Importing OpenClinica Data in SAS: It's Possible! — Our Experiences

Jules van der Zalm, OCS Consulting, the Netherlands

INTRODUCTION

Because of its open source nature, OpenClinica is a favourite choice for many CRO's, small size pharmaceutical companies and universities. With the Community Edition, OpenClinica offers a free EDC and clinical data management system. The Enterprise Edition provides additional features and services such as formally tested software, data rules, upgrades and patches and customer support. With various export formats, data collected in OpenClinica's EDC component can (supposedly) be easily imported, transformed and analysed in a variety of applications such as SPSS, R, Excel or SAS.

Our customer is a manufacturer of medical devices. They use OpenClinica amongst some other EDC systems. For one of many clinical trials that they execute, our customer cooperates with five universities, each of which represents a study center. With the universities as main provider of resources for managing this clinical trial, OpenClinica is the EDC system of choice.[1] This clinical trial is the subject of this paper.

DISCLAIMER

It is not the intention of this paper to provide a solution for any of the issues discussed. For most of the problems the cause was not found, either because the required knowledge was not available in the team or because there wasn't sufficient time available and a workaround would be faster. The intention of this paper is to share the problems that we encountered on our journey and the solution that we chose, and possibly start the discussion on a better integration between OpenClinica and SAS.

[1] http://www.clinicaltrials.gov/ct2/show/NCT01519908

XML AND ODM DEFINITIONS

When exporting data from OpenClinica to XML, OpenClinica uses CDISC ODM (Operational Data Model). In this model data is presented in a relational way rather than in flat tables which are very common in the pharmaceutical industry. For those who are new to XML, this is a very simple example of how one subject's demographic data could be represented in XML:

```
<subject>
 <usubjid>SVT_01-001</usubjid>
 <age Unit="years">24</age>
 <sex>F</sex>
 <race>Caucasian</race>
</subject>
```

If you want to read more about XML, Raymond Ebben's paper *XML by eXaMpLe* is a good start: http://www.phusewiki.org/docs/2008/PAPERS/TU03.pdf

The CDISC ODM model as used by OpenClinica distinguishes metadata and subject data. Both are presented in the XML file that can be opened and read in any text editor. The most important metadata entities in this relational model are:

- Study: contains the study name, e.g. MC2013 or S91-002
- StudyEvent: usually contains a visit name, e.g. Visit3 or Termination
- FormDef: the name of a form, e.g. Demographics or Adverse Events
- ItemGroupDef: a collection of items (questions) on a form, e.g. DM or VS
- ItemDef: an item (question) on a form, e.g. Age or Weight at birth

IMPORTING THE DATA IN SAS

SAS comes with support for ODM XML data built in. PROC CDISC was designed to read in these XML files and create SAS data sets for each domain defined in the XML. It supports ODM versions up to 1.2, though at the time of writing this paper we are at ODM version 1.3. Fortunately OpenClinica can export (i.e. generate) to both versions. It is important to know that PROC CDISC processes only the patient data and not the metadata. To also import the metadata we have created an XML Map (using SAS XML Mapper) and use the SAS XMLv2 libname engine.

After importing both the subject data and the metadata, bespoke SAS programs were written to convert the raw SAS data sets into analysis datasets. These steps are the majority of the work before the tables, listings and figures can be generated.

During the process of importing the XML files to SAS and writing the bespoke SAS programs that generate analysis data sets we encountered various issues with the OpenClinica data. The following few paragraphs will describe each of these issues and in which way we either solved it, or worked around it.

ISSUE #1: ONLY ONE SITE PER DATA FILE

When we received our first extract from OpenClinica, we received one XML file containing the data from all centers. Soon enough this resulted in our first error extracting the data: for unknown reasons, OpenClinica uses the StudyOID-field to store the Site ID, and PROC CDISC can not handle multiple StudyOIDs in a single XML file. Research indicated that we should use OpenClinica to export per site, after which you will get one XML file per site and each XML file will have to be read in separately.

Since we're now reading data from five different sources that represent the five different centers, we get five sets of data which in theory each have the same data structure. However, we can not be certain of that and for that reason we created a script that identifies the difference, should there be any. This script extracts the metadata from each XML file and places it in a separate SAS library. This metadata is then compared using SASs' PROC COMPARE. If differences are found then these are reported and should be either explained or resolved. Various metadata files are manually excluded from the comparison since they will be different per file by definition, for example if they contain the name of the center or the number of records in a domain.

Once we have confirmed that metadata is consistent over all files, or that differences can be explained, it's safe to read the subject data in those separate files. Each file represents data from one site. Each site collects the same data, but of course for different subjects. When we want to use these data sets for further processing, the data from all sites must be combined. A SAS program has been designed to read the separate XML files using PROC CDISC that combines data sets with the same name into one large data set. The end result is a folder with the raw data, which is the combined data from the XML files from each of the sites.

ISSUE #2: PROC CDISC AND ODM v1.3

The ODM structured XML files come in different versions. OpenClinica can export versions 1.2 and 1.3. However, PROC CDISC only supports v1.2. In our case that meant that we could not use item information that describes the parent item. This would have been especially useful for sub-questions on a CRF. In our study there were a few cases where CRF's with long lists of checkboxes (such as Medical History where there are prespecified illnesses) were used. Some of these checkboxes had sub-questions belonging to them, such as "Date of Onset". Given the structure of this data, with the many checkboxes, our preferred programming method to convert this data to analysis data was a method that would be generic, where we would not have to specify all the different variable names. However, from the data nor the metadata, could it be derived which parent item belonged to an item, since this information simply is not available in our v1.2 ODM XML data.

In our case we were lucky to have a technical team who had already written parsers for v1.3 ODM XML for their SQL environment. As we had access to those SQL databases we could read their raw imports of the data, which did give us insight in the parent-child relationship of items. In the scenario where we would not have this SQL database, we would have had to 'manually' read the v1.3 ODM files and extract the parent-child relationships from them programmatically.

ISSUE #3: NON-UNIQUE SASFIELDNAMES

Once we got around the issue of only one site per data file, the following issue arised: these were various variables in the XML data that had the same SASFieldName. SASFieldName is the attribute of an item in OpenClinica that is used when SAS reads the XML file to SAS data sets. It is an 8 character name that replaces the ItemOID as PROC CDISC cannot handle more thn 8 characters and within OpenClinica that value has a maximum length of 32 characters. The SASFieldName is directly derived from the ItemOID and where possible, it is simply truncated to 8 characters. However, that may of course result in duplicate variable names when the first 8 characters are the same for several ItemOIDs. If that is the case, OpenClinica will generate a new SASFieldName that should be unique. However, in our situation, the SASFieldName was only unique if the casing was considered as well (e.g. BASESUPP and BaseSupP). SAS does not distinguish one from the other and sees it as a duplicate variable name.

There seemed to be now way to resolve this issue in OpenClinica. Experts on that area informed us that this is the way OpenClinica creates SASFieldNames and that it can not be changed. What we did to allow SAS to read the XML files is the following. We used SAS infile statements to read the XML file (after all it is a structured, but plain text file) into a SAS data set. Using regular expressions we identified each SASFieldName. All SASFieldNames were uppercased (to remove the case sensitivity) and written into a separate data set. the program then writes all duplicate values to an Excel spreadsheet, in which we manually provide new, alternative SASFieldNames. The next time we would run this program it picks up this Excel spreadsheet and replaces the original SASFieldNames with our alternatives. Should any new duplicate variable names have been added to the data, the program detects that, informs the user and adds it to the Excel spreadsheet.

After all duplicate SASFieldNames have been given alternative names, the program writes a new XML file containing no duplicates, which can then be read in by our import programs.

ISSUE #4: CHANGING CRFs IN OPENCLINICA

During the course of the study, the study database manager was required to change several items on different forms. This would eventually result in up to five or six different versions of the same form. Since data had already been entered in older versions of the same form, we would get data from “one” CRF in different shapes. In one instance, a formatted variable was given different values for the same checkbox; the checkbox called NONE on the form used to have value NONE in the database, but that was changed to 0. In other cases single fields on the CRF were captured in two or three different variables, which then had to be combined into one variable before they could be used.

The major drawback of changing CRFs is that you are required to update your program when the CRF changes. Programs that have already been completed and validated lose their validated status when you change them, so not only do you have to invest effort in keeping your programs up to date with the CRF, also you have to spend additional time redoing the validation.

ISSUE #5: CHANGING SASFIELDNAMES

After completing the programming for nearly all of our data extraction programs we received a fresh export of the OpenClinica data, as we had every few weeks in the time before. After processing the XML files through all our preparation programs and converting the XML into SAS data sets, we executed the extraction programs that convert the raw SAS data into analysis data sets. To our surprise, none of the programs would work.

It turned out that the SASFieldNames had changed for nearly 100 out of approximately 500 variables, so many of the variable name references in the programs were no longer valid, for example, references used in drop or keep statement, derivations and calculations and BY statements in procedures. We would have to go through each individual program manually to find out which variables had undergone this name change, and what the new name of the variable is. As the OpenClinica team could not retrieve the original SASFieldNames in new exports, this would be the fastest way to overcome this issue. However, as we have not found the reason why this happened, we can also not be sure it will not happen again! So if we would replace the old SASFieldNames with new SASFieldNames everything would work again, but only until the names changes again in a future export.

In order to require this change only once and then never again, we decided to do the following. After converting the XML data into SAS data sets using PROC CDISC in SAS, we executed a bespoke utility that renamed the SAS variables from their SASFieldName to the ItemOID. This ItemOID has a maximum length of 32 characters which SAS can handle just fine. Doing so would mean that all our variable names would need to change in the extraction programs, but at least we were certain that we only needed to do this once since the ItemOID supposedly never changes.

In order to facilitate the updating of the extraction programs, we wrote a short macro to which you can provide a, or a series of, SASFieldName(s), and it will return the corresponding ItemOIDs which you can then use to replace to previous item names in the program.

An alternative approach would be to automate this process, by reading in each SAS program in a data step and replacing the SASFieldNames automatically, but we decided not to do this in order to keep full control over what happened in the programs. In situations where programs used study metadata to generate analysis data sets sometimes we needed to change a bit more than just the variable names in the programs, and an automated process could not do that, or unwanted changes would not be noticed.

LESSONS LEARNED: THE ESSENCE OF DATABASE DESIGN

A properly structured database design is essential if you do not want to spend hours finding the data that you need to create each domain data set. The way OpenClinica, ODM and PROC CDISC work is that you get one SAS data set for each ItemGroup that is defined in OpenClinica.

In our case, this resulted in approximately 40 SAS data sets, many of which contained very similar data that is supposed to go into the same domain. Should we get a data set per domain we would have reduced this number to approximately 20 data sets, making the processing a lot easier.

On the other hand there were many data sets that were called UNGROUPED##, where ## is an increasing number. We could only guess what CRF was represented in each of these, by looking at the variables and the labels and comparing that to our (manually) annotated CRFs.

The database was designed so that each visit would be represented in one (or more, but ideally one) data set, rather than each domain. For example, when creating the analysis data set for the Concomitant Medication domain, we would have to gather data from at least five different source data sets which had a very similar, yet slightly different structure and content. Each data set would contain Concomitant Medication data from a different visit. Ideally all Concomitant Medication data had been collected into one data set and have a variable indicate (e.g. VISITNUM) from which visit a record origins.

The lesson learned here for us is that, for future studies, the database design team and the study programmers should cooperate actively during the setup phase of the database in order to provide the best foundation from which the analysis data sets are created. This would result in properly defined data sets straight from the database, that need only little changes to be used as analysis data.

CONCLUSION

Definitely this has been a very exciting exercise from a programming perspective. However, we were somewhat disappointed to see that there are so many issues to tackle before we are able to load data from OpenClinica in SAS properly. However, with the complete set of workaround programs that we created we did set up an environment that can be used for future studies that have the same approach.

This was our first experience with OpenClinica, and possibly more will follow. It's fair to conclude that is will be essential to have close cooperation between the different teams in order to prepare the technical study setup. This will potentially result in much less interaction required by the programmer, and allow for a better integration between OpenClinica and XML.

ACKNOWLEDGEMENTS

Thanks to the people within our customer with whom we have cooperated, whose names unfortunately cannot be mentioned in this paper. Thanks to Panagiotis Metaxas, my former colleague of OCS Consulting, for helping to collect an overview of all the things that happened over the past months. And finally, thanks to everyone that we have worked with during this project.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jules van der Zalm
OCS Consulting
P.O. Box 3434
5203 DK 's-Hertogenbosch
+31 (0)73 523 6000
sasquestions@ocs-consulting.com
http://www.ocs-consulting.nl/