUCTURE would automatically apply such a change. If the change was more radical than the error reports which would be produced gave us the means to produce a very precise estimate of the impact of that change. In this instance also we had succeeded in delegating to the machine a large part of the task of implementing change.

THE OVERALL MACRO LOGIC



THE CLIENT'S STANDARD TEMPLATE DATASET

A PROC CONTENTS was run on the Client's Standard Template database, and OUT= was used to create a Standard Template dataset. This would be used by the %STRUCTURE macro in order to enforce the Client's standards. This dataset contained the details for each of the possible types of dataset required by the Client e.g. LAB AE DEMOG etc.

PARAMETERS

%STRUCTURE is defined with the following parameters

TEMPLATE Defines the particular type of dataset to be created e.g. LAB AE DEMOG etc.
DSIN Input dataset to be processed - a work file containing all the fields required for the output dataset
DSOUT Output dataset to be created - the name of the permanent file to be written by the macro
EXCLUDE List of variables present on the template dataset to be set to missing on the DSOUT for this study
UNIQUE Is a check to be performed that observations on the output dataset are unique? (Y/N)

PREAMBLE

The macro begins by deleting any pre-existing error reports, and by then establishing the required sort keys for the datasets which may be created. If a problem is found by %STRUCTURE then an error report will be created by the macro in the directory which will contain the output dataset. We only need to list the files in the output directory in order to verify that no problems have arisen. In other words if a directory listing of the output directory does NOT show any files beginning with the words Structural_Errors then we do not have any problems. We know that we have adhered to the Client's specifications. If we do have a problem, then that problem is described in the relevant Structural_Error file.

```
* Delete any error report previously created;

data _null_;
call system("rm ../../Structural_Errors_&dsout..lst");
run;

* Assign the sortkey to be applied to the final dataset;

%let by = suno sitpaten;

%if %upcase(&dsout) eq EFFPFTAS %then %do;
    %let by = suno subid efftype;
%end;

%if %upcase(&dsout) eq EFFPFT %then %do;
    %let by = suno sitpaten timept efftype ;
%end;
```

The sort keys for a further eleven possible datasets are defined in the same manner.

VERIFY TEMPLATE DATASET EXISTS

* Extract template dataset records from the Client's Standard Template ;

```
data structure;
set clients.standard_template(where=(memname="%upcase(&template)"));
run;
```

```
proc sort data=structure;
by name;
run;


%noobs(dsn=structure,mvar=scount); /* This macro puts the number of observations in the DSN parameter into  */
                                   /* the global macro variable named by the MVAR parameter  */

* Check that template dataset records do exist ;

%if &scount eq 0  %then  %do;
   data repl;
   attrib txt format=$80. length=$80;
   txt="Template for &template not present on the Standard Template dataset";
   output;
   stop;
   run;

   filename struct "../../Structural_Errors_&dsout..lst";
   proc printto file=struct new;
   run;

   proc print data=repl;
   title "Structural Validation Error Report for &template";
   run;

   proc printto ;
   run;

 %end;
```

If there is a typo in the TEMPLATE keyword this step produces a Structural Error report.

REMOVE EXCLUDED VARIABLES FROM TEMPLATE DATASET

```
* Remove variables from the template dataset if we know that they are not present for a particular study ;
* but include them on the excluded metadata dataset so that code to force these variables to missing can ;
* be generated                                                                                            ;

   %if &exclude ne   %then  %do;
     data exclude;
     attrib name format=$32. length=$32;
     %let i=1;
      %do %until (%scan(&exclude,&i) eq );
         name=input(upcase("%scan(&exclude,&i)"),$varying32.);
         output exclude;
      %let i=%eval(&i+1);
      %end;
     stop;
     run;
     proc sort data=exclude;
     by name;
     run;
     data excludedmetadata;
     merge structure(in=s) exclude(in=e);
```

```
by name;
if e and not s then put "ERR" "OR: Entry on Exclude list not on Standard Template " name ;
if e then output excludedmetadata;
run;
proc sort data=excludedmetadata;
by varnum;
run;
```

CREATE CODE TO ASSIGN NULLS TO EXCLUDED VARIABLES

This creates code of the form
varname1 = .;
The assignnulls dataset will read and executed in the penultimate step below

```
data assignnulls(keep=txt);
set excludedmetadata end=finish;
attrib txt  format=$200. length=$200;
if type eq 1  then do;
  txt = trim(name) !! "=.;";
end;
else  do;
  txt = trim(name) !! "='';";
end;
output;
if finish  then do;
  txt = "run;";
  output;
end;
%end;
```

ESTABLISH STRUCTURE OF WORK FILE

* Extract the structure from the User's Work File. In principle a proc contents of the input work file should be identical to the proc contents of the client's standard template;

```
proc contents data=&dsin noprint out=userfilestructure;
run;

proc sort data=userfilestructure;
by name;
run;
```

COMPARISON OF WORKFILE AND TEMPLATE DATASET

* Compare the template and the User's Work File in order to ensure that the User's Work File contains all the template variables.

```
%if &exclude ne  %then  %do;
  data differences notexcluded;
  merge userfilestructure(in=u) structure(in=s) exclude(in=e);
  by name;
  if s and not (u or e) then output differences;
  if s and not e then output notexcluded;
```

```
  run;
%end;
%else  %do;
  data differences notexcluded;
  merge userfilestructure(in=u) structure(in=s) ;
  by name;
  if s and not (u) then output differences;
  if s then output notexcluded;
  run;
%end;
```

REPORT DIFFERENCES BETWEEN WORKFILE AND TEMPLATE DATASET

```
%noobs(dsn=differences,mvar=dcount);

* Are there any differences? If so report them    ;

%if &dcount ne 0 %then  %do;

  data repl(keep=txt);
  set  differences;
  attrib txt format=$80. length=$80;
  if _n_ eq 1  then do;
    txt="The following fields are missing from the input  file &dsout";
    output;
    txt=name;
    output;
  end;
  else  do;
    txt=name;
    output;
  end;
  run;

  filename  struct "../../Structural_Errors_&dsout..lst";
  proc printto file=struct new;
  run;

  proc print data=repl;
  title "Structural Validation Error Report for &template";
  run;

  proc printto ;
  run;

%end;
```

This step produces the key error report which identifies discrepancies between the client's standards for a particular dataset, and what we were actually creating.

CREATE CODE TO APPLY REQUIRED STRUCTURE TO DERIVED DATASET

This step creates code of the form

```
data derived.ae;
attrib aedate length=8 label='AE Date' format=date9.;
```
with an ATTRIB statement for each of the required fields.

* Create the step which will impose the Client's standards upon the User's Work File ;

```
proc sort data=structure;
by varnum;
run;

data attriblist(keep=txt frmat lngth);
set structure end=finish;
attrib txt format=$200. length=$200;
attrib frmat format=$40. length=$40;
attrib lngth format=$40. length=$40;
if _n_ eq 1 then do;
  txt="data derived.&dsout;";
  output;
end;
if format = '' then do;
  if type eq 1 then do;
    if formatl = 0 then do;
      frmat = compress(put(length,3.)) !! '.';
    end;
    else do;
      frmat = compress(put(length,3.)) !! '.' !! compress(put(formatd,3.));
    end;
  end;
  else do;
    frmat = compress('$' !! compress(put(length,3.)) !! '.');
  end;
end;
else do;
  if formatl ne 0 then do;
    frmat = compress(format !! compress(put(formatl,3.)) !! '.');
  end;
  else do;
    frmat = compress(format !! '.');
  end;
end;
if type eq 1 then do;
  lngth = compress(put(length,3.));
end;
else do;
  lngth = "$" !! compress(put(length,3.));
end;
txt="attrib " !! trim(name) !! " length=" !! trim(lngth) !! " label=" !! quote(trim(label)) !! " format=" !! trim(frmat) !! ';';
output;

run;
```

CREATE CODE TO RETAIN REQUIRED VARIABLES

This step creates code of the form

```
set aework(keep=
aedate
All of required fields are included in this keep list
);
```

run: If there are no variables to be excluded.

```
  data keeplist(keep=txt);
  set notexcluded end=finish;
  attrib txt format=$200. length=$200;
  if _n_ eq 1 then do;
    txt="set &dsin(keep=";
    output;
  end;

  txt=name;
  output;

  if finish then do;
    txt = ");";
    output;
    %if &exclude eq   %then %do;
    txt = "run;";
    output;
     %end;
  end;
```

GENERATE AND EXECUTE CREATED CODE

* Invoke the step which will impose the Client's standards upon the User's Work File and create it from the input work file;

```
  data _null_;
  set attriblist keeplist
    %if &exclude ne %then %do;
      assignnulls
    %end;
      ;
  call execute(txt);
  run;
```

OPTIONALLY CHECK FOR DUPLICATED RECORDS

* Sort the final dataset ;

```
  proc sort data=derived.&dsout;
  by &by;
  run;

%if %upcase(&unique) eq Y %then %do;

* Check that there are no duplicates on the final dataset for the specified sort key;

data _null_;
```

```
set derived.&dsout;
by &by;
%let i=1;
%let r=;
if not(
  %do %until (%scan(&by,&i) eq );
    &r
    last.%scan(&by,&i)
    %let i=%eval(&i+1);
    %let r=%str(or);
  %end;
  ) then put "WARN" "ING: Sort keys are not unique _n_=" _n_ ;
run;

%end;
```

CONCLUSION

This technique would also have been possible if the client's standards had have been described in any other electronic medium. In these cases producing the Standard Template dataset would have involved more work, but would still have been possible. The key point is that if standards are described in an electronic medium, then considerable time and effort can be saved by automating the imposition of standards. To do this results in benefits not only in terms of time saved, but also in improvements in quality. Those benefits will have an impact on the bottom line.

ACKNOWLEDGMENTS

My thanks go in the first instance to Dr Kurt Löffler and to Beate Hientzsch of Accovion GmbH without whose support this work would not have been possible, and secondly to the many members of staff at Accovion who made my time there both productive and enjoyable.

CONTACT INFORMATION

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

Paul Kearney
Online Success Ltd
16 Hornbeam Close
Penwortham
Preston
PR1 0PT
Work Phone: 0044 (0) 7801358197
Email: PKEARNEY2000@GOOGLEMAIL.COM

COPYRIGHT INFORMATION