PhUSE 2011

Paper PP07

A Simple Interface for defining, programming and managing SAS edit checks

Virginie Freytag, Clin Data Management, Rouffach, France
Michel Toussaint, Clin Data Management, Rouffach, France

ABSTRACT

If SAS ® is still widely used for data cleaning, EDC technology has however deeply changed the schedule of data cleaning management. The validation plan should now be defined at an early stage of the project, while programs must be ready to send off-line queries as soon as the data collection has started, in conformity with first-stage specifications.

During the course of the trial, it is not rare to modify the validation plan, and this along with the subsequent programming changes, should be managed as well.

In this context we have designed an optimized SAS workflow interfaced with a simple web-based application, allowing the simultaneous definition of edit checks and the automatic generation of the corresponding SAS source codes. The validation plan and underlying programming are then entirely managed through this user friendly web application offering an integrated solution for the fast, efficient and traceable implementation of edit checks.

INTRODUCTION

Our company Clin Data Management (CDM) provides fully integrated electronic data capture (EDC) solutions for clinical trials and SAS is used for most data management and statistical activities.

To ensure the accuracy of collected data and the validity of subsequent analysis we require a complete, adapted and efficient data cleaning process. With the use of EDC technology this starts at the earliest stage of the project and leads to data managers being faced with new challenges.
On the one hand the validation plan which gives an exhaustive description of all the checks to be applied to the data must be available at the beginning of the eCRF development stage. SAS programmers on the other hand must implement the corresponding SAS edit checks within very short timelines. This can rapidly becomes a challenge with the high number of edit checks usually needed to clean a clinical study database.
It is not rare that the validation plan is changed during the course of the study and subsequent programming changes must be managed as well.

Regarding the above issues we needed a solution to simultaneously allow:

- the definition of the validation plan
- fast, accurate and standardized SAS programming of corresponding edit checks
- the easy update of the validation plan and subsequent programs during the course of the trial

We started identifying requirements of the SAS programs used to target the inconsistencies stressing the flexibility of programming.
Next we designed a simple web interface which in combination with an SAS program enables management and automatic implementation of the validation plan.

This paper describes the main features of our solution.

REQUIREMENT SPECIFICATIONS

EDIT CHECKS

Different types of inconsistencies can be found in a study database and, depending on their complexity, we can define three types of checks:

- MM checks: missing values
- MC checks : missing values depending on the value of another item
- CC checks : inconsistency between several items

The two first are rather easy to implement using SAS code and appropriate *if/then* statements, whereas the CC checks can be more complex.

We identified the following issues for MM/MC edit checks and designed SAS edit checks programs accordingly:

- various types of items should be considered: character, numerical as well as date/datetime items
- some edit checks could deal with several simultaneously missing items
- several conditions should be considered for MC checks

As a result we defined two different edit check program templates, that can be automatically implemented.

VALIDATION PLAN

The main difficulty for the management of the validation plan relies on the match of programming and specifications.
The data manager must ensure that all defined edit checks are accurately implemented and that all implemented checks are defined.
A simple solution is to store the complete validation plan information in a database, and to use the updated information stored to execute SAS edit checks at a later time.

WEB USER INTERFACE

The need for a user friendly interface is obvious. Automatic code generation requires parameters to be entered and can be efficiently managed by a single web-form.

SOLUTION OVERVIEW

Our solution is composed of:

- a simple **Active Server Page Visual Basic** (ASP VB) application with the following functionalities:
  - creating a new edit check
  - updating an existing edit check
  - deactivating an existing edit check

The Validation Plan Management application also integrates security features with management of both the application and the file user access restriction.

- an **Oracle table** shown in Figure 2, which corresponds to the validation plan for the study: all edit checks and corresponding descriptions entered by the user are stored in this table, so that it can be used to execute all the off-line edit checks set up for the study.

It is important to note that information like message and recipient are not hard coded in edit check programs but retrieved from the validation table while programs are executed.
Management of dynamic messages (messages depending on the value of a specific item) is ensured by using a syntax adapted to identifying specific items.
This means that edit checks programs are not to be modified after the update of message or recipient information.

- an **SAS program** for creating and updating the validation plan table, and generating automatic source code depending on the type of check.

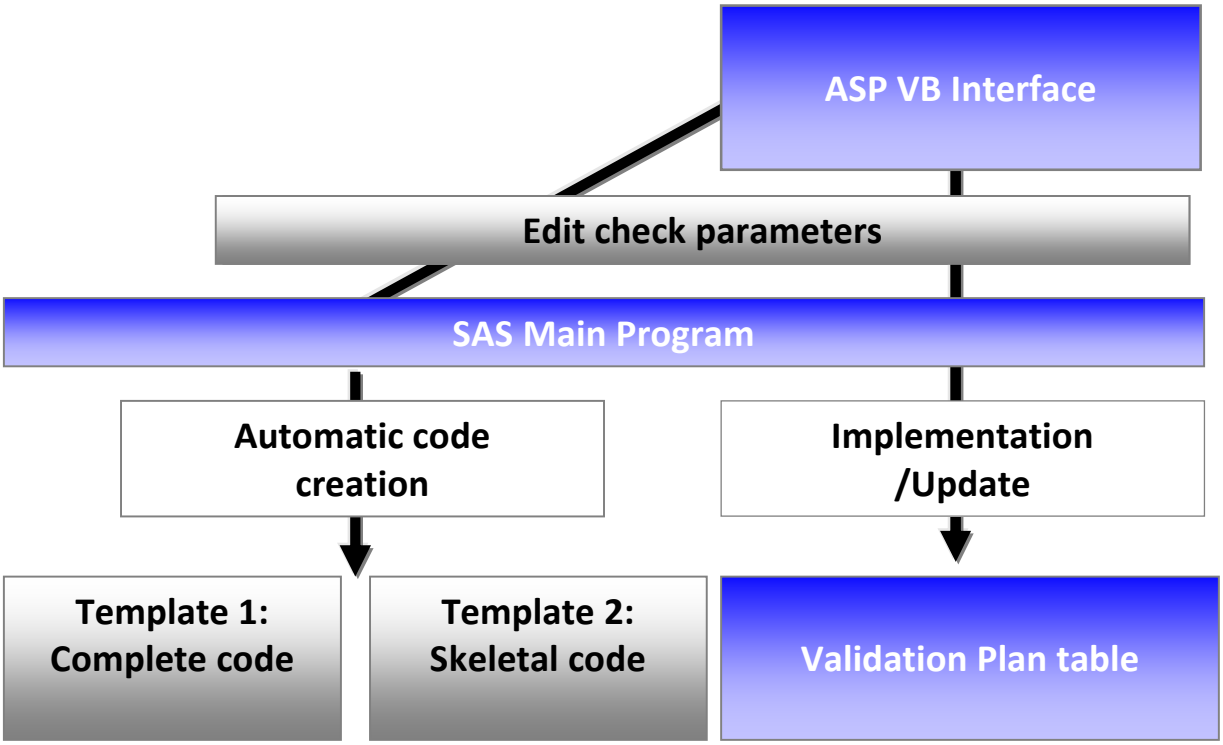

Figure 1: solution overview

CHECK	PANEL	DESCRIPTION	MESSAGE	CRITERIA	R	O
MC0X	PANEL1	At visit X, if ITEM02 is YES then ITEM01 must be present.	At visit <VISIT> ITEM02 is completed but ITEM01 is missing. Please complete.	At visit X ITEM02=1 AND ITEM01=null.	INV	FALSE
MM0X	PANEL1	At visit X, ITEM03 must be present.	At visit <VISIT> DATE1 is <missing/incomplete>. Please complete.	At visit X DAY1=null OR MONTH1=null OR YEAR1=null	INV	FALSE
CC0X	PANEL1	If medical history is treated then at least one concomitant treatment must be entered.	Medical history n°<NB> is treated but no concomitant treatment has been entered.	TREATED=1 AND CONCTRT=null	DM	FALSE

Figure 2: short contents of the Oracle table. O: Obsolete. R: Recipient

CREATING A NEW EDIT CHECK

The user can describe on-line checks (edit checks that are implemented within the EDC application) and off-line checks (SAS edit checks applied to the study database).

For each new check, the user enters the check description

- Unique identification for the check
- Description of expected consistency of the data
- Criteria for retrieving inconsistency
- Message to be sent in case of inconsistency
- Recipient of the potential query (in Figure 2, INV stands for Investigator and DM for data manager)

and for off-line edit checks, programming parameters that will be used for automatic SAS program creation:

- Type of control (MM, MC, CC)
- Concerned panel
- Item(s) name(s) for MM/MC checks entered as: *ITEM1, ITEM1* ***OR*** *ITEM2, ITEM1* ***AND*** *ITEM2* (...)
- Optional condition for MC checks entered as*: ITEM3 > 5 AND ITEM3<10*

After completion the user executes the SAS program which creates a new record in the validation plan table for each check and generates an SAS program for each off-line check.

At the end of the specification stage, the SAS programming tasks are more or less complete or well advanced and match the exhaustive validation plan.

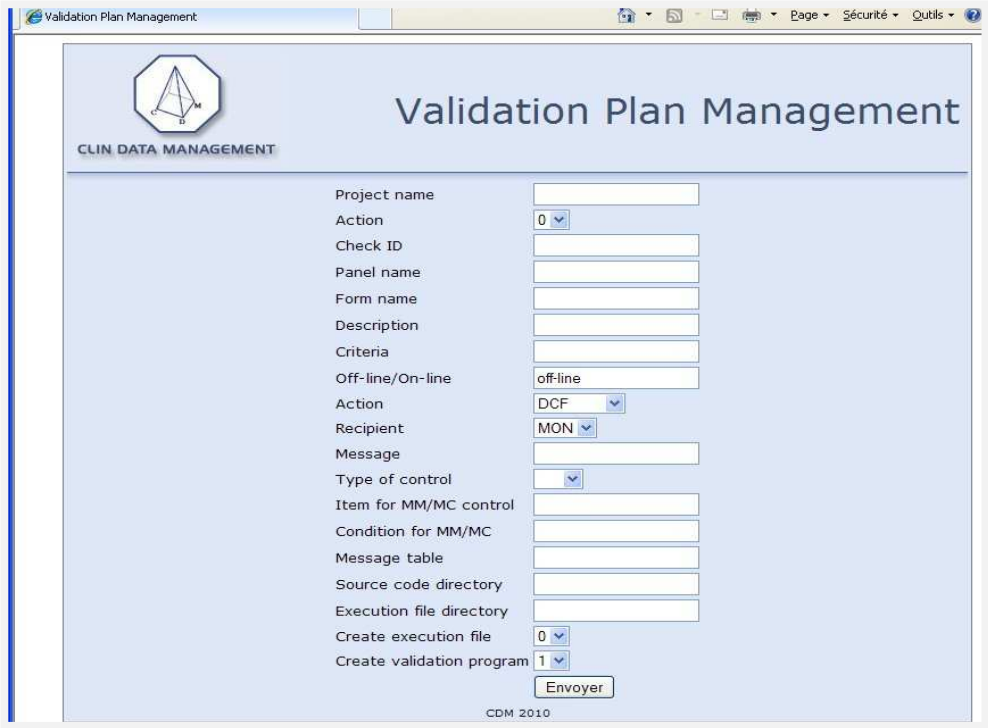

Figure 3: Application screenshot

UPDATING THE VALIDATION PLAN

For managing changes to the validation plan the user identifies the check to be changed, and modifies the attributes (message or recipient) so that the validation plan table is updated accordingly and no change needs to be made on the existing SAS program

DEACTIVATING AN EDIT CHECK

The user has also the possibility to deactivate an edit check during the trial. The check is not deleted, but an obsolete status is then associated to the identified check in the Oracle table, indicating that the check won't be executed anymore.

THE SAS SIDE

SAS PROGRAM FOR CREATING CODE

All functionalities are implemented in a single and rather basic SAS program. The program loads parameters entered through the web-form and manages the validation plan and automatic code creation with appropriate and classical *if/then* statements, SQL procedures and *put* instructions. The dummy code hereafter is presented for reflecting the ease of programming:

```
...
/*STEP 1 : Parameters Import and initialisation of corresponding macro variables*/
...     %put _TYPE;                    /*Functionality*/
        %put _TYPECC;                  /*Type of check*/
        %put _WHERE;                   /*Condition for MM check*/
        %put _CHECK_ID;                /*Check identification*/
        %put _PANEL;                   /*Panel Name*/
        %put _ITEM;                    /*Item(s)*/
        %put _DESCRIPTION;             /*Check description*/
        %put _CRITERIA;                /* Check criteria*/
        %put _MESSAGE;                 /*Check message*/
        %put _RECIPIENT;               /*Recipient for the message*/
        %put _FOLDER;                  /*Directory for SAS programs created*/
/*FUNCTIONALITY 1: creating a new edit check*/
%if &_TYPE.=0 %then %do;
       proc sql ...                        /*    Creating the record in validation plan table
      /*In case of off-line control, generate corresponding SAS code */
        %if (&_OFFLINE.=1) %then %do;
               filename mycode "&_FOLDER.\&_MACRO..sas";
               data _null_;
               file mycode;
               put
               '..................................................... ' /
               …
            ;
               /*Complete code for MC/MM*/
               if "&_TYPECC."^="CC" then
               put
               '.....................................................'/ ;
              /* Skeletal code for CC*/
               else put
               '.....................................................' /
           ...;
        %end;
%end;
/*FUNCTIONALITY 2: Update of the edit check*/
%if &_TYPE.=1 %then %do;
        proc sql ...
%end;
/*FUNCTIONALITY 3: Deactivating edit check*/
%if &_TYPE.=2 %then %do;
        proc sql ...
%end;
```

CASE 1 – READY TO USE CODE FOR BASIC CHECKS

Figure 4 presents the template for the code generated in case of controls of types MM/MC. For better understanding, macro variable names initialised in the main program have been indicated with the following syntax *<&_name>*; of course in programs generated, the macro variables are replaced by their values so that the code generated is ready to use.

Firstly, the company's standard SAS program header is written as well as comments in the macro body and structure of the program. This results in both the standardisation of programming and advantageous time-saving.

The macro %MISSING_ITEM which is called first for targeting inconsistencies, offers the following functionalities:

- items of several types can be managed: character, numerical as well as date/datetime; for the latter case the standardisation of the date items names is assumed in the study database
- several items of the same type can be managed simultaneously, depending on the item string given in the argument: *ITEM1=. **OR** ITEM2=. , ITEM1=. **AND** ITEM2=. (...)*
- the macro can manage conditions on other items: the missing record for the entered item(s) are targeted depending on the value of one or several other item values. The condition is entered simply using classical SAS 'where' statement, allowing high flexibility of use

The next macro, %FORMAT_TEXT aims at loading the appropriate message from the validation plan table and offers the possibility to manage dynamic items, as illustrated by the following example:

VISIT	VALIDATION PLAN MESSAGE	FINAL MESSAGE
INCL	At visit <VISIT> Weight is missing. Please complete.	At visit INCLUSION Weight is missing. Please complete.

The last macro, %LOAD_DCFS, is called for loading queries in the queries management table (not presented in this paper).

```
/*****************************************************************
*   CLIN DATA MANAGEMENT
*   PROGRAM NAME      : <&_MACRO>
*   STUDY             : <&_PROJECT>
*   PROJECT           : QUERIES MANAGEMENT
* ---------------------------------------------------------------
*   AUTHOR            : &sysuserid
*   CREATE DATE       : &fdate
*   VERSION           : 1.0
* ---------------------------------------------------------------
*   DESCRIPTION       : Consistency check <&_MACRO>
* ---------------------------------------------------------------
*   PARAMETERS        :
*       LIB      =    Study data library
*       DELWORK =     Delete WORK contents
* ---------------------------------------------------------------
*   MODIFICATIONS :
*   [xx.xx.xxx] V x.x :
*****************************************************************

%MACRO &_MACRO(lib=&_gLIB,delwork=1);

/*1. Find inconsistency*/
    %MISSING_ITEM(input=&lib..<&_PANEL>,
                  item=<&_ITEM>,
                  output=%str(TMP(where=(<&_WHERE>))));

/*2. Load Message from Validation plan table*/
    %FORMAT_TEXT(input=TMP,check=<&_MACRO>);

/*3. Inconsistencies loading in global inconsistencies table*/
    %LOAD_DCFS(table=TMP,check=<&_MACRO>);

/*Delete work contents*/
%if (&delwork=1) %then %do;
    &_gdel_work;
%end;

%MEND;
```

Figure 4: SAS code template generated for MM/MC edit checks

CASE 2 – SKELETAL CODE FOR MORE COMPLEX CHECKS

For more complex checks another template code is generated as shown in Figure 5. The difference with the previous template remains only in the first part of the program, where the %MISSING_ITEM macro is replaced by a "TO BE COMPLETED" comment, indicating that the program needs to be completed by an SAS programmer. Again this approach ensures standardized and efficient programming.

```
%MACRO &_MACRO(lib=&_gLIB,delwork=1);

/*1. Find inconsistency*/
    /*TO BE COMPLETED - TO BE COMPLETED - TO BE COMPLETED - TO BE COMPLETED */

/*2. Load Message from Validation plan table*/
    %FORMAT_TEXT(input=TMP,check=<&_MACRO>);

/*3. Inconsistencies loading in global inconsistencies table*/
    %LOAD_DCFS(table=TMP,check=<&_MACRO>);

/*Delete work contents*/
%if (&delwork=1) %then %do;
    &_gdel_work;
%end;

%MEND;
```

Figure 5: Template macro for CC checks

CONCLUSION

The combination of the ASP VB interface and the SAS program for automatic code generation offers an excellent solution for fast and standardized programming of edit checks, even for non SAS specialists.
The workflow and SAS programs implemented are nevertheless quite straightforward and the solution relies mainly on the automatic code template design and associated SAS macros.
Similar solutions could be easily developed for other data management processes, particularly in a context where pressure for reducing timelines is constant.

ACKNOWLEDGMENTS

Thanks to our colleagues Vincent Bippus and Pierre Diether for their contribution to the development of this solution.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the authors at:

Authors Names : Virginie Freytag and Michel Toussaint
Company : Clin Data Management
Address : Zone artisanale, Rue d'Alsace
City / Postcode: 68 250
Work Phone: +33 (0)3 89 49 56 60
Fax:: +33 (0)3 89 78 51 81
Email: v.freytag@cdm-trials.com and m.toussaint@cdm-trials.com
Web: www.cdm-trials.com