Introduction
Demographics and safety are key components of most clinical study reports. The reporting of this data is similar from study to study and therefore can be standardized. This standardization creates an opportunity for SAS macro-driven reporting. This paper discusses a simple methodology for developing clinical reporting macros. The methodology consists of five steps.

- Report Definition with Users
- Macro Specification & Review
- Coding
- User Testing and Acceptance
- Validation and Documentation

Once developed and validated, reporting macros may be used in programming, incorporated into a SAS/AF reporting application or incorporated into SAS PH/Clinical as report templates.

Use of standard macros reduces report development time and can improve quality by decreasing the opportunity of programming errors.

Development Methodology Overview
The macro development methodology consists of five steps. In this paper we use an Adverse Event Summary Report Macro as an example.

- The first step is to define the report with the report users. In the case of clinical reports this step is performed by statisticians, medical writers or medical reviewers.

- The second step is to produce and review the macro specification document. This step is performed by report producers, such as statisticians or statistical programmers. The programmer also produces a validation document, specifying programming steps and data manipulation performed in the macro.

- The third step is coding and testing. The coding is performed as specified in the report macro specification.

- The fourth step, user testing and acceptance, is performed by the report producers (statisticians or statistical programmers). The final reports are then reviewed and accepted by the report users (medical writers or medical reviewers).

- The final step is validation and documentation. Validation is performed by a second programmer. Technical and user documentation is finalized and included in the macro user manual.

Report Definition
Report definition is the most crucial step in the development process. This step must include all users involved in developing the clinical study report. The report users may be statisticians, medical writers, regulatory reviewers or medical reviewers. The product of this step is a report template (Appendix 1).

The report template describes the standard report and options. A good starting point for developing the report templates is to compile all of the adverse event summary reports which have been produced within the last two years. Categorize the reports into types and create one report template for each type of report. A word of caution, if there are many different adverse event reports, don’t try to combine all of the reports into one report macro. One fits-all report macro is hard to define and will be a maintenance nightmare.

Although the reports are standard, they are "Flexible Standards". The overall format of the reports are consistent, specific portions of the reports are flexible. The flexible portions of the reports include:

- Titles
- Footnotes
- Column and Row Variables
- Column and Row Labels
- Dosing or Treatment Labels
- Number of Doses Displayed
- Number of Time Points Displayed
- Groupings (i.e. by center or overall)

These portions of the reports are driven by the study design and the specific reporting needs of the study. This concept of Data Driven Reporting means that the data drives the report structure.
Macro Specification

The macro specification document (Appendix 2) describes the parameters used in the macro call and is an outline for coding the macro. This step must include any user involved in producing the report. The report producers may be statisticians, clinical programmers or statistical programmers. The developers’ task is to design the macro specifications using the report template as a guide.

Note that when developing a series of report macros, it is very important to keep the macro parameters and design consistent across all macros. This consistency across macros facilitates ease of use for the users of the report macro system. Additionally, it is important to develop a tool kit of utility macros which are used in developing report macros. Examples utility macro may be a titles macro, footnotes macro or a standard page format macro. The macro specification is used as a starting point of the documentation.

Coding and Testing

Coding and testing is the easiest part of the job. An important part of coding and testing is to create a testing database. This database should contain datasets from many different types of studies. For example, data from phase 1, phase 2 and phase 3 studies. This data must be static and should reside in a separate write protected directory. This data will also serve as validation database.

We follow a few simple guidelines when developing report macros.

- Use a structured programming technique. Most companies have not quite implemented object orientated programming (OOP) techniques, so use the next best thing.

- Develop data-driven code, allowing the data to determine the report structure. Pre-process the dataset to create formats, labels and groupings. If possible, DO NOT hard code values, labels, column widths, etc.

- Use a toolkit of utility macros to develop the report macros. Utility macros are building blocks for report macros, develop to replace procedures or code that is used in several other report macros. Try not to create a macro-hell development environment containing 200 utility macros where many are similar or project specific. For example, currently we use 20 utility macros in our tool kit.

- Use SAS procedures to process data. For example, when calculating counts or means use Proc Means or Proc Univariate. This will ensure consistency of results.

- If possible, use Proc Report for formatting and displaying reports. Proc Report is a very powerful procedure that provides many features that are hard to duplicate using Proc Print or Data_Null_and Put statements.

Thorough testing should be performed on each code module and on the final macro. The developer should test, if possible, all error situations and all macro options using at least three different datasets.

The last steps of coding and testing is to develop draft user and validation documentation. Our user documentation is similar to the first page of the macro specification in Appendix 2; it lists macro parameters, default values and options. It is important to include many example outputs in the documentation. This documentation will be used by the users during user testing and acceptance.

Validation documentation provides a technical description of processing performed by the macro and is used during validation to replicate results returned by the macro.

User Testing and Acceptance

User testing and acceptance is performed by the report producers and report users. Before testing can start, the users must be trained. Training consists of reviewing the documentation with the report producers, demonstration of use of the macro with 'real' data familiar to the users.

The report producers, such as statisticians, statistical programmers or clinical programmers can then start testing. We encourage the report producers to reproduce reports from finalized studies and create reports for current ongoing studies. It is important for the users to understand that the report macro is NOT validated and output should not be included in final studies. This testing should include reviewing the documentation.

The test reports created by the report producers, can then be passed on to the report users (statisticians, medical writers, regulatory reviewers or medical reviewers) for review and acceptance.

For most report macros, this step commonly results in changes or enhancements to report specification or macro parameters. Do not be discouraged, this happens because
initial specifications are rarely perfect. Try not to make the changes one at a time. Meet with the users and compile all changes, then implement all modifications at one time. The user testing and acceptance process should be repeated after code changes are completed.

Validation and Documentation
Validation and documentation should be simple and quick after the user testing and acceptance is completed. In the first step, the draft documentation, produced in the coding step is finalized and included in the Report Macro Manual. Validation is the final step. A good starting point for validation is to have a written validation procedure or SOP.

At DataCeutics, Inc. the validation is performed by a second programmer utilizing a validation database. As described above, this database must be static and should reside in a separate write-protected directory. Additionally, there is a separate directory used to store all programs and logs used in validation. The DataCeutics, Inc. validation steps are as follows:

- Using the macro being validated, produce several reports using many variations of macro calls. It is important to test all credible combinations of parameter settings.
- Using the validation document as a record of how data is processed, a second program is written to replicate the results expected from the report macro.
- The results of the two programs are compared.
- The programs and results are saved so they can be used as proof of validation and to be used in future revalidation.

Revalidation should be performed after operating system changes, software upgrades or enhancements to the report macro. When the validation tests are successfully completed, the report macro can be moved into production.

Conclusions
The clinical report macro development methodology described in this paper is simple and straightforward. It allows us to create a flexible report macro in a short time period. As in most software development, agreement with the users on a set of complete and clear specifications is the most important part of the process.

Once developed and validated, clinical report macros will greatly reduce the time needed to produce the common demographics and safety reports which are included in most clinical study reports. This allows the users to spend more time on the important key analyses. Quality is enhanced by reducing the quantity of coding done for individual reports (controlling opportunity for errors), and standardization of report appearance and content is enforced by use of a single programmatic method.

The data-driven aspect of the reporting macros allows them to be easily incorporated into a SAS/AF reporting application or into SAS PH/Clinical as report templates.

Biography
Paul Gilbert is Vice President of DataCeutics, Inc., a consulting group specializing in software solutions, system integration, programming and support in the areas of clinical data management and statistical reporting. His fourteen years' experience includes clinical data management, implementing and maintaining BBN/Clintrial, designing SAS-based biostatistics reporting systems, managing SAS-based NDA programming support and CANDA integration. Paul can be contacted at 610-970-2333.

Steve Light, a Senior Consultant at DataCeutics, Inc., is responsible for SAS report macro development and biostatistics/c clinical SAS programming support. His ten years of experience with clinical information systems include SAS systems development and validation, NDA driven project management, programming and clinical data management.
Appendix 1
Adverse Event Report Summary Template

Appendix/Table xx

Title 1
Title 2
Title 3

Optional by- line

<table>
<thead>
<tr>
<th>WHO Body System</th>
<th>WHO Preferred Term</th>
<th>DOSE A (N=)</th>
<th>DOSE B (N=)</th>
<th>DOSE C (N=)</th>
<th>TOTAL (N=)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Systems</td>
<td>Any AE</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
</tr>
<tr>
<td>Body As A Whole -</td>
<td>Accident</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
</tr>
<tr>
<td>General Disorders</td>
<td>Fever</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
</tr>
<tr>
<td>Central And Peripheral</td>
<td>Dizziness</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
</tr>
<tr>
<td>Nervous System Disorders</td>
<td>Headache</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
</tr>
<tr>
<td></td>
<td>Tremor</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
<td>nnn (xxx%)</td>
</tr>
</tbody>
</table>

Report Options:

1) Titles are optional and there can be up to seven titles.
2) Footnotes are optional and there can be up to ten footnotes.
3) BY line is optional.
4) (N=), the number of patients in dose group is optional.
5) TOTAL column is optional.
6) nnn, can represent the counts of patients or counts of adverse events.
7) (xxx%), the percent of patients in the dose group is optional.
8) Columns for BODY SYSTEM and PREFERRED TERM are optional. You can use both or one column.
9) DOSE, the column descriptor is optional. You can use other variables besides DOSE.
Appendix 2
Design Specifications for the %AE1 Macro

%AE1 (in=, title=, footnote=, pageby=, column=dose,
total=y nlabel=y, count=patients, row=whobody
whopre, props=, suffix=none, type=l,
pagenum=1, footnum=1, appnum=titles.lis);

\textbf{in=} input data set. This parameter is required with no default.

\textbf{title=} Number and text for titles used on table. The number is specified followed by the text enclosed in double quotes. Valid title numbers are 4 through 10. This parameter is optional with no default. \textit{Use the \%titles macro.}

\textbf{footnote=} Number and text for footnotes used on table. The number is specified followed by the text enclosed in double quotes. Valid footnote numbers are 1 through 9. This parameter is optional with no default. \textit{Use the \%foots macro.}

\textbf{pageby=} name of variable(s) used to PAGEBY on report. More than one variable can be specified with spaces between. This parameter is optional and there is no default.

\textbf{total=} Optional display of the total column. The default is Y (yes), use N for no display.

\textbf{column=} name of the column variable. This is optional and the default is dose. If dose is specified, then dose will be formatted with the \%formatm macro using the doseFmt variable. The calculation of \textit{(N=)} in the label is based on a combination of the pageby and column variables where N is the total number of patients in that pageby and column group.

\textbf{nlabel=} option to display \textit{(N=)} in the dose label. This is optional and the default is Y (yes), use N for no.

\textbf{count=} display counts of the number of patients (patients) or the total number of events (events). This parameter is required, the default is patients.

\textbf{row=} name of the row variable(s). This is optional and the default is row=whobody whopre, which produces a display of body system and preferred term. Row=whobody produces a display of body system. Row=whopre produces a display of preferred term.

\textbf{props=} Proc Report options.

The following parameters are passed to the \%rstart macro if suffix is not NONE.

\textbf{suffix=} output file suffix as used by \%rstart. The default is NONE, which causes no \%rstart to execute.

\textbf{type=} type of output to be created (l=landscape, p=portrait). This is required and the default is l (landscape).

\textbf{pagenum=} starting page number submitted to \%rstart. The default is 1, use NO for no page numbers.

\textbf{footnum=} number of footnote for standard documentation footer. This is submitted to \%rstart. The default is 1.

\textbf{appnum=} name of titles file submitted to \%rstart. The default is TITLES.LST

540
Appendix 2
Design Specifications for the %AE1 Macro

Developer Notes

1) The variables dose and dose_fmt are required on the in=dataset when column=dose. These are used to create the dose label when column=dose. The (N=) for the label is calculated by combining the variables from pageby and column.

2) Use Proc Report to create output.

3) The format for N and % is nnn (xxx%), for example 85 (15%). When the percentage is lower than 1 % then use the (<1%) designation, i.e. 5 (<1%). The percentage is calculated using the total number of patients in the pageby= and column= group. This in the number represented in the (N=) portion of the dose label. This is done because there may be missing values for some patients.

4) The width of the body system and preferred term columns, defined in the row= parameter, should be <= 25 and should automatically wrap. There is an option to specify a length for the row= if there is a page fit problem. The option is row=var1 length1 var2 length2, for example row=whobody 20 whopre 20.

5) When using the 'count=patients' parameter, the data should be subsetted to contain only one unique body system and/or term per patient. This is to count the number of patients with a specific body system and/or term. The sorting sequence should be a combination of pageby=, column=, patid and row= variables.

6) Place a blank line between each body system.

7) The variables patid (patient id) and ae_flag (Y/N flag to signify if a patient had an adverse event) must be contained in the adverse event dataset. These are used for calculation counts of any adverse event and % of patients.

8) The macro design should follow a structured design.

- Parse out all macro parameters and perform error checking to assure that dataset and required variables are present.
- Process dataset. Subset patients if count=patients. Create one record for each patient, using the ae_flag variable. This one record per patient is used to produce counts of All Systems and Any AE. Create additional records in the dataset to represent the Total column, if required. Create N= for the dose label. Create counts (nnn) and percent (xxx%) using proc means, combine into one variable.
- Create required format for dose or other variables using the %formatm macro.
- Process macro parameters required for %rstart macro call and perform the %rstart call.
- Perform call to the %titles macro.
- Perform call to the %foots macro.
- Create and execute the proc report code. Include the proc report options.
- Perform clean-up. Delete temporary datasets and delete macro variables.