MWSUG 2012 – Paper PH14

Using SAS® Software to Aid in the ICD-10 Code Set Implementation

Alexander Pakalniskis, M.S., Cedars-Sinai Medical Center, Los Angeles, CA
Nilesh Bharadwaj, B.S., Cedars-Sinai Medical Center, Los Angeles, CA
Caren Jones, MPH, Cedars-Sinai Medical Center, Los Angeles, CA
Alein T. Chun, Ph.D., Cedars-Sinai Medical Center, Los Angeles, CA

ABSTRACT

Created by the Data Quality Management Unit (DQMU) of the Cedars-Sinai Medical Center (CSMC) Resource and Outcomes Management (ROM) Department, the ICD-9 to ICD-10 Conversion Utilities (IICU) support the controlled implementation of the Department of Health and Human Services (HHS) mandate to replace ICD-9-CM/PCS codes with ICD-10-CM/PCS codes. Designed to identify all programs and owners affected by this impending data change, the tools help ensure ongoing report quality, reduce the time and effort required for change implementation, and provide transparency for project management.

INTRODUCTION

Cedars-Sinai Medical Center (CSMC), located in Los Angeles, California, is one of the largest private academic hospitals in the Western United States. With nearly 1,000 beds, this medical facility's annual patient volume is more than 50,000 inpatient admissions and over 450,000 outpatient visits. Its work force comprises about 12,000 employees, 2,100 physicians on the voluntary staff and 300 in the clinical faculty.

As part of its portfolio of responsibilities, the Resource and Outcomes Management (ROM) Department at CSMC provides *ad hoc*, routine, and scheduled reporting on patient care quality and efficiency, as well as support for data quality management issues. Much of this work utilizes the CSMC-wide Data Warehouse (DW), a large relational database that contains clinically-oriented administrative data at the patient level. Most ROM reports originate with SAS-embedded SQL queries to the DW.

Currently, when a patient is treated at our hospital, diagnoses are codified using the International Classification of Diseases, Ninth Revision, Clinical Modification (ICD-9-CM). This code system is based on the official version of the World Health Organization's (WHO) ninth revision of the International Classification of Diseases. WHO designed this coding system so that a patient's morbidity and mortality information could be classified for statistical purposes.

In the United States, remuneration for services provided in a clinical setting is heavily dependent on how medical records of a patient are coded. Since WHO never developed a comparable code set for medical procedures, the Centers for Medicare & Medicaid Services (CMS) developed and maintains this coding system, referred to as ICD-9-PCS (Procedure Coding System).

In January of 2009, the Department of Health and Human Services (HHS) mandated that ICD-9-CM/PCS codes be replaced with ICD-10-CM/PCS codes by October 1, 2013. While this deadline has been pushed off to October 1, 2014, it is known that there will be a major impact place on many aspects of quality clinical and public health reporting once this recodification takes. The changes will certainly impact patient medical record coding and monitoring at our hospital. The department which maintains contracts with medical insurers and healthcare providers will be affected in a major way. As SAS analysts and programmers, we realized that we needed to prepare for this event so that on October 1, 2014 our programming and reporting would be correct and complete both for periods both prior to and following this cutoff date.

The expansion of code length from five to seven alpha-numeric characters in ICD-10 aims to parallel advancements in medicine made since the 1970's, when ICD-9 was introduced. Expanded CM codes will enable greater specificity in identifying disease etiology as well as anatomic site and severity, while expanded PCS codes will allow for identifying the body system, the encounter root operation, the affected body part, the approach used to provide the procedure, and the various devices used in the procedure. This greatly increased specificity will enhance claims processing and reimbursement, as well as health care data analytics (cost, utilization, outcomes). These enhancements in turn will greatly improve operational efficiency and analytics processing.

IICU

[In order to simplify the understanding of how the IICU work, all SAS programs discussed herein along with any input and output datasets used, are housed in a subfolder of the C Drive named ROMR7511. Furthermore, the discussion is limited to diagnoses. The contents of this subfolder are available upon request.]

We recognized that, due to the unprecedented level of data change involved in this conversion, an automated tool would be required to identify all programs affected by impending changes in DW table structure and definition. This was the primary impetus for the creation of IICU. As a first task, we needed to determine how much work would arise in our department to accommodate for this coding transition. More specifically, we needed to be able to plan ahead and create a workable project plan for managing this transition based on priority, complexity, and available resources. We also needed to be able to communicate a timeframe for this work, structuring it into logical phases.

Multiple additional benefits were realized as a result of the development of IICU. Using these utilities, an email notification can be automatically sent to programmers identifying all their programs affected by data change and providing useful information regarding the change. Additionally, management now has the ability to evaluate data change impact and status at individual programmer, departmental, and project levels. IICU also provides ongoing support for data quality control and change management.

In order to determine the complexity of program modifications resulting from ICD-9 to ICD-10 conversion, we needed an inventory of programs currently creating clinical reports for our user community. Such an inventory, called the Programmer Documentation Table (PDT), already existed and as a SAS file. This inventory contains information such as programmer name, program name, program network location, and program description. For the purposes of illustration, we will consider one programmer's (Elmer Fudd's) program set consisting of SQL-embedded SAS programs. Let us suppose that this PDT contains 7 records associated with Programmer_Name = Elmer Fudd. Let us suppose that one of the 7 records looks like the record described in Figure 1.

Variable	**Example**
Programmer_Name	Elmer Fudd
Program_ID	ROMR2237
Program_Folder	\\cshsmedaff \ROMR2237\
Program_Name	ROMR2237.sas
Program_Description	Grouping Table for Classic DRG & APR groupings

Figure 1. *PDT Variables*

UTILITY 1

The first utility in the IICU toolset determines which of Elmer Fudd's 7 programs uses ICD-9 diagnosis codes. To make this determination, the utility accesses VISIT_DIAGNOSIS, the DW table housing references to ICD-9 diagnosis codes, organized by patient encounter.

Traditionally, SAS is viewed as a good tool for data analysis and reporting. However, it is not limited to just that. There are novel uses of SAS, such as the ability to parse other SAS programs using string manipulation. This first Utility will utilize that ability.

The three key pieces of information for this first utility are Programmer_Name, Program_Folder, and Program_Name. The following SAS code will determine how many of the 7 programs attributed to programmer Elmer Fudd mine the pertinent ICD-9 relational database table:

```
libname PROGS "c:\\ROMR7511\";

data TEMP1;
 set PROGS.PDT;
if  Programmer_Name='Elmer Fudd';
run;

data   _NULL_;
 set   TEMP1;
call symput('n',trim(left(input(put(_n_,5.),$5.))));
length progrm $255;
progrm=upcase(trim(left(Program_Folder)))||upcase(trim(left(Program_Name)));
call symput('prog'||trim(left(input(put(_n_,5.),$5.))),trim(left(progrm)));
run;
```

At the end of this SAS data statement, there will be 8 macro variables created: &n will have the value of 7 (the number of records in SAS data file PROGS.PDT) and &prog1 through &prog7 will contain all 7 program folder locations. We can now analyze each of the 7 programs sequentially using the value of &n to course through the list of programs:

```
%macro ut1;

%do i = 1 %to &n;
    filename r&i "&&prog&i" lrecl=255;

    data temp&i(keep=prog);
    length prog $255;
    infile r&i truncover;
    if  _n_=1
        then prog=" ";
    input @1  lin   $char255.;
    if  index(upcase(lin),"VISIT_DIAGNOSIS") not = 0
        then prog="&&prog&i";
    if  prog not = " ";
    run;
%end;

data tempall;
 set %do i = 1 %to &n;
         temp&i
     %end;
     ;
run;

%mend ut1;

%ut1;
```

In the above-listed macro, each of the 7 (&n) programs is read line by line to determine whether the token "VISIT_DIAGNOSIS" is found. If this token is found, that program name is written out to the data set TEMPALL. It turns out that Elmer Fudd is responsible for 3 programs that mine the VISIT_DIAGNOSIS table in CSMC's DW.

UTILITY 2

If SAS programs simply accessed relational database tables to list variable values, Utility 1 would provide sufficient support for ICD-9 to ICD-10 conversion. ICD-9 identified SAS programs could be automatically rewritten by replacing old ICD-9 table and field names and descriptors with new ICD-10 information.

Unfortunately, most SAS programs not only access variables, but also specify which variable values are of interest, for example, records of patients with some form of diabetes. Such a SAS program might be mining the VISIT_DIAGNOSIS table for ICD-9 codes 250.1, 250.2, and 366.41. Because these variable values will be changing, frequently without a one-to-one mapping, it is these SAS programs which must be identified, the level of conversion complexity evaluated, and every effort made to minimize manual conversion, not only to save programming time, but also to increase programming quality.

Having identified 3 programs for which programmer Elmer Fudd is responsible and which contain reference to the table VISIT_DIAGNOSIS, we needed the ability to extract "PROC SQL" statements and parse the "WHERE" clause, if present, to identify which ICD-9 codes are mined.

The 2009 Annual Conference Proceedings of Western Users of SAS Software featured an article entitled "An Automated Tool for Data Change Implementation" which shows how to implement this parsing. This very parsing tool is used as Utility 2 to identify, by affected SAS program, which ICD-9 codes are mined in each program.

UTILITY 3A

CMS maintains a web site in which many helpful transition aids are available. One section in this web site (http://www.cms.gov/Medicare/Coding/ICD10/2012-ICD10-CM-and-GEMs.html) provides downloads and support for translating (producing "General Equivalence Mappings") from ICD-9 DX to ICD-10 DX and vice versa. In particular, the 2012 Diagnosis Code Set General Equivalence Mappings (GEM) download provides two files (2012_I10gem.txt and 2012_I9gem.txt) which can be used to build a data base of DX translations. Both GEM files are dated December 2011 on the CMS web site. Three other files in this area provide documentation on how these ".txt" files should be used.

(Similarly, another CMS web site (http://www.cms.gov/Medicare/Coding/ICD10/2012-ICD-10-PCS.html) provides downloads and support for translating from ICD-9 PX to ICD-10 PX and vice versa. Creating a PX translation data base follows the same steps and similar coding as for the DX translation data base.)

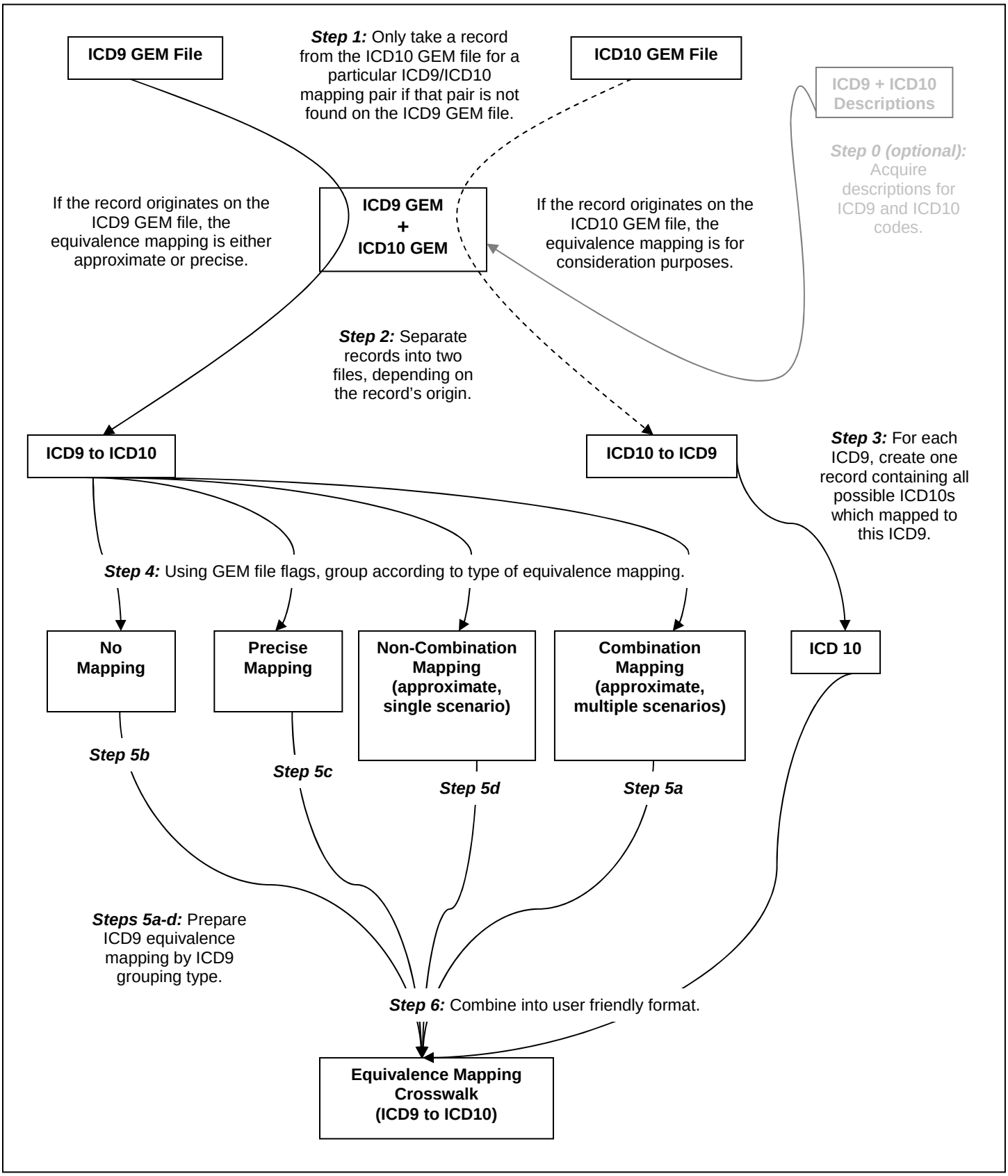

Figure 2: *Process Flow Chart for Utility 3A (ICD9 DX to ICD10 DX)*

We first read in two SAS files, one containing ICD-9 CM code and description and the other containing ICD-10 CM code and description:

```
/* Step 0; optional */
libname mayn "c:\\ROMR7511\";

proc sort data=mayn.tendescsdx nodupkey;
  by icd10_cm_code;
run;

proc sort data=mayn.ninedescsdx nodupkey;
  by icd9_cm_code;
run;
```

Next, all seven ICD-9 GEM file variables were mined from the ".txt" files downloaded from the CMS web site.

```
/* Step 1a */
data nine2tendx(keep=icd9_cm_code icd10_cm_code
                     approximate_flag no_map_flag combination_flag
                     scenario choice_list directo);
length directo $8;
infile "c:\ROMR7511\2012_I9gem.txt" lrecl=19;
input
       @001  icd9_cm_code              $char5.
       @007  icd10_cm_code             $char7.
       @015  approximate_flag          $char1.
       @016  no_map_flag               $char1.
       @017  combination_flag          $char1.
       @018  scenario                  $char1.
       @019  choice_list               $char1.
       ;
directo='Forward';
run;
```

The ICD-9 GEM file (downloaded from the CMS web site) contains 23,912 records and shows the ICD-9 DX Code with its ICD-10 DX Code equivalent. The other variables indicate whether the translated equivalent is precise (or approximate), whether the ICD-9 DX Code has (or does not have) an ICD-10 DX Code equivalent, whether the ICD-9 DX Code's translated equivalent is a combination of ICD-10 DX Codes, whether there might be several equivalent ICD-10 DX Code combinations (scenarios) for an ICD-9 DX Code, and whether a translated ICD-10 DX Code equivalent might comprise a list of codes from which one should be chosen. For our reference, as we built our translation database, we created a flag (DIRECTO), indicating which GEM we were using (ICD-9 to ICD-10 or ICD-10 to ICD-9).

We then read in only 2 of the ICD-10 GEM file variables (CMS web site ".txt" files):

```
/* Step 1b */
data ten2ninedx(keep=icd9_cm_code icd10_cm_code directo);
length directo $8;
infile "c:\ROMR7511\2012_I10gem.txt" lrecl=19;
input
       @001  icd10_cm_code              $char7.
       @009  icd9_cm_code               $char5.
       @016  no_map_flag                $char1.
       ;
if  no_map_flag not ='1';
directo='Backward';
run;

proc sort data=ten2ninedx nodupkey;
  by icd10_cm_code
     icd9_cm_code;
run;
```

When we read in the 78,840 records of the ICD-10 GEM file (for use in the ICD-9 to ICD-10 translation database), we did not read all the CMS supplied flags because these flags apply only when the translation occurs in the ICD-10 to ICD-9 direction. We did however, read in the No Map Flag and deleted the record if there was no mapping from ICD-10 to ICD-9 and then eliminated duplicate records. This left us with 78,014 records. The variable DIRECTO indicates from which file

we mined this record. We reasoned that if there was an ICD-10 to ICD-9 mapping not mentioned in the ICD-9 to ICD-10 GEM files, this would provide additional potentially useful mapping information.

At this point we combined the information gleaned from the GEM files into one and found appropriate descriptions for both the ICD-9 and ICD-10 codes:

/* Step 1c */

```
data possible_translations1;
 set nine2tendx
     ten2ninedx;
run;
```

/* Step 1d; optional */

```
proc sort data=possible_translations1;
  by icd10_cm_code;
run;

data possible_translations0;
merge possible_translations1(in=a)
      mayn.tendescsdx;
   by icd10_cm_code;
if  a;
run;

proc sort data=possible_translations0;
  by icd9_cm_code;
run;

data possible_translations1;
merge possible_translations0(in=a)
      mayn.ninedescsdx;
   by icd9_cm_code;
if  a;
run;
```

The resulting file contained 101,926 (=23,912 + 78,014) records. Adding the descriptions to both the ICD-9 and ICD-10 codes was optional, although we chose to implement this (much to our later delight, as will be seen later).

We next preferentially chose the ICD-9 to ICD-10 equivalence mappings from the ICD-9 GEM file.

/* Step 1e */

```
proc sort data=possible_translations1;
  by icd9_cm_code
     icd10_cm_code
     descending directo;
run;

data pt2(keep=icd9_cm_code icd10_cm_code directo);
 set possible_translations1;
  by icd9_cm_code
     icd10_cm_code
     descending directo;
if  first.icd10_cm_code;
run;

proc sort data=pt2 nodupkey;
  by icd9_cm_code
     icd10_cm_code
     descending directo;
run;
```

/* Step 2 */

```
data ten2nine(keep=icd9_cm_code icd9_cm_desc icd10_cm_code icd10_cm_desc)
     nine2ten(keep=icd9_cm_code icd9_cm_desc icd10_cm_code icd10_cm_desc
                   approximate_flag no_map_flag combination_flag
                   scenario choice_list);
merge possible_translations1(in=a)
      pt2(in=b);
   by icd9_cm_code
      icd10_cm_code
      descending directo;
if  a;
if  b;
if  directo='Backward'
    then output ten2nine;
 else
    output nine2ten;
run;
```

This left us with 83,467 records. We separated the ICD-9 to ICD-10 mapping records from those that map ICD-10 to ICD-9 (using the self-crafted variable DIRECTO). We were left with 23,912 mapping records from ICD-9 to ICD-10, and 59,555 mapping records from ICD-10 to ICD-9.

We first processed those general equivalence mappings that we mined from the ICD-10 GEM file. Since there were 59,555 of these mapping records it was apparent that there were some ICD-9 codes that would show a relationship (requiring further scrutiny) with more than one ICD-10 code. We chose to create one record for each ICD-9 code, by concatenating all possible ICD-10 codes (separating them by the token “OR”):

/* Step 3 */

```
proc sort data=ten2nine nodupkey;
  by icd9_cm_code
     icd10_cm_code;
run;

data t29(keep=icd9_cm_code icd9_cm_desc gr1);
length gr1 tmp $32767;
retain gr1;
 set ten2nine;
  by icd9_cm_code
     icd10_cm_code;
if  first.icd9_cm_code
    then gr1=trim(left(icd10_cm_code))||
               ' '||
               trim(left(icd10_cm_desc));
 else
    do;
        tmp=trim(left(gr1))||
            ' OR '||
            trim(left(icd10_cm_code))||
            ' '||
            trim(left(icd10_cm_desc));
        gr1=trim(left(tmp));
    end;
if  last.icd9_cm_code;
run;
```

We were left with 4,646 records. An example record was for ICD-9 code 0068 (AMEBIC INFECTION OF OTHER SITES) which showed a mapping to “A063 Ameboma of intestine” OR “A0681 Amebic cystitis” OR “A0682 Other amebic genitourinary infections.”

ICD-9 DX 0068 → ICD-10 DX (A063 or A0681 or A0682)

Figure 3. *Example mapping derived from the ICD-10 GEM file*

We now focused on the ICD-9 to ICD-10 GEM file information. Using the definitions of the GEM file flags, we separated the general equivalence mappings found in the ICD-9 to ICD-10 Gem files into four groupings which were later processed separately.

/* Step 4 */

```
data combo(keep=icd9_cm_code icd9_cm_desc icd10_cm_code icd10_cm_desc
                approximate_flag scenario choice_list)
     nomap(keep=icd9_cm_code icd9_cm_desc icd10_cm_code icd10_cm_desc
                approximate_flag)
     precise(keep=icd9_cm_code icd9_cm_desc icd10_cm_code icd10_cm_desc
                 approximate_flag)
     nocombo(keep=icd9_cm_code icd9_cm_desc icd10_cm_code icd10_cm_desc
                 approximate_flag);
set nine2ten;
if  no_map_flag = '1'
    then output nomap;
 else
if  approximate_flag = '0'
    then output precise;
 else
if  combination_flag = '0'
    then output nocombo;
 else
    output combo;
run;
```

The No Map Flag, when valued at 1, indicates that there is no reasonable mapping from ICD-9 to ICD-10. There were 425 such records. The Approximate Flag, when valued at 0, indicates that the mapping provided is precise. There were 3,533 such records. The Combination Flag, when valued at 0, indicates that the mapping is a direct mapping, not a combination mapping. There were 17,694 such records. The remaining 2,260 records were “combination” records, i.e., an ICD-9 code was mapped into 2 or more ICD-10 codes.

The processing of the 2,260 combination records was the most complex:

/* Step 5a */

```
proc sort data=combo;
  by descending choice_list;
run;

data _null_;
 set combo;
  by descending choice_list;
if  _n_=1;
call symput('choice2',trim(left(input(put(choice_list,5.),$5.))));
run;

proc sort data=combo nodupkey;
  by icd9_cm_code
     scenario
     choice_list
     icd10_cm_code;
run;
```

For these 2,260 combination records, we determined which mapping scenario had the most possible lists of choices. For this version of the DX Gem files, that macro variable (&choice2) was determined to be 3.

```
data combo2(keep=icd9_cm_code icd9_cm_desc scenario gr1-gr&choice2 approximate_flag);
length gr1-gr&choice2
       tmp      $32767;
retain gr1-gr&choice2;
 set combo;
  by icd9_cm_code
     scenario
     choice_list
     icd10_cm_code;
array grps(&choice2) $ gr1-gr&choice2;
if  first.scenario
    then do;
             do i = 1 to &choice2;
                grps(i)=' ';
             end;
             hh=0;
         end;
if  first.choice_list
    then do;
             hh+1;
             grps(hh)=trim(left(icd10_cm_code))||
                      ' '||
                      trim(left(icd10_cm_desc));
         end;
 else
    do;
        tmp=trim(left(grps(hh)))||
            ' OR '||
            trim(left(icd10_cm_code))||
            ' '||
            trim(left(icd10_cm_desc));
        grps(hh)=trim(left(tmp));
    end;
if  last.scenario;
run;
```

The 2,260 records defining the combination translations collapsed into 747 crosswalk records. An example record was for ICD-9 code 80626 (CLOSED FRACTURE OF T7-T12 LEVEL WITH COMPLETE LESION OF CORD) which showed two possible mapping scenarios. One mapping scenario was to “S24113A Complete lesion at T7-T10, init” AND one of “S22069A Unsp fracture of T7-T8 vertebra, init for clos fx” OR “S22079A Unsp fracture of T9-T10 vertebra, init for clos fx.” The other mapping scenario was to “S24114A Complete lesion at T11-T12, init” AND “S22089A Unsp fracture of T11-T12 vertebra, init for clos fx.”

ICD-9 DX 80626 → ICD-10 DX (S24113A and one of (S22069S or S22079A))
ICD-9 DX 80626 → ICD-10 DX (S24114A and S22089)

Figure 4. *Example combination mapping derived from the ICD-9 GEM file*

The ICD-9 DX codes with no ICD-10 DX code mapping equivalents were processed next:

```
/* Step 5b */
data nomap2(keep=icd9_cm_code icd9_cm_desc gr1 approximate_flag);
length gr1 $32767;
 set nomap;
gr1=trim(left(icd10_cm_code));
run;
```

These 425 records with no acceptable mapping from ICD-9 to ICD-10 were reformatted. An example record was for ICD-9 code 36570 (GLAUCOMA STAGE, UNSPECIFIED) which was mapped to “NoDx.”

ICD-9 DX 36570 → ICD-10 DX (NoDx)

Figure 5. *Example of ICD-9 Code with No ICD-10 Mapping on the ICD-9 GEM file*

The next grouping of mappings involved those that were precise. These 3,533 records were also simply reformatted.

/* Step 5c */

```
data precise2(keep=icd9_cm_code icd9_cm_desc gr1 approximate_flag);
length gr1 $32767;
 set precise;
gr1=trim(left(icd10_cm_code))||
    ' '||
    trim(left(icd10_cm_desc));
run;
```

An example record was for ICD-9 code 6949 (UNSPECIFIED BULLOUS DERMATOSIS) which was mapped to “L139 Bullous disorder, unspecified.”

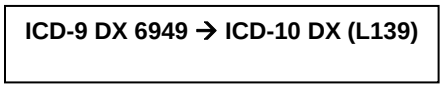

Figure 6. *Example of ICD-9 Code with an Exact Mapping on the ICD-9 GEM file*

/* Step 5d */

The last grouping of mappings, those that did not require a combination of ICD-10 codes to accurately represent an ICD-9 code, was processed next:

```
proc sort data=nocombo nodupkey;
  by icd9_cm_code
     icd10_cm_code;
run;

data nocombo2(keep=icd9_cm_code icd9_cm_desc gr1 approximate_flag);
length gr1 tmp $32767;
retain gr1;
 set nocombo;
  by icd9_cm_code
     icd10_cm_code;
if  first.icd9_cm_code
    then gr1=trim(left(icd10_cm_code))||
               ' '||
               trim(left(icd10_cm_desc));
else
    do;
        tmp=trim(left(gr1))||
            ' OR '||
            trim(left(icd10_cm_code))||
            ' '||
            trim(left(icd10_cm_desc));
        gr1=trim(left(tmp));
    end;
if  last.icd9_cm_code;
run;
```

9,967 viable No Combination records were derived from the 17,694 raw GEM records. An example record was for ICD-9 code 0278 (UNSPECIFIED OTHER SPECIFIED ZOONOTIC BACTERIAL DISEASES) which was mapped to “A288 Oth zoonotic bacterial diseases, not elsewhere classified.”

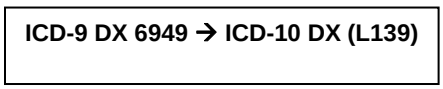

Figure 7. *Example of ICD-9 Code w/ a Non-Combination Mapping on the ICD-9 GEM file*

/* Step 6 */
All that was left to do at this point was to combine all five intermediate files (one from the ICD-10 GEM file and four from the ICD-9 GEM file):

```
%macro gg;

data dxs9to10(keep=precision icd9_cm_code icd9_cm_desc scenario gr1-gr&choice2);
length precision $20;
retain precision
       icd9_cm_code
       icd9_cm_desc
       scenario
       gr1-gr&choice2;
 set combo2
     nocombo2
     nomap2
     precise2
     t29;
if  approximate_flag='1'
    then precision='Approximate';
 else
if  approximate_flag='0'
    then precision='Precise';
 else
if  approximate_flag=' '
    then precision='For Consideration';
label %do i = 1 %to &choice2 - 1;
          gr&i = "One DX from This List (&i) and"
      %end;
      gr&choice2="One DX from This List (&choice2)"
      ;
run;

%mend gg;

%gg;
```

19,318 records went into our mapping database. We used a macro so that we could label each ICD-10 diagnosis list accordingly. Note that we introduced another value to the Approximate Flag (“For Consideration”), which identifies those records that were derived from the ICD-10 GEM file.

All that is left to do is to write this mapping crosswalk into an accessible SAS file:

```
proc sort data=dxs9to10 nodupkey;
  by icd9_cm_code
     scenario
     gr1
     gr2
     gr3;
run;

data mayn.dxs9to10;
 set dxs9to10;
  by icd9_cm_code
     scenario
     gr1
     gr2
     gr3;
if  first.icd9_cm_code
and last.icd9_cm_code
    then scenario=' ';
run;
```

UTILITY 3B

Having created our ICD-9 to ICD-10 crosswalk, we developed utilities to access it in a variety of ways. For example, when we want to look at an ICD-9 DX related to diabetes, we use:

```
data k;
 set mayn.dxs9to10;
if  index(upcase(icd9_cm_desc),'DIABETES') not = 0;
run;
```

When we want to look at one diagnosis, we use a simple data step as follows:

```
data k;
 set mayn.dxs9to10;
if  icd9_cm_code='80600';
run;
```

The above query generates four records, meaning that there are four possible scenarios to which an ICD-9 diagnosis could be translated.

There are a variety of ways that this crosswalk can aid programming staff.

CONCLUSION

By performing automated impact analyses, IICU saves programmer time and avoids errors natural to manual processes. In addition, the information it gathers provides transparency to project and resource managers. Initially created to help identify the impact and scope of ICD-9 to ICD-10 diagnosis and procedure change, IICU has proven its value as a tool for ongoing program quality management and control. Its continued use ensures uniform report quality.

ACKNOWLEDGEMENTS

Thanks to Mirga Girnius, Ph.D. for editing and to Bruce Davidson, Ph.D. for reviewing this paper and providing their helpful remarks.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.
Contact the authors at email address pakalniskisa@cshs.org.