SAS and MedDRA – A Winning Combination
A “Home Grown” application for the Medical Coding of Adverse Events with MedDRA
Elaine Dempsey, Rho Inc., Chapel Hill, NC

ABSTRACT
The introduction of the MedDRA dictionary as an authoritative source for the coding of adverse events has presented a challenge to many pharmaceutical companies and CROs responsible for doing this coding. As a small but growing CRO, we had to make the decision whether to spend our money on a vendor application or to invest our money and time in the resources within the company to develop a MedDRA coding tool of our own. The latter option was chosen and SAS would be the tool. Our coding system was built using SAS version 8.2 and leveraging the power of the object-oriented AF/Frame technology. Base SAS, SAS/SQL and SAS/SHARE were also used from within OO environment. The intent of this paper is to demonstrate the winning partnership between SAS and MedDRA.

INTRODUCTION
With the advent of MedDRA as the new “gold standard” for medical coding of adverse events, most pharmaceutical companies and Contract Research Organizations who had used Costart for so long are now being forced to stop and re-evaluate their coding practices along with their electronic data management and automatic coding systems. By means of this paper I would like to discuss the changes and effort required to do successful medical coding using the new MedDRA standard and to examine the possibility of using SAS to implement such changes in our electronic systems.

COSTART – OUT WITH THE OLD
With a very few exceptions, Costart is basically a simple, one-level coding dictionary. In a standard SAS application, an adverse event from the study data is read in and compared to the Costart dictionary. If an exact match to an adverse event record in the Costart dictionary is found then one costart low level term, one body system code and one body subcategory are returned. The choices may not be entirely realistic but they are certainly easy. For instance, note this example where we want to code an adverse event of “Peripheral Edema”. The Costart dictionary returns two almost identical records. The only difference is in the spelling of “edema”.

MEDDRA – IN WITH THE NEW
When MedDRA came along, we were introduced to multi-level hierarchical medical coding. A low level term and a body system were no longer enough to correctly and specifically code an AE. There were now several layers of choices in between all leading to the appropriate Body system organ class to associate with the adverse event. There were also multiple accepted spellings for the same term. As in the example above, note what the Meddra dictionary returns for “Peripheral Edema”.

BUILDING A CODING SYSTEM
In our case, as a small but growing CRO, we had to make the decision whether to spend our money to purchase a vendor application or to invest our money and time in the resources within the company to develop a MedDRA coding tool of our own. Since we already had the full complement of SAS products available for use, it was decided that we would build our own tool and leverage the resources that we already had. And so a major undertaking was born.

APPEARANCE
We started out by designing a user interface that exhibited economy and clarity. The interface should make it easy for the user to navigate through the levels of MedDRA and have a clear view of the choices to be made. The GUI was built using SAS/FRAME objects (with SCL behind them) along with both visual and non-visual class objects. We relied heavily upon existing SAS class methods but also wrote many of our own classes and methods to accomplish specialized tasks.

The frame should be heavily populated with data tables, listboxes and/or combo boxes to give a clear view of what is going on and it should all fit on one screen, if possible, to avoid additional keystrokes from paging from one screen to another. The interface needs to remind the coding personnel of what study and protocol they are working on. It should also be very clear which
datastream is being coded, whether it be adverse events, concomitant medications, medical history, etc. A status count of how many of the terms have been coded and how many are left to be coded will give the user an idea of the progress that has been made on the coding task. And, a request was made for a feature to be able to tell whether the individual mappings have been made by dictionary exact-match auto-encoding or through manual decisions by data management personnel. Although the GUI landscape is a bit crowded, it is felt that this best suits the needs of the professional medical coder. The resulting interface design looks like this:

For the benefit of the use who handles multiple projects at the same time, the top portion of the frame identifies (in text boxes) the project and datastream that is currently being worked on. A count of the terms coded and uncoded is also displayed in text boxes. This allows the user to keep track of their progress and the counts are updated as each term is coded. These categories are also color-coded for more intuitive viewing.

A listbox object at the upper left holds a list of all unique verbatim terms (term listbox) or adverse events so that the user may choose which one is to be coded. When a unique term is selected the data table at the upper right is populated with all records containing that term. If a specific record from this data table is selected, the information in the blue area pertaining to that record is filled in.

A tabber object is used so that four data tables can be displayed in the space of only one. The first tab contains a table displaying all the lower level terms in the coding dictionary which, in this case, is MedDRA. The next three tabs contain all the mappings made by the coder. A mapping is a manually selected match between the adverse event recorded in the study to a low-level term in the dictionary or from a low-level dictionary term to a higher level dictionary term. It can be applied to all records fitting the criteria or it can be applied to only one record because of a specific circumstance. The second tab contains all non auto-encoded low level mappings. The third tab contains all non auto-encoded high level mappings and the fourth tab contains all non auto-encoded record specific mappings. There is a filter input field on each tab to allow the user to do a “search” on a specific string that they type in and a pushbutton for clearing that filter.

There are also pushbuttons to allow the user to assign a mapping and to delete a mapping.

FUNCTIONALITY.
When the data to be coded is read in, the frame is populated like this:

At the upper left, a combo-box is used as a means of allowing the user to move from one datastream to another by just clicking on the arrow and selecting which one they want. To change from coding adverse events to coding concomitant medications is as easy as clicking on the combo-box arrow.

Below the datastream selector is a list-box filled with all the unique adverse event terms available for coding. Selecting an item in this list-box will fill in record specific information in the data table to the right and in the text boxes in the blue area below. If the record has already been coded the coding selections combo-boxes will be populated. The items in the list-box can be filtered into coded/uncoded lists by clicking on the pulldown menu labeled “Terms” and making the appropriate selection.

At the bottom of the frame is the tabber with each tab populated with its respective data. Switching from one data display to another is as easy as clicking on the appropriate tab.

The pulldown menus on the top left contain navigational tools for displaying the data in different ways and for branching off and displaying data that is not included on the frame but that may need to be referenced to get needed information. They provide a way to view the entire MedDRA dictionary and the entire dataset being worked on.

PRACTICAL APPLICATION
Here is an example of a very small study containing four adverse events.

Of course, there are some simple preliminary setup procedures that are necessary to enable the coding system to locate the necessary files for the study being worked on. These will not be discussed in detail here because there are many ways that it could be done depending upon your setup. We chose to have what we call a “Dictionary Information” dataset to contain all the necessary setup information which can be referenced at any time by the coding application. After this mechanism is in place, the first step is to select the study that you would like to do medical coding on.
When a study is selected and we proceed into coding, we are then asked which set of CRF data we would like to perform medical coding on. In this case we have a choice of coding Adverse Events or Concomitant Medications.

After choosing to code Adverse Events the first thing that SAS is programmed to do is compare the AE dataset with the MedDRA dictionary to find exact matches, i.e. autocoded matches. After it goes through the entire dataset the results are then displayed.

In this particular study there are only four adverse events. The count boxes on top indicate that two of these (both of them being “Headache”) have been coded automatically by an exact match in the dictionary. One event (Peripheral Edema) has been partly coded, or coded at a low level only. And, one event (Elevated Glucose) has been left completely uncoded.

Clicking on “Peripheral Edema” will fill in the partial coding that was done and will give the user choices to complete the coding as follows:

When the user makes all the necessary choices, the rest of the coding will be done automatically. After the Assign button is pressed and the assignment made, the term will then be considered completely coded and the counts in the boxes at the top will change.

To manually code an uncoded term, in this case “Elevated Glucose” we have no partial coding to start with. So, the user must attempt to find the closest matching term in the dictionary by using the Filter. The user may try to filter on the word “Glucose” and see what results it brings.
Filtering on the word “glucose” brings back many results and it is up to the coder to select the dictionary match that best fits the adverse event record being coded for the study.

In this case, I will choose “Glucose increased” as the low level term and the rest of the hierarchy will be filled in automatically. Press the assign button and the entire dataset of adverse events has been coded. Granted, this is a very small study but the intent of this paper and demonstration is to show that, large or small study, the new MedDRA structure is not as intimidating as it may seem.

CONCLUSION
Although, in the pharmaceutical field, SAS software is generally thought of for its statistical power, it is a largely untapped resource for its other many features such as screen building and object oriented development that we need to keep up with the latest Information and Technology advances in the pharmaceutical world. SAS and MedDRA were truly meant for each other.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the author at:
Elaine Dempsey
Rho, Inc.
100 Eastowne Drive
Chapel Hill, NC 27514
(919) 929-6200 x300
(919) 408-0999
edempsey@rhoworld.com
www.rhoworld.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are trademarks of their respective companies.