PharmaSUG 2023 - Paper MM-205

Automating SDTM: A Metadata-Driven Journey

Keith Hibbetts, Eli Lilly and Company;

ABSTRACT

SDTM is one of the deliverables in every clinical trial. Efficient creation of SDTM can save hundreds of hours of effort, yet automation of SDTM has proven to be a very difficult proposition. This paper details the journey Eli Lilly has been on pursuing SDTM automation. To use a car analogy, you need four wheels and an engine to drive. On this particular journey, the four wheels are: a robust set of standards, a metadata repository to store and maintain those standards, a set of generic macros for data set creation, and a programming process to utilize those macros. The engine is metadata. By defining a metadata model that not only defines the source and target but also the logic to convert the source to the target, we can build out the rest of the components to make this vision a reality. A proof-of-concept project based on this idea achieved 96% automation of SDTM variables in a test study. Now we're on the road to making this concept a production-ready reality.

INTRODUCTION

Here at Eli Lilly, we recently found ourselves with a big opportunity. The metadata repository (MDR as a generic acronym) that we had implemented several years ago was reaching the end of its technology lifecycle and needed to be replaced. The previous MDR only maintained our data collection standards. It had no study-level component. All downstream standards, such as the Study Data Tabulation Model (SDTM) and the Analysis Data Model (ADaM) were maintained in spreadsheets. No significant automation gains had been made in SDTM or ADaM creation at the study level. We wanted to take this opportunity to realize significant gains in efficiency in SDTM and ADaM creation, with specific focus on SDTM since the potential gains were greater in that space.

This paper will focus on the SDTM automation journey. In the abstract, I used the car analogy of 4 wheels and an engine. Those elements were:

- Wheel 1: a robust set of standards. This wheel was already in place. Lilly has placed a great deal of focus on standards creation and maintenance for many years. Data collection standards were in place for more than 700 data collection forms (internally referred to as Data Element Definitions or DEDs). SDTM standards were also in place for every SDTM domain relevant to those data collection standards.
- Wheel 2: a metadata repository to store and maintain these standards. We did not have this wheel: our previous MDR could not support SDTM standards, and that MDR was reaching the end of its lifecycle. We needed to implement a new MDR system with a broader scope of standards that it could support, as well as having a study-level component to create and maintain study-level specifications.
- Wheel 3: a set of generic macros for data set creation. We did not have this wheel. Generic, in this context, refers to macros that can be re-used across compounds, studies, and domains.
- Wheel 4: a programming process to utilize those macros. Without the macros themselves in place, we of course did not have this wheel.
- The Engine: Metadata. This wheel was partially in place. Our data collection standards were in place, our SDTM domain standards were in place, and the SDTM standards contained instructions on how to create the SDTM. But the transformation metadata (description of how the raw data would be processed into SDTM) needed to be much more granular to enable efficiency gains through automation.

GUIDING PRINCIPLES

With the key components identified, we also needed some guiding principles in what we wanted to accomplish. As relates to SDTM automation, those principles were:

- Produce submission ready SDTM domains. The end-result needed to be SDTM data sets and transport files that were ready to be submitted, and to be used as input to ADaM data sets. We did not want to produce “SDTM-Like” data that would need to be further refined to be submission ready.
- Save time in SDTM specification writing. While there are always parts of SDTM specifications that are highly protocol-specific, we want to minimize the amount of time needed to complete these SDTM specifications.
- Save time in SDTM programming. Similar to the specifications, there will always be a need to customize certain part of the code that produces the SDTM, we want to automate as much as possible to reduce the time spent by the study team writing code. In addition, when some custom code is necessary, it should easily be inserted into the process.
- Error checking should be built in “behind the scenes”. Each step of the way, error checking should be occurring without requiring additional steps. Simply by taking the steps to create/refresh SDTM, the study team should get the benefit of being alerted to any potential issues.
- The overall process must be flexible enough to adapt to upstream changes. Rather than being built on the assumption that all systems that collect and deliver raw data will always be exactly as they are today, the overall system should easily be able to adapt to any such changes.
    - Related to this principle, we made the conscious choice to not try to integrate the MDR with our EDC system directly, nor change our study build/edata processes any more than absolutely necessary. This had 2 major advantages. First, it ensured our SDTM automation solution would not be tied to any specific EDC system. Second, we would be able to achieve the SDTM automation faster by not having to take the time to integrate those 2 systems together.
- Make the right thing to do also the fastest thing to do. Timelines pressures are real, which can leave people in a position to choose between the fastest solution and the highest quality solution. The best way to mitigate is to ensure that the fastest solution is also the highest quality solution.
- The system must be flexible enough to allow study teams to react to the reality of their study. It’s impossible to foresee every scenario that every study team will ever face. There must be enough flexibility to allow the study team to react, even when that situation dictates that non-standard logic must be used.
    - Related to this principle, this meant the macros in the process must be 100% metadata-data driven. This means, when determining how a variable would be created, the macros never make assumptions about the logic, but instead use the transformation metadata to determine what logic to execute.

PROOF OF CONCEPT

Now that we had established what we wanted to accomplish, we needed to prove that it was feasible. We knew that we were going to need a vendor partner, as the MDR component would require the kind of software development expertise that is not prevalent within a sponsor company. There are several MDR products on the market, but we knew none would meet our exact needs out of the box. So we conducted parallel proof of concept projects with the following parameters:

- We would choose one Lilly study, and de-identify the raw, SDTM and ADaM data from that study.
- Select 3 ADaM data sets from the study (ADSL, plus one BDS class and one OCCDS class data set).

- Identify all SDTM domains needed to support those ADaM data sets selected. There were 11 domains needed.
- Identify all raw data sets needed to support those 11 SDTM domains. There were 17.
- Design a metadata model to hold the transformation metadata needed to enable automation. This design was done internally
- Build generic macros and a programming process to utilize the metadata to create the SDTM and ADaM data sets. These were built internally
- Select 2 vendor partners to build the MDR component: one vendor which would be developing a new system from scratch, and one vendor with an on-the-market MDR product that would be customized for our needs.
- Provide the same metadata model to both vendors.
- Set a 3 month window to complete the project.

The outcomes that we were looking to analyze were:

- Could the MDR product hold all of the metadata needed for the global standards for those in-scope items, including the transformation metadata for building SDTM from the raw data and to build ADaM from the SDTM data?
- Could the MDR hold the study-level metadata separate from the global standards?
- Could linkages between the standards (DED->SDTM, SDTM->ADaM) be demonstrated?
- Could the metadata be extracted in both machine-readable and human-readable formats (note, this could either be one format readable by both machines and humans, or two separate formats)?
- Could we use the macros developed and the programming process to recreate the selected SDTM and ADaM data sets? If so, what percentage of variables were we able to automate? The percentage was calculated by finding the total number of variables across all of the SDTM domains (counting a QNAM in a supplemental data set as a variable), then measure how many of those variables were created by the generic macros without need for custom code related to that variable. A similar calculation was done for ADaM.

Ultimately both parallel projects were successful. Both MDR solutions were able to demonstrate all of the points above, and the same level of automation was achieved with each solution. Those levels were 96% of the SDTM variables, and 81% of the ADaM variables. While we do believe that the level of SDTM automation will stay at or near that level once expanded out to our entire portfolio, we do anticipate that the level of ADaM automation will drop when expanded out.

Either product could have potentially been part of our solution. We chose to implement the on-the-market MDR product, which was the TCS ADD Metadata Repository offering from Tata Consulting Services (TCS).

With the concept proven, and the vendor partner chosen, we could begin the real journey to SDTM automation gains. Internally, we branded this effort as STAMP, which stands for STandards Automation Metadata Process. The STAMP branding encompasses the MDR, the macros, and the entire process built around them.

THE METADATA

Serving as the engine, the transformation metadata is the catalyst for the rest of the system. While this paper will not exhaustively explore all the metadata concepts in STAMP, it will highlight the most critical concepts.

DOMAIN LEVEL METADATA

Metadata is stored about each SDTM domain. Some of the key domain level metadata elements are:

- The name of the domain
- The domain label
- The name and label of the supplemental data set (if any) for the domain

VARIABLE LEVEL METADATA

Attributes are also defined for each variable within a domain. Note that STAMP treats QNAM values intended for the supplemental data set as a domain variable. Thus, the variables listed in a domain contain both the variables in the parent data set (i.e. AE) and the possible QNAM values in the supplemental data set (i.e. SUPPAE). Some of the key variable level metadata elements are:

- Variable name
- Variable label
- Type (character or numeric)
- Length
- Order that the variable should appear in the domain
- Whether the variable is required (Y/N)
    - Note that this slightly different than the CDISC CORE attribute definition. While any variables with a CORE attribute of REQUIRED or EXPECTED will have a Y value for this attribute, we may also consider some variables with a CORE attribute of PERMISSIBLE or suppqual variables as required.
- Is the variable a suppqual variable (Y/N)
    - When Y, this indicates the variable will be output as a QNAM in the supplemental data set instead of in the parent data set of the domain.
- Acceptable Null Percentage (0-100)
    - This attribute defines a maximum percentage of records that should realistically have a null value, assuming this variable is being populated in the domain. This is used in a check during the macro execution. A message is generated anytime the percentage of records with a null value exceeds the acceptable null percentage, as that could indicate an issue with either the specifications or the programming.
- Sequence Variable
    - This attribute is used to identify which variables should be used in the sort step when creating the xxSEQ variable for the domain.
- Drop From XPT (Y/N)
    - This attribute highlights one of the innovations within STAMP. A variable can be defined as part of the SDTM domain that is in the output data set but not included in the .xpt file. This allows us to make the SDTM creation process more efficient in the following ways:
        - Record level traceability is enabled. This is a concept that we are borrowing from ADaM and applying to SDTM. Additional variables are added to identify where in the raw data a particular record comes from. When data issues manifest and are first caught in SDTM, this can help you more quickly identify the source in the raw data.
        - Creation of the RELREC domain is more efficient by not treating it as a silo. For example, a form that collects concomitant medication data (which thus maps into the CM domain) might collect the group id of the adverse event the medication

was given in response to. That should result in a pair of records being created in RELREC to link a record in CM (the concomitant medication) to a record in AE (the adverse event the medication was given in response to). In STAMP, the information about the adverse event is included in the CM domain. After all other SDTM domains have been created, a RELREC-related macro runs which checks all other SDTM domains for such relationship-type variables, and uses them to create paired records in RELREC. This is much more efficient than RELREC processing going back to the raw data to find such relationships, and then reverse-engineering the location of those records in SDTM.

        - Similar to RELREC, when a comment is collected on a form that comment is included in the domain. For example, if a comment was collected on a lab record, that comment is included in the LB domain. After all other SDTM domains are created, a comments macro runs to harvest all of these comments and create records in the comments domain (CO).
    - Because the .xpt files are the files included in a regulatory submission, and also used for running validation in Pinnacle21 Enterprise, dropping the variables from the .xpt files creates no validation or submission issues.
    - To ensure these variables are not inadvertently used during ADaM creation, the variable name themselves include the characters NO_USE_IN_ADAM. Note that because the variables are not included in the .xpt file, they are also not subject to a variable name length limitation of 8.

TRANSFORMATION METADATA

The transformation metadata is stored at a level of one record per SDTM variable per TAGGED_FORM. A TAGGED_FORM in STAMP is an input data set that is contributing records to the SDTM domain. By storing the transformation metadata in this manner, it allows the metadata to be both extremely granular and allows for easy import from global standards to study level specifications. Note as well, every SDTM variable has transformation metadata populated for every TAGGED_FORM, even if that particular source does not collect anything that populates that particular variable. This ensures there is never any question about whether the variable was overlooked in the transformation metadata mapping process.

Some of the key elements of the transformation metadata are:

- Transformation type.
    - This is the most important element of the transformation metadata, as it is what drives what type of logic is used to populate that SDTM variable for records coming from that TAGGED_FORM source.
        - Approximately 100 transformation types were created as part of the STAMP system.
            - Some are highly reusable across many variables in many domains. Examples of this include:
                - DIRECT – used when an input variable has the exact value needed in the output SDTM variable
                - CONSTANT – used when all output records should have the same constant value
                - NULL – used when all output records should have a null value.
                - CONCAT – used when multiple input variables should have their values concatenated.
            - Some are used for a specific type of variable across multiple domains. Examples of this include:

        - EPOCH – used for our default logic for deriving EPOCH
        - STUDYDAY – used for variables that are expressing the amount of time between the subject's reference start date and the data point.
        - SEQ – used to create the xxSEQ variable
    - Some are very specific to particular SDTM variables and/or TAGGED_FORMS.
  - The transformation type determines whether the variable will be created automatically by one of the STAMP macros, or whether it will be created by custom code from the study team.
  - It will determines (along with additional metadata mentioned below in some cases) when the variable will be created during the programming flow.
- Input Library
  - Only relevant for certain transformation types, this identifies the library the input data is located in. Typically the values will be either:
    - RAW – indicating the data is in the raw library
    - SDTM – indicating the data comes from another SDTM domain, or is dependent on another variable within the same domain
- Input Data set
  - Also only relevant for certain transformation types, this identifies the name of the data set containing the input data
- Input Variable
  - Also only relevant for certain transformation types, this identifies the name of the input variable.
- Additional parameters
  - Some transformation types require additional parameters to be populated in order to give the macros all of the information needed to create the variable. These parameters vary depending on what the transformation type is. For example, the CONCAT transformation type contains the following parameters:
    - CONCAT_VARS – the names of the variables to be concatenated, listed in the order they should be concatenated
    - PREFIX – the prefix to put on the output value, if any
    - SUFFIX – the suffix to put on the output value, if any
    - DELIM – the delimiter used to separate the values in the output, if any

MISCELLANEOUS METADATA TABLES

Additional tables are needed to store metadata for other special cases. A few of these tables include:

- TRANSPOSE
  - In situations where a single input record can create multiple output records, some variables in the SDTM domain may require different logic across those output records.
    - For example, a questionnaire with 10 questions will create 10 output records. The logic to populate variables such as QSTESTCD, QSTEST, QSORRES, and potentially other variables will differ across those 10 records.

    - The transpose table identifies the key variable that triggers the creation of the output record, and then contains the transformation metadata for each of the impacted SDTM variables.
        - Using the same example as above, TRANSPOSE would have a row for each of the 10 questions. Within that row it would define how QSTESTCD, QSTEST, and QSORRES should be populated on each of those output records.

- CT_CONVERT
    - In situations where an input variable does not contain the values needed in the output variable, but each value in the input variable can be mapped to a specific output variable, CT_CONVERT holds that input value-output value mapping.
- UNIT_CONVERSION
    - Holds the formulas for converting from one input unit of measure to another.

ROBUST STANDARDS

For a system like STAMP to be possible, you need a robust library of standards to driver your data collection. This provides the sources for the transformation mapping. It is also critical that those data collection standards are followed at the study level, thus they must be a built with an appropriate level of flexibility. Lilly has maintained data collection standards for many years, so this wheel was available when the STAMP project began.

METADATA REPOSITORY

As mentioned earlier in the paper, Lilly partnered with TCS for the metadata repository element of the project. This partnership included significant customization of the MDR to allow it to store the specific metadata model designed for this SDTM automation solution.

The metadata repository is a key component because trying to maintain metadata with this volume and level of granularity would be extremely difficult to do using spreadsheets or other similar technology. The metadata repository not only provides a robust database to store all of this metadata, it provides a user interface that includes built-in functionality that improves the quality of the metadata being maintained. For example, when populating transformation metadata, the transformation type column is populated via a drop down, meaning the user does not need to memorize the exact name of each transformation type. When the transformation type is selected, if any additional parameters are defined for the transformation type they appear in a dialog box which includes a description of each parameter. Another example is that any input variable being selected is done through a dialog box that connects to the standard for that source (whether that source is a DED or an SDTM domain), so the user is selecting from a list of variables instead of typing in the variable name.

But perhaps the biggest advantage of using a metadata repository is that the metadata itself can be linked. This provides two major advantages. The first is that it makes assessing the impact of changes to standards feasible with little effort. For example, if a change is being considered to a variable on a DED, you can very quickly find what SDTM variables are using that variable. The other advantage is that the metadata repository contains a study-level module for SDTM specifications. Because the metadata is linked, and the transformation metadata is so granular, SDTM specifications become a much less time-consuming task. Simply selecting and importing the data collection standards being used for the study and executing a function within the metadata repository (which takes approximately fifteen minutes to complete) will import in all needed SDTM domain metadata and transformation metadata from the global standards. At that point the SDTM specifications are 90-95% complete, as the only parts that require updates are things that are very protocol specific, such as applying the correct arm codes in demographics, defining how elements start and stop in the Subject Elements (SE) domain, timepoint information from protocol schedules, and defining constant dosing intervals in the Exposure (EX) domain.

While only a small percentage of the specifications will need to be updated on a typical study, the study team has complete access to update any of the metadata at the study level. This allows the study team to have the flexibility to react to any scenario faced by the study.

Once SDTM specifications are completed, they are exported from the MDR and saved in our statistical computing environment, where they serve as the metadata driving the SDTM creation. Any changes to the specifications needed after initial completion are made in the MDR and then re-exported.

AUTOMATION MACROS AND THE PROGRAMMING PROCESS

These two concepts will be discussed together, as they are highly intertwined.

Each study follows the same programming process. The programs used in the study consist of a driver program that calls programs for individual domains (in some cases, inter-related domains may be processed within the same program).

DRIVER PROGRAM

The driver program sets all of the paths needed, sets all options needed (ex. Version of the MedDRA dictionary), and calls the first STAMP macro in the process, SDTM_SPEC_IMPORT.

SDTM_SPEC_IMPORT reads in the specifications file (which was exported from the MDR) and prepares that metadata for use by the rest of the process. Below are a few examples of the types of functionality included in this macro:

- Checks all specifications metadata for completeness, and generates messages for any potential issues:
    - Example: if a particular row of transformation metadata is using the CONCAT transformation type, but the CONCAT_VARS parameter is not populated, generates a message.
- Compares the transformation metadata against the data in the raw library, and makes adjustments on the fly based on what it finds:
    - Example: one of our AE DEDs includes a variable that collects the related imaging agent of the adverse event. The SDTM AE variable AERLIMA stores this information, and the transformation type to populate AERLIMA is DIRECT (meaning the input variable contains the exact value needed in the output variable). However, this item on the DED is not mandatory. If a particular study uses this AE DED but did not include this variable, the transformation metadata for AERLIMA for this form will get changed on the fly to a NULL transformation type.
    - Another example: many DEDs have both a date and time variable, but the time is often not mandatory and is frequently not collected on the study. xxDTC variables in SDTM should contain both date and time if both are collected, and typically the transformation metadata for xxDTC variables will use the DATE transformation type. DATE has a parameter that lists either a single variable (if only date is collected) or two variables (if date and time are collected). If SDTM_SPEC_IMPORT finds that the time variable is not present in the raw data, it will remove it from the parameter so only the date variable remains.
    - This saves tremendous time in the specification definition process. Instead of manually spending time to adjust the specifications based on whether optional variables were included in the data collection, the SDTM specifications can assume all variables are included and let SDTM_SPEC_IMPORT adjust the specifications based on what is actually contained in the raw data.
    - This also means SDTM specifications do not have to be adjusted if a post-production change in data collection adds a variable into a form.

- After all such adjustments are made, drops any SDTM variables that contain only NULL transformation types in the transformation metadata.
    - SDTM variables do not have to be manually from the specifications if none of the input sources used in the study are populating that variable.
- Compares all variables contained in the raw data library to a complete list of variables used in the transformation mappings, and generates a list of any variables not used. These could represent an oversight in the SDTM specifications.
- Determines which macro will create the variable (based on the transformation metadata).

DOMAIN PROGRAMS

The driver program calls programs for each of the domains required for the study. At the end of the program execution, the full SDTM domain is available, data sets and .xpt files for both the parent data set and (if applicable) the supplemental data set.

While a few domains have specialized processing, most follow the same basic logic flow. That logic flow involves consistent use of a small number of STAMP macros, with opportunities to insert custom code in between those macro calls. This logic flow is what enables such a high degree of flexibility. And because the macros themselves are 100% metadata driven, most changes to SDTM specifications will not result in a need to update the programs themselves. The macros are using the metadata to drive the creation and execution of code to produce the output data sets every time they are executed. So while a specification change will typically result in a macro producing and executing different code, it rarely results in the need for a human to update the domain program itself.

Below is an example of a typical domain program. This is the actual program used to create the medical history domain (MH) in the proof of concept. For context, the MH program in the original study used in the proof of concept project consisted of approximately 240 lines of code:

```
%mh;

%sdtm_raw_data_process(domain=mh, prid=mhpresp1001);
%combine_sdtm(domain=mh, prid=mhpresp1001,
input_dataset=mh_mhpresp1001_mac1output);

%sdtm_raw_data_process(domain=mh, prid=mh7001);
%combine_sdtm(domain=mh, prid=mh7001, input_dataset=mh_mh7001_mac1output);

%seq(domain=mh, input_datasets=mh_mhpresp1001_mac2output
mh_mh7001_mac2output);

%sdtm_output(domain=mh, input_dataset=mh_seq_output);

%mend mh;
```

As you can see, the amount of code needed is greatly reduced. In this example, the domain was 100% automated, as no custom code was needed to create any of the SDTM variables. If custom code had been needed to derive any variables, it simply would have been inserted in between two of the macro calls. Below is a summary of each of the macros appearing in the code above:

- SDTM_RAW_DATA_PROCESS
    - Is run for a specific input data set
    - If more than one input data set is contributing records to the domain, this macro will be called separately for each of these input data sets.
    - Will create any SDTM variable that only depends on information available in the raw data.

        - Which variables those are is determined by SDTM_SPEC_IMPORT based on the transformation metadata.
        - In most instances, the majority of SDTM variables are created by this macro.
- COMBINE_SDTM
    - Is run for a specific input data set
    - If more than one input data set is contributing records to the domain, this macro will be called separately for each of these input data sets.
    - Takes a data set in the work library as input
        - If no custom code is needed, this will be the data set output by SDTM_RAW_DATA_PROCESS.
        - If custom code is needed, this will be the data set in the work library which contains both the variables created by SDTM_RAW_DATA_PROCESS and the variables derived in the custom code.
    - Will create variables that depend on information contain in other SDTM domains, or that depend on another variable within this domain.
    - Which variables those are is determined by SDTM_SPEC_IMPORT
- SEQ
    - Is run only once for the domain
    - The input is one or more data sets in the work library containing all SDTM variables for all input sources. If more than one data set is input, appends into a single data set.
    - Creates the xxSEQ variable for the domain
- SDTM_OUTPUT
    - Is run only once for the domain
    - The input is a data set that contains all SDTM variables for the domain.
    - Applies all attributes (labels, lengths, variable order, etc.)
    - Conditionally drops variables with only null values.
    - Outputs both the parent (i.e. MH) and supplemental (i.e. SUPPMH) data sets and .xpt files.

OTHER BENEFITS

Besides the high degree of automation, this process also provides additional benefits to the study team. This include:

- Issue checking
    - Each macro used in the process is checking for potential issues, and generates messages around those issues. At the end of the driver program, those messages are output in an excel file. The excel file is nicely organized into three tabs, one which contains issues that must be resolved, one which contains issues that must be reviewed (and may need to be resolved), and one is just informational and thus not require review.
- Define.xml metadata
    - Because all of the specifications and output data are available, each time SDTM is created or refreshed, the define.xml metadata is output automatically without any additional steps needed by the study team.

GETTING STARTED

Because the entire process is driven by the metadata in the SDTM specifications, once those specifications are completed the creation of the programs needed is largely automated. At the beginning of a study, a macro called BUILD_FIRST_DRAFT_PROGRAMS is called which reads in the specifications metadata and builds first drafts of the driver and domain programs. This includes placeholders and/or initial values for all paths/options in the driver program, calls to the domain programs in the driver program, all macro calls within the domain programs, and comments inserted for all variables requiring custom code. With this single macro call, approximately 95% of the SDTM programming is complete.

A NOTE ABOUT SDTM VALIDATION

Within the STAMP process, SDTM data is validated via independent replication. Two versions of each of the STAMP macros were created, each written independently. One version is for use in the primary-side programming for a study, and the other is for use in the independent-replication side programming. This allows independent replication to occur, but with both sides getting the advantage and time savings of automation.

CONCLUSION

In conclusion, this is an exciting time at Lilly. The work to build the SDTM automation portion of STAMP is complete, and our first live studies using the STAMP process have just had first patient visit as of the time this paper was written. The journey isn't over, but it has reached an important milestone that will allow us to start reaping the benefits of SDTM automation.

CONTACT INFORMATION

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

Keith Hibbetts
Eli Lilly and Company
Keith.hibbetts@lilly.com