
Problems & Solutions from the CDSI Support Archives

Wayne Taylor
Clinical DataFax Systems Inc.

Problem Agenda

“How can we ”

- overcome problems encountered when creating QC notes in edit checks.
- implement conditional visits.
- check for assessments that are required even if the patient terminates before the assessment was scheduled to occur.
- implement complicated CRF work flow tasks.

Topic 1: Edit Checks & QC Notes

Edit checks can create QC notes, and display them on the screen so that the user can accept or reject them.

What problems can arise?

What solutions are available?

An Example

Blood Pressure Screening Form

Screen 1:	Date	<input type="text"/>	<input type="text"/>	<input type="text"/>	Reading 1:	<input type="text"/>	<input type="text"/>	mmHg
		day	month	year				
					Reading 2:	<input type="text"/>	<input type="text"/>	mmHg
					Reading 3:	<input type="text"/>	<input type="text"/>	mmHg
Screen 2:	Date	<input type="text"/>	<input type="text"/>	<input type="text"/>	Reading 1:	<input type="text"/>	<input type="text"/>	mmHg
		day	month	year				
					Reading 2:	<input type="text"/>	<input type="text"/>	mmHg
					Reading 3:	<input type="text"/>	<input type="text"/>	mmHg

According to the study protocol:
Screen 2 must occur 1 or more days after Screen 1

A Simple Edit Check

```
edit CheckScreen12() {
  string m,s ;

  # Other edit checks kick in if either date is missing or illegal
  if( dfmissing(S1DATE) || dfmissing(S2DATE) ) return ;
  if( !dflegal(S1DATE) || !dflegal(S2DATE) ) return ;

  # If screen1 precedes screen2 there is no problem
  if( S1DATE < S2DATE ) return;

  # Create an error message to be included in the QC query
  m= "Screen 2 must follow Screen 1 by at least 1 day. "
    + "Screen 1 date = " + (s=S1DATE) + ". "
    + "Screen 2 date = " + (s=S2DATE) + ". " ;

  # Display the QC note on the users screen
  dfaddqc(@T,2,m,1,2,"") ;
}
```

Problem: OK or Cancel?

When a QC note pops up the user must apply it by clicking OK or reject it by clicking Cancel, but the user may not understand what the edit check is about.

Examining the QC note to determine what the problem is takes time and may not be easy if the query is long & complicated.

Users may be tempted to just apply every QC note that pops up, which may not always be the correct thing to do.

Solution #1

Use dfask to:

1. show the user a description of the problem,
2. ask the user if a QC note needs to be added.

```
# Create a clear message describing the problem
m= "Screen 2 must follow Screen 1 by at least 1 day.\n "
  + "Screen 1 date = " + (s=S1DATE) + ".\n "
  + "Screen 2 date = " + (s=S2DATE) + ".\n " ;

# Show message and ask if a QC should be added
if( dfask(m+"\n\nAdd a QC note?",1,"Yes","No") ==1 )
  dfaddqc(@T,2,m,1,2,"") ;
```

Problem: multiple QCs

dfaddqc does nothing if a QC note already exists on the designated field!

This is OK if the problem is unresolved and has already been flagged with a QC note.

It is not OK if:

- the problem has re-emerged,
- the edit check has identified a new problem

Solution #2

Warn the user when dfaddqc fails.

```
# Show message and ask if a QC note should be added.
# Warn the user if an attempt to add a QC note fails.
if( dfask(m+"\n\nAdd a QC note?",1,"Yes","No") ==1 ) {
  if( !dfaddqc(@T,2,m,1,2,"")
    dfwarning("Unable to add a QC note.");
}
```

Problem: user confusion

But, now the user is interrupted and asked if a QC should be added even when a QC note has already been added for the problem.

An edit check can test to see if a QC note already exists, but how can we decide whether it is for the same problem or a new problem?

Solution #3

Put a QC identifier in the note field of each QC note created by an edit check.

```
# Create a unique problem ID for each QC note
string QCID="QC1.2";

# Do nothing if a QC note already exists for this problem
if( dfmatch(QCID,dfqcinfo(@T,DFQCNOTE),"") return;

# Show message and ask if a QC should be added
# If a QC note is created put the problem ID in the note
if( dfask(m+"\n\nAdd a QC note?",1,"Yes","No") ==1 ) {
  if( !dfaddqc(@T,2,m,1,2,QCID)
    dfwarning("Unable to add a QC note.");
}
```

Problem: when QCID exists

Aborting the edit check when QCID already exists is not always be the right thing to do.

If QCID exists and QC status = resolved, but the problem still exists:

- the QC may have been resolved incorrectly,
- the problem may have re-emerged after being fixed, or
- the problem may have been reviewed and accepted as a protocol exception.

Solution #4

When resolving a QC note, tag the QC identifier with the resolution type.

Tag the QCID as follows:

QCID=fixed ... if the problem has been fixed
 QCID=allowed ... if the problem still exists in the database but it has been accepted as a protocol exception.

Solution #4

Before asking if a QC note should be added check for an existing QC note with the current problem ID

```
# Note, this is pseudo code, real code would use dfmatch and dfqcinfo
if(QC note ~ "QC1.2") {
  if(QC note ~ "QC1.2=allowed") return;
  if(QC note ~ "QC1.2=fixed") { dfwarning("..."); return; }
  if(QC status is "resolved") { dfwarning("..."); return; }
  return; # QC status is "unresolved"
}

if( dfask(m+"\n\nAdd a QC note?",1,"Yes","No") ==1 ) {
  if( !dfaddqc(@T,2,m,1,2,QCID)
    dfwarning("Unable to add a QC note.");
}
```

Why do we need QC1.2=fixed ?

Question:

When resolving a QC is it necessary to tag the QCID with “=fixed”. Isn't it adequate to just resolve the QC note?

Answer:

This would be OK if only one QCID could ever be add to a field. But this is not the case.

QC status (resolved/unresolved) determines whether the current QC note is sent to the investigator. The current problem may be different from old problems.

e.g. we might have QC1.1 for illegal dates and QC1.2 for screening date2 <=screening date1.

Problem: This is too much!

Question:

OK, I see the need to address these issues, but won't this make writing edit checks terribly complicated?

Answer:

The edit check language is extensible, so solutions #1-4 can be standardized in a user defined function.

Still Simple Edit Check

```
edit CheckScreen12() {
  string m,s ;

  # Other edit checks kick in if either date is missing or illegal
  if( dfmissing(S1DATE) || dfmissing(S2DATE) ) return ;
  if( !dflegal(S1DATE) || !dflegal(S2DATE) ) return ;

  # If screen1 precedes screen2 there is no problem
  if( S1DATE < S2DATE ) return;

  # Create an error message to be included in the QC query
  m= "Screen 2 must follow Screen 1 by at least 1 day.\n "
    + "Screen 1 date = " + (s=S1DATE) + ".\n "
    + "Screen 2 date = " + (s=S2DATE) + ".\n " ;

  # Run QC note function
  myaddqc(m,2,m,1,2,"QC1.2") ;
}
```

Solution #5

myaddqc:

An implementation of myaddqc was introduced at the fall 2001 QC edit check training course.

It is available from CDSI for anyone who wants it from the DFUG 2002 web site.

It requires new functionality implemented in release 3.5-003

Solution #5 SOPs

This solution requires the use of QCIDs

- All problems must have a QCID (code identifier), entered in the private note field.
- If it is necessary to modify an existing QC note to describe a new problem, the new QCID must be added to the note field, but the previous QCID must also be preserved. In such cases I recommend that the current QCID be entered at the beginning of the QC note.
- When resolving QC notes users must tag the correct QCID(s) with “fixed” or “allowed”.

e.g. the note field of a currently unresolved QC note

QC4.1, QC4.3=fixed, QC4.2=allowed -memo 25/15/02

Topic 2: Conditional Visits

How can we get DataFax to generate missing plate QC notes for CRFs that are due at a conditional visit?

Example:

- at each follow-up patients are sent for an MRI if there is diagnostic suspicion of a stroke.
- in such cases the MRI results must be received within 7 days of the follow-up visit

Solution #1

Define conditional visits using the conditional plate map: DFcplate_map

```
IF|visit#|plate#|field#|value
visit list|plate list
```

Example: conditional plate map

If at visits 10,20,30,..80, field 18 (dxstroke) on plate 33 equals 2 (YES), then plates 55 and 56 are due at visit# = visit# + 2.

```
IF|10|33|18|2
12|55,56
IF|20|33|18|2
22|55,56
etc.
```

But: this will only work if visits 12, 22, 32, ..82 are defined in the visit map as required visits, with other required plates.

How DF_QCupdate works

Step 1. for each patient, it determines which visits have arrived and creates overdue visit QC notes for any overdue visits.

Step 2. for each visit that has arrived (i.e. at least 1 plate is in the database) it creates missing plate QC notes for any required and conditional plates that are missing from that visit.

Thus, if all plates for a visit are conditional DataFax will not complain about missing plates if they are all missing.

Solution #2

Use the conditional plate map only for CRF pages that are conditionally due at required visits already defined in the visit map.

Use an edit check for conditional visits, i.e. use `dfaddmpqc` to create a code 23 (missing plate) QC note for the CRFs due at conditional visits.

Example: conditional visits

```
#Run on exit from plate 33 at each follow-up visit
edit MRIcheck() {
  string s, m;
  number days, v=@[6]+2;
  #If clinical diagnosis = stroke and follow-up occurred >7 days ago
  #add missing plate QCs for the MRI report (plates 55,56)
  if( dxstroke == YES && (days=(dftoday()-vdate))>7 ) {
    m = "MRI is Overdue: " + (s=(days-7)) + " days." ;
    dfaddmpqc(ID,v,55,m,1,2,"") ; dfaddmpqc(ID,v,56,m,1,2,"") ;
  }
}
```

Problem: data corrections

What if the data fields, dxstroke and/or vdate, are corrected during a subsequent review of the CRF?

If the old values led to the creation of a missing plate QC note the last time the record was signed off, it will remain in the database, even if the new values do not warrant it.

Solution #3

```
#Run on exit from plate 33 at each follow-up visit
edit MRIcheck() {
  string s, m;
  number days, v=@[6]+2;
  #If a stroke is suspected and follow-up occurred >7 days ago
  #add missing plate QCs for the MRI report (plates 55,56)
  if( dxstroke == YES && (days=(dftoday()-vdate))>7 ) {
    m = "MRI is Overdue: " + (s=(days-7)) + " days." ;
    dfaddmpqc(ID,v,55,m,1,2,"") ; dfaddmpqc(ID,v,56,m,1,2,"") ;
  }
  #Error correction, for changes in dxstroke and/or vdate
  else { dfdelmpqc(ID,v,55) ; dfdelmpqc(ID,v,56) ; }
}
```

Problem: key changes

What if the patient ID (field 7) or visit number (field 6) need to be corrected?

If a code 23 QC note was previously created for the old keys it will remain in the database, even if it is no longer warranted.

Note: if the keys have been corrected, it is possible that the old keys do not even correspond to a data record. In such cases an old code 23 QC note would be stranded in the database; what we call a free floating QC note.

Solution #4

```
#Run on exit from plate 33 at each follow-up visit
edit MRIcheck() {
  string s, m;
  number days, v=@[6]+2;
  number id0 = dfvarinfo(@[7],DFVAR_ENTER_VALUE) ;
  number v0 = dfvarinfo(@[6],DFVAR_ENTER_VALUE) ;

  #If a stroke is suspected and follow-up occurred >7 days ago
  #add missing plate QCs for the MRI report (plates 55,56)
  if( dxstroke == YES && (days=(dftoday()-vdate))>7 ) {
    m = "MRI is Overdue: " + (s=(days-7)) + " days." ;
    dfaddmpqc(ID,v,55,m,1,2,"") ; dfaddmpqc(ID,v,56,m,1,2,"") ;
  }
  #Error correction, for changes in dxstroke and/or vdate
  else { dfdelmpqc(ID,v,55) ; dfdelmpqc(ID,v,56) ; }
```

Solution #4 continued

```
# If we are beyond new data entry (DFSTATUS>0 or DFLEVEL>0)
# Remove QCs previously created using erroneous keys
if( DFSTATUS>0 && (@[7]!=id0 || @[6]!=v0) ) {
  v0 = v0+2; dfdelmpqc(id0,v0,55) ; dfdelmpqc(id0,v0,56) ;
}
}
```

Note:

The above error correction will only occur if the edit check is executed interactively. Key changes will not be picked up when this edit check is run in batch mode.

Topic 3: Early termination woes

Example:

At baseline, patients go for a lab test which is defined using visit type r (due by next visit).

```
0|B|Baseline|1|9|0|0|1-10|||  
1|r|Lab Test|15|9|0|0|15|||  
2|S|1 mon visit|20|9|30|5|10,16|||  
etc.
```

Suppose that the lab test is required even if the patient terminates before the first follow-up visit.

Problem

DataFax does not consider a visit overdue if the patient terminates before the visit is due.

Thus if the patient terminates before the first follow-up (visit 2 due at 30 days) the lab test will not be called overdue.

Solution #1

Make the lab CRF (plate 15) due at the baseline visit (visit 0), and remove visit 1.

```
0|B|Baseline|1|9|0|0|1-10,15|||
1|r|Lab Test|15|9|0|0|15||| ... delete
2|S|1 mon visit|20|9|30|5|10,16|||
etc.
```

But then missing plate QC notes will be created for the lab test (plate 15) as soon as the baseline CRFs (plates 1-10) arrive. This might annoy the investigators.

Solution #2

```
#Add this edit check to each and every termination date field.
#Run on plate exit.
#Check for Baseline lab report (visit 1, plate 15)
edit CheckBlab() {
  if( !dfmissing(@T) ) dfaddmpqc(ID,1,15,"",1,2,"") ;
}
```

But what about corrections to the termination date?
Can we do this?

```
if( !dfmissing(@T) ) dfaddmpqc(ID,1,15,"",1,2,"") ;
else dfdelmpqc(ID,1,15) ;
```

Solution #3

```
#Add to each and every termination date field. Run on plate exit.
#Check for Baseline lab report (visit 1, plate 22)
#Use terminated(), a user defined function that checks all
#termination dates and returns true if any of them are set.
edit CheckBlab() {
  if( !dfmissing(@T) ) dfaddmpqc(ID,1,15,"",1,2,"") ;
  else if( !terminated(ID) ) dfdelmpqc(ID,1,15);
}
```

terminated – a user defined function

```
# Check each termination date, and return:
# 1 (true) if any termination date is set, otherwise return 0 (false).
# For example suppose that termination dates appear as:
# variable tdate on plate 33 at visits 1~15, and
# variable vdate on plate 44 at visit 20, and plate 55 at visit 21
number terminated(number id) {
  number v=1;
  while(v<=15) { if( !dfmissing(tdate[id,v,33]) ) return(1); v=v+1; }
  if( !dfmissing(vdate[id,20,44]) ) return(1);
  if( !dfmissing(vdate[id,21,55]) ) return(1);
  return(0) ;
}
```

What about ID key changes?

As for MRIcheck, we need to correct any cases where erroneous keys led to the creation of an erroneous code 23 QC note.

For CheckBlab we only need to consider changes in the patient ID key, because the termination date is what's relevant not the visit number, i.e. correction of the visit number would not lead us to question whether or not the patient had terminated, only changes to the termination date and ID matter.

Solution #4 – but this has a problem

```
#Add to each and every termination date field. Run on plate exit.
#Check for Baseline lab report (visit 1, plate 22).
#Include terminated() a user defined function that checks all
#termination dates and returns true if any of them are set.
#Check for changes to the patient ID number (field 7).
edit CheckBlab() {
  number id0 = dfvarinfo(@[7],DFVAR_ENTER_VALUE) ;
  if( !dfmissing(@T) ) dfaddmpqc(ID,1,15,"",1,2,"") ;
  else if( !terminated(ID) ) dfdelmpqc(ID,1,15);
  if( ID != id0 ) {
    if( terminated(id0) ) dfaddmpqc(id0,1,15,"",1,2,"") ;
    else dfdelmpqc(id0,1,15);
  }
}
```

Problem

The current record with the old ID is still in the database and will be used, inappropriately, even on plate exit.

Thus we must either abandon function terminated, and write a custom edit check for each and every termination date, or ...

always run dfdelmpqc when the ID changes and plan to recover any inappropriate removals by running the edit check again in batch mode before sending out QC reports.

Solution #5

```
#Add to each and every termination date field. Run on plate exit.
#Check for Baseline lab report (visit 1, plate 22).
#Include terminated() a user defined function that checks all
#termination dates and returns true if any of them are set.
#Check for changes to the patient ID number (field 7).
edit CheckBlab() {
  number id0 = dfvarinfo(@[7],DFVAR_ENTER_VALUE) ;
  if( !dfmissing(@T) ) dfaddmpqc(ID,1,15,"",1,2,"") ;
  else if( !terminated(ID) ) dfdelmpqc(ID,1,15);
  #if ID has changed remove missing plate QC for old ID
  #any inappropriate removals will be recovered in batch run
  if( ID != id0 ) dfdelmpqc(id0,1,15);
}
```

dfaddmpqc SOPS

When edit checks use dfaddmpqc:

1. include code to delete previously created QCs, if the condition no longer applies due to changes in the data being tested.
2. include code to delete previously created QCs, corresponding to old key fields if there are changes in the keys.
3. run these edit checks interactively to ensure that key change corrections (2. above) occur.
4. also, if the test condition spans multiple plates, run these edit checks in batch before sending QC reports. This will recreate any QCs removed inappropriately (in 3. above).

Topic 4: Work Flow

How can work flow be automated for conditional reviews?

Example:

A CRA must review all cases where a patient was ineligible during screening, but nevertheless proceeded to the baseline visit.

In such cases the CRA must review all screening and baseline records.

An Example

Blood Pressure Screening Form

Eligibility Criteria

1. Age 16-79 years old.
2. Male, or if female, not pregnant and using a reliable contraceptive method.
3. Taking blood pressure medication.
4. Major surgery or other medical emergency in the past 6 months.
5. Patient has signed informed consent and agrees to return for monthly follow-up visits.
6. Systolic blood pressure 160-200 mmHg or Diastolic blood pressure 95-105 mmHg on Reading 3 at both Screen 1 and Screen 2.



note: this appears on plate 1 at visit 0

Solution #1

Define a Task using retrieval by variables which retrieves cases where the patient has failed any of the eligibility questions.

But, retrieval by variables can only test fields on a single plate and can only retrieve data records for that same plate.

Solution #2

Use retrieval by program.

Write a program which will retrieve all screening and baseline records for ineligible patients with baseline data.

Solution #2

Program Steps:

- export the eligibility data
- identify ineligible patients that have not yet been reviewed
- check to see if any of them went on to the baseline visit, and if so
- write DRF records, ID|Visit|Plate, for the screening and baseline CRFs that need to be reviewed

Example: UNIX Shell Script

```
#!/bin/sh
#CRA must review screening (visit 0, plate 1) and baseline (visit 1, plates 2~4)
#data of ineligible patients who proceeded from screening to baseline.
#CRA raises records from level 1~3 to level 4 when performing this review.

#1: identify the ineligible patients still at validation levels 1~3
$DATAFAX_DIR/bin/DFexport.rpc -s primary -v 1~3 -f "7,27~32" 253 1 - | \
awk -F\ '{ if($2==1 || $3==1 || $4==2 || $5==2 || $6==1 || $7==1) print $1 }' > xids

#2: if any of the ineligible patients found above in step #1 have baseline data
# print a DRF record "ID|Visit|Plate" for the records to be reviewed
for ID in `cat xids`
do
    $DATAFAX_DIR/bin/DFexport.rpc -s primary -l $ID -f 7 253 2 - | \
    awk '{ printf"%d|0|1\n", $1; for(p=2;p<=4;p++) printf"%d|1|%d\n",$1,p; }'
done
/bin/rm xids
```

Solution #2

Using the Retrieval Program:

1. Make the program file executable

```
chmod +x CRAreview
```

2. Install it in the study work directory

```
mv CRAreview /opt/studies/mystudy/work
```

3. Instruct the user to use validation retrieval mode "By Program" in the validation tool and specify CRAreview as the program to run.

Problem: performance

The program may be slow because:

- it's running on old hardware
- the database is huge
- the program is complex
- the program runs on a different system (e.g. SAS on another computer) and the results must be sent back to the DataFax server.

Solution #3

Using cron run the program at specified times and save the results in a DataFax retrieval file.

e.g. to run the program at 2am every day

```
crontab -e  
0 2 * * * ~/work/CRAreview > ~/work/CRAreview.drf
```

The CRA would then use validation retrieval mode "By DataFax Retrieval File".

Note: when running cron jobs define environment variables, like \$DATAFAX_DIR in the program file, or execute your .profile file at the top of the program file, or hard code environment variables.

Summary

- edit check QC creation can be standardized in a user defined function like myaddqc.
- conditional visits can be implemented using dfaddmpqc but remember to include error correction.
- when the test condition for dfaddmpqc involves several plates, run the edit check interactively and in batch to achieve appropriate error correction.
- complicated CRF work flow tasks can be implemented using the By Program retrieval mode or using DataFax Retrieval Files created by periodic program execution using cron jobs.