PhUSE 2016

Paper IS05

Good Programming Practice and Its Importance in the New World of Metadata

Jennifer Chin, Eisai Ltd, Hatfield, UK
Shafi Chowdhury, Shafi Consultancy, Uxbridge, UK
Mark Foxwell, PRA Health Sciences, Reading, UK

ABSTRACT

In clinical trials, we need to process and analyze voluminous quantity of information and its metadata appropriately. As we move into an electronic environment, and couple with the ever increasing complexity of metadata, we need to develop an effective and efficient strategy to utilise and to disseminate information by auto generating analysis programs. The application of Good Programming Practice (GPP) in such environments, which use combination of complex macros, metadata and auto generating macros is essential to efficiency of work, ease of maintenance of programs over time, portability and, readability of programs for utility within the organisation and regulatory submission.

This presentation emphasizes the need for GPP for harmonious working of macros or automated programs, metadata and auto-generation programs in combination.

INTRODUCTION

As we step into an electronic environment, information sources seem to be growing, and data information spreads much faster with each passing day. Proper information management is a necessity to ensure that important research is not lost or duplicated unnecessarily. Applying the metadata concept to your information will allow you, and those you share information with, to search across multiple files for the exact piece of information you need easily.

Metadata and metadata-driven utilities are effective tools that reduce or entirely eliminate many of the programming and management problems inherent in the conventional project work flows.

Metadata programming is simply a way of writing algorithms that leverage metadata to control how automated program behaves. Automated programs can automatically update on data or options change, allowing you to create templates for repetitive tasks or to perform batch operations from the user interface. To automate tasks, we need to think technical, best practices to apply programmatically of ways to accomplish proficiency, to maximize and to increase efficiency.

THE METADATA WORLD

Metadata is defined as “structured data about data”. These are data which are used to analyze trial data. Metadata summarizes basic information about data, which can make finding and working with particular instances of data easier. For example, author, date created and date modified, file name and file size are examples of very basic document metadata. Having the ability to filter through that metadata makes it much easier for someone to locate a specific document.

There are three main types of metadata:

- Descriptive metadata describes a resource for purposes such as discovery and identification. It can include element such as title, abstract, author, and keywords
- Structural metadata indicates how compound objects are put together, for example, how pages are ordered to form chapters.
- Administrative metadata provides information to help manage a resource, such as when and how it was created, version number, file type and other technical information, and who can access it.

The structure of the metadata reference in this paper is administrative metadata which consists of metadata of parameters such as program names, output names, standard templates, study end points, populations and other key relevant parameters. By applying an auto-generated SAS program (autogen.sas) that would read the metadata which triggers the creation of databases that contain key words like dataset names, variable names and produce a suite of macros programs that are described in the metadata. This administrative metadata helps to manage information resources as well as other technical information for purpose of appropriate file management and preservation.

Metadata is often used to control the workflow. Workflow in this paper is referred to the process that surrounds a content management system. The content is whatever you want to create, edit, collate and deploy. High-quality metadata makes it all work consistently and reliably.

There is the basic metadata required for an output:

Type	tlfnum	Title
Table	1.1.1	Patient analysis sets and reasons for exclusion - RS
Table	1.2.1.1	Demographic data - TS
Table	1.2.1.2	Demographic data - FAS
Table	1.2.1.3	Concomitant diagnoses by MedDRA SOC and preferred term - TS

Then there are the details about which program produces the output and where the output is stored:

pgmloc	pgmname	outno	outloc	outname	logname
pgm/ctr/tpopu.sas	tpopu.sas	T1	lst/tpopu_t1.lst	tpopu_t1.lst	tpopu.log
pgm/ctr/ct/demo.sas	demo.sas	T1	lst/demo_t1.lst	demo_t1.lst	demo.log
pgm/ctr/ct/demo.sas	demo.sas	T2	lst/demo_t2.lst	demo_t2.lst	demo.log
pgm/ctr/condiag.sas	condiag.sas	T1	lst/condiag_t1.lst	condiag_t1.lst	condiag.log

Finally there is the metadata required to generate the output:

shell name	Parameter 1	Parameter 2	Parameter 3
sh_popu1	pop=%STR(RANDFL='Y')		
sh_dem1	pop=%STR(TRTFL='Y')	vars=AGE #SEX #RACE HT WT	
sh_dem1	pop=%STR(FASFL='Y')	vars=AGE #SEX #RACE HT WT	
sh_cd1	pop=%STR(TRTFL='Y')		

These data can be used by a generic program to create programs with structured headers and macro calls to each

of the table, listing or figure shells.

```
/*=====================================================================
*
* STUDY                 : Study                    CREATION DATE: 12JUL2016
*
* PROGRAM               : demo.sas
*
* PROGRAMMER            : John Smith
*
* DESCRIPTION           : Create following outputs:
*                           - T1 Demographic data - TS
*                           - T2 Demographic data - FAS
*
* USER REQUIREMENTS     : SAP
*
* SAS VERSION           : SASV9.4
*
* COMMENTS              :
*
*---------------------------------------------------------------------
*
* REVISION HISTORY
*
* DATE                  :
* PROGRAMMER            :
* REASON FOR CHANGE     :
*
*=====================================================================*/


 ********************************************************************;
 ***
 *** Output No. T1
 *** Title: 1.2.1.1 Demographic data - TS
 ***
 ********************************************************************;

%sh_dem1 ( outno = T1
          ,pop=%STR(TRTFL='Y')
          ,vars=AGE #SEX #RACE HT WT
          );


 ********************************************************************;
 ***
 *** Output No. T2
 *** Title: 1.2.1.2 Demographic data - FAS
 ***
 ********************************************************************;

%sh_dem1 ( outno = T2
          ,pop=%STR(FASFL='Y')
          ,vars=AGE #SEX #RACE HT WT
          );


********************************************************************;
***                                                              ***;
***                   End of Program                             ***;
***                                                              ***;
********************************************************************;
```

To take data analysis to the next level with metadata, we strongly recommend the use of metadata to generate programs which are created by applying logical strategies and adhering to Good Programming Practice (GPP). This document is available at the PhUSE Wiki.

For guideline and more details, go to:
http://www.phusewiki.org/wiki/index.php?title=Good_Programming_Practice

METADATA CHALLENGES

In high functional metadata-driven software, the interrelationships within the metadata become complex and maintenance becomes challenging. If, by accident, a metadata element is defined incorrectly, subtle or major malfunctions manifest in the system's operation; for example, errors may creep into the data. It is therefore desirable to build programs that minimize the possibility of incorrect metadata. We describe here a rule-based approach to good programming practice to metadata management that allow a system designer to define complex constraints that prevent inconsistencies from occurring when changes are made to the metadata or automatically cascades a series of corrective or maintenance actions to maintain consistency.

One of the aspects of metadata challenges is to ensure that we are exploiting the metadata that we are collecting to the fullest, possible extent and that the metadata is easily accessible and effectively used across the organization. It is well known that metadata changes half way through projects as well as data changes and updates required on completed studies. Any changes will need to be kept up-to-date so that new metadata is incorporated. We need to be able to ensure that changes or updates to metadata should not in any way affect the quality of the programs that are used to read or analyze the data.

At time we have to face the challenge of dealing with disparate information sources that make up the corporate data landscape. In order to use metadata effectively, we must create a consistent and easy-to-understand format across the wide variety of sources. Couple with consistency in metadata format, we have to create analysis programs that are dynamic, and also easy to review, update and maintain. It is important to bear in mind that some users are likely to have limited knowledge of the programs that are associated with the metadata as they are not necessarily the original creators. Moreover, challenges are not limited to prior knowledge, metadata programming should be written using good programming practices in mind as programs can be used across the globe by a variety of programmers of different skill sets and experiences.

Enforcing business rules for metadata is an important aspect of maintaining data integrity and usability. Effective and clear communication with all of the internal stakeholders who have access to, or are using data, is vital to leveraging metadata.

WHAT ACTIONS CAN WE TAKE?

Half the battle of writing programs that work is to adhere to good programming practices. One of the practices that are highlighted is to write programs that are easy to read. A primary argument for writing programs that are easy to read is that doing so makes it easier to prevent and/or find errors. Programmers are encouraged to test each part of the program before proceeding to the next part with small datasets and with representative data.

Good programming practices start with a good, clear design. Update the design documents or the specifications regularly. Create additional design documents before adding major new features or functionality.

The program under development should be at all times functioning. The development process consists of adding new functionality without breaking existing functionality or disrupting overall design.

Work has to be divided into small incremental steps that can be typically accomplished and code-reviewed in one day. Even large-scale rewrites should be made incremental.

Every line of code written or modified should undergo peer review. The smallest team must contain at least two programmers so that they can code-review each other's changes. The programming manager also plays a crucial part to ensure they monitor compliance to ensure that the team adheres to an agreed style whether it is a style that is totally automated or it is part automatic and part manual process.

We recommend macros programming as macros can save you time by letting you automate relatively simple tasks that you need to perform often, as well as complex procedures that consist of many steps. Macros can be powerful tools that can greatly reduce the time that you will need to finish your work and can eliminate the need to remember all the steps in a tedious procedure. Generic macros generate programs that adhere to Good Programming Practices (GPP) in which it has the following features.

- Header: who wrote it, program name and reason
- Structured program: input, process and output
- Comments: explain what the next part of the program does and why, relate to the specification
- Datasets: use meaningful names, do not overwrite, use different names
- Indent: use consistent spacing

As metadata get more voluminous and complex each day, we need to ensure skill sets are matching with the demand for more sophisticated and complex programming. It should be recognized that programmers will need to periodically be trained to improve their programming skills that are in line with good programming practices.

We also strongly recommend the implementation of a clear and easy to follow guidelines or work practices document for programming in the organization. The GPP Guidance document is clear, concise and easy to follow and can be downloaded from the GPP website.
The reference and link to the PhUSE Wiki for guideline and more details is:
http://www.phusewiki.org/wiki/index.php?title=Good_Programming_Practice

WHAT DIFFERENCE CAN GPP MAKE?

Any programmer dealing with frequent changes to program specifications are prepared to cope with all things that are frustrating and error-prone. Changes to report headers and footers, dataset contents, and other aspects of client deliverables have to be communicated effectively and implemented correctly. Effective programming which follow the guidance document on good programming practices would be the best way forward to better and easier way to create high-quality outputs.

Superior coding techniques and good programming practices are hallmarks of a professional programmer. The bulk of programming consists of making a large number of small choices while attempting to solve a larger set of problems. How wisely those choices are made depends largely upon the programmer's skill and expertise. The good programming practices provide guidance for programmers to create programs with clarity and readability which provide high quality and reliability outputs.

This paper presents examples of some fundamental coding techniques and provides a collection coding practices from which the GPP guidance document outlined. The coding techniques are primarily those that improve the readability and maintainability of code, whereas the programming practices are there to provide standards and best practices to increase performance enhancements with easy to maintain and update to changes.

A suite of good programs are tools that are critical to analyses. We can have the best-designed metadata in the world, but if you don't provide tools for users to display and use it, the metadata are close to useless. The tool type we reference in this paper to access the metadata are implemented as SAS programs and macros. As this is highlighted in the GPP guidance document the use of macros and/or defensive programming allow greater flexibility and improve efficiency in production and reliable quality control of outputs.

A proper documentation for the programmers on tools will provide knowledge on the relevant macros, their applications, limitations and functionality for mapping parameters. The programmer does not have to know the structure of the program as this is handled by the macro. What the programmers do need to know is how to use the macros. Hence, it is essential to build robust and dynamic macros that work in the complex metadata environment, and to achieve this programs should adhere to GPP.

All Good Programming Practice (GPP) applies to macro writing, but the following are key importance.

Header and comments:-

- Explain clearly what the macro does, explain all parameters and give an example call
- Macros variables are generally local
- Build in very clear error trapping and consider custom messages to log

If the program encounters unexpected data or metadata behavior, this needs to be identified quickly by users.

Equally important is the use of sensible and consistent naming convention. When we name a macro in a program, the macro name alone should give an idea of what it does, although commenting would always be useful.

GPP enhances the tasks undertaking by the developer for the designs and structures of the macros. The key considerations are the life time of the programs, the reusability of the program within a compound or across different departments within an organization.

There are huge benefits of the metadata-driven approach. If the metadata changes, the tools pick up the changes and present the modifications to the program. The program itself does not have to change. In addition, there are benefits of using the macros from a system and validation perspective. The tools will be validated before being used by the programmers. The tools are generalized, and can be used by any project that uses properly formatted variable-level metadata. If properly constructed, the tools will also perform checks, issue messages, and in general be more thorough than if similar functionality were embedded within an individual study's program.

CONCLUSION

Dataset, variable characteristics and other metadata are held in a format that can be easily accessed programmatically. Access tools such as the macros autogen.sas allow the metadata to be used any number of times without the possibility of incorrect transcription from the specification source.

If the access tools are used, modifications to a metadata value will be picked up automatically in all appropriate locations. Consistent use of the metadata access tools ensures that a change that needs to occur "n" times does, indeed occur "n" times.

Designing, developing, and writing programs to make the metadata readily accessible to programmers can be time-consuming and costly. However, if the programs are properly written with good programming practices in mind, they can be modified to accommodate new variables, new options, etc. relatively easily.

.

CONTACT INFORMATION

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

Jennifer Chin
Eisai Limited
Mosquito Way, Hatfield,
Herts, AL10 9SN
Work Phone: +44 (0) 845 676 1217
Fax:+44 (0) 845 676 1482
Email: Jennifer_chin@eisai.net