

National Committee for Quality Assurance HEDIS 2.5 Data Submission Forms: A Subclassed Extended Table Approach.

Joe Whitehurst, Whitehurst Associates, Inc.

Abstract

Freimuth and Yindra presented a SUGI 21 paper entitled, "Receiving and Distributing Data Nationally: A SAS® Solution to HEDIS Reporting". In that paper they described how the SAS System was used to develop a client-server solution to HEDIS reporting. In this presentation, I will demonstrate a system with similar functionality but which incorporates some of the SAS System's Object Oriented Programming techniques for creating the user interface. Specifically, I will show how a subclassed Extended Table permits the creation of all of the data entry/display FRAME entries without requiring any code specific to each FRAME, i.e., none of the data entry/display FRAMES have an associated SCL entry. This greatly simplifies the process of maintenance as well the development of new data entry/display FRAMES.

Background

The National Committee for Quality Assurance (NCQA) is an independent, not-for-profit organization dedicated to assessing and reporting on the quality of managed care plans, including Health Maintenance Organizations (HMOs) and Managed Behavioral Health Organizations (MBHO's). NCQA is governed by a Board of Directors that includes employers, consumer and labor representatives, health plans, quality experts, regulators, and representatives from organized medicine. NCQA's mission is to provide information that enables purchasers and consumers of managed health care to distinguish among plans based on quality, thereby allowing them to make more informed decisions. Currently, NCQA's efforts are organized around two separate activities, accreditation and performance measurement (report cards), which are complementary strategies for producing information to guide choice.

Approximately 50% of the nation's 547 HMOs have been reviewed as part of the accreditation process and recently NCQA released draft standards for accreditation of MBHOs for public comment. Performance

measurement activities began in 1989 with efforts by employers and health plans to specify an integrated set of uniform, standardized performance measures that could be used to document the quality and value of health plans and eventually allow rational comparisons among competing plans. In 1993 these efforts resulted in the release of what is known as the Health Plan Employer Data and Information Set (HEDIS 2.0/2.5)--a set of more than 60 integrated performance measures that has now become the industry standard. In 1994, NCQA launched the Report Card Pilot Project in collaboration with 21 health plans and key employer, consumer, health policy and labor representatives to test the feasibility of implementing a system of standardized performance measures that could provide timely information to purchasers, consumers, health plan executives, and others regarding the quality of care and service in managed care plans. The success of this pilot project led the NCQA to conclude that performance measures that are rigorously produced, audited and displayed in common formats provide useful information on health care performance (NCQA WEB Site, 1996).

HEDIS Report Card Application

Because of the variability of existing reimbursement and compensation structures and the wide range of administrative and financial information systems that are the source data for most of the HEDIS measures, most health plans face significant challenges in producing HEDIS Report Cards that conform to evolving HEDIS specifications. Freimuth and Yindra (1996) have already documented the use of the SAS System to develop a comprehensive reporting system that conforms to the HEDIS specifications for a large health care insurer. They describe in detail the development process, including all the information systems challenges they encountered, "beginning with batch programs running on the mainframe against sequential files, and evolving into a flexible adhoc reporting system running under a PC-

based GUI interface, accessing data from local and host data stores, and distributing via e-mail " (Freimuth and Yindra, 1996 p. 711).

Whitehurst Associates, Inc., (WAI) was contracted by another large health care insurer to develop a system with similar functionality. Because this large health care insurer had already begun the process of establishing a Health Care Information Warehouse which could serve as a source for much of the data required to produce the HEDIS measures, the focus of WAI's efforts was on the development of an application that manages the process of creating the HEDIS Report Cards. The creation of this application is the topic of this paper.

Some Technical Specifications

In addition to the HEDIS 2.5 specifications (updated HEDIS 2.0 specifications with no new measures nor any deleted) which define HEDIS measures in detail, NCQA also provided detailed descriptions and pictures of a set of data submission forms designed to standardize the submission of HEDIS data. These forms provide health plans with a standard format for data entry (for data obtained from sources other than the Warehouse or to correct data obtained from the Warehouse) and submission, but do not define HEDIS measures in detail—the detail is in the HEDIS 2.5 specifications

What the Project Director Demanded

In discussions with the Project Director, WAI learned that she wanted an application with data entry/display screens that mirrored as closely as possible, the actual NCQA supplied forms (approximately 60 screens now with many more to come). She also wanted these data entry/display screens to allow recalculation of row and/or column totals, the computation of measures from two or more data items, the automatic validation of each cell to ensure that it could only contain valid entries, and the creation of an audit trail that recorded every addition and/or change to any data. In addition, the screens must allow the words "SUPPRESSED" to appear in any particular cell of a form, footnotes to be entered that are tied to the data for a particular network,

and an unlimited amount of free form narrative text to be entered or pasted from the clipboard and spell checked that is also tied to the data for a particular network. All these data, including which cells have been suppressed, must be stored on and accessed from a mainframe by multiple, concurrent users with edit/update authority running an application on LANs scattered around the country. The Project Director also requested that users be able to export/print any subset of the HEDIS measures to spreadsheets (or any other format including SAS Datasets) and to choose for each measure whether to export/print footnotes and/or the narrative text. The Project Director insisted also that there be no redundant code, that is to say, for any given processing, the same code should perform the processing whatever the context. Finally, the Project director wanted all this functionality to be controlled by pull-down menus and/or mouse clicks on a GUI.

What WAI Did TO Satisfy the Demands of the Project Director

After learning that the client was currently using version 6.08 on both the mainframe and PC platforms, WAI recommended that the client install version 6.11 on the PC in order to exploit the new FRAME objects for data display/edit. Since this could not be done company wide, WAI undertook the task of developing the application using version 6.10 which was scheduled to be installed.

Although the project's severe time constraints argued for a Rapid Application Development (RAD) iterative approach to the design and coding of the application, maintenance considerations suggested that exploitation of the SAS System's object-oriented programming capabilities might yield valuable advantages. Inspection of the NCQA data submission forms revealed that, while the layout of the forms differed widely in terms of numbers of columns and/or rows of data, the kinds of calculations and data validations to be performed, and the placement of row and column totals, all the forms consisted of rows and columns of data. So, although time constraints would not permit the kinds of analysis and design work typically associated with an object-oriented approach to application

development, significant advantages could still be realized by using some of the object-oriented programming techniques available in the SAS System. Prior to version 6.11, the FRAME Extended Table Class was the most versatile object for creating a data display/entry object for rows and columns of data. Accordingly, WAI recommended that the application be developed using a subclassed Extended Table. What follows is a description of how a subclassed FRAME, a subclassed Extended Table and 5 subclassed Text Entry widgets were used to provide all the functionality demanded by the Project Director. None of the FRAMEs for displaying/entering data have an associated SCL entry. All of these FRAMEs share the same set of methods associated with the sub-classes. To avoid dealing with 60 (and growing) sets of variable names in the SCL methods, WAI decided to write all the SCL code using SCL list references. All the variable names, row and column identifiers, scrolling and paging information, and suppression options for each measure were stored as SLISTS and loaded as needed. Additionally, all the data items were loaded into SCL lists as soon as they were retrieved from the mainframe. Thus all the code for creating row and column totals, calculation of new values, and displaying data items involved references to list items rather than SCL variables. Below are examples of such code:

Moving Data From a SAS Dataset To an SCL LIST

```
do while(fetch(statdid)^=1);
  do l=1 to listlen(varnames_lid);
    rc=setnitemc(getniteml(cnt_lists_lid,
      getitemc(varnames_lid,i)),
    put(getvarn(statdid,varnum(statdid,
      getitemc(varnames_lid,i))),
      best10.),rowid);
  end;
end;
```

Calculating Columns of Ratios

```
if input(getnitemc(members_lid,
  getitemc(rowid_lid,j)),comma10.)^=0
then do;
  calcr=setnitemc(getniteml(calc_lists_lid,
    getitemc(calnames_lid,i)),
    put(1200*(input(left(getnitemc(
```

```
getniteml(cnt_lists_lid,
  getitemc(varnames_lid,i)),
  getitemc(rowid_lid,j))),comma10.)/
input(getnitemc(members_lid,
  getitemc(rowid_lid,j)),comma10.)),
  10.2),getitemc(rowid_lid,i));
```

end;

Refreshing an Extended Table Using SCL Lists

```
if 0<itemno<=listlen(rowid_lid) then do;
  do i=2 to listlen(widgetid_lid);
    widgetid=getniteml(widgetid_lid,i);
    if getnitemc(widgetid,'NAME',1)=
      getnitemc(special_colnames_lid,
        getnitemc(widgetid,'NAME',1),1,1,'0') then do;
      if
        getnitemc(widgetid,'NAME',1)='MEMBERS'
      then call send(widgetid,'_set_text_',
        getitemc(MEMBERS_LID,itemno));
      else if getnitemc(widgetid,'NAME',1)=
        'DESCRIPT' then call
        send(widgetid,'_set_text_',
          getitemc(descript_LID,itemno));
        filled_widgets+1;
      end;
    else do;
      j=i-filled_widgets;
      if j<=listlen(colnames_lid) then do;
        name=getitemc(colnames_lid,j+
          hscroll_value);
        if name='VAR' then do;
          if getnitemc(getniteml(cnt_lists_lid,
            getnitemc(varnames_lid,name)),
            getitemc(rowid_lid,itemno),
            2,1,'NONE')=
            'SUPPRESSED' then do;
            call send(widgetid,'_set_text_',
              getnitemc(getniteml(cnt_lists_lid,
                getnitemc(varnames_lid,name)),
                getitemc(rowid_lid,itemno),2));
            end;
          end;
          else if name='CALC' then do;
            if getnitemc(getniteml(calc_lists_lid,
              getnitemc(calnames_lid,name)),
              getitemc(rowid_lid,itemno),2,1,'NONE')=
              'SUPPRESSED' then call
            send(widgetid,'_set_text_',
              getnitemc(getniteml(calc_lists_lid,
                getnitemc(calnames_lid,name)),
                getitemc(rowid_lid,itemno),2));
            else call send(widgetid,'_set_text_',
```

```

        getnitemc(getniteml(calc_lists_lid,
        getnitemc(calcnames_lid,name)),
        getitemc(rowid_lid,itemno),1));
    end;
end;
end;
end;

```

Sending SCL List Data to EXCEL Via DDE

```

do j=begin to end;
  name=getitemc(colnames_lid,j);
  if name=:'VAR' then do;
    used_columns+1;
    if getnitemc(getniteml(cnt_lists_lid,
    getnitemc(varnames_lid,name)),
    getitemc(rowid_lid,i),2,1,'NONE')=
    'SUPPRESSED' then do;
      rc=fput(excelprt_fid,'SUPPRESSED');
      rc=fput(excelprt_fid,'09'x);
    end;
  else do;
    rc=fput(excelprt_fid,
    getitemc(getniteml(cnt_lists_lid,
    getnitemc(varnames_lid,name)),i));
    rc=fput(excelprt_fid,'09'x);
  end;
end;
else if name=:'CALC' then do;
  used_columns+1;
  if getnitemc(getniteml(calc_lists_lid,
  getnitemc(calcnames_lid,name)),
  getitemc(rowid_lid,i),2,1,'NONE')=
  'SUPPRESSED' then do;
    rc=fput(excelprt_fid,'SUPPRESSED');
    rc=fput(excelprt_fid,'09'x);
  end;
  else do;
    rc=fput(excelprt_fid,
    getitemc(getniteml(calc_lists_lid,
    getnitemc(calcnames_lid,name)),i));
    rc=fput(excelprt_fid,'09'x);
  end;
end;
end;
rc=fwrite(excelprt_fid,' ');

```

Creation of the Subclasses

Creation of the subclasses is straightforward and well documented on pages 91-119 in "SAS/AF® Software FRAME Application Development Concepts" First Edition. Information found there will not be repeated here. Instead, I will describe for you some of the enhanced functionality of the subclassed

Extended Table-- horizontal scrolling, for instance--and the methods used to achieve some of this enhanced functionality. In addition, in the presentation, I will demonstrate for you how subclasses are used to create a new data display/entry screen for a new HEDIS measure--so you know it must take less than an hour.

The Subclassed Extended Table

Pages 2-3 in the Extended Table Class section of "Object-Oriented Programming with the SAS System" documents the flow of control in an Extended Table when its associated SCL program executes:

1. The INIT section runs.
2. The GETROW section runs.
3. The program waits for user input.
4. If a user modifies a row in the table, the PUTROW section runs for that row, the MAIN section runs, and the GETROW section runs again to display the table with the modified value.

So what happens when there is no SCL program associated with the extended table? Nothing very interesting, unless one or more of the set of methods supplied for the Extended Table Class are overridden. In fact, several of these supplied methods are specifically designed to be overridden and currently do nothing at all. Overriding a method, in the simplest case, merely means supplying some alternative code for the method which executes in addition to or instead of the method's default code. For examples, the `_GETROW_` and `_GETROW_LABEL_` methods can be overridden with code for a user-defined method that completely replaces or only extends the GETROW labelled section in a SCL program. Similarly, the `_PUTROW_` and `_PUTROW_LABEL_` methods can also be overridden with code for a user-defined method that completely replaces or extends the PUTROW labelled section in a SCL program. The code above in the section, **Refreshing an Extended Table Using SCL Lists**, is a simplified version of the HEDIS Report Card Application code that overrides the `_GETROW_` method our subclassed Extended Table inherited from the Extended Table Class. This method will be called automatically by any extended table widget created from our subclassed

Extended Table using the same rules the Extended Table Class uses to call the GETROW label. By using SCL list references for all expressions as opposed to SCL variables and creating a subclassed Extended Table, only one block of code is required for refreshing this application's Extended Table widgets no matter how many instances of our subclassed Extended Table we instantiate. So, if circumstances ever change in ways that require our Extended Tables to do something differently, we only need to change our custom methods which reside in only one place.

Enabling Horizontal Scrolling for an Extended Table

Prior to version 6.11, the horizontal scroll bars optionally included with an Extended Table could not be made to function. Since displaying some HEDIS measures requires more columns than will ever fit on one screen, WAI enabled horizontal scrolling on the subclassed Extended Table. Well, not exactly. What WAI actually did is subclass the Horizontal Scroll Bar Class and override its `_Select_` method with the following code:

```
envlist_id=envlist('G');
table_id=getnitemn(envlist_id,'TABLE_ID');
label_values_id=getnitemn(envlist_id,
'LABEL_VALUES_LID');
frame_id=getnitemn(_self, '_FRAME_');
call send(frame_id, '_get_widgets_',
frame_widgets_id);
call send(_self, '_get_value_', hscroll_value);
rc=setnitemn(envlist_id, hscroll_value,
'HSCROLL_VALUE');
incsize=getnitemn(_self, 'INCSIZE');
rc=setnitemn(envlist_id, incsize, 'INCSIZE');
do i=1 to listlen(label_values_id);
  call send(getnitemn(frame_widgets_id,
nameitem(label_values_id, i)),
'_SET_TEXT_',
getnitemc(label_values_id,
nameitem(label_values_id, i),
(hscroll_value/incsize)+1));
end;
call super(_self, '_select_');
```

Without getting too much into the details of the code, it should be apparent that we are using the method to query the horizontal scroll bar widget regarding its horizontal position (`hscroll_value`) and increment size and using this information to

send the appropriate text column descriptions to the text entry widgets on the currently active FRAME. This same value (`hscroll_value`) is passed to the routine that refreshes the extended table widget and used to increment an index that determines which column of data will be displayed where. But it looks like horizontal scrolling to me and the Project Director.

Conclusions

WAI must agree with Freimuth's and Yindra's 1996 assessment: "The SAS System met the challenges of HEDIS reporting on many different fronts." With the just released HEDIS 3.0 specifications running over 400 pages, the clear intent of NCQA to bring the nations' 100 million members of MBHOs under the HEDIS reporting umbrella, and all the other changes afoot, HEDIS reporting is definitely a fast moving target. The SAS System is just what the information systems Director's doctor ordered for this developing crisis—an integrated, platform independent, software system capable of reading, writing, managing, and analyzing data and presenting information in any format whatsoever. Add object-oriented applications development programming techniques and WAI will bet on it.

ACKNOWLEDGEMENTS

SAS, SAS/AF, are registered trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration

References

Freimuth, Sue & Yindra, Chris. "Receiving and Distributing Data Nationally: A SAS Solution to HEDIS Reporting", Proceedings of the Twenty-First Annual SAS Users Group International Conference, Chicago, Illinois, 1996, pp. 711-717.

NCQA WEB SITE: <http://www.ncqa.org>

SAS Institute Inc., SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1995

SAS Institute Inc., SAS/AF Software: FRAME Application Development Concepts, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1995.

Your comments, questions, and suggestions
will be welcomed by the author.

Joe Whitehurst, Whitehurst Associates, Inc.
341 Eureka Drive
Atlanta, Ga. 30305
(404) 240-0729
joewhite@mindspring.com