Paper AD17

Customizing SAS® Data Integration Studio to Generate CDISC Compliant SDTM 3.1 Domains

Tatyana Kovtun, Bayer HealthCare Pharmaceuticals, Montville, NJ

John Markle, Bayer HealthCare Pharmaceuticals, Montville, NJ

Chris Treglio, Bayer HealthCare Pharmaceuticals, Montville, NJ

ABSTRACT

DI Studio provides an interface to the SAS® Metadata Server for designing and running metadata driven processes. This paper summarizes our experience in using the out-of-box features of DI Studio for a CDISC compliant SDTM submission. It also describes enhancements we made to DI Studio to create a more business oriented, user-friendly, and secure environment for the generation of SDTM general observation classes. It provides some details on the implementation of all the steps required to convert our data to SDTM format, from creating mapping specifications, through designing and running domain jobs, to validating SDTM domain datasets, and, finally, generating define.xml (CRT) file.

Specifically, the paper describes

- advantages and drawbacks of using DI Studio as SDTM generation tool
- our repository hierarchy and metadata tree structure adapted for SDTM related tasks
- an overview of a working application for SDTM production, including additional modules (Java plugins) that were built specifically to initialize a new study repository, assist in defining mapping specifications, and generating define.xml
- an 'opportunistic' approach to complement SAS Metadata with an externally stored auxiliary table that holds our business metadata to 'drive' an automated SDTM generation
- successful practice in building job templates, that are completely reusable across SDTM domains as well as across submission projects.

Finally, we are going to present our 'pros' and 'cons' of using DI Studio as a tool for SDTM generation and suggestions how to make DI environment more robust and more attractive for experienced SAS programmers.

INTRODUCTION

As CDISC becomes the preferred FDA (soon to be required) format for SDTM submission, the pharmaceutical industry is becoming increasingly interested in developing robust and cost effective CDISC compliant submission process. One of the areas of research in this direction is SAS9 Business Intelligence Platform. The platform components that are of particular interest are Metadata Server, which is the core of this architecture, and the client specifically designed for data transformation tasks – Data Integration Studio. In this paper we will compare three different approaches to prepare CDISC SDTM submission with the focus on our innovative techniques implemented through CDISC focused customization of DI Studio. The scope of this paper does not include generation of Trial Design models.

BI PLATFORM VS. TRADITIONAL SAS/MACRO CODING

In broad terms, SDTM generation is a process of transforming collected data (source data) to standardized format (target data).

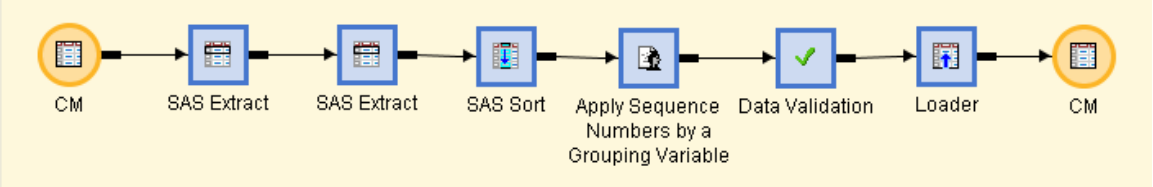
Each study is unique. This implies uniqueness of collected data, complicating industry efforts to develop and maintain standardized databases (sources). Output standards also change occasionally with version upgrades. Consequently, programs created by means of base SAS coding – even reinforced by an assortment of standard macros – are hardly reusable across different studies, require a substantial manual effort, are prone to errors, and in the end do not provide a cost effective solution.

On the contrary, metadata driven processes supported by new architecture, specifically by the Metadata Server and Data Integration Studio, significantly reduce the task of manual coding as DI Studio takes on the role of code

generator. Reusability and maintenance also impriove, because a significant part of any SDTM related data processing contains the same sequence of steps, most of which are available out-of-box standard DI transformations.

These common steps are: extract data from sources, join data, apply mapping logic, sort in the required order, assign sequence number (the xxSEQ variable), validate with PROC CDISC, load to SDTM domain, and generate define.xml.

Fig.1 Example of a Typical SDTM Job Flow



Although most of these transforms are standard, you would still need to perform some manual configuration, such as mapping definition, parameter definition, and options to adapt these transforms to the specific characteristics of your input and output data. Whether light or extensive configuration needs to be applied depends upon the complexity of your business logic. When mapping logic is complex, which is often the case, this manual configuration sometimes involves nested CASE expressions, or nested SAS functions, or both. Using the base Data Integration Studio product, these expressions need to be coded in the Expression Builder, making the process error prone and time consuming.

We completed that journey in our first pilot DI Studio based SDTM project.

USING DI STUDIO TO CONVERT OPERATIONAL DATASETS TO CDISC DATA STRUCTURES (PILOT IMPLEMENTATION)

Pilot Implementation Highlights

- The basic idea was to explore ETL Studio standard features for Integrated Study Summary submission.
- The business prepared spreadsheets (one per domain) containing referential guidance on mapping between Berlex domain variables and STDM.
- A separate Job was created for each SDTM domain.
- Derived variable were defined within Extract transformation Mapping tab. All expressions were manually typed in Expression Builder.
- After each Job was developed, it was submitted for execution to generate SDTM table.
- A Java Plug-in was developed to enable users to add supplementary metadata to meet CDISC requirements, to generate the define.xml file, to navigate through the metadata, and to validate for completeness.

Results from Pilot Study

Positive:

- Overall, the initiative proved successful: all intended CDISC compliant SDTM domains were created using exclusively DI tools within a reasonably short period of time
- Centralized metadata storage allowed for efficient define.xml production.
- Nice visual environment with easy to read job flow diagrams
- Secure Change Management environment
- SAS supplied backup and restore tool (%omabakup)

Negative:

- The initial process of creating mapping spreadsheets was tedious, mapping information was not part of the automated process, used just for reference

- ETL Mapping tab and Expression Builder do not provide 'friendly' environment for entering complex mapping logic, prone to typos, errors and difficult to debug
- Transposing data from horizontal to vertical structure with subsequent derivation – very wearying 'hardcoding' task
- Jobs were project (input data) specific, hardly reusable across different studies
- From the beginning, the jobs were built individually, one per domain, and even though there were strong structural similarities, the steps inside each job had to be individually configured (manual effort) for the specifics of input and output data.

DEVELOPING DI BASED SDTM APPLICATION TO AUTOMATE CDISC COMPLIANT SUBMISSION

To circumvent what was identified as deficiency in using just out-of-the-box DI Studio features, we decided to build our own components – Java plugins – specifically to facilitate SDTM generation. That was the first and most extensive area of customization.

In addition, experience gained during the pilot study helped to realize the roles of different steps in SDTM generation. Some steps in job flow diagram were identified as domain specific, and some as generic, i.e. actually common to all domains. The idea of generic job modules came forward.

The next step toward customization was to enable these generic modules to adapt themselves to the specifics of input-output data. This was achieved through the automated integration of externally stored mapping specifications to SAS macro code.

NB: Enhancing DI GUI with custom built plugins is fully supported by DI Studio design. However, the idea of generic (input and output independent) jobs is not supported by current release. (Fig.1 provides an illustration that input and output table icons are necessary component of a job flow) Nevertheless, engaging Base SAS as another API to the Metadata Server helped to bypass limitations of the current DI release.

HOW IS OUR METADATA ORGANIZED?

When designing metadata driven processes, it is extremely important to organize metadata in a straightforward way.

On the Metadata Server, metadata is arranged in repositories. Repositories can be dependent upon other repositories, or can be self-contained (although not absolutely self-contained). The core repository, upon which all other repositories depend, is the Foundation repository, which stores the most general information necessary for the rest of the metadata build-up to function properly. This essential information includes physical machine name, port numbers, Workspace Server (SAS processor) name, security information. All other repositories are arbitrary (called custom) and are dependent of Foundation repository.

For our purposes, it was decided to build four-level repository structure:

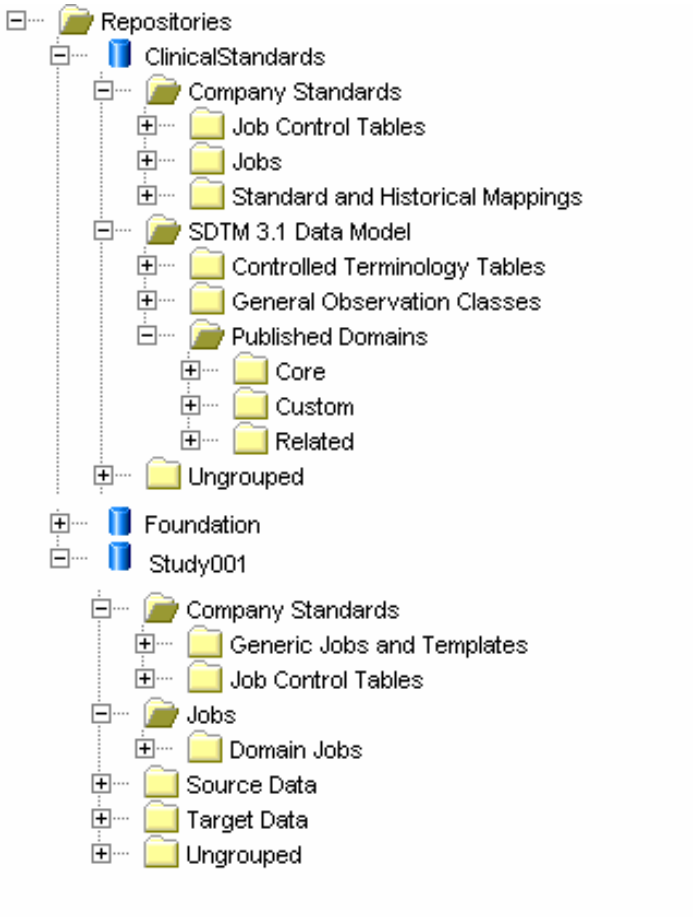
1) Foundation Repository – as a necessary component of the metadata server – which is always present
2) Clinical Standards Repository, to hold external (CDISC) and internal (company specific) standards, which depends on the Foundation repository
3) Study Repositories, which will store all study related metadata needed for SDTM generation and depends on Clinical Standards
4) Various Project repositories, one per user per study, where our users could do their work.

Foundation repository is created immediately after the metadata server installation.

Clinical Standards repository tree was created manually. Its main sub-trees are SDTM 3.1 Data Model and Company Standards. SDTM 3.1 Data Model contains CDISC SDTM domains metadata, including Control Terminology Lookup tables and Published Domains. Company Standards contains company specific metadata used for SDTM generation. Below we describe in details what we consider to be Company Standards.

Study repository, one for each study, is created and initialized by the Clinical Administrator Assistant, a custom Java-plugin. Part of the content of this repository gets created during initialization, including some objects copied from Clinical Standards repository (Job Templates, for example). The rest of the content is either created manually by copying and modifying existing metadata objects, or get actualized as an intended by-product from a related activity.

Fig.2 Metadata Structure in Clinical Standards and Study repositories



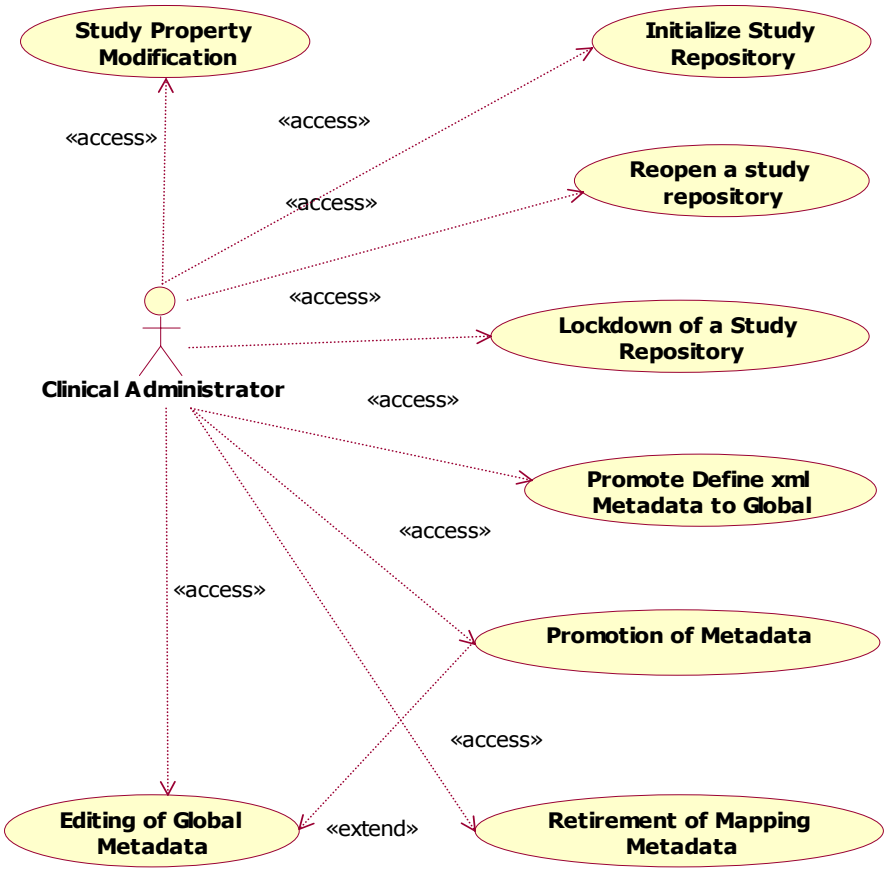
APPLICATION COMPONENTS

Table 1. Comparative overview of three custom Java plugins:

	Clinical Administrator Assistant	**Mapping Assistant**	**CRT XML Generator**
Used By	Clinical Administrator	Clinical Programmer	Clinical Programmer
Repository Updated	Clinical Standards repository, Study Repository	Study repository, external SAS datasets called “Control Tables”	Study repository
Purpose	Update Standards, Initialize Study repository, Move Standards between Clinical Standards and Study repositories Security Administration, Lockdown Study Repository	Create, Update, and Validate Mapping Control Tables	Enter Additional Metadata as Extended Attributes, Generate define.xml

Clinical Administrator Assistant

Fig.3 Functionality of Clinical Administrator Plugin



Restricted to those users within the business who are designated as “Clinical Administrators,” this plugin handles the following tasks

- Study Repository initialization, performed at the beginning of a submission preparation, creates or copies the appropriate metadata objects into the Study Repository.
- “Standards” Editing enables administrators to establish mapping standards, which take effect before manual mappings are made. Often mappings from our source data to the CDISC SDTM structure follow a predictable pattern, which makes “standard” definition useful.
- Study Lockdown (upon completion) removes all permissions from the study, so we can be sure that the data that was submitted to the FDA is always the version we have stored historically. By ensuring that our repository is identical to the state it was in at submission, we know that we can regenerate he submission at any time.
- Miscellaneous Properties, among which is the specific version of CDISC SDTM with which the submission will comply.

CRT XML Generator

Fig.4 Functionality of CRT XML Generator Plugin

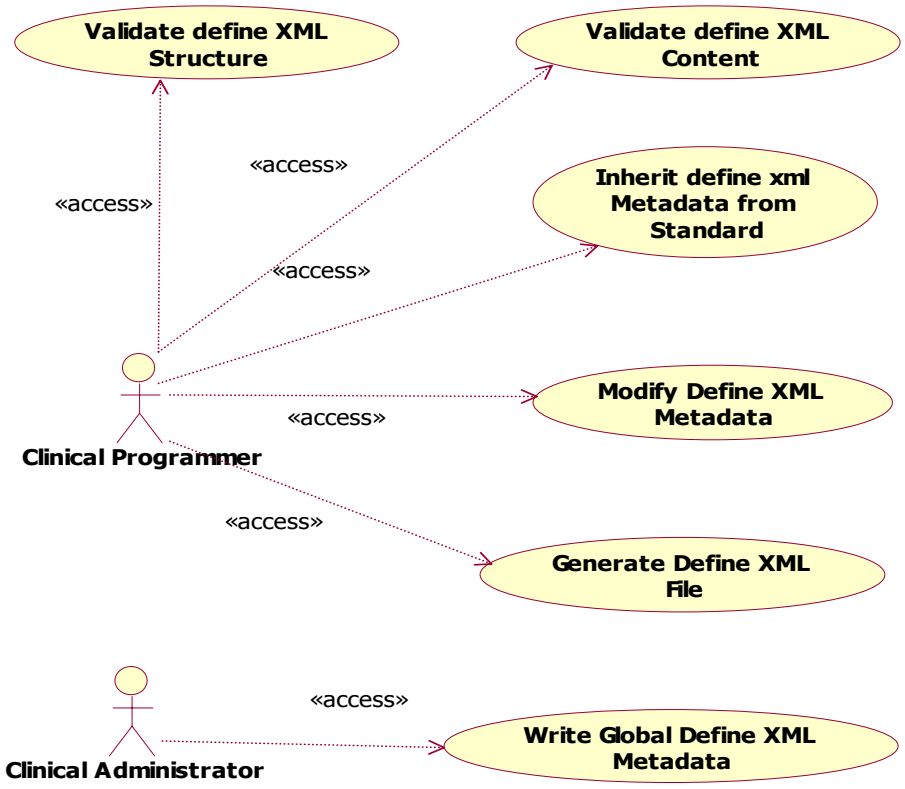
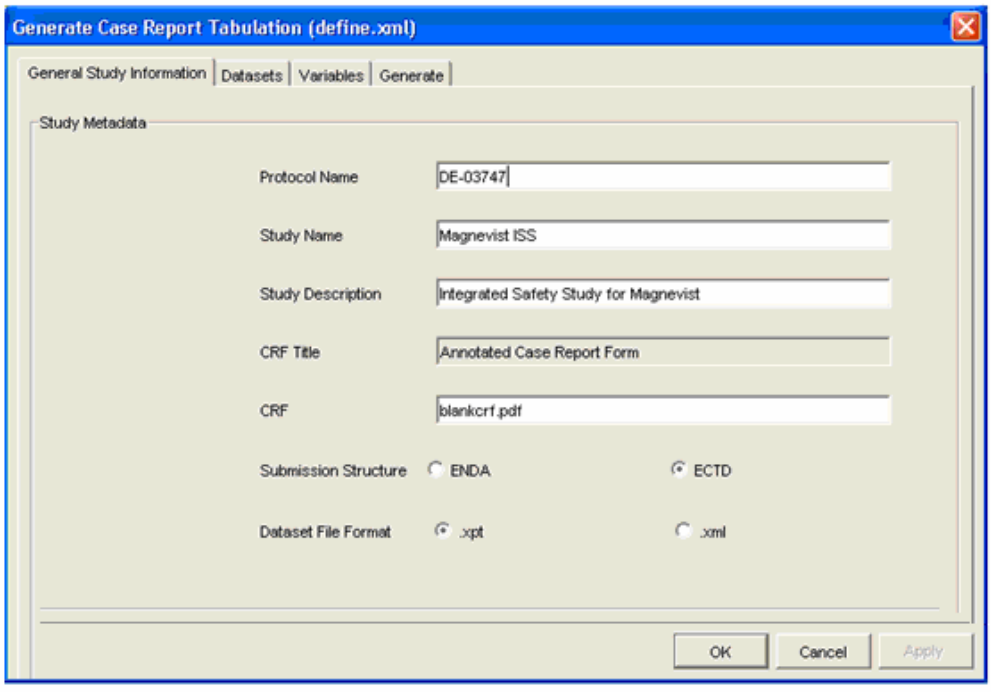


Fig.5 CRT (Define.XML) Generator Interface.



The CRT XML Generator (also known as the “Define.xml Generator”) offers a graphical user interface to enter and modify the metadata necessary to build the “define.xml” case report tabulation, as well as to generate and validate the define.xml.

The metadata required for the define.xml falls into three categories: study-level metadata (such as the version of the SDTM that is being submitted, or the codelists in use), table level metadata (such as the “role” of a particular dataset), and variable level metadata (such as the datatype, or comment, associated with a variable). Some of the metadata required is available by default from the metadata server, like variable types. For other pieces of metadata, our plugin builds custom metadata objects within the Study repository. For table and variable level metadata, our plugin attaches “ExtendedAttribute” objects to the object in question. For Study level metadata, our plugin builds “TextStore” objects which are all contained within a custom hidden Tree object in each study repository called “STUDY_METADATA”.

Mapping Assistant

The Mapping Assistance is the most sophisticated GUI extension that we’ve coded to Data Integration Studio. It guides our users through the creation of mapping specification in the format compatible with SAS syntax rules. As a result, the content of the table can be integrated into DI generated SAS code.

As mentioned above, two main deficiencies of the Pilot Implementation were:

a) The tedious task of manual creation of Excel spreadsheets with mapping information. These specifications simply served as guidelines and were not part of any automated code generation.

b) The even more tedious task of entering extensive derivation logic into DI Expression Builder.

To rectify these two deficiencies, the new application turns the informal “mapping specification” into what we call Mapping Control tables, which act as input to the SAS code generated by DI Studio on-the-fly. More about Mapping Control Table see section

The control tables take the place of the informal mapping “specifications” provided in the pilot. The Mapping Assistant provides more the robust and user-friendly environment for entering, validating, and storing derivation logic creation. As a result, the debugging, and maintenance of SDTM jobs is much more efficient and provide for their greater reusability. Also, because the time consuming and error-prone translation of the specification into code is removed, our mapping process takes less time and is more accurate. For the purpose of documentation, technical specifications can be generated automatically from the mapping control tables with simple SAS code.

HOW DOES MAPPING CONTROL TABLE GET CREATED AND UPDATED?

Our Mapping Assistant guides users first through table-to-table and then through variable-to-variable mapping process.

Fig.6 Table-to –Table Screen

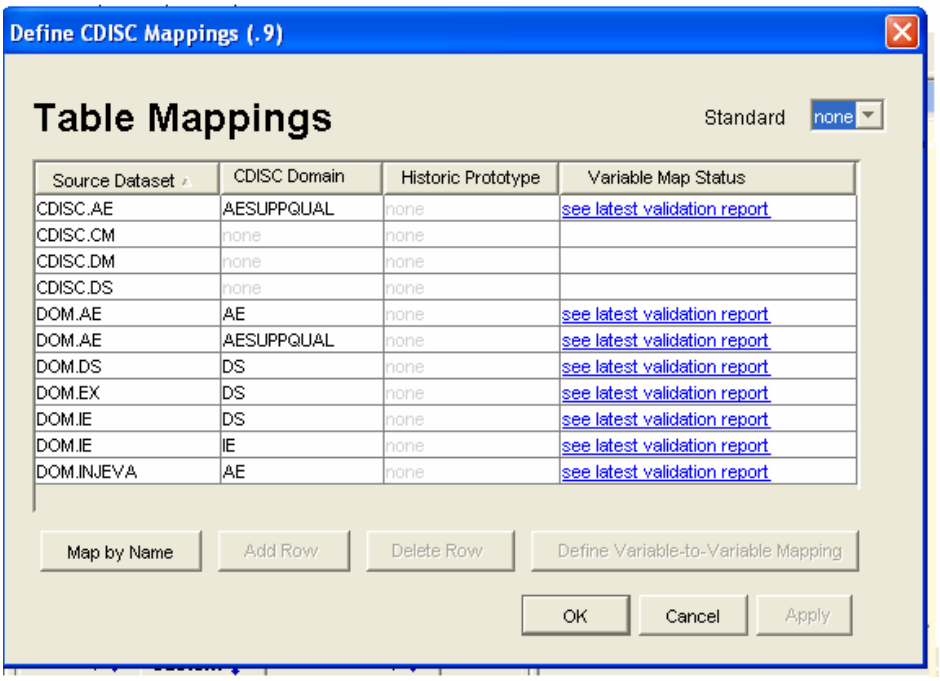
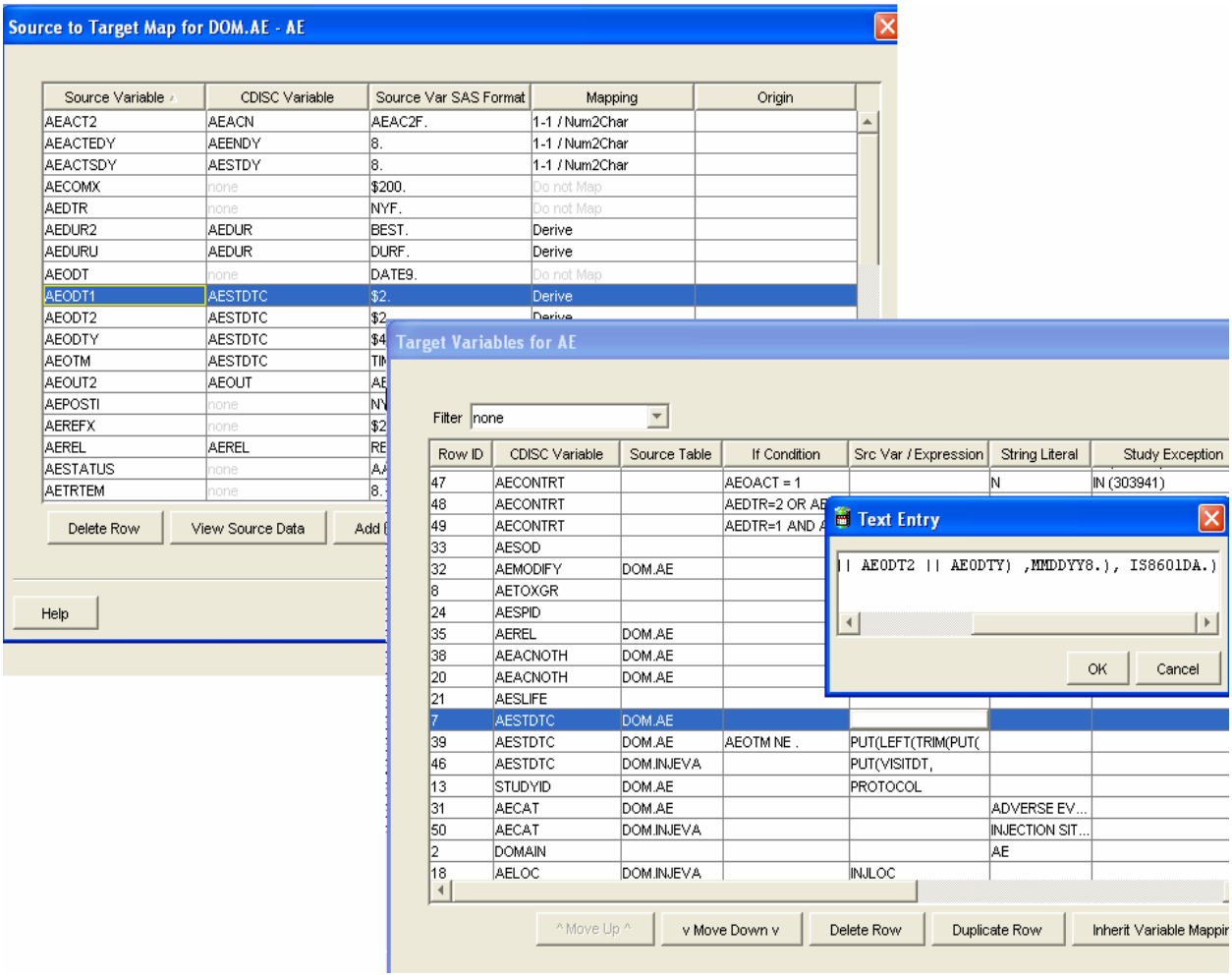


Fig.7 Variable-to-Variable two-page Wizard



Out of 19 fields of the Mapping Control table, 6 are fully editable through Mapping Assistant GUI, 3 are menu-driven, and the rest are populated 'behind the scene' by the plugin's interaction with the metadata server. This significantly reduces manual effort. For example, Source variable informats and target variable formats are read directly from Study repository and do not require any INPUT.. or PUT functions to be applied in the mapping expressions.

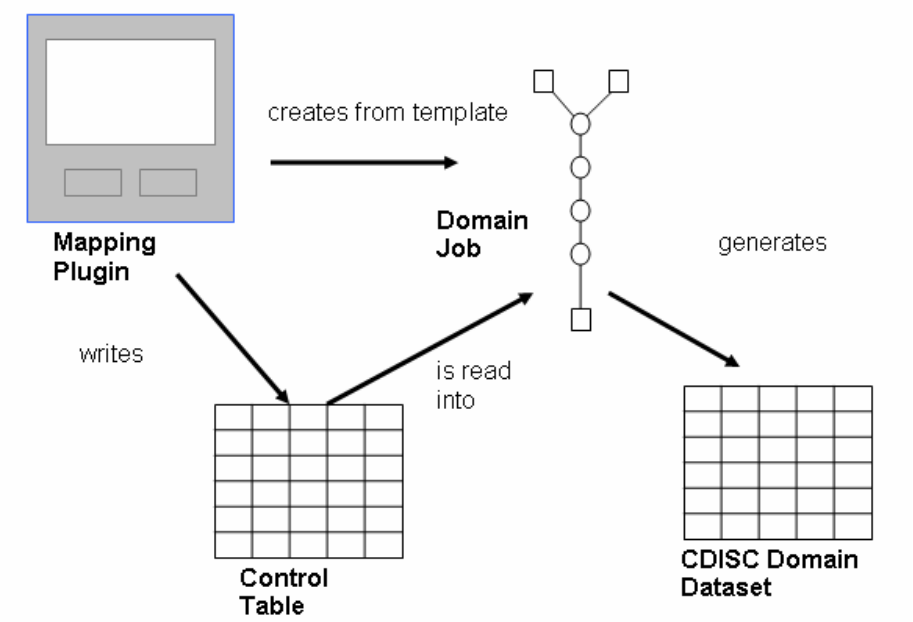
The manual effort is further reduced by our inheritance capabilities, which allow users to browse through already-built specifications and inherit mappings from previous studies where appropriate. Some convenience features include the ability to browse source data, filter content, perform quick mapping by name, etc.

When all mappings for the given SDTM domain are defined, the plugin saves user-entered derivation logic to the Mapping Control Table, and runs a validation report. The validation report checks SAS syntax in the derivation conditions and assignment statements, as well and catches some other discrepancies and/or conflicting assignments before they are used in SAS code by DI Studio.

For more information on how Mapping Control table governs data transformation within DI Studio job, see next section.

MAPPING CONTROL TABLE

Fig. 8 Mapping Control Table Centered Process

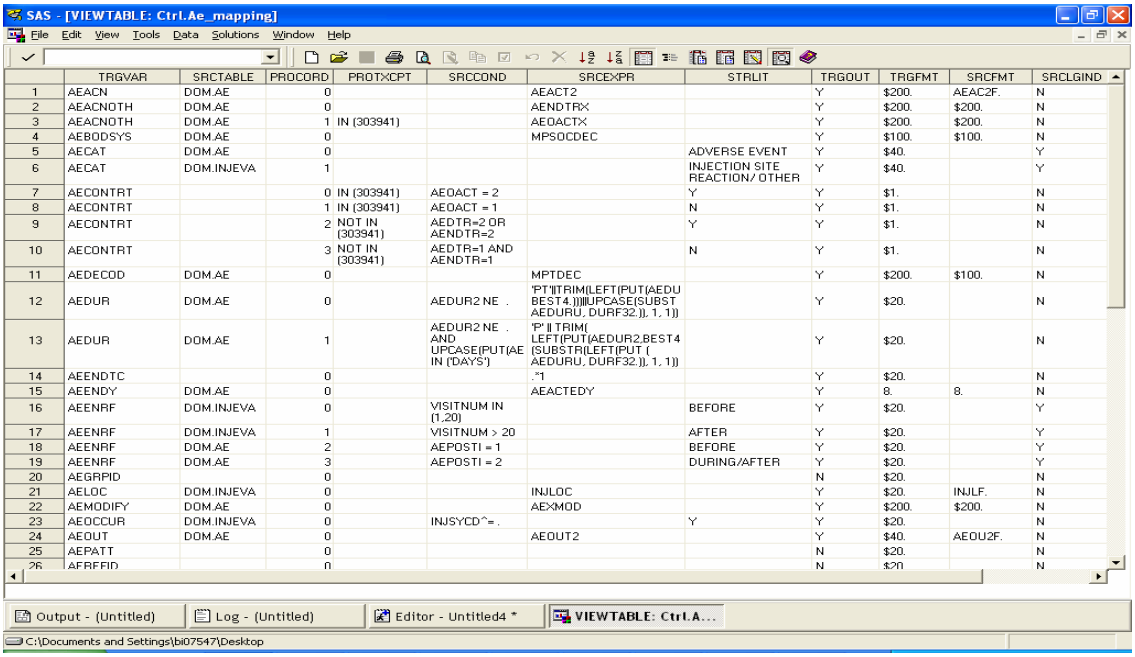


The Mapping Control Tables are the core component of the SDTM Application. These are SAS datasets that store mapping specifications in the format readily available for data transformation processes. Mapping Control Tables serve double role as mapping specifications as well as 'parameter feeders' for macros inside DI Studio jobs that create and populate SDTM variables.

Our application stores as many Mapping tables as there are SDTM datasets to be submitted, which hold all necessary information to convert our source data (analysis datasets) to SDTM format. The tables are created and modified by the Mapping Assistant plugin. As SDTM job accesses Mapping Control table, this table becomes a data transforming 'driving force'.

The Mapping Control table is target focused, containing at least one record per SDTM variable. Multiple records for the same target variable is also possible, representing the different mapping conditions for the variable.

Fig.9 Example of Mapping Control table in SAS Viewtable window.



	TRGVAR	SRCTABLE	PROCORD	PROTXCPT	SRCCOND	SRCEXPR	STRLIT	TRGOUT	TRGFMT	SRCFMT	SRCLGIND
1	AEACN	DOM.AE	0			AEACT2		Y	$200.	AEAC2F.	N
2	AEACNOTH	DOM.AE	0			AENDTRX		Y	$200.	$200.	N
3	AEACNOTH	DOM.AE	1	IN (303941)		AEOACTX		Y	$200.	$200.	N
4	AEBODSYS	DOM.AE	0			MPSOCDEC		Y	$100.	$100.	N
5	AECAT	DOM.AE	0				ADVERSE EVENT	Y	$40.		Y
6	AECAT	DOM.INJEVA	1				INJECTION SITE REACTION/ OTHER	Y	$40.		Y
7	AECONTRT		0	IN (303941)	AEOACT = 2		Y	Y	$1.		N
8	AECONTRT		1	IN (303941)	AEOACT = 1		N	Y	$1.		N
9	AECONTRT		2	NOT IN (303941)	AEDTR=2 OR AENDTR=2		Y	Y	$1.		N
10	AECONTRT		3	NOT IN (303941)	AEDTR=1 AND AENDTR=1		N	Y	$1.		N
11	AEDECOD	DOM.AE	0			MPTDEC		Y	$200.	$100.	N
12	AEDUR	DOM.AE	0		AEDUR2 NE .	'PT'||TRIM(LEFT(PUT(AEDUBEST4.))||UPCASE(SUBSTAEDURU, DURF32.)), 1, 1))		Y	$20.		N
13	AEDUR	DOM.AE	1		AEDUR2 NE . AND UPCASE(PUT(AEIN ('DAYS')	'P' || TRIM(LEFT(PUT(AEDUR2,BEST4(SUBSTR(LEFT(PUT (AEDURU, DURF32.)), 1, 1))		Y	$20.		N
14	AEENDTC		0			.*1		Y	$20.		N
15	AEENDY	DOM.AE	0			AEACTEDY		Y	8.	8.	N
16	AEENRF	DOM.INJEVA	0		VISITNUM IN (1,20)		BEFORE	Y	$20.		Y
17	AEENRF	DOM.INJEVA	1		VISITNUM > 20		AFTER	Y	$20.		Y
18	AEENRF	DOM.AE	2		AEPOSTI = 1		BEFORE	Y	$20.		Y
19	AEENRF	DOM.AE	3		AEPOSTI = 2		DURING/AFTER	Y	$20.		Y
20	AEGRPID		0					N	$20.		N
21	AELOC	DOM.INJEVA	0			INJLOC		Y	$20.	INJLF.	N
22	AEMODIFY	DOM.AE	0			AEXMOD		Y	$200.	$200.	N
23	AEOCCUR	DOM.INJEVA	0		INJSYCD^=.		Y	Y	$20.		N
24	AEOUT	DOM.AE	0			AEOUT2		Y	$40.	AEOU2F.	N
25	AEPATT		0					N	$20.		N
26	AEREFID		0					N	$20.		N

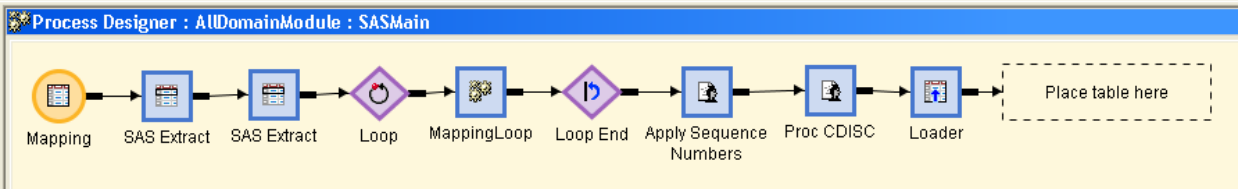
SDTM JOBS

Each SDTM job contains custom designed Mapping Transform. This transform makes an essential step in every SDTM job. Mapping transform executes a loop. In each iteration, a record is read from the Mapping Control table, the values are passed to macro parameters, which in turn populate IF…THEN… statements in SAS data step. As a result of this iteration one of SDTM variable is either first defined, or updated.

Despite the intention to design a standard process requiring minimum manual effort, a good SDTM tool must also provide a certain level of flexibility. To a great extent, this flexibility is achieved through the Mapping Control table. Another point where of standardization and flexibility are balanced is through the modular SDTM job structure.

The Mapping step, Sequence Number Assignment step, Proc CDISC validation step and final Loader are a part of every SDTM generation. These steps can be assembled in a standard module (called AllDomainModule) that can simply be inserted into any SDTM generating job (for any domain in any study).

Fig.10. AllDomainModule



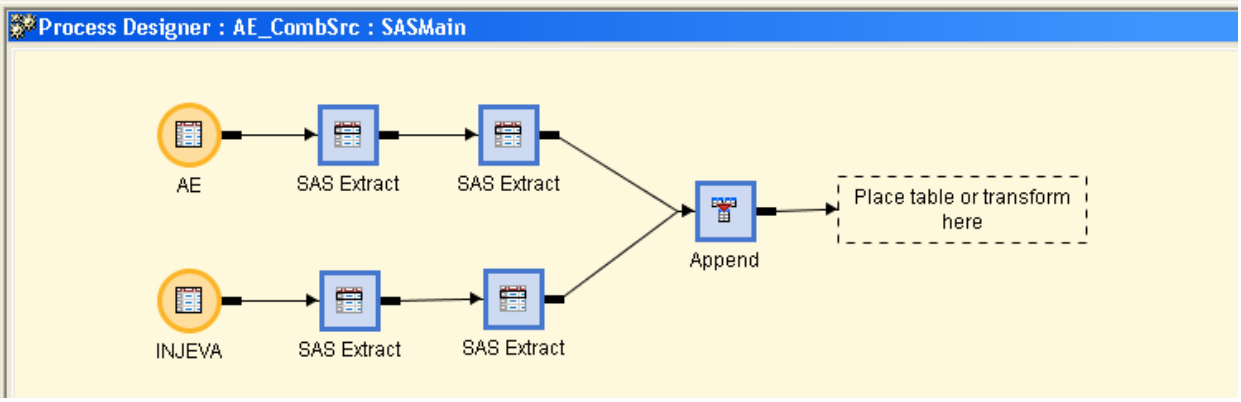
However, this module can apply its data transformation only if incoming data meets certain requirements. Among these requirements are:

- The incoming data must be combined in a single source. Depending on primary sources, this can be either horizontal (merge) or vertical (append) data union
- Data selection and cleansing should already be performed

The job module responsible for these tasks is completely study and domain specific, and is built by a clinical programmer using standard DI transformations as building blocks. By our naming convention, we call it DomainName_CombSrc (i.e.AE_CombSrc, CM_CombSrc, etc).

Since no mapping needs to be performed during this stage of data flow, this step is, in general, an easy and fun task. The only restrictions imposed on the users is to build this Combined Source job on a standard template which includes hidden post-processing that renames the output dataset to a preset name, so it can be picked up by the succeeding standard module.

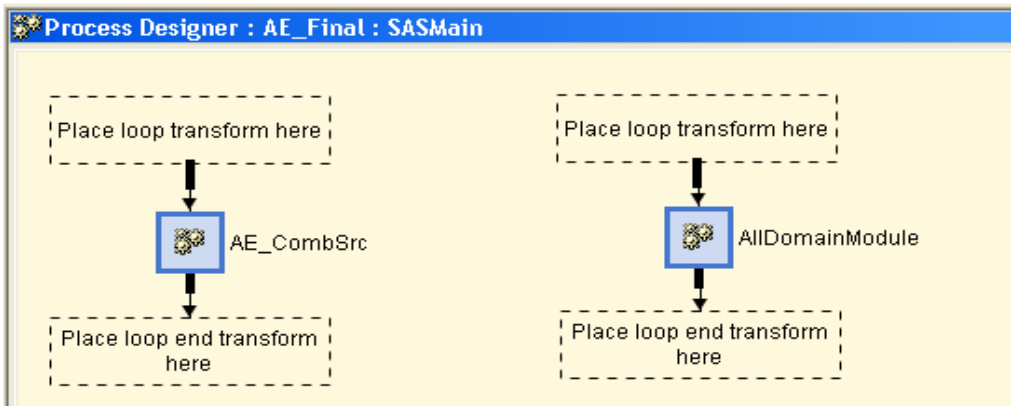
Fig.11 Example of a Combined Source Module (AE_CombSrc)



The final Append Step of this job module creates an input dataset to the subsequent AllDomainModule processing.

The shell job that puts these two modules together is called DomainName_Final.

Fig.12 Example of a Shell Job – Modular Construction



The template for this job contains the AllDomainModule and a hidden pre-process that uses SAS API to exchange information with Metadata Server. Why do we need to communicate with metadata Server independent of DI Studio, which is by design an interface to SAS metadata? Unfortunately, the current release of DI Studio does not allow design and use of generic processes similar to our AllDomainModule. Because our AllDomainModule does not contain any table metadata object (a table icon) for a target dataset, DI Studio is ignorant of which dataset it is to expect at the end. This information is to be communicated to Metadata Server using SAS API. This pre-process is also generic – it configures itself 'on-the-fly' using the domain name which is the part of the DomainName_Final job as a 'hook' to pull the rest of domain specific 'information chain'. This includes library names and paths, target dataset name, its sort order and indexes, variable attributes.

CONCLUSION

The chart below represents the outcomes and conclusions we came upon our experimentation with out-of-box DI Studio and our customization .

Table 2. Comparative overview of three approaches to CDISC SDTM generation.

	SAS/Macro Coding	**"Out-of-box DI Studio"**	**CDISC Customized DI Studio**
Facilitates Mapping	No	No	Yes
Manual coding effort	Heavy	Medium	Medium-Light
Reusability across studies	Limited	Limited	Core process fully reusable
Reusability across domains	No	No	Core process fully reusable
Technical Specs as by-product	No	No	Yes

WHAT LIES AHEAD?

Our future plans include:

- Extending Inheritance features
- Trial design data sets generation
- ADAM
- Expanding mapping functionality to enable generation of analysis datasets out of primary sources
- Designing internal tools for facilitating date/time variables conversion to the CDISC required format.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please contact usme at:

Tatyana Kovtun
Bayer HealthCare Pharmaceuticals
Montville, NJ USA
E-mail:
tatyana_kovtun@berlex.com

John Markle
Bayer HealthCare Pharmaceuticals
Montville, NJ USA
E-mail:
john_markle@berlex.com

Christopher Treglio
Bayer HealthCare Pharmaceuticals
Montville, NJ USA
E-mail:
chris_treglio@berlex.com