PhUSE 2016

Paper CS01

Boosting the efficiency of data review by using SAS Integration Technologies and VBA

Giuseppe Di Monaco, UCB BioSciences GmbH, Monheim, Germany

ABSTRACT

Identifying and repairing software errors can be a costly business. In addition, aside from being an important economic factor, errors can contribute to stress in the workplace, especially when users are required to perform their tasks within short timeframes. This paper presents a software module called Prerequisite which uses the SAS ® IOM (Integrated Object Model). Written in VBA (Visual Basic for Applications) with embedded SAS, Prerequisite performs fault blocking actions such as the detection, isolation, identification, and subsequent resolution of errors caused by unexpected inputs such as wrong variable names. Hence, run-time errors that would otherwise arise from problematic inputs can be reduced or eliminated, and users can focus on amending the inputs rather than having to review lengthy log files to identify issues. A module of this nature can be particularly useful when working with flexible standard structures such as ADaM datasets.

INTRODUCTION

PRIMARY AND SECONDARY OBJECTIVE OF THE PAPER

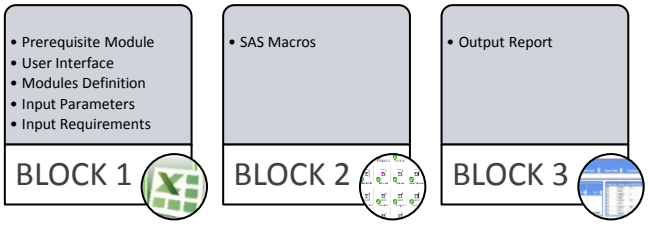
The primary objective of this paper is to show how a “minimum” combination of VBA, SAS Base and SAS IOM can be used to create a module (Prerequisite) that can identify runtime errors. The secondary objective is to show an improvement of the application achieved by implementing a GUI (Graphical User Interface) to assist the user with isolating and resolving errors. Due to space constraints, I will provide guideline on how to create the code of Prerequisite module in the appendix and will leave it to the reader to re-create the code for the GUI.

THE CHALLENGE OF UPDATING A CONFIGURATION FILE

Microsoft Excel is widely used in the pharmaceutical industry, especially for exchanging data into and out of SAS. Another common use for programmers is to use Excel as a configuration file for SAS macros. Having an external configuration file can be an extremely useful method of configuring the initial settings and of passing parameters to SAS macros since this avoids having to make constant changes to SAS programs. Parameters need to be amended frequently when working with highly flexible input data structures. This is especially true when there is a need to make updates in to the configuration file due to runtime errors caused by unexpected inputs. Usually these changes are made manually and the risk of making mistakes is high. After such manual changes, programmers have to re-run their programs and subsequently review lengthy log files to identify issues or to verify that the changes have had the desired effects. This is an error-prone and very time-consuming process.

Prerequisite is part of an application I have created called Data Review Tool. This application ensures that the data is as clean and consistent as possible. The application comprises three main blocks (see **Image 1**). The first block is a spreadsheet (dr_tool.xls) which includes the Prerequisite module, a GUI, a definition of the standard SAS macros with their attendant parameters, and finally the requirements of the input parameters to be used by the Prerequisite module. The second block contains a set of standard macros and the third is the output report created by the application.

Image 1



Prerequisite is used by each SAS macro (see block 2 of **Image 1**) and can check the correctness of the input parameters stored in block 1. The intent is to make users aware that an error would occur so corrective action can be taken before the Data Review tool itself is run. Before delving deeper into the objectives of the paper, let us have a look at the configuration file and SAS errors that might occur.

FIRST APPROACH (WITHOUT PREREQUISITE)

A MACRO SHOULD NOT RUN AT ALL

A solution which is very popular amongst SAS programmers is to write SAS code aimed at stopping macro execution if the required inputs do not exist and instead outputting a suppression message to the log. This was also the very first solution adopted to address this problem in the Data Review Tool. Before running any macros, the tool used a SAS driver program to perform these checks on the input parameters. This solution works here and the above error is avoided since the driver program detects this scenario and outputs a message like the following to the log:

```
ERROR: ***********
ERROR: The TARGET variable (ARMCD) is not in the AD._ALL_ dataset(s) you requested
ERROR: Please check the spelling of the variable and/or dataset name
ERROR: ***********
```

This solution is only acceptable when the number of such errors is low, the programmer who is running the macros knows the design of the tool, is able to interpret the macro errors and hence make the necessary changes. Imagine a scenario, however, where there is limited time to run the application over dozens of macros each with multiple parameters. The resulting logs are long and hard to read. Consequently changes must be made to the configuration file, the application re-run, and the logs reviewed again until they are clean. Only then is final review of the output reports possible. This was the scenario that convinced me to find a more efficient solution.

SECOND APPROACH (WITH PREREQUISITE)

PREREQUISITE

The Prerequisite module was developed using Visual Basic for Applications (VBA) and the SAS IOM to run SAS and retrieve information from SAS. (Guideline on how to create the code for this module is provided in the appendix to this paper.) Before going into the details of this module, we shall briefly consider SAS IOM Technologies.

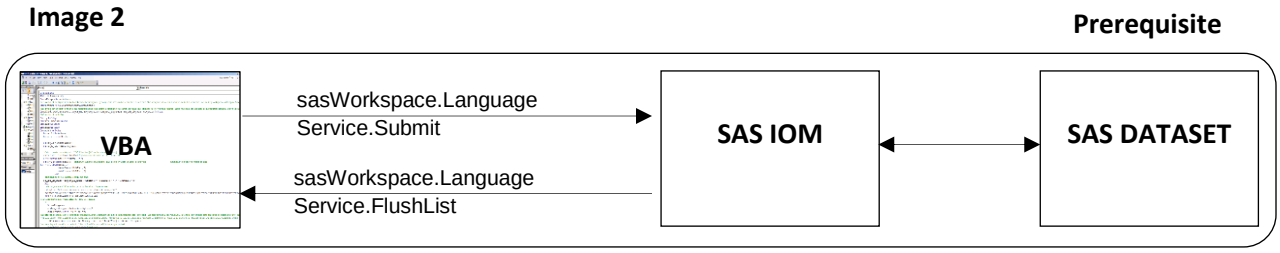
INTEGRATED OBJECT MODEL

SAS IOM provides interfaces to facilitate client/server communication between multiple vendor architectures. Built into SAS, these interfaces are available to client applications in which SAS is executed as an object server. IOM allows the use of industry-standard languages and communication protocols to develop client programs that access services on IOM servers. Four types of IOM servers are available: SAS Workspace Server, SAS Stored Process Server, SAS OLAP Server and SAS Metadata Server. In this project I have only used the SAS Workspace Server. This Object server was first released in SAS 8.2. It represents a session with the SAS System and is functionally equivalent to the execution of base SAS software in batch mode. It can be seen as an on-demand SAS execution environment that is instantiated upon client request. The interface provides access to Foundation SAS software features such as the SAS language, SAS libraries, the server file system, results content, and formatting services.

Two components are available for creating objects on IOM servers. The first is called SAS Workspace Manager and the second is the SAS Object Manager. I used the SAS Workspace Manager for creating and managing the SAS Workspace Server. The SAS Workspace Manager executes on the client side and enables, for example, a client application to connect to a server simply by using a logical name. I did not need to use any object for the connection since the client application (VBA module of Excel) resides on the same cluster of servers as the SAS system.

To submit SAS code, I used an IOM component called LanguageService. This component provides methods to retrieve log and list output as **Image 2** shows

Image 2



PROBLEM

CONFIGURATION FILE

This file contains the information to be passed to the SAS macros. Amongst this information are the macro definitions, their parameters and the locations of datasets to assign SAS libraries. These macros are designed as general-purpose and they can be used to check consistency of data in different scenarios by just modifying input parameters via the configuration file.

There is no restriction on the number of SAS macros and related parameters to be checked by the Prerequisite module. For brevity in this paper I will show a call to one macro only (TRACESINGLEVARIABLE). This is designed to trace a value of a variable (or a set of variables) across domains. In this case the aim is to trace the value of ARMCD from SDTM to ADaM.
By design, four parameters are required when calling this macro, TARGETDS, TARGETV, SECDSET and SECVAR. The parameter SELECT should be considered as conditional meaning that it must be equal to "X" if and only if all required variables exist. All other parameters such as CONDITION and ODISPVAR are optional and in this specific case of TRACESINGLEVARIABLE nothing needs to be checked since they have been left blank. (See **Table 1**).

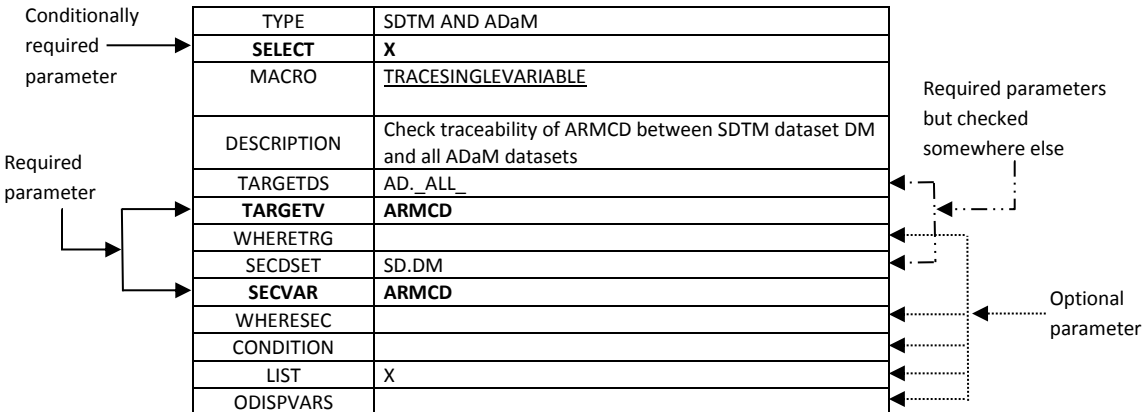
Note that TARGETDS and SECDSET were checked earlier in the process by another module. In fact, it would not make sense to check existence of a variable if the dataset that contains the variable does not exist. Assume that TARGETDS and SECDSET have already passed the presence test. Hence only TARGETV and SECVAR need to be checked for existence and only if they exist will the conditional parameter SELECT have the value "X".

Focus on SELECT, TARGETV and SECVAR parameters.

- **SELECT**: Select with an "**X**" a macro to run in case SD.DM.ARMCD and AD._ALL_.ARMCD exist.
- **TARGETV**: In this case **ARMCD**, is the target variable that must exist within at least one dataset in the library AD specified in TARGETDS field.
- **SECVAR**: In this case **ARMCD**, is the target variable that must exist in the DM dataset in the library SD specified in SECDSET field.

Table 1

TYPE	SDTM AND ADaM
SELECT	**X**
MACRO	TRACESINGLEVARIABLE
DESCRIPTION	Check traceability of ARMCD between SDTM dataset DM and all ADaM datasets
TARGETDS	AD._ALL_
TARGETV	**ARMCD**
WHERETRG	
SECDSET	SD.DM
SECVAR	**ARMCD**
WHERESEC	
CONDITION	
LIST	X
ODISPVARS	
GRADE	Recommended
.....	
PARAMETER n	



The content of the parameter TARGETV is ARMCD and this variable must exist in one of the datasets defined in the TARGETDS field. AD._ALL_ stands for all datasets present in the AD library. Hence this macro expects that at least one ADaM dataset contains an ARMCD variable in the library AD.
Similarly, the content of the parameter SECVAR is ARMCD and this variable must exist within the dataset SD.DM defined in the SECDSET field.

If these parameters do not exist when the macro TRACESINGLEVARIABLE runs, SAS will return errors. Here is an example of a SAS error that would occur if a variable specified within the "TARGETV" column was not present in any of the datasets specified in TARGETDS column:

```
ERROR: The following columns were not found in the contributing tables: ARMCD
```

(**Note**: Data Review Tool can customize any SAS macro and parameters for a study or indication, TRACESINGLEVARIABLE macro and related ARMCD parameters have been chosen just as an example.)

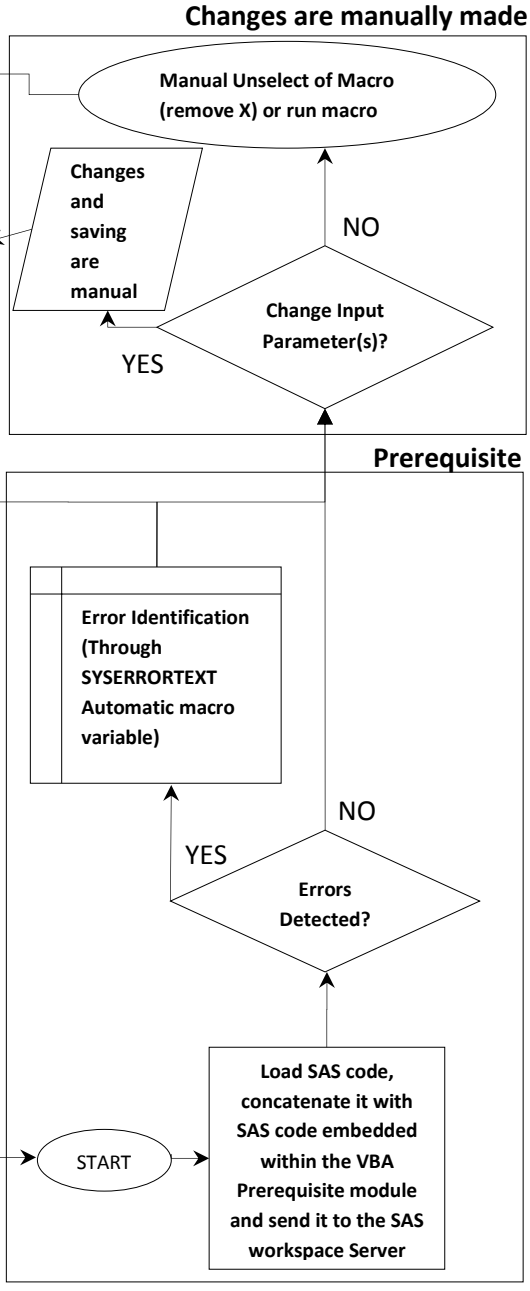
PROCESS FLOW FOR PREREQUISITE

As for the configuration file, for brevity, the process flow will be shown for one SAS macro only. There are two extra fields in **Table 2**, PREREQUISITE RESULT and SAS REQUIREMENTS. The first will be either blank or will hold the SAS error returned by the PREREQUISITE module. SAS REQUIREMENTS contains SAS code designed to generate an error if variables in TARGETV and SECVAR do not exist. In other words, Prerequisite module loads the SAS code contained in SAS REQUIREMENTS, concatenates it with some other SAS code embedded within the VBA Prerequisite module (see appendix for more details) and submits it to the SAS Workspace Server. If the submitted code does not produce any error, then the PREREQUISITE RESULT field is left blank and this macro is considered ready to run. On the contrary, if submitted code produces an error, then this error is stored in PREREQUISITE RESULT. This error is retrieved from SAS using the component LanguageService. Instead of retrieving the error directly from SAS log, Prerequisite retrieves this error from a SAS list, a trick used to filter out all messages that SAS writes in to the log. (E.g.: SAS messages, etc…). Guideline for the code for LanguageService is given in the appendix to this paper. Here we see that the value of the automatic macro variable 'syserrortext' is printed to a SAS list and then redirected to Excel using sasWorkspace.LanguageService.FlushList.

Table 2 **Configuration file**

TYPE	SDTM AND ADaM
SELECT	**X**
MACRO	TRACESINGLEVARIABLE
DESCRIPTION	Check traceability of ARMCD between SDTM dataset DM and all ADaM datasets
TARGETDS	AD._ALL_
TARGETV	**ARMCD**
WHERETRG	
SECDSET	SD.DM
SECVAR	**ARMCD**
WHERESEC	
CONDITION	
LIST	X
ODISPVARS	
GRADE	Recommended
PREREQUISITE RESULT	The variable ARMCD in the DROP KEEP or RENAME list has never been referenced.

SAS REQUIREMENTS
 	
(code to force a SAS error in case TARGETV and/or SECVAR do not exist)
	proc sort data = sashelp.vcolumn (keep = libname memname name) out = errdetect;
	where libname = "AD" and name in ("**ARMCD**");
	by libname memname name;
	run;
	
	data errdetect1;
	set errdetect;
	by libname memname name;
	run;
	
	proc sort data = errdetect1 out = errdetect1;
	by memname name;
	run;
	
	proc transpose data=errdetect1 out=errdetect1 (drop = _NAME_ _label_) let;
	by memname ;
	var name;
	id name ;
	run;
	
	data errdetect2;
	set errdetect1(keep = **ARMCD**);
	run;
	
	data errdetect3;
	set SD.DM(keep = **ARMCD**);
	run;



FINAL APPROACH (WITH PREREQUISITE AND GUI)

PROCESS FLOW FOR GUI AND PREREQUISITE

When the run of Prerequisite ends, launches a GUI that shows the contents of the configuration file together with the results from Prerequisite. This GUI was created to reduce the steep learning curve needed to make changes to configuration file. New users can now learn this GUI easily since it is visually intuitive. Another great benefit of using this GUI is that it reduces the potential for making mistakes. When the GUI appears, the user has two choices: either fix the error or leave the macro unselected. Once this step has been completed, a driver program can run all modules in a batch and we can be reasonably sure that the trapped errors will not occur.

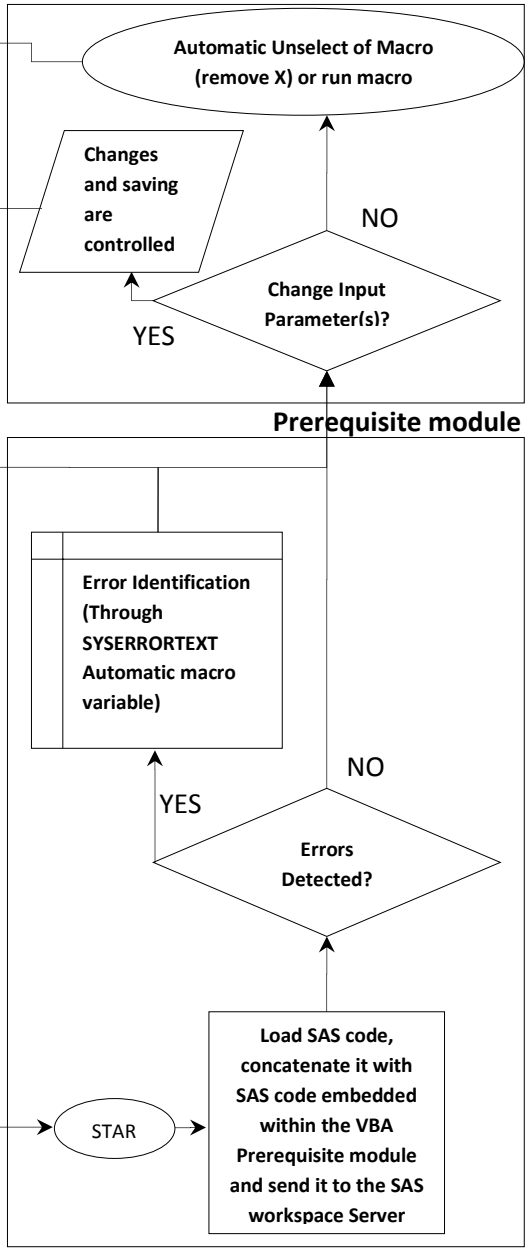
Table 3

Configuration file

TYPE	SDTM AND ADaM
SELECT	**X**
MACRO	TRACESINGLEVARIABLE
DESCRIPTION	Check traceability of ARMCD between SDTM dataset DM and all ADaM datasets
TARGETDS	AD._ALL_
TARGETV	**ARMCD**
WHERETRG	
SECDSET	SD.DM
SECVAR	**ARMCD**
WHERESEC	
CONDITION	
LIST	X
ODISPVARS	
GRADE	Recommended
PREREQUISITE RESULT	The variable ARMCD in the DROP KEEP or RENAME list has never been referenced.

SAS REQUIREMENTS
 	
(code to force a SAS error in case TARGETV and/or SECVAR do not exist)
	proc sort data = sashelp.vcolumn (keep = libname memname name) out = errdetect;
	where libname = "AD" and name in ("**ARMCD**");
	by libname memname name;
	run;
	
	data errdetect1;
	set errdetect;
	by libname memname name;
	run;
	
	proc sort data = errdetect1 out = errdetect1;
	by memname name;
	run;
	
	proc transpose data=errdetect1 out=errdetect1 (drop = _NAME_ _label_) let;
	by memname ;
	var name;
	id name ;
	run;
	
	data errdetect2;
	set errdetect1(keep = **ARMCD**);
	run;
	
	data errdetect3;
	set SD.DM(keep = **ARMCD**);
	run;

GUI



FUNCTIONAL REQUIREMENTS OF THE GUI

The interface was designed and developed mainly to increase the usability of the Data Review Tool (see numbers on **Image 3**). It can perform the following actions:

1. **Identification** of the error through the column PREREQUISITE
2. **Isolation** of the error by automatic deselecting of the column SELECT (removal of "X")
3. **Controlling** enterable fields to avoid human error. In particular, colors have the following meaning:
   - 3.1. Dark gray is for non-missing and non-enterable fields. These are fixed-value fields used by the macro.
   - 3.2. Light Gray is for missing and non-enterable fields. These are fields that are not used by the macro.
   - 3.3. White is for enterable fields. All white fields can be changed.
   - 3.4. Macro Label contains a direct link to the SharePoint web page (see **Image 3**) where information related to that particular macro is stored. The link takes the user to the relevant location where information about that macro is stored. This web page stores all test cases previously performed with expected input and output. This method facilitates the error resolution.
4. **Update** macro parameters.

   This interface can change the configuration and return the result to the spreadsheet in a way that is completely transparent to the user as described in the process flow Process Flow of the GUI and Prerequisite (see **Image 4 and Table 4**). The user has two possibilities: either leave the configuration as it is in **Table 3** or make changes. For instance, assume that the column ARMCD has been added to the ADSL dataset only. The user can type "AD.ADSL" in the TARGET field, then re-select this macro by putting an "X" into the SELECT column and finally SAS REQUIREMENTS may need (in some exceptional cases) to be adjusted so that TARGETDS obtains the value of AD.ADSL instead of AD._ALL_. The color background of the changed field changes to yellow to visually notify the user that she/he is about to change the original values of those parameters. When the user hits save, the changes are stored in the configuration file. (**Image 4** and **Table 4**)

Image 3

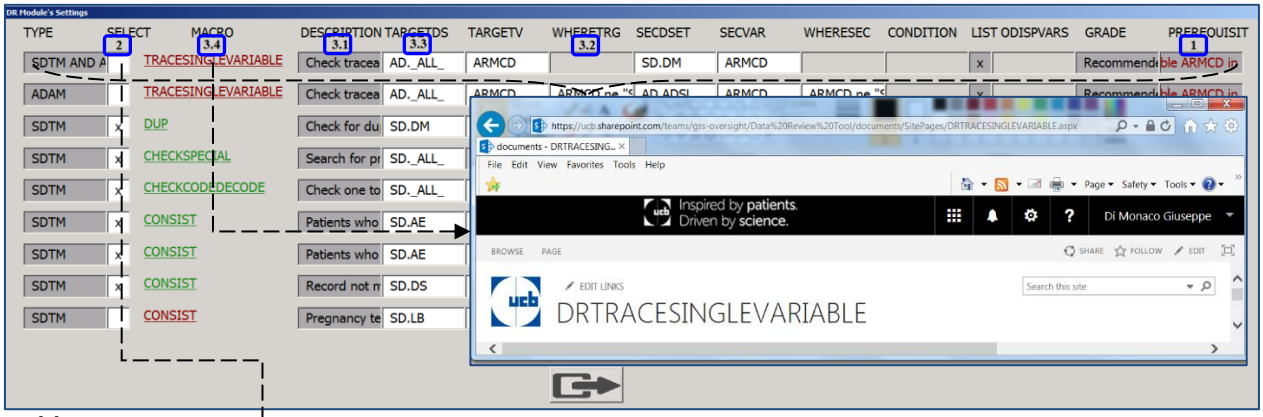


Table 3

TYPE	SDTM AND ADaM
SELECT	
MACRO	TRACESINGLEVARIABLE
DESCRIPTION	Check traceability of ARMCD between SDTM dataset DM and all ADaM datasets
TARGETDS	AD.ADSL
TARGETV	**ARMCD**
WHERETRG	
SECDSET	SD.DM
SECVAR	**ARMCD**
WHERESEC	
CONDITION	
LIST	X
ODISPVARS	
GRADE	Recommended
PREREQUISITE RESULT	The variable ARMCD in the DROP KEEP or RENAME list has never been referenced.

Image 4

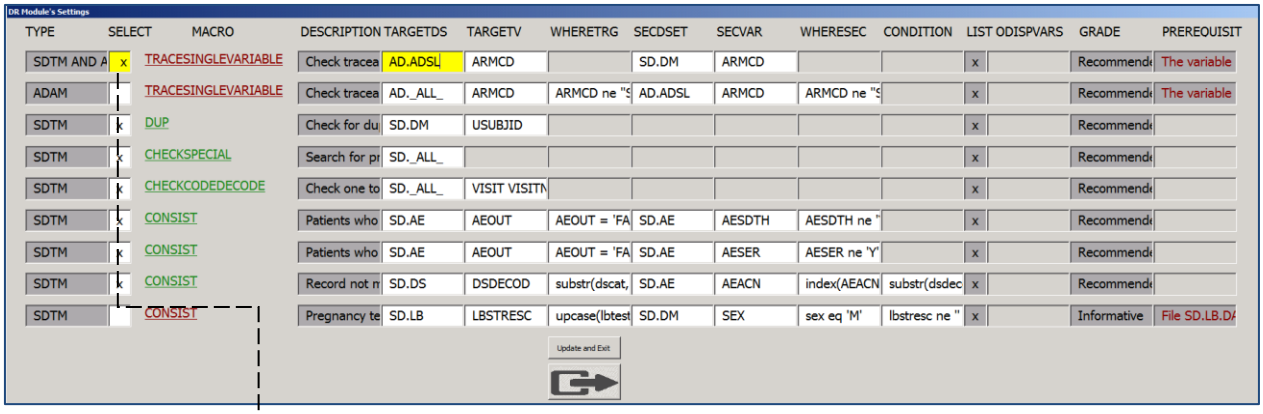


Table 4

TYPE	SDTM AND ADaM
SELECT	**X**
MACRO	TRACESINGLEVARIABLE
DESCRIPTION	Check traceability of ARMCD between SDTM dataset DM and all ADaM datasets
TARGETDS	AD.ADSL
TARGETV	**ARMCD**
WHERETRG	
SECDSET	SD.DM
SECVAR	**ARMCD**
WHERESEC	
CONDITION	
LIST	X
ODISPVARS	
GRADE	Recommended
PREREQUISITE RESULT	The variable ARMCD in the DROP KEEP or RENAME list has never been referenced.

NOTE: Once the input parameter has been fixed, on the text run the error in the PREREQUISITE RESULT will not be present. (Provided the updated parameters really do exist.)

IMPROVEMENT & INTEGRATION WITH OTHER TOOLS

FROM VBA TO ASP.NET

The entire project could be migrated from VBA to ASP.NET (Active Server Pages .NET). SAS System features can be utilized from the ASP.NET framework. This framework would allow full access to SAS System facilities from an ASP.Net web page with the advantage of using a modern, web-based technology. ASP.NET is server-side technology so its code executes on the server before sending the results to the browser. Also, ASP.NET drastically reduces the amount of code required to create applications. In addition, when using ASP.NET, the application could be expanded to include new functionalities such as populating and/or checking of metadata information from a repository.

CONCLUSION

Recall the original problem that I wanted to solve. As stated in the introduction, having an external configuration file can be extremely useful to keep source code untouched and yet allow configuration of the initial settings of programs, in this case the SAS macro parameters. However, when running these SAS macros over ADaM datasets I got frequent errors due to the highly flexible ADaM data structure. So I wanted to avoid such scenarios. Specifically, I wanted to avoid having to review logs for errors caused by input parameters since it was labor-intensive to review the logs, resolve the errors, and then wait for the complete re-run of the application only to start again at the beginning. In addition, in case of errors, I also wanted to avoid having to update the configuration file manually. Apart from being an error-prone process, it also required a deep knowledge of the structure of the configuration file. Hence reviewers needed to be trained.

The basic idea to solve this problem was to electronically check and update SAS macros parameters before running the entire application so that in case of error, the application would assist the user by not only blocking the execution of macros certain to fail, but also by facilitating the resolution of errors and the updating of the configuration file in a controlled manner.

Thus my solution was the combination of the configuration file, the IOM and the GUI application. The final result is of great benefits to those charged with reviewing data.

This method is not just limited to checking input parameters to SAS macros. It could equally be applied each time there is need to check expected programs inputs.

I hope that the examples provided in this paper will serve as an inspiration for you to explore the technologies described and how they can improve daily tasks.

REFERENCES

Developing Client Applications
http://support.sas.com/rnd/itech/doc/dist-obj/index.html

A Data Science tool for Data Reviewing
http://www.lexjansen.com/phuse/2015/pp/PP01.pdf

LanguageService Object
http://support.sas.com/rnd/itech/doc/dist-obj/comdoc/ilngca.html

ACKNOWLEDGMENTS

The author would like to thank all colleagues who provided support for the testing phase of the Prerequisite module and the user interface. A Special thank goes to Colin de Klerk for his help during the preparation of this paper.

CONTACT INFORMATION

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

Giuseppe Di Monaco
UCB BIOSCIENCES GmbH
Alfred-Nobel-Straße 10
40789 Monheim, Germany
Tel: +49.2173.48.2091
Fax: +49.2173.48.1947
Email: Giuseppe.DiMonaco@ucb.com
Web: www.ucb.com

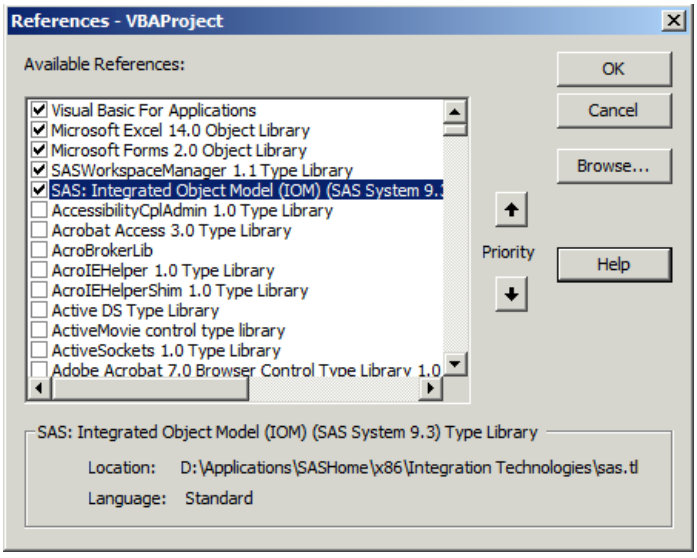
APPENDIX

REQUIREMENTS FOR PREREQUISITE

ENABLING VBA

When working with Excel, the first problem may be enabling VBA since VBA is disabled by default. To enable VBA, go to “File”, “Options” and select “Customize the ribbon”. Then tick the developer box. Also allow Excel to run programs included with spreadsheets. This is done by going to “File”, “Options”, then selecting “Trust Center” and then “TrustCenter Settings”. Select “Macro Settings” and finally select “Enable all macros”. The Developer tab should be now available in the ribbon. In Excel 2013 access the VBA editor by clicking on the VBA icon under the Developer ribbon. To insert a module, go to “Insert”, ”Macro”, “Module”. A default name (Module1, Module2, etc...) is given to the new module but it can be renamed. Copy the Prerequisite module into the editor of Module1 and save the spreadsheet.

VBA & SAS IOM (References)



In order to access IOM servers from a Visual Basic project, references are needed to at least two type libraries. Reference them from the Tools (VBA) menu, find the references menu item and search for the type library labelled "SAS: Integrated Object Model (IOM)" and click the check box to add the reference to your project.

Similarly, find the type library labelled "SASWorkspaceManager" and click its check box to add this reference to the project. Click OK to activate these changes. After referencing these libraries, you can use them immediately and the definitions are available to the Visual Basic object browser. (If you do not see these as options in your reference list, then the integrated technologies packages are not correctly installed.)

NOTE: It is possible to use another object instead of the Workspace Manager. This is the Object Manager interface which gives more advantages such as new features available with Version 9 Integration Technologies.

GUIDELINE ON HOW TO CREATE VBA PREREQUISITE MODULE

1. Declare an object variable to reference a Workspace. By doing so you create a variable to reference the workspace but you still need to create the workspace. For creating workspaces, you can use Workspace Manager.
2. Create a variable to hold SAS code options.
3. To assign SAS libraries, extract paths and libref information from the spreadsheet and store it in to a variable.
4. Create a variable to hold SAS code stored in SAS REQUIREMENTS field (see **Table 3** above).
5. Create a variable containing code for printing the value of 'syserrortext' content in to SAS list
6. For each definition of macro stored in the spreadsheet do the following steps:
    a. Create a workspace variable containing the object variable declared in point 1
    b. Concatenate variables containing:
        i. SAS options
        ii. SAS libraries
        iii. SAS code stored in SAS REQUIREMENTS
        iv. SAS code for printing 'syserrortext'
    c. Submit the concatenated variables using sasWorkspace.LanguageService.Submit.
    d. Retrieve the error from list by using sasWorkspace.LanguageService.FlushList and store it in to PREREQUISITE RESULT field of the excel spreadsheet.
    e. Reset all variables to null.
    F. Close the workspace.