Paper 169-27

The Web Data Entry System: Methods for Web Development and SAS® Data Management

Paul A Thompson, Sarah Littlewood, Avril J Adelman and J Philip Miller
Division of Biostatistics, Washington University, St. Louis, MO

ABSTRACT

Processing data using the SAS/IntrNet® system requires working with SAS®, HTML, and JavaScript code simultaneously. The Web Data Entry System (WDES) is a macro-based system which develops web page systems and manages the web-oriented data handling process. The system simplifies and speeds development by using SAS macros to very quickly process a modified HTML file, producing a processed HTML file, the SAS dataset, and the code to process the interaction between the two. The system simplifies and speeds management of transactions using a comprehensive system of SAS macros. The WDES uses a metadata strategy, processes the final HTML file and uses the SAS resolve function to simplify modification of files. This approach is currently supporting two multi-site clinical trials in which the Division of Biostatistics, Washington University, serves as the data center.

INTRODUCTION

The internet offers unprecedented opportunity for remote interaction with users. In the Division of Biostatistics, Washington University, we are managing data from remote clinics in several multi-site clinical trials (including the CRISP project studying polycystic kidney disease and the EXCITE trial studying stroke rehabilitation techniques). In these projects, we must provide users at multiple remote sites with the full data editing experience, including methods which allow users to add new observations to datasets, edit existing observations, rarely delete observations, and list values from the dataset. In return, we must provide feedback and information to the clinics (i.e., recruitment status, case information, interim results).

In our environment, dynamic (rather than static stored) HTML pages are generally used, for several reasons. First, maintaining static pages is difficult when multiple clinics are used, which each require a separate page. Second, the pages must allow collection of data (using a form version of the screen) and printing of the data (using a printable version of the screen). These are accommodated most directly using dynamic pages. Finally, the system needs to work in an environment where aspects are changing frequently. Use of a dedicated HTML editor, such as FrontPage®, is not consistent with these needs for dynamic page construction. Development and code management is difficult in an environment in which frequent modifications are made. The SAS/IntrNet approach works well in this situation and with these requirements.

To satisfy these basic needs, the Web Data Entry System (WDES) has been developed. It performs two functions:

1. It is a development system to produce HTML forms, automating SAS code production and final HTML page preparation simultaneously.
2. It is a macro-oriented, metadata-based method for web-based data management.

OTHER METHODS

When setting up methods to work with SAS/IntrNet to gather data on the web using HTML-based forms in complex and multi-form projects, several approaches are often tried initially:

1. The PUT statement can be used to write HTML screens in the DATA step.
2. The HTML pages may be kept in files, and modified using scan, substr and other character functions.

Using these different approaches, it eventually becomes difficult to maintain a multiple-form system. Here are some problems commonly encountered:

1. Frequent HTML form changes are very difficult to manage.
2. PUT statements involve single and double quotes to properly write statements. These invariably are difficult to maintain, result in unclosed quotes and are difficult to read.
3. Once the initial author quits, maintaining code is difficult.

These problems call for a radically different approach, which systematically solves the problems of maintainability, development and system management of forms. These solutions are provided by the WDES.

KEY IDEAS FOR THE WDES

The basic ideas for the WDES are as follows (some points are not unique to WDES):

- **Web to SAS to web:** An HTML screen or page is used to get data from a user and send it to SAS, which performs a task and returns feedback to the user in another HTML screen.
- **The SCAD:** The *gestalt* of code (SAS, HTML and JavaScript) and data used to support the processing of a single form is called the SCAD (**S**creen, **C**ontrols, **A**nd **D**ataset). It includes these components:
  1) Screen: an HTML-based form presented on the web.
  2) Controls: the SAS code which moves data between the web form and the SAS dataset.
  3) Dataset: a SAS dataset (two-level name, variables and their characteristics).
- **Macros for common processes:** The very common standardized data management processes (moving data from SAS dataset to web form and *vice versa*, printing data, checking for observations) are performed using macro functions. This ensures that code is easily maintained. Macro functions for data processing enable systematic and common modifications in all programs by macro changes.
- **Metadata for projects and SCADs:** Metadata is "data about data." A metadata system is used to provide the macro functions with various types of information. The two type of metadata, maintained using a web interface, are:
  1) **Project-wide:** A project is a large data collection effort, generally involving multiple forms and data collection instruments, directed to fulfilling some joint mission. Projects involve common directories for SAS datasets, format libraries and stored compiled macro libraries. They often involve shared lists of subjects and participants, and thus need a common location for such information.
  2) **SCAD-specific:** Within the project, each SCAD has unique metadata for dataset information, variable information and form information. The SCAD-level metadata is similar to the output datasets produced by PROC CONTENTS, but is more detailed and has several other types of information.
- **Common representation method:** A standardized method is used to name macro variables. This ensures that the system users are always clear on macro names.
- **Modular HTML files:** The final modified HTML file is divided into "pagelets" which are sections of the HTML file that function in a similar fashion. Thus, the header part, BY variable portion, main variable section and decision choice portion of the HTML file are placed in different HTML pagelet files. Thus, multiple SCADs can use shared components.

- **HTML page modification:** HTML pages are modified by including SAS macro references on the pages. The SAS resolve function is used to substitute the macro value for the reference, avoiding the use of character functions.

DEVELOPMENT OF WEB PAGES

BASIC IDEAS

The SAS/IntrNet process requires different types of code:

1. HTML code composes the screen form;
2. JavaScript code works with the HTML code; and
3. SAS code handles the data in the SAS environment.

One approach to development in such situations is the "literate programming" approach (Knuth, 1984). This approach uses a canonical source document which produces code in several different languages. Thus, when the canonical source document is modified, all derived documents are also modified.

In WDES, HTML, JavaScript and SAS code is derived from an HTML file called the "driver HTML" file, which is a file containing all of the form components for the SCAD. In the preparation process, it is read and interpreted by special macro functions to perform all other functions. It requires further preparation for final use in SAS/IntrNet. The entire code development process is controlled from a web-based system called the "Project Control" system. This approach uses web tools to develop other web tools, in a semi-automated manner.

SCAD DEVELOPMENT PROCESS

HTML driver file preparation: The driver HTML file is the key tool in WDES, as it is used to construct all other parts of the SCAD. The form, prior to web enablement, **should be** mature and stable. While the file can be modified after preparation, it is easier to work with stable forms.

The driver HTML file consists of standard HTML components (i.e., form elements such as TEXT fields, RADIO buttons and SELECT lists). The most important aspect of the driver HTML file is the form section (the code between the <FORM> and </FORM> tags). The beginning and ending sections of the form may both be left off the driver HTML file. Often, a project will have standardized beginning and ending sections. HTML code in the beginning section will be needed to ensure that different SCAD forms have a consistent, project-wide "look-and-feel." The project will also often have a consistent ending section, including a SUBMIT button and a tool to select the next task to perform. The standardized beginning and ending sections can both be added to the driver HTML file by the "Project Control" system.

Any of a number of HTML editors may be used to write the driver file. The user may choose to use the emacs editor or another plain text editor, which provides very good control of the development process, but requires understanding HTML at the code level. Detailed construction of the HTML code:

```
<INPUT TYPE="TEXT" NAME="Newname">
```

can be assisted using the HTML major editing mode. When using this mode, the emacs editor will assist the user by prompting for various text components.

Microsoft® Word® (either XP® or 2000®) may also be used. Word includes a number of tools for the creation of HTML files, accessed using:

```
View->Toolbars->Web Tools
```

The user should save all files in two forms.

- Standard HTML format, with all Microsoft HTML and XML controls, using "save as Web Page (*.htm;*.HTML)." Future modifications of the form are done with this version.
- Simplified or filtered form ("save as Web Page, Filtered (*.htm;*.HTML)') in which many of the extraneous Microsoft tags are removed. This file should be then listed as the driver HTML file in the SCAD metadata dataset.

The "Project Control" system: Development of the SCAD components in WDES is performed using the "Project Control" system. This is a web-based interface which performs a number of functions, including:

- managing the project-wide metadata;
- managing the SCAD -specific metadata;
- initially extracting variables from the driver HTML file;
- adding formats to the format library directly; and
- using the project-wide and SCAD-specific metadata to construct the final components of the SCAD.

A new project is defined by entering project metadata into the "Project Control" system, including information about directories (i.e., for SAS datasets and JavaScript files), and project LIBNAMEs. All information about the project is stored in this tool, and thus can be used during SCAD development.

1. The project is defined by specifying project metadata.
2. SCAD-specific metadata is next added.
3. The driver HTML file is read by the "Project Control" system, and variables are located in the HTML file. These are stored in the SCAD-specific metadata set. This information is presented to the user for further definition and modification.
4. Once the variables are fully described with the "Project Control" system, the final SCAD is constructed.

Project-wide metadata: Project-wide metadata are defined and managed at the start of a project, and can be modified later. Most of the metadata for the project are directory names and locations, including locations of SAS datasets, JavaScript files, cascading style sheet files, public file dumps, and other components needed for a complete project. Additionally, descriptive information is included in the metadata file (i.e., project title). The "Project Control" system is then used to create the directories listed in the project-wide metadata file.

Extraction of variables from the HTML file: The driver HTML file is processed by a WDES macro, controlled from the "Project Control" system. During this process, all variables in the HTML file are identified, and placed into the SCAD -specific metadata dataset. Variables are identified by finding HTML form components (i.e., RADIO, SELECT, CHECKBOX, TEXT, TEXTAREA items). The NAME field is read for the variable name.

As the variable names are determined, other aspects of the items are used to make an initial tentative determination of variable characteristics. For instance, a RADIO button item will be examined to see if the VALUE choices are numeric or character, and the TYPE of the variable will be set accordingly. TEXT items which have large SIZE or MAXLENGTH values will be classified as character, but those with shorter lengths (under 15) will be classified as numeric.

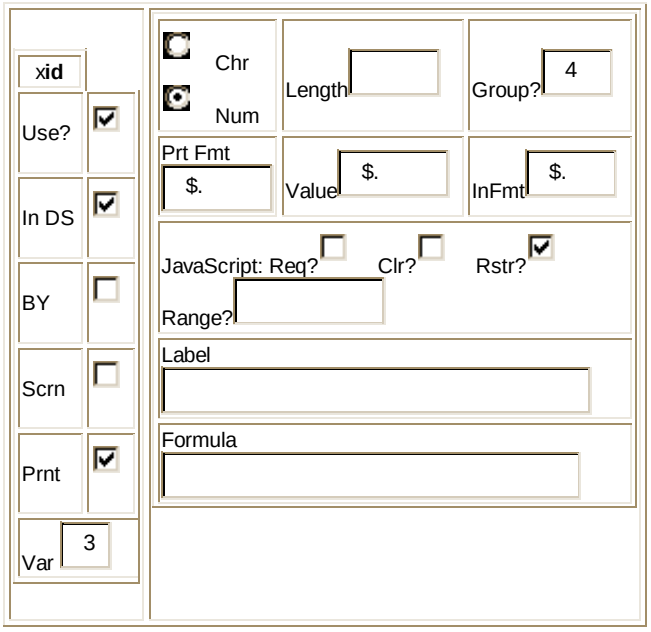
``Variable modification'' screen: The variable information is presented to the user for further evaluation and possible modification. The "Variable Modification" screen is an HTML web form, so that designers can use the system on either a remote or local basis. Variable characteristics (i.e., type, informat and printing formats, BY status, label) may be entered and modified using this tool. Once the modifications are completed, SUBMIT initiates the process of storing the values in the SCAD-specific metadata dataset. An example of the "Variable Modification" screen is shown in Figure 1. At this point, all required information for setting up the HTML screen, processing the information from the screen and

editing the data is known to the WDES.

Using this screen, the designer may also specify the skip pattern for the HTML screen. As the values are entered into the HTML screen, JavaScript code (added by the "Project Control" system) shifts the focus to items, using the specifications entered in the "Jump to" field. Since focus is very difficult to detect with RADIO buttons and CHECKBOX items, a small blue arrow is provided (again automatically by the "Project Control" system) which points at the current focus item.

The "Variable Modification" screen may be accessed at any time. New variables may be added to the SAS dataset using this tool. For this purpose, formulas for computed variables are a part of the SCAD -specific metadata dataset. Thus, the entire process of SAS dataset construction may be performed using the Variable Modification" screen, followed by further preparation using the Project Control system. More information about this tool is given in Thompson (2002b).

Figure 1: Variable Modification" screen section



SCAD preparation: Using the information in the project-wide and SCAD-specific metadata datasets, the driver HTML file is processed again (using the "Project Control" system), in conjunction with the information entered into the SCAD-specific metadata dataset. The following functions are performed:

- SAS code is set up to add new observations, recover existing observations for editing, list values and delete observations from the main SAS dataset. This macro code may be modified by the user later to add other functions to the processing of the SCAD information. A standardized set of macro functions is used to perform the data processing tasks. The various choices are made using the "Project Control" system. This is done by reading a pattern file from the system, and modifying it according to the information recovered from the metadata files.
- The final version of the HTML web page is set up, which involves producing a web page version with form elements (for data entry and modification) and a web page for printing (for presentation after data entry or modification has been performed). For instance, this code defines a TEXT item:

```
<TD><INPUT TYPE="TEXT" NAME="_tx"
    VALUE="&_xtx">-Text of message</TD>
```

This line contains a TEXT form element. The system would process that line from the driver HTML file, and produce a second line:

```
<TD>&_ftx-Text of message</TD>
```

The second line would be placed into another pagelet file, used for printing the results of operations on the SAS dataset. The changes in the lines produced by the "Project Control" system preserve the structure of tables and other control information, and print the information appropriately.

- JavaScript code is added to the HTML code. For this purpose, a core set of JavaScript functions is needed. These are provided with WDES. The JavaScript code enables the browser to:
  1) check ranges for numeric values;
  2) require that certain widgets have values, including TEXT boxes, SELECT lists and RADIO buttons;
  3) skip items based on the selection of earlier items; and
  4) clear and restore values for TEXT boxes.
- Cascading style sheet references are added. Many projects have a consistent "look-and-feel," involving a selection of background color, font for text and other factors. These are usually defined in a cascading style sheet (.css) file, which is shared across all SCADs in the project. Locations for the .css file and JavaScript files are stored with the project-wide metadata.
- RADIO buttons, CHECKBOX items and SELECT lists are prepared for editing. This is done by a "pre-slug" process. Macro references are inserted into the HTML code. These are a simple function of the variable name and the value of the button or selection list; if the name is "sex" with value "1," the "pre-slug" term inserted into the pagelet file is &_ysex1. During the setup for the edit process, the "pre-slug" term corresponding to the current value is set to CHECKED (for RADIO buttons and CHECKBOX items) or SELECTED (for SELECT lists). The value is then inserted into the final HTML page using the resolve technique.
- The SAS dataset is set up, including labels, formats, lengths and other required information. Computed variables, which are a direct function of other variables in the dataset, can be added as well (i.e., calculating a SAS date).
- The processes for automatic backup of the data are set up. The backups may be done in several places if necessary.
- The data audit method, using the SAS "audit trail" mechanism, is set up. The SAS "audit trail" system maintains a separate indication of all transactions involving the dataset. This system is useful, but does not survive replacement operations on the dataset (i.e., operations involving SORT, SET, MERGE, or UPDATE). Thus, the system also automatically "extracts" the audit trail and stores this in a permanent, standard SAS dataset. This version of the audit trail is permanent.

Final preparation: At this point, the file is ready for processing. The user must provide an HTML "on-ramp" for access to the start-up version for the form (from which edit, addition, listing, etc., operations can be initiated). WDES sets up a test version of this page automatically. All other functions have at this time been set up by the "Project Control" system. The designer must now check the results.

Overall flow of control: The overall flow of control in the "Project Control" system is shown in Figure 2 below. Arrows represent processing by the designer in the context of the "Project Control" system. The designer is responsible for the "driver HTML" file. The arrows represent processing by the "Project Control" system. The designer is also responsible for reviewing metadata following initial processing.

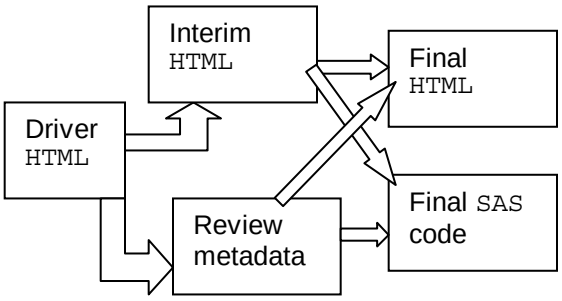


Figure 2: Flow of control in WDES

WEB-BASED DATA MANAGEMENT

WDES is also used to manage web page data processing. The system is built using a number of macro programs for the processing of the web information, which perform common functions simply and consistently. The macro functions are facilitated by the metadata system.

TASK SELECTION FOR PROCESSING

Information processing for a given form is controlled by a macro for task selection (which is written by the "Project Control" system for each SCAD). Variable _procmain is used to select the SCAD for processing, and Variable _procsub selects the specific task (add, edit, etc.) from the available choices in the SCAD. Values for _procsub are pre-set by the "Project Control" system. Other options may be added. For instance, if a special alternative listing method is needed, it may be added by selecting an unused values for _procsub (e.g., 8), and the relevant code added to the selection macro at an appropriate place. This preset selection macro approach allows standard methods to be done in a standard way, while enabling the user to add other options in a very flexible and simple manner. The value for _procsub is built into the task selection on the web pages, set up by the "Project Control" system, schematically shown here:

```
%macro _select;
preparation code
%if (&_procsub = 0) %then %do;
back to onramp
%end;
%if (&_procsub = 1) %then %do;
add new observation
%end;
%if (&_procsub = 2) %then %do;
get existing values for edit
%end;
%if (&_procsub = 3) %then %do;
get existing values for delete
%end;
%if (&_procsub = 4) %then %do;
list observations
%end;
%if (&_procsub = 6) %then %do;
add edited values back to dataset
%end;
%if (&_procsub = 7) %then %do;
finish delete
%end;
%if (&_procsub = 8) %then %do;
optional additional task
%end;
%if (&_procsub = 5) %then %do;
show page with "Add", "List", "Edit" tasks
%end;
ship page from queued pagelets to _webout
%mend _select;
```

In each of the sections above (in which the value of _procsub is used to select a choice), an action is performed. This usually involves moving values in or out of a SAS dataset. Each selection also involves queuing up a selection of pagelet files. The pagelet file queue is set up by the "Project Control" system, but may be modified by the user later.

This selection macro is set up by the "Project Control" system. Thus, the semi-automated methods involved in the "Project Control" system are used to pre-set the basic, standard tasks needed for data management. The system is designed to do basic common tasks in a standard manner, freeing up the designer to write code for non-standard, more difficult data management tasks.

MACRO FUNCTIONS

In WDES, common data management processes (adding, editing listing, and deleting observations from datasets) are performed using macro functions. These macros are supported by the SCAD metadata. The system uses approximately 95 macro functions.

The macro functions rely upon one basic assumption: Values are moved in and out of the SAS dataset one observation at a time. If the designer wishes to work with multiple observations, this must be set up using specially-written code. Internally, data management tasks are performed using the MODIFY statement, which alters a dataset in place, which preserves audit trails.

Standard approaches for each operation are:

- **Addition:** Values for a new case are entered into the web page and returned to SAS. The SAS dataset is checked for unique values of BY variables, and the observation is added.
- **Editing/deletion:** BY variables are used to select one case for editing/deleting. Values are obtained and presented on the output web page. The user sends back instructions and modified data (store modified values/delete case).
- **Listing:** Observations from the entire dataset are listed.

TRANSFER OF VARIABLE VALUES TO AND FROM SAS

In using SAS/IntrNet, the web browser sends back "name-value" pairs from the web page to the SAS/IntrNet Broker®. The data are input into SAS as macro variables (the HTML name is the macro name and the HTML value is the macro value). The WDES macro functions for data entry use these macro variables to add or modify observations in the SAS dataset.

A parallel process is used to send values from SAS back to the web browser. Within the SAS dataset, variable values for the selected observation are converted into values of macro variables. These are then written to the output web page using the resolve method discussed below. To avoid confusion in the processing of values, one set of macro variables (with one macro variable per SAS dataset variable) is used to get values from the web back to SAS and a second set is used to put values from SAS back to the web. In addition, a third set of variables is used for the formatted values of the variables. This ensures that no confusion is made between values input to the SAS dataset and values output from the SAS dataset. This macro-variable based approach (for both input and output of dataset values) enables most (if not all) pages to be written without the use of character functions.

MACRO METADATA VARIABLES

Macro variables communicate metadata information to the WDES macros. Metadata macro variable names are composed of two parts: a short name for the SCAD, used as a prefix, and the main portion for the metadata name, used as a stem or base for the full name. The entire set of macro metadata variables is produced automatically using the "Project Control" system.

The macro metadata variables solve an important problem: How can code be written to enable several different macro variables to be resolved from a single invocation? The approach taken defines the

macro variable with two parts, one which does not vary (the stem or base) and one which varies at each call (the short name for the SCAD). Thus, the final resolution, which takes place after the macro function is called, results in multiple macro variable names after macro resolution, although only a single macro name is printed in the macro function.

The dataset name information macro stem is "xmset." If the short name for a SCAD was "a," the full macro for the dataset name for this SCAD is "_axmset." During the processing when this macro is used, the value for _ltr is passed to the macro function through the macro call. The name can be recovered using the construction &&_&_ltr.xmset. If:

```
_ltr=a, _axmlib=MAINLIB, _axmset=SASSET
```

Thus, a SAS statement such as:

```
DATA &&_&_ltr.xmlib..&&_&_ltr.xmset;
```

would be used, and then would resolve to:

```
DATA MAINLIB.SASSET;
```

Two periods in the macro reference statement are required.

KEY NOTIONS FOR WDES

The `resolve` function: When preparing a page for publication, lines must be formed up and published to either a static or dynamic page. Thus, some approach must be used to vary the content of lines published, based on SAS information. In WDES, a simple and elegant method is used for source modification.

During the page publication process, lines are read from pagelet files. These lines are set up during SCAD preparation with SAS macro references (i.e., &_ref) set in places in which varying information is to be placed. The SAS resolve function is used to substitute the macro value for the macro reference at the page publication time. If we are reading the line SLINE from an input source, and publishing the line OLINE, we would use this code:

```
DATA _NULL_;
INFILE PAGELTS LRECL=240 PAD; FILE _webout;
LENGTH OLINE SLINE $1000;
INPUT SLINE $CHAR200.;
OLINE=RESOLVE(SLINE); LLONG=LENGTH(OLINE);
PUT OLINE $VARYING1000. LLONG;RUN;
```

This method forces the resolution of macro reference in variable SLINE and places the result in variable OLINE. Thus, if variable SLINE includes macro references, values for those references would replace the references as the output variable OLINE is set up. This method can be used to perform the following tasks (and others as well):

- set up feedback messages following tasks; and
- place existing values into the HTML page for editing.

There are several important advantages to using this method. First, the HTML code is very easy to work with. This avoids the use of multiple levels of quotations and other string-manipulation tools. The code is merely written as HTML code with the added SAS macro references. Thus, these pages are very easy to read and maintain. Second, the HTML page can be changed flexibly. The sole requirement is that the same macro references must be used. The macro references may be moved around or relocated without great problems, as the resolve function will place the information wherever the reference is found.

Here are two examples of this method.

1. If he following lines were stored in a pagelet file:

```
<H1>&_pagetitle</H1>
<H2>&_operreport</H2>
```

and if values for these macro variables are:

```
_pagetitle= Patient Registration
_operreport = Edit operation successful
```

then passing the lines above through the resolve statement would result in the following lines being sent to the final output:

```
<H1>Patient Registration</H1>
<H2>Edit operation successful</H2>
```

In the actual text of the decision process, we would select the value for _operreport based on operation results.

2. Lines involving form elements can also have preset macro references:

```
<INPUT TYPE="TEXT" NAME="_m" VALUE="&_xm">
```

and if _xm=Tylenol, processing with the resolve function would result in the following text:

```
<INPUT TYPE="TEXT" NAME="_m" VALUE="Tylenol">
```

Pagelet files: When maintaining files of code to be published, other difficulties still remain.

1. Similar information is to be published for several locations
2. Clinic 1 needs the Clinic 1 patient list, while Clinic 2 should not see these patients.
3. The Clinic 1 patient list (but not the Clinic 2 list) must be rewritten when a new patient is added.
4. When editing existing information, BY variables should not be edited. If this were allowed, it becomes more difficult to preserve dataset integrity.

One simple solution lies in the use of HTML pages divided into useful pieces, which can be separately modified or altered. In WDES, these sections are termed "pagelets". Each pagelet is a section of an HTML page which is structurally and conceptually self-contained. A collection of pagelet files are published sequentially to form the final page.

This approach to page construction has useful implications:

- Lists of patients can be stored in pagelet files which contain the clinic number as a part of the name. Thus, the SAS macro technology can be used to **automatically select** the proper file, by including the macro reference in the selection of the pagelet file.
- Pagelet files, which represent information used in many places, can be stored in a shared subdirectory. Thus, patient lists are stored in a subdirectory called shared, which is available to all other SCADs.
- ID variables are placed into "ID pagelets," while normal non-ID variables are placed into "variable" pagelets. Thus, when editing, the ID pagelets are presented in "printing" mode, while standard data are presented as INPUT form items.

For instance, many clinic name lists may be maintained as SELECT lists _list1, _list2, etc. These may be modified whenever a new case is added to the given clinic. If the pagelet list includes a specification such as _list&_clin, the correct clinic list is selected when the value of _clin is set.

``Pre-slugging" RADIO and SELECT items: When presenting information for editing, TEXT items are easy to work with, as the existing value can be placed into the VALUE tag, as was shown above in the resolve function section. The situation is more complex with RADIO, SELECT, and CHECKBOX items. Here, the

edit process must begin with current value pre-selected. This is done using a "pre-slug" technique. For each RADIO, SELECT, or CHECKBOX item, the form is set up with macro references by the "Project Control" system. During the publication process, the values for the special macro variables are set to either CHECKED or empty. Thus, the item which has CHECKED will be selected on the HTML form which is sent to the browser.

Consider two RADIO buttons:

```
<INPUT TYPE="RADIO" NAME="_sex" VALUE="1"
&_ysex1>Male
<INPUT TYPE="RADIO" NAME="_sex" VALUE="2"
&_ysex2>Female
```

Assume that the value of _sex=1. Prior to the publication process involving the resolve statement, the following statement would be processed:

```
%let _ysex&_sex=CHECKED;
```

Macro substitution leads to:

```
%let _ysex1=CHECKED;
```

The publication process with the resolve statement results in:

```
<INPUT TYPE="RADIO" NAME="_sex" VALUE="1"
CHECKED>Male
<INPUT TYPE="RADIO" NAME="_sex" VALUE="2">
Female
```

which would lead to the presentation with the Male button checked when the edit screen is presented.

LIMITATIONS OF THE WDES

The WDES has disadvantages. The system is a macro-based system, which can be somewhat confusing at first. The code for the selection macro is difficult to read. It involves little standard SAS code. Thus, there is a learning curve associated with the use of the system.

The system is oriented toward HTML. Thus, the system is very good at creating an HTML form which is modeled after the printed form. The system is not oriented toward the production of forms from SAS items. Thus, when variables in different datasets are defined, care must be taken to use the same specifications for the same variables in different datasets. WDES has a special technique for defining common variables in a shared file.

CONCLUSION

The WDES is a system for the development and maintenance of web pages. It is based on metadata storage of information with processing performed by macro functions. The metadata information is also maintained in macro variables, to enable the system to access it very quickly and simply.

Development is driven by a driver HTML file. Initial processing of the file extracts variables, which are further defined by the user using a simple interface, in which all components of the variable are modified simultaneously. Final processing produces a modified HTML file and all supporting code to enable the file to be immediately used in a data entry context. The processing also produces the code to support processing of data for basic tasks.

Normal processing is supported by macro functions, stored in a compiled macro catalog. A macro for task selection is used that is written by the system. Since the task selection is done by macro, customization of the tasks is quite straightforward. The data processing macros use the macro metadata information.

The WDES is currently the production system for two multicenter clinical trials. As such, it is not a potential system, but a working SAS solution. While there is a learning curve associated with the system, the payoff lies in the production of a data management system, rather than a collection of disparate unconnected and separate code tools. The system is further supported by several manuals (Thompson, 2002a, 2002b).

REFERENCES

Knuth DE (1984} Literate Programming. *The Computer Journal, 27*, 97-111.

Thompson PA (2002a) *Reference Manual, Web Data Entry System, Version 1.50*. St. Louis MO: Division of Biostatistics, Washington University School of Medicine.

Thompson PA (2002b) *User's Guide to the Web Data Entry System, Version 1.50.* St. Louis MO: Division of Biostatistics, Washington University School of Medicine.

ACKNOWLEDGMENTS

Preparation of this article was supported by EXCITE (Extremity Constraint-Induced Therapy Evaluation - Grant # IR01 HD37606) and CRISP (Consortium for Radiologic Imaging Studies of Polycystic Kidney Disease, Grant # 5U01 DK56961). SAS, SAS/IntrNet and SAS/IntrNet Broker are registered trademarks of SAS Institute, Inc in the US and other countries. ® indicates USA registration. Microsoft Word is the registered trademark of the Microsoft Corporation. ® indicates USA registration.

CONTACT INFORMATION

Contact the author at:

Name: Paul A. Thompson, Ph.D.
Associate Professor, Division of Biostatistics
Address: Washington University School of Medicine
Campus Box 8067, 660 S. Euclid St.
St. Louis, MO 63110
Telephone: (314) 747-3793
Fax: (314) 362-2693
E-mail paul@wubios.wustl.edu