CDISC 360 Metadata-Driven Data Transformation Engine for Automation and Transparency

Author

Gregory Steffens

CDISC 2020 European Interchange

1 - 2 April 2020

Introduction

The value of data standards is generally accepted as a fact. Data standards facilitate more consistent study data designs; sharing code across studies; aggregating data and the submission of data to regulatory agencies and researchers, in a data design that can be more quickly understood. But, an even greater value, that is facilitated by standards, emerges from greatly improved quality, efficiency and transparency of the creation of the data. To fully attain this second type of value from standards, three technology components are required – 1) a complete set of types of information in the standards; 2) a standard metadata design in which to store this information and 3) metadata-driven, multi-use software that creates the data and define files. The required metadata consists of data state metadata, that describe the data attributes, and map metadata, that describe the flow of data from one data state to another. Today, CDISC delivers partial data state standards, e.g. what SDTM looks like, what ADaM looks like. The data flow standards, stored in map metadata, do not exist in the CDISC standard. Once the gaps in the metadata are filled, a DTE (data transformation engine) software solution can automate the data creation and description, as it flows from one data state to another. This paper describes the metadata design and data transformation engine, that is being used in the CDISC 360 project, to demonstrate the automation and transparency that are achieved. This technology implements an infrastructure for future AI and machine learning systems, that manage the metadata content in an MDR (metadata repository). The objectives of the CDISC 360 project, described below, are achieved, with a new kind of metadata design and a DTE software solution.

CDISC 360 aims to support standards-based, metadata-driven automation across the end-to-end clinical research data lifecycle and represents a significant next step toward realizing an increased return on investment in standards implementation that our stakeholders expect – substantially improved efficiency, consistency, and re-usability.

Directly accessible metadata

The metadata is designed as a relational database. So, the metadata are directly accessible with familiar programming languages, like SAS. There is no need to learn a new technology or language, to access the metadata. One result of the CDISC 360 project is that a program will be made available to extract CDISC

standards into this relational metadata format, including data state metadata and map metadata. The metadata design is documented in the 360 project and will be made available through other means as well. The metadatabase consists of metadata sets that describe attributes of data sets, variables, row-level (VLM) variables, valid values (CT), natural-language descriptions, code snippets for derivations, etc. Providing the information, found in this metadata design, is a core objective of the CDSIC 360 project – to provide enough information about the standards to enable the development of software like the DTE.

Data State Metadata – describing attributes of the data to be created

A data state is the set of data sets that are created as permanent data libraries. Describing a data state includes the definition of the attributes of the data, but not how to create the data from predecessor data through transformations and derivations. Data state metadata stores the description of each data state, such as the CDISC data states CDASH, SDTM and ADaM. The data state is fully described, including all the data-state attributes required for the automation and transparency objectives of the system of metadata, standards and software. This metadata design is proven to include all these attributes, through the use of it by the DTE automation in many study projects. It has evolved over years of refinement, to a concise design and set of attributes required for data, define file creation and other applications. This design is optimized for programmatic access, leaving the population of it to a user interface – created in workstreams 1 and 2 - and the publication of it to several human-friendly formats, including the define.xml and an html define file of my design. The design of data state metadata does not assume any data standard, but fully supports the CDISC data standard. This design results in a much wider scope of application, beyond CDISC. It has been used for data transfer specifications, data review checks, aggregated data specifications, and many other data flows that are not CDISC complaint. If there is a requirement to create SDTM from data that is not CDASH compliant, the metadata design and SDTM data-state metadata content can still be used. A single metadata design describes any relational data state and standard. There is no need to modify the design, such that one design applies to SDTM and another design to ADaM, for example. The only difference between SDTM and ADaM metadata is the content of the metadata, not the design of the metadata.

Map Metadata – describing how the target data state is created from the source data state

The map metadata stores the description of the flow of data from one data state to another – from a source data state to a target data state. The map metadata contains no attributes about the data states, as these are all described in the data state metadata. Map metadata is a physically distinct metadata from data state metadata, which allows for a much higher flexibility in defining data flows. It is a common mistake to include mapping information in data state metadata, but this results in a data flow that is “hardcoded” into the metadata design, severely limiting the scope of application of the technology. If the data flow changes, this kind of metadata design is unable to describe it. Separating

map metadata from data state metadata results in the ability to keep the data states and target data state metadata unchanged, even when the source or target data does not conform to standards or expectations.

Data Transformation Engine – the software that implements the data flow described in data state and map metadata

Transformations

The DTE has the built-in capability to perform any data transformation and to decide which data transformation to apply to each instance. The metadata provides the data set and variable names to apply its transformations to. There is no need to specify the type of data transformation in the metadata, because the DTE makes that decision itself based on the data set and variables defined in the map metadata as sources of each target data set and variable. No syntax is required in the metadata to describe any data transformation. A data transformation is defined as a flow of a data value from a source to a target data. The flow may put the source value in a different data set and variable name, in a SUPPxx data set, in a tall-thin data set from a short-wide data set, retype the variable it is stored in, change the format between SAS dates and ISO dates, or any other type of data transformation.

Derivations

A data derivation is described as the creation of a data value in the target data that does not exist in the source data. To accomplish this, one or more source variables are mapped to a single derived target variable and a code snippet applied. The DTE takes care of the required merging and preprocessing of source data to gather the source variables required to create the derived variable.

A new way to describe and implement data flow

Currently, data flow is typically described in documents (SAP) or metadata (often *ex post facto*) that do not include enough attributes to support full automation nor transparency. The implementation of data creation is hidden in programming code, usually SAS code. There is a discussion in industry about changing from SAS to some other language, like R and python, and to move to non-relational metadata and data designs. Adopting new technology is a natural part of the evolution of software development, even for study programming. But adopting a new language will not yield the results to accomplish our objectives. The point is not simply to switch languages, using the same process as we use with SAS. The invention of new code for each study is the root cause of the inefficiencies and quality challenges we

have today. Changing languages from SAS to R will not solve problems, in itself, any more than describing the problem in French or German, instead of English, will solve problems. We need to change the manner in which we create data. How can we better describe a data flow to the computer, than using some programing language or another? The answer is that we need to describe data flow in transparent metadata, instead of in opaque programming code. Subject matter experts in clinical data design, who do not know programming, can populate well-designed metadata, using a user interface. Powerful multi-use software, like the DTE, can then read the metadata and implement the data flow. This greatly decreases the amount of programming code we create and validate for each study and project, not simply changing the programming language we use, but fundamentally changing the way we describe data flow in enough detail to implement that data flow.

Transparency and Traceability

Because the metadata contains all the information, required by the DTE to create the target data state from the source data state, we have complete transparency and traceability of the entire data flow. Software can read the metadata to create reports that describe the data flow in any format we choose and with the full set of information required to create the data. Subsets of the information can also be reported for different reviewer skill sets and interests. A chain of any length, where each link of the chain consists of a data state, through map, to a target data state, can be defined. The target of one link becomes the source of the next link. Thus, the data flow from raw data, extracted from EDC and third-party data sources, to SDTM and then to ADaM can be represented with the metadata. Defining a post-ADaM data set that is then formatted into a report, extends the range of application of the metadata and DTE to include tables, listings and figures.

Metadata collection – collecting metadata content in human friendly format

The metadata is designed to optimize for software access. It follows good relational design principles of the same sort that we all use in study data design and that are used in the CDISC data standards. This design is correct because the intent is that metadata be directly accessed by software. But this design can be unfamiliar or even nonintuitive for users who don't understand relational data design. Even users who do understand the design may find it inefficient to directly populate the metadata manually. For these reasons, it is important to provide a user interface for the collection and management of metadata content. This UI will present metadata collection interfaces that are optimized for humans, rather than for software. The user interface can include higher levels of sophistication, including AI and machine learning, that assist the human user in managing metadata content for standards and for study data specifications. It is a serious mistake to allow the metadata design to be influenced by human interface considerations – the consumer of metadata is software. In fact, the higher-level functions of AI required well-designed metadata in order to succeed. Therefore, the user interface is an essential

component of the technology solution. In the CDISC 360 project, workstreams 1 and 2 are responsible for this user interface. After collecting information from the human user, it exports the information to software-friendly designs, such as my relational metadata design.

Metadata publication – presenting metadata content in human friendly format

A define file is an example of a publication format of metadata content. It reads metadata, optimized for software access, and publishes the content in a human-friendly format. The define xml schema is an example of metadata and its style sheets are examples of publication formats. The CDISC 360 recognizes the limitations of this xml metadata and is designing an alternative metadata. The define xml schema is limited to data-state attributes and does not adequately address mapping information. I've designed a define file in html format. There are several advantages of the html define file, as compared to the define xml file. The define html file presents all the information required to create the data flow, including all data state and map metadata content. It does so in a more intuitive interface than define.xml, not being limited by the attributes of the define.xml schema and the associated style sheets. This define html file is used to describe data standards and project data specifications for internal and external use. It should be created before programming begins, and used as a data specification. The define html presents all the information required to create each variable in a single location, which I refer to as the variable detail section of the define html file. The variable information is collected from source data state metadata, map metadata and target data state metadata and presented in one location that is viewable and able to be printed. The define xml style sheets follow the physical design of the underlying xml and therefore does not optimize for human-friendly format. It requires multiple hyperlink clicks to view the information associated with a variable, to see comments, controlled terminology, methods, etc.

Conclusion

A new way of describing and creating data is required; in order to attain the objectives of efficiency, transparency and traceability that we have defined as shared industry objectives. These objectives are part of the CDISC 360 project. I have described a technology solution, that includes a single, standard metadata design and a reusable software component, named the data transformation engine. This technology greatly reduces the about of code required at the study and project level and is an example of a fundamentally different way to implement the creation and description of data. This new way of working will increase efficiency, transparency and traceability of data creation.