PharmaSUG 2020 - Paper AD-130

A Novel Solution for Converting Case Report Form Data to SDTM Using Configurable Transformations

Sara Shoemaker, Matthew Martin, Robert Kleemann, David Costanzo, Tobin Stelling
Statistical Center for HIV/AIDS Research and Prevention (SCHARP), Fred Hutch

ABSTRACT

Converting case report form (CRF) data to SDTM is a complicated process, even when data are collected in CDASH format. Conversion requires many data set manipulations and demands flexibility in the order of execution. In addition, different domain types require different actions on the source data, e.g. Findings domains require transposition of data records. Many conversion solutions address this complexity by performing data pre-processing, mapping, and post-processing as disparate pipeline sections. Most include programming in a language such as SAS where blocks of code can obscure the details of the transformation from data customers.

This paper describes a solution for SDTM conversion that uses a method termed “Configurable Transformations”. This model both achieves conversion using one consistent pipeline for all phases from CRF data to SDTM and provides visibility into the data transformations for non-programmers. This is achieved by a human readable configuration that uses a small set of simple transformation step types to produce derived data sets. These resulting configurations can be defined by data analysts and can be understood by data customers.

Our group was able to successfully map and convert all CRF data for an HIV prevention study using this model with no need for procedural code. This paper will go into the details of the Configurable Transformation model and discuss our use of it in converting data for a study..

INTRODUCTION

The conversion from study databases and other data sources with varied formats to CDISC Study Data Tabulation Model (SDTM) presents numerous issues to be solved. A single SDTM domain can require the merging of data from many different case report forms (CRFs), several processing steps are required to calculate variables such as the study day for a given visit, and different types of domains require different orders of operation (e.g. findings domains require transposition). In reviewing solutions for the conversion of CRF data to SDTM, we found the following drawbacks:

Procedural code – less visibility: Although languages like SAS can provide flexibility and powerful functionality, it can be difficult for downstream data customers to be able to find out what data are being used as the source for a given data set. Additionally, procedural code obscures the implementation of the data transformation for everyone but a SAS programmer.

Disparate pipeline sections – less flexibility: Another solution is to have a pipeline where each section performs a discrete task, such as preprocessing, mapping, and post-processing. This presents drawbacks such as: the need to support, maintain, and train on multiple systems; possible data type handling issues between sections; possible loss of metadata during handoff; and decreased ability for real-time data processing.

Mixed mapping – confusing and complicated: In addition to specifying variable mappings from source to destination data sets, some organizations use a spreadsheet read by the system to define various data set actions such as merging, sorting, and filtering. Although this solution centralizes all actions into one spreadsheet, the mixing of general functions with variable mapping information creates a confusing and complicated document.

OUR APPROACH – CONFIGURABLE TRANSFORMATIONS

The approach we chose was to design a system where the programmer chooses from a set of basic data set transformations that can be performed in any order (“transformation steps”). These steps are defined in a both human-readable and machine-readable manner in a configuration file and can be understood by staff that do not have a programming background. Each variable mapping is broken out into its own spreadsheet and the developer can use as many steps as necessary to define a transformation, not being constrained to performing them in a certain order. We termed this model “Configurable Transformations”.

HIGH LEVEL DESCRIPTION OF OVERALL SYSTEM

THE DELPHI SYSTEM

The Configurable Transformation model is used to convert data to SDTM at SCHARP within the context of a system named “Delphi”, which you can see diagrammed in Figure 1. The Delphi system is comprised of the following components:

- A validated platform for lightweight data set transformations (“derived data sets”).
- Datamarts (databases) for delivering imported and derived study data sets.
- A web tool for viewing data set information.
- A suite of tools for administrators, data set developers, and data standards users.

NOTE: In our organization, the data customers are clinical and statistical programmers and data standards analysts function as the data set developers (of SDTM).

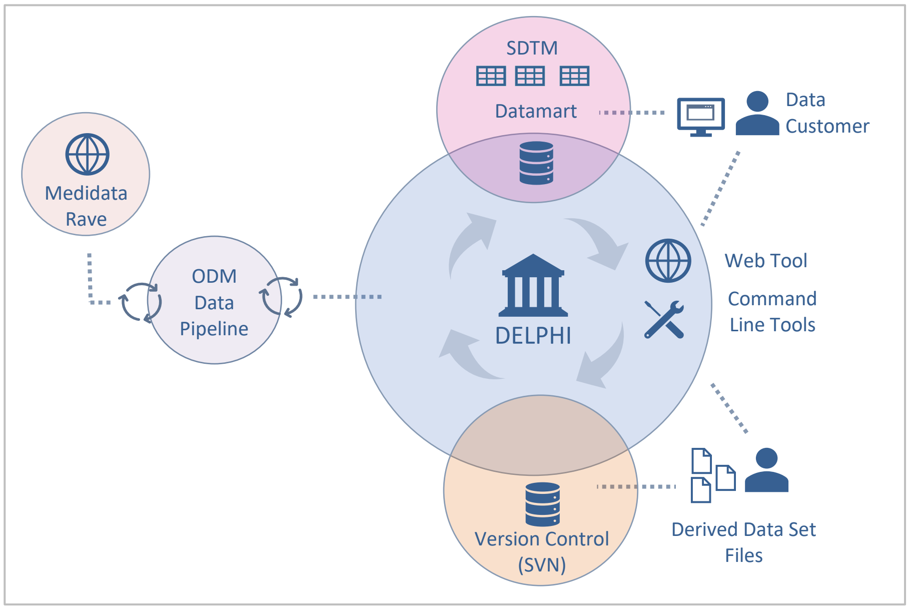

Figure 1. Delphi System Diagram

DATA SET TYPES IN DELPHI

One of Delphi’s key functions is to make study data available to roles within SCHARP who need it for data reports or analysis. These study data are organized into data sets, accessible as objects in the datamart under schemas (a schema usually corresponds to a single study). The data sets in Delphi can be understood as falling into one of the following types:

1. **Imported data sets**: These data sets contain data that are presented to Delphi by a pipeline and are imported into the system. The data in these data sets are not transformed other than being organized into a data set.
2. **Derived data sets**: These data sets take as their starting or “input” data other data sets present within Delphi (both imported and derived). These input data undergo a transformation and are then made available in the datamart as a derived data set.
3. **Delphi data sets**: Two data sets in each schema are special data sets that are neither imported nor

derived and are created by the system to give users access to metadata about the imported and derived data sets. These data sets provide table-level and variable-level metadata for all tables in a schema.

4. **Page-Level data sets**: Delphi supports a special kind of derived data set termed page-level data sets. These data sets contain data from the Medidata Rave (clinical data) platform organized so that each data set represents data from one form in Rave. These are the data sets most often used by customers to access clinical data and are easier for users to consume than the imported data, which is organized in a key-value fashion.

THE DATA SET CONTRACT

To support data management best practices, Delphi requires that each data set managed by the system must have an associated "data dictionary" (the set of table level and variable level attributes that collectively define a data set). For derived data sets, the data dictionary is part of a data set's definition files and is specified by a data set developer. For imported data sets, the data dictionary is managed by the system. Data sets in Delphi are guaranteed to conform to the data dictionary, making it a solid contract. Non-conforming data sets cannot exist in the system.

In Delphi, all variables must have a specified type. Data sets present in the database or specified within data dictionary files on the file system all comply with a defined set of standard types in use by SCHARP. One of these standard types is the CDISC DATE type, which allows us to deliver data in the subset of ISO 8601 formats preferred by CDISC

EXPLICIT DEPLOYMENT

Another important concept in the design of Delphi is that derived data sets only appear in the datamart after they are deployed – a manual action that requires separate permissions and authentication. Changes to data set files can be committed to the repository at any time at the data set developer's discretion, but the data set will only appear in the datamart after it is deployed. These actions are logged and produce an audit trail that can be viewed in the web tool.

DATA SET PERMISSIONS

Data security and integrity are essential aspect of a data platform. To achieve this, the Delphi system assigns permissions to imported data sets based on their data type and then follows the principle of least privilege to determine access to derived data sets. For example, different and more restrictive roles are assigned to imported randomization data sets than other imported study data sets. If a derived data set sources data from both randomization and study data sets, the resulting derived data set requires the user to have access to both source data sets to access the new derived data set.

TRANSFORMATION VISIBILITY

One of the benefits of the Configurable Transformation model is that the input data sets are specifically called out in a file that can be read by the Delphi. The system therefore understands the dependency tree for the data sets it manages and can use that knowledge to display information to the user. In the case study described below, the system-generated diagram of inputs to the operational DM domain is shown (Figure 5). Data customers can use this diagram to understand which data sets contribute to the domain in question. The diagram can also be used to navigate to information about the input data sets.

The Delphi system is also aware of the different steps that are being used in the transformation. A feature currently slated for a future version of our web tool is the ability to diagram the transformation so that customers will be able to have visibility into what operations are being done on the input data sets.

CONFIGURABLE TRANSFORMATION INTERPRETER

Transformations defined with the Configurable Transformation model are executed by a Java command line tool referred to as the "Configurable Transformation Interpreter". This tool operates on files organized into one directory per data set ("data set directories") and can execute the transformation either on the file system or within the Delphi system itself. Resulting data sets are then available either on the filesystem as .csv files or in the datamart where they can be accessed by customers.

THE CONFIGURABLE TRANSFORMATION MODEL

THE DATA SET DIRECTORY

A data set directory consists of prescribed set of files that is expected by the Configurable Transformation Interpreter. It contains all the information needed to execute a data set transformation. There is one directory per data set, and these directories are stored in a code repository from which they can be checked out onto the file system. The Delphi system also accesses the code repository, and executes the transformations described in the data set directories.

Summary of Directory Files

Table 1 summarizes the files in the data set directory, as well as the purpose of each. The key files that the user most often interacts with are highlighted in blue. An example of the *transformation.json* content is shown later in this paper in the discussion of the case study (Figure 4).f

Filename	Contents
*.directory-version.json**	Contains a number specifying the version of the directory structure. Allows structural changes to be made while maintaining backward compatibility.
*.full-dataset-name.json**	Contains the full data set name for this directory
*.transformation-type.json**	Specifies the method of transformation for this directory – in this case the type is Configurable Transformation
derived.csv	The generated file containing the data set that is the result of the transformation
derived.variables.v1.csv	Contains information about the data set's variables, including name, label, type and optional mapping expression
derived-attributes.json	A set of key-value pairs specifying data set level attributes
input-datasets.txt	Contains a list of all data sets that are used as input for this transformation
lib-dependencies.csv	The names of code libraries and their associated versions that are referenced in the transformation
README.txt	Information for data customers about how the transformation is implemented
transformation.json	Specifies the steps that happen in the transformation. Each step is a block of JSON code (JSON object)

**these files are populated on data set creation and the user does not need to interact with them*

Table 1. Data Set Directory Files

EXPRESSION MAPPING

In transforming data sets to SDTM, it is convenient to be able to define a derived variable value using an expression. For example, in our implementation of SDTM we create the USUBJID by prepending a study ID to the SUBJECT. A mapping spreadsheet is a common tool for accomplishing this on a data set level. In the Configurable Transformation model, the Variable Map step type uses a spreadsheet to map the values in rows contained in an input data set to values in rows in a resulting output data set.

For Delphi, we wanted to leverage an existing expression language instead of creating our own. We chose Apache JEXL as the expression language as it is clear, well documented and well maintained and satisfies the following criteria:

- Is human and computer-readable
- Handles various data types
- Provides string, numeric and logical operations

JEXL is used with ‘Filter’, ‘Variable Map’ and ‘Add Derived Variable’ step types described below. We also created JEXL library functions organized into libraries for reuse of more complicated expressions. Table 2 shows an example of the use of JEXL in a variable mapping spreadsheet with some variables from the AE domain.

Name	Label	Type	Expression
DOMAIN	Domain Abbreviation	CHARACTER(2)	"AE"
USUBJID	Unique Subject Identifier	CHARACTER(15)	"300084"+asCharacter(SUBJECT)
AEACNOTH	Other Action Taken	CHARACTER(200)	AETRTOTH_RAW == "1" ? AETRTOSP : ""
AEREL	Causality	CHARACTER(40)	AEREL_DECODED.toUpperCase()
AEOUT	Outcome of Adverse Event	CHARACTER(40)	{ "1" : "RECOVERED/RESOLVED", "2" : "RECOVERING/RESOLVING", "3" : "RECOVERED/RESOLVED WITH SEQUELAE", "4" : "NOT RECOVERED/NOT RESOLVED", "5" : "FATAL" }[AEOUT_RAW]
EAEYN	Reported as Expedited AE	CHARACTER(1)	EAEYN

Table 2. Example Variable Mapping from AE Domain

TRANSFORMATION STEP TYPES

Within a Configurable Transformation, the discrete data operations are modeled as “transformation steps”. Those steps are executed in the order that they are listed in the *transformation.json* file. Each of the steps takes as its input the output of the step before it – except for the “start with”, “merge” and “union” step types which allow the user to specify which data sets are being used. The step passes its resulting data set along to the next step in the transformation (see Figure 2).

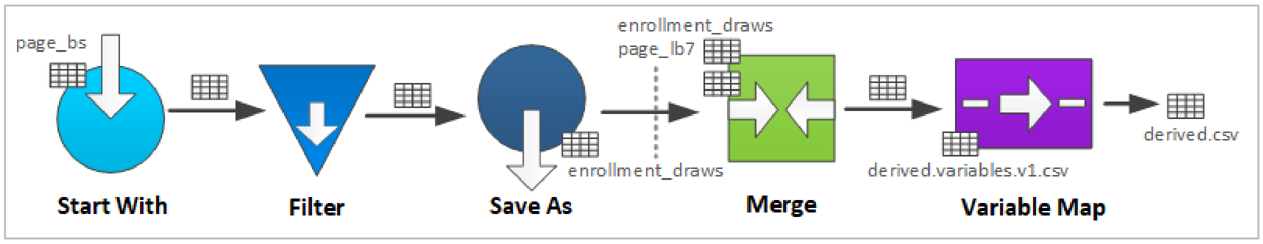

Figure 2. Transformation Steps and Data Set Handover

All steps in a transformation have a “transformation step type”, with different step types providing different behavior. Table 3 displays a summary of the different step types supported by Delphi 2.0. Some of the step types are common data operations, while others were designed for problems specific to the conversion of CRF data to SDTM domains. Each step type takes a particular set of parameters which specify, for example, which column to sort on or which data sets to merge together.

Symbol	Step Type	Description
🟪	Variable Map	A Variable Map uses a .csv file to map values to the current data set’s variables. The mapping can use an arbitrary JEXL expression that can reference any variable value in the same row of data as well as literals.
🟪	Add Derived Variable	The Add Derived Variable step type is a simpler version of the variable map step type. It allows the user to specify an expression that defines a new derived variable for each row of data.
⬢	Drop Variables	The Drop Variables step type allows the user to remove columns in the data set that are not needed.
▼	Filter	A Filter step has the effect of either keeping or filtering out only the rows whose variable values meet the specified criteria.
◆	Sort	A Sort step allows sorting of the rows in the data set based on one or more of the data set’s variables (in ascending or descending order)
⬢	First in Group	This step allows the user to group rows together based on specified variables, then keep only the first of the rows and discard the rest. Performing a Sort step before this one will ensure the rows are in the correct order.
🟩	Merge	The Merge step type provides the ability to join two or more data sets that share a common primary key (like a JOIN in SQL). The resulting data set will contain all non-key variables in their own column, with their variable name prefixed by the source data set.
🟩	Union	Union combines two or more data sets where each row in the input data sets becomes a row in the next data set. This has the effect of stacking the data sets on top of each other.
🟩	Append Lookup Variables	This step type adds variables from a specified data set to the current data set. Rows are compared based on “compareOn” criteria, and data are only added when values from the second data set match values in the first.
⬢	Sequence	Sequence provides the ability to add a variable representing the numbered order of the row in the data set based on a specified grouping and sort order. These sequence numbers are required by the SDTM standard.
⬬	Aggregate	Aggregate allows the Data Set Developer to turn a data set that consists of key value pairs into a wider data set where the keys are variable names.
●	Start With	This step type must be the first step in a transformation and must also follow a Save As step. It specifies which data set will be used in the transformation described by the following step.
●	Save As	A Save As step assigns a name to the current data set, making it available for use by steps later in the transformation.

Table 3. Configurable Transformation Step Types

DISCOVERIES

After the model was first designed, we created a prototype interpreter and started using it to map SDTM domains for a study. Both during this period of initial use of the model, as well as the earlier phase of gathering requirements for converting data to SDTM, we made several discoveries about the problem space. Some of these discoveries are discussed below.

VARIABLE MAPPING ALONE WAS INSUFFICIENT

When we were initially exploring solutions, we had thought that the CRF data sets (page_level data) could be transformed into SDTM data sets by simply performing a variable map that included use of an expression language. We found that there was significant need for re-arranging the data before that variable map action took place. Calculating study days, assembling subject visit information, producing baseline references, as well as other variables added quite a bit of complexity to the transformation. Even when the data were collected using CDASH standards, the CRF data needed many steps in order to be transformed.

TEMPORARY INTERMEDIATE DATA SETS ARE USEFUL

Another discovery was that it was helpful to create temporary intermediate data sets while designing a transformation. By adding a “save as” step type, these data sets were made available to steps later in the transformation (the sections of code that result in a saved data set were termed “segments”). If each of these intermediate data sets is created using a separate data set directory, the transformation code is more difficult to maintain.

PROCEDURAL WAS CODE NOT NEEDED

The Delphi system was designed to support different types of transformations, with the plan to eventually support procedural languages such as RStudio or Python. We had assumed when starting on this project that some of the more complicated domains would need to be transformed using procedural code. However, we found that all the domains that were required to model the CRF data could be implemented using the Configurable Transformation model. This result demonstrated that the model is flexible enough to work as a SDTM solution

CONTROLLED TERMINOLOGY MAPPING CAN BE CLEARLY SHOWN

One of the requirements for a transformation of clinical data to SDTM is for certain variables to be mapped to CDISC Controlled Terminology. Originally, we had planned to encapsulate the terminology in code libraries. However, in order to clarify the mapping, a decision was made to represent the mapping by using a case statement as shown below. This allows it to be clearer to others reading the variable mapping file which CRF field choices map to which controlled terminology variables. Table 4 shows the definition of the AEOUT variable and the mapping expression that applies the controlled terminology.

Name	Label	Type	Expression

AEOUT	Outcome of Adverse Event	CHARACTER(40)	{
			    "1" : "RECOVERED/RESOLVED",
			    "2" : "RECOVERING/RESOLVING",
			    "3" : "RECOVERED/RESOLVED WITH SEQUELAE",
			    "4" : "NOT RECOVERED/NOT RESOLVED",
			    "5" : "FATAL"
			}[AEOUT_RAW]

Table 4. Mapping of Controlled Terminology via JEXL

CASE STUDY: HIV PREVENTION TRIAL

The first study targeted for transformation to SDTM by our organization was an HIV prevention trial. The transformation was initially performed by a member of our software development team during the development of the Delphi system in order to work out issues in the model and provide a jump-start to the team that would be handling the SDTM transformation going forward. After development of the system, the transformations were completed by the SCHARP data standards analysts. The following section focuses on the implementation of that study.

EXAMPLE OF DEMOGRAPHICS (DM) DOMAIN IMPLEMENTATION

Different domains require different ordering of the transformation steps. Figure 3 shows the strategy we used for implementing the SDTM demographics (DM) domain via Configurable Transformations. In Phase 1, the steps are organized into 4 segments, with each saving out a temporary data set. Segments A-D prepare data sets for subjects who have special circumstances such as fatal adverse events, discontinuation, or were screened out. In Phase 2, the temporary data sets are merged in with data from the enrollment, demographics, and disposition forms. In the final phase all necessary data are properly filtered, steps are executed to incorporate the study ID, and then the final variable mapping takes place.

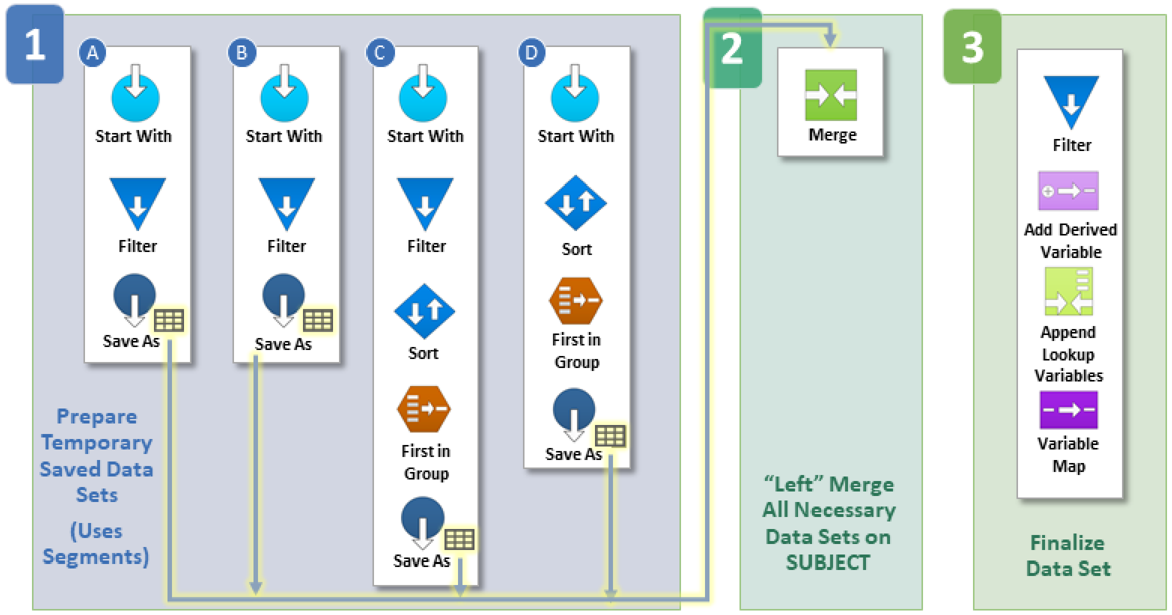

Figure 3. Strategy for Transformation of DM Domain

Transformation JSON Code for a Segment

As shown in Table 1, the *transformation.json* file contains a block of JSON for each transformation step. Figure 4 shows an excerpt from that file for the DM domain: the JSON used for transforming Segment C from Phase 1 displayed in Figure 3. The purpose of this segment is to create a data set containing one row for each participant who discontinued receiving the study product. The example code displays the different parameters being passed to the various step types and that comments may be used when defining a step:

```
{
    "_comment": "Start with data from the Product Hold / Discontinuation Log.",
    "stepType": "start with",
    "namedDataset": "page_dstrt"
},
{
    "_comment": "Keep records where subject is permanently discontinued",
    "stepType": "filter",
    "keepWhen": "(DSCONT_RAW == '2' or DSCONT_RAW == '3') and EXSTDAT_RAW == ''"
},
{
    "_comment": "Sort so most recent log line per subject is first.",
    "stepType": "sort",
    "orderBy": [
           {    "variableName": "SUBJECT",
                "order": "ascending" },
           {    "variableName": "ITEM_GROUP_REPEAT_KEY",
                "order": "descending" }
     ]
},
{
    "stepType": "first in group",
    "groupBy": ["SUBJECT"]
},
{
    "stepType": "save as",
    "namedTemporaryDataset": "dstrt_distinct_subject"
},
```

Figure 4. JSON Code for Temporary Data Set

Input Data Set Diagram for DM domain

As previously discussed, the Delphi web tool provides the user with a diagram which displays the input data sets used as sources for a given derived data set. Figure 5 shows the diagram for the Demographics domain, with the direct inputs emphasized. This diagram makes it clear which data sets are being used as sources and how many different forms are involved in producing this domain.

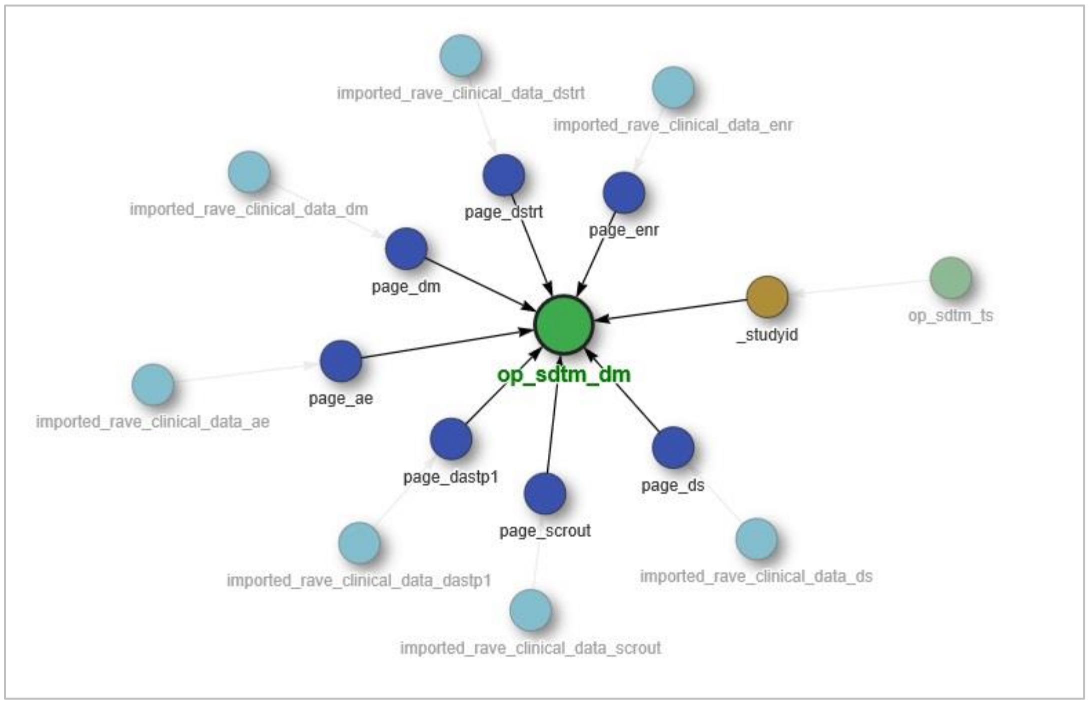
imported_rave_clinical_data_dstrt
imported_rave_clinical_data_enr
imported_rave_clinical_data_dm
page_dstrt
page_enr
page_dm
_studyid
op_sdtm_ts
op_sdtm_dm
page_ae
imported_rave_clinical_data_ae
page_dastp1
page_ds
page_scrout
imported_rave_clinical_data_dastp1
imported_rave_clinical_data_ds
imported_rave_clinical_data_scrout

Figure 5. Visualization of Input Data Sources for Demographics Domain

NUMBER OF STEPS REQUIRED FOR A DOMAIN TRANSFORMATION

After completing a study mapping using this model, we were surprised by the complexity of the transformation required. Although 60% of the domains were moderately complex or simpler (requiring 30 or fewer steps) the other 40% of the domains present progressively more complicated challenges: five of the domains require more than 70 steps and two over a hundred (see Table 5 and Figure 6).

The number of steps does not necessarily represent the complexity of a transformation. For example, the transformation for the LB domain, which has 129 steps, is conceptually easier to understand than other domains because it uses segments to create temporary data sets for each test type.

More step types can be added in the future if it is determined that they can reduce the transformation complexity and still retain the desired visibility.

NOTE: The trial design domains (TS, TA, etc.) contain metadata about the study and do not source from the clinical data. We chose to manage those data sets using a different type of transformation, so they are not represented in the data.

Moderately Complex or Simpler

Description	Domain	Num Steps
Associated Persons DM	APDM	8
Associated Persons PE	APPE	9
Medical History	MH	9
Adverse Events	AE	10
Protocol Deviations	DV	10
Concomitant Medications	CM	11
Inclusion/Exclusion Criteria Not Met*	IE	13
Exposure	EX	16
Associated Persons RS	APRS	18
Subject Characteristics*	SC	19
Demographics	DM	20
Associated Persons VS*	APVS	21
Disposition	DS	21
Device Identifiers	DI	24
Drug Accountability	DA	30

More Complex

Description	Domain	Num Steps
Subject Visits	SV	34
Reproductive System Findings*	RP	39
Associated Persons SC*	APSC	42
Clinical Events	CE	52
Associated Persons AE	APAE	57
Microbiology Specimen*	MB	72
Subject Elements	SE	88
Questionnaires*	QS	95
Vital Signs*	VS	110
Laboratory Test Results*	LB	129

* Findings Domains

Table 5. : Number of Transformation Steps per Domain

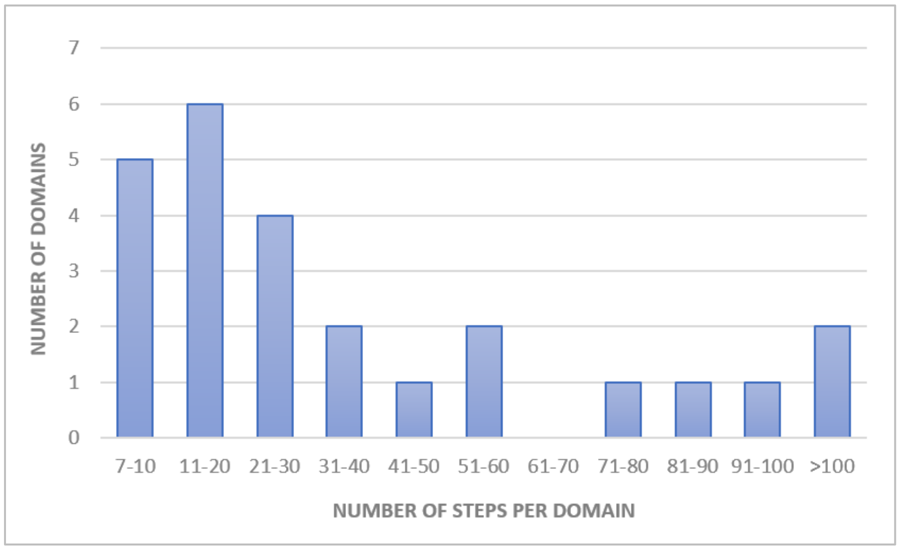

Figure 6. Distribution of Transformation Steps per Domain

PERSISTENT INTERMEDIATE DATA SETS

In our implementation of this study, we chose to create a few data sets termed "intermediate" that were not consumed by downstream customers but were only used as inputs for the SDTM data sets. We found that only a few of these data sets were needed across different domains and were useful to maintain. The rest of the preparatory data sets were managed using temporary data sets created through segments. We did find, however, that in some cases the strategic use of persistent intermediate data sets could simplify the final transformations. One example of this was a data set used to calculate EPOCH.

NOTE: Our system provides the ability to hide intermediate data sets from the downstream data customers.

CONCLUSION

In conclusion, we were able to implement a flexible model for SDTM data set transformation based on a set of basic data set transformation step types specified using JSON. The system uses a Java-based command line tool to interpret the configuration files and JEXL for variable mapping expressions that can be used via a mapping spreadsheet. We were able to map all the required domains for an HIV prevention study using this model and did not need to use any procedural code to create our operational SDTM. The conversion of CRF data to SDTM presents a complex challenge, and the final transformation code does reflect that complexity.

Our Configurable Transformation model uses a consistent approach to transform the data from CRF data sets to SDTM data sets, without the need for separate pipelines. The model also allows the system to visualize input data sets so that the customer knows where the data in each data set comes from. It also provides the ability to allow for the future diagramming of transformation steps for the user.

Finally, the Configurable Transformation model can be used in concert with an overall system to provide a validated platform for delivering derived data sets to customers using a datamart.

REFERENCES

McCallum, S. and S. Chan. 2011. "An Excel Framework to Convert Clinical Data to CDISC SDTM Leveraging SAS Technology." *Proceedings of PharmaSUG 2011*. Chapel Hill, NC: PharmaSUG. Available at https://www.pharmasug.org/2011-proceedings.html

van Bakel, B. 2016. "DIY: Create your own SDTM mapping framework." *Proceedings of PhUSE 2016*. Kent, UK: PhUSE. Available at https://www.phuse.eu/annual-conference-2016

ACKNOWLEDGMENTS

The authors would like to acknowledge Maura Kush for sharing her knowledge of SDTM, assisting in the mapping and conversion of the case study trial, and providing valuable input during requirements gathering for the system. We would also like to acknowledge Kobie O'Brian and Robin Schulmeister for their thorough work on refining the SDTM specifications and completing the conversion of the case study to SDTM, Corbin Charpentier for his software development contributions to the Delphi system, and Kathy Bowen for providing essential testing support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors via sc.delphi@scharp.org.