Paper SI02

Mind the Gap: Raw Data Model

Julie Smiley, Oracle, San Antonio, US

ABSTRACT

Increasingly more organizations are implementing metadata management solutions as part of their overall clinical data standards and information governance strategy. The objective is to manage standards, including mappings and transformations across the clinical data lifecycle to drive efficiencies and automation in study setup and programming. However, one of the major barriers in achieving this goal is the gap between data collection and SDTM. While CDASH is the standard for data collection, it only addresses the needs for defining the data collection forms, but there is no standard for raw data output from source systems. In this session, we will discuss this gap, the impact it has on the overall data flow, and strategies for mitigation.

INTRODUCTION

With growing regulatory challenges and competition from generics and digital trials, pharmaceutical organizations are under constant pressure to get to submission faster, with greater quality in less time and with fewer resources. One potential opportunity for gaining significant efficiencies is automating the clinical data flow, especially between data collection and SDTM. To achieve this, not only is there a need to manage standards for different phases of the clinical data lifecycle, but also the mappings and transformations between these phases. There have been multiple approaches offered and claims made by solution vendors, especially related to metadata management and/or clinical programming solutions. While many have moved towards this vision, very few organizations have actually achieved success. One major reason for this is the gap between data collection standards and SDTM. Data collection standards are typically mapped to SDTM, but their intent is to define how data is collected in the user interface (UI), not how it is extracted. Most data collection systems do not output the data in the same format it was specified for collection, leaving a gap that causes the mappings to break. This paper will explore this gap further and will propose methods for overcoming this barrier.

UNDERSTANDING THE GAP

Many organizations are in the process of implementing significant changes across clinical development to gain efficiencies, reduce risk, and save time and money. These changes often include adding a data or standards governance organization that develops, manages, and ensures compliance to clinical data standards. From the technical perspective, this often includes an investment in a standards management solution capable of producing data standards specifications for either human or computer consumption to support not only compliance to standards, but also automation of the data flow to gain efficiencies. However, even for those organizations that have put significant time and investment in these changes, very few have been able to fully automate the clinical data flow and see their expected return on investment.

One major reason for this is that there is a gap between data collection standards (CDISC and/or internal) and the study data tabulation model (SDTM). Data collection standards are used to create specifications for the data collection system. They define how the data is collected from investigator sites, but do not prescribe how the data is extracted from its source system. When management of these standards or study specifications includes mappings and/or transformation instructions from data collection to SDTM, the missing raw data model can result in breaks in the overall data flow. For SDTM programmers reading these specifications, it can result in significant time and effort to manually interpret and program to these specs with so many discrepancies. For systems that implement the standards programmatically, the gap can cause disruptions in the data flow, or worse, result in inaccurate, incomplete, or inconsistent data in SDTM.

Data collection systems normally have a proprietary physical model where the data is stored, with no access privileges for the users. Instead, access to views or a method for extracting the data into tables or datasets is provided. The format for these tables varies across vendors and solutions, but oftentimes the data collection forms are not a one-to-one match to the extracted tables, and items on the forms are not a one-to-one match to the variables in the extract tables. Let's look at some examples of these rules and their impact on the data flow.

EXAMPLE 1: REPEATING AND NON-REPEATING FORM SECTIONS

Data is often collected on a form that has repeating items or attributes, but there is heading information that does not repeat for the form as a whole. An example of this is a lab form for a particular set of tests, such as a chemistry, hematology, or urine panel. The repeating items are the tests with their associated results and units (and potentially other attributes depending on the type of tests). However, the panel is collected at the same time, so the form header has the date and time of collection. In the data collection system, the non-repeating items are in one extract table (LB_NR) and the repeating items are in another table (LB) as illustrated in Figure 1 below. If the specification only showed these as a single form (LB) that is mapped to LB in SDTM, then there is no mapping from the LB_NR table. This can result in missing data in SDTM or possibly having data appear to be duplicate if the same test showed the same results on different dates/draws within the same visit without the date and time present. This could lead to many queries to clarify, as well as a lot of time and effort by the sponsor and site to try to resolve these data issues.

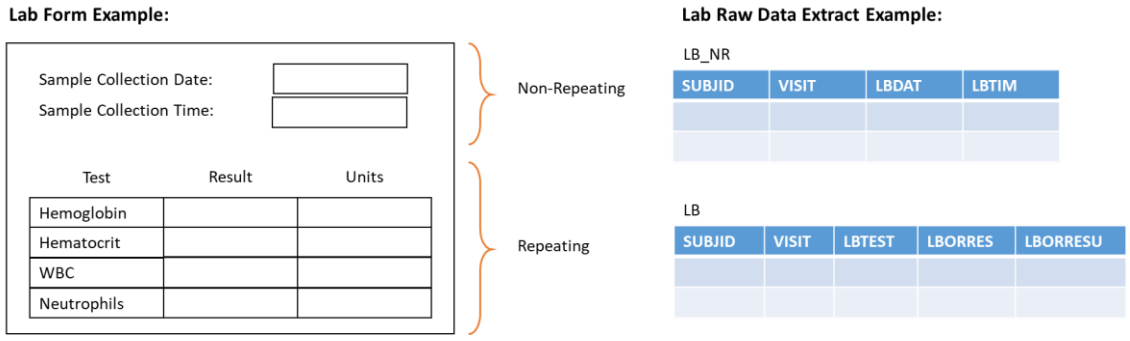


Figure 1: Example of repeating and non-repeating form sections using labs

EXAMPLE 2: MULTIPLE FORM VARIANTS

Sometimes the same data is collected multiple times on a patient throughout the trial, but the specific data points or codelist associated with an item might change. Vital signs (VS) is a good example of this, as many times the height of a patient is only collected at the screening visit, but all other vitals are taken at every visit. In this case, many data collection systems see this variation in the form to be different forms, which in turn have different extract tables as shown in Figure 2. If the standards or mapping specifications only show VS from Data Collection to VS in SDTM, then there is the potential for data from both VS1 and VS2 (all variations not named VS) to not make it into SDTM without manual effort and interpretation. Again, this could lead to a significant number of discrepancies and manual effort to resolve these issues if not addressed in the specifications.

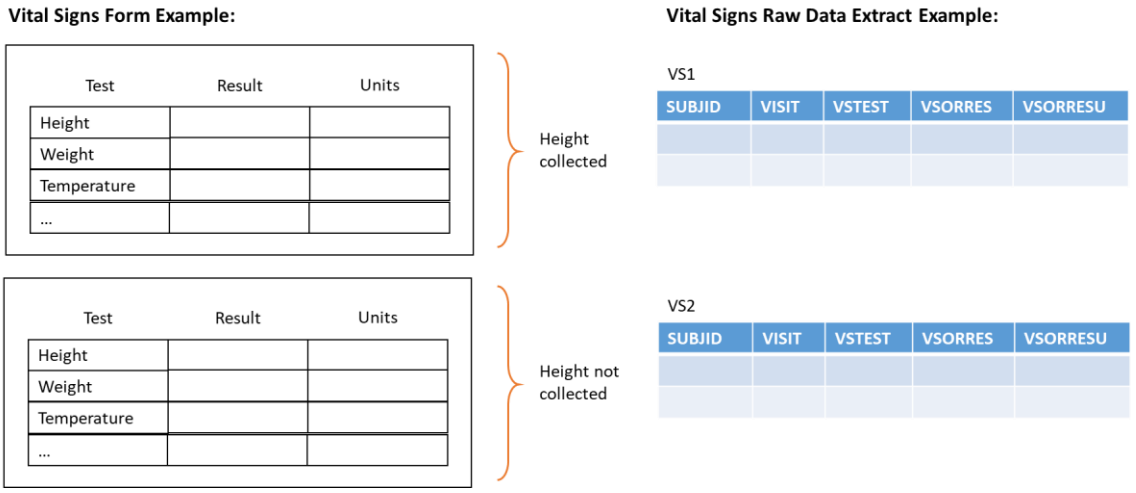


Figure 2: Example of multiple form variants using vital signs

EXAMPLE 3: REPEATING AND NON-REPEATING FORM SECTIONS ACROSS MULTIPLE VARIANTS

In addition to multiple sections or variations of a form, sometimes there can be multiple variations of a form with multiple sections, resulting in the same type of data being extracted in exponentially more tables than specified. Continuing with the lab example above, sometimes the date vital signs are taken is collected as well, which means each variation of the form also has multiple sections if collected in vertically (or normalized) as repeating and non-repeating sections (as opposed to collecting the data horizontally or in a non-normalized way), as depicted in Figure 3. This of course only compounds the issues described above.

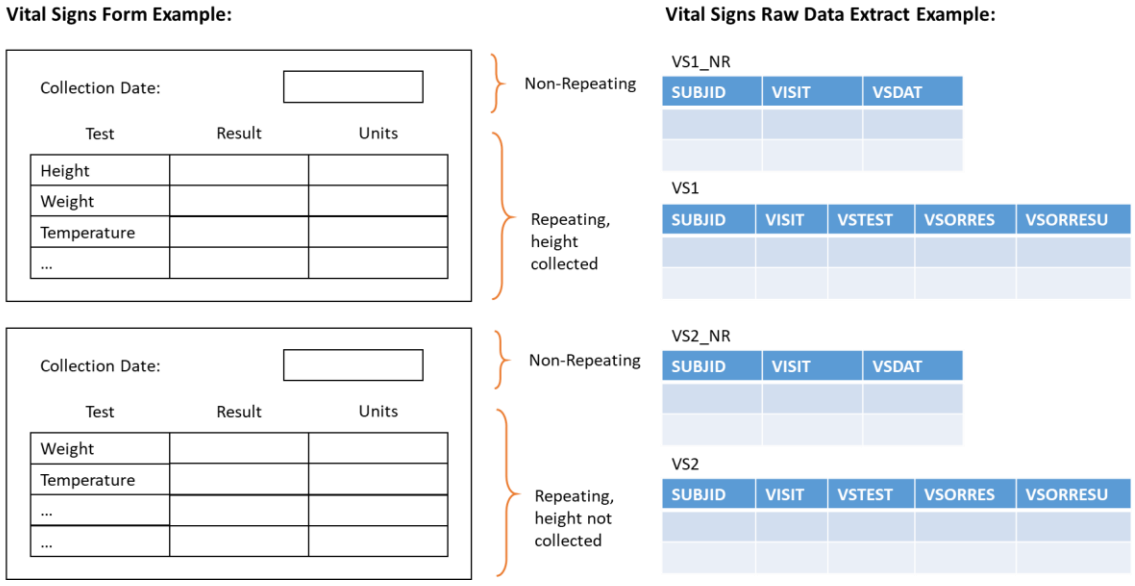


Figure 3: Example of multiple form variants with multiple sections using vital signs

EXAMPLE 4: MULTIPLE VARIABLES PER ITEM

Finally, most data collection systems add variables into the extract based on each items data type to support multiple use cases. For instance in demographics a codelist is used for collecting sex and ethnicity. Each of these coded items has two different variables in the extract; one for the code and another for the value as shown in Figure 4. This

is another case that must be addressed in the specifications, as they generally only map the named item to SDTM (e.g., DM.SEX > SDTM.SEX). However, SDTM typically is expecting the code, not the value, but in the example below the code is in SEX_C, so SDTM receives data that it doesn't understand, causing errors.

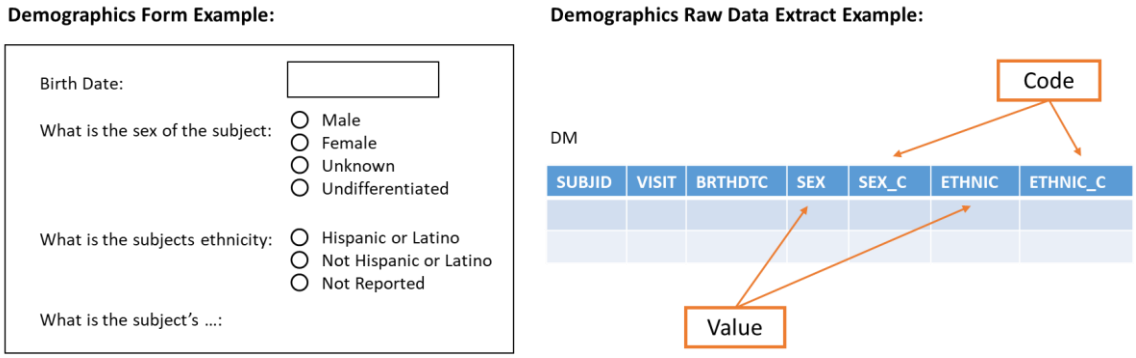


Figure 4: Example of multiple variables by data type using demographics

MINDING THE GAP

While these examples illustrate the gap between data collection and SDTM, this is not inclusive of all potential raw data model or extract issues. It is important to understand every data collection system's extract rules to ensure your organization has addressed all potential data flow issues. There are many methods to ensure that your SDTM datasets have all the data correctly transformed from your data collection system. Below are several common methods with their advantages and challenges, which are depicted in Figure 5 below.

METHOD 1: MANAGE OPERATIONAL METADATA AS PART OF THE DATA COLLECTION STANDARD

One way that many companies manage this is by adding the data collection system operational metadata to their internal data collection standard to ensure that the mappings are complete to SDTM. While this method works, it takes a significant amount of effort to create and manage standards with this level of operational metadata, especially if the organization uses more than one data collection system or does not have a formalized, object oriented system for managing standards and specifications, such as an MDR.

METHOD 2: MANAGE THE RAW DATA MODEL AS A STANDARD

Another method for managing this is to manage the raw data model as a separate internal standard between data collection and SDTM. This is similar to method 1, but gives separation between the standard and operational metadata, since they are managed as different models. However, it would cause duplication between the data collection standard and raw data model standard in order to add the operational attributes, which would result in even more effort to manage these as two separate standards and sets of mappings (one from data collection and another to SDTM). Further, each time additions are made to either the data collection standard or SDTM, they would also need to be made to the raw model. This amount of duplication can result in quality issues, as well as increased resource needs for maintenance.

METHOD 3: PROGRAMMATICALLY CONVERT RAW DATA TO THE DATA COLLECTION STANDARD

The third method is to continue managing mappings and transformations from data collection standards to SDTM without addressing the raw data format, but instead have a clinical programming team that converts the data to the expected format before starting on SDTM programming. This is what most organizations have done historically; they use the data collection extract rules to programmatically merge the raw data to the data collection format so that all of the downstream mappings and transformations work as expected. While this methodology requires far less effort in managing the standards/specifications, it takes a lot of up-front programming effort to do these conversions on every study. Also, traceability of the data from its original source can get lost without very clear documentation, and it delays getting data into SDTM format, although some programming automation (e.g., with macros) can reduce this.

METHOD 4: VIRTUALLY MAP TO SDTM FROM THE RAW DATA MODEL

Finally, the last and recommended option to address this gap is to programmatically generate the mappings from the raw data model by applying the EDC extract rules to the data extract metadata (i.e., data dictionary), as well as internal rules on which variables to use from the extract for different data types (e.g., the code or value variable for coded items). It is not necessary to manage the raw data structure as a model within the standards management

system though. Instead, the rules can be built into the SDTM mapping specification so that each time it is generated, whether manually for human consumption or programmatically for full automation, the mappings and transformations are from the raw data model instead of the data collection standard. This approach can bridge the gap between data collection and SDTM and since the programming is rules-based, it can be applied to any study without additional manual programming effort. This has advantages over the first two methods because no additional effort is required to create and maintain the operational metadata (either with the data collection model or as a separate raw data model). It also has advantages over the third method, as data lineage is maintained since SDTM is programmed from the raw data model (where the data is actually stored) versus to the data collection model. Further, it is a one-time only programming effort versus the third method, which requires additional programming for every study. This improves efficiency and reduces resources in both standards management and clinical programming.

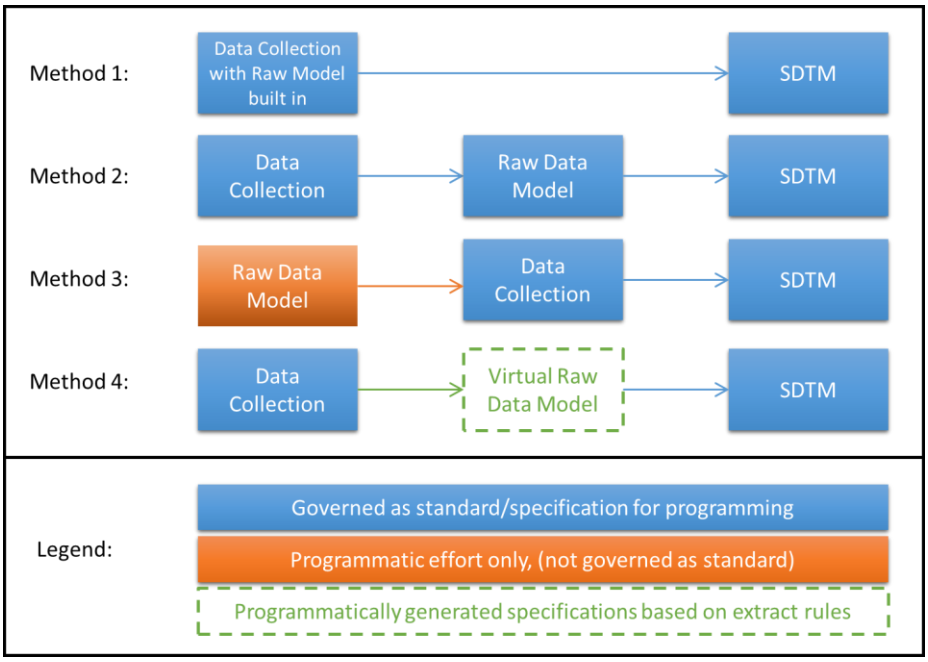


Figure 5: Methods for bridging the gap between data collection and SDTM

Using the lab example above (in Figure 1), let's take a look at how to convert the data collection specifications to a raw data model with mappings to SDTM. Using one of the previous examples, there is a non-repeating form section (LB_NR) and repeating form section (LB). The data collection system rule is that it adds the suffix _NR to the form name to create the parent non-repeating form, which should be joined to its child form of the same name without the suffix. Using Excel formulas to demonstrate this, the following was done:

1. Programmatically used the lab form specification and added in a new column at the beginning to concatenate the form and item name and sort by the FORM.ITEM column, as seen in Figure 6.

Lab Data Collection Form Specifications:

FORM.ITEM	FORM	ITEM	DATA TYPE	CODELIST	SDTM Mapping	SDTM Mapping Instructions
LB.LBDAT	LB	LBDAT	CHAR		LBDTC	Concatenate LBDAT and LBTIM to an ISO 8601 format
LB.LBRES	LB	LBRES	CHAR		LBRES	
LB.LBTEST	LB	LBTEST	CHAR	TESTCD	LBTEST	
LB.LBTIM	LB	LBTIM	CHAR		LBDTC	Concatenate LBDAT and LBTIM to an ISO 8601 format
LB.LBUNIT	LB	LBUNIT	CHAR	UNIT	LBUNIT	
LB.SUBJID	LB	SUBJID	CHAR		DM.SUBJID	
LB.VISIT	LB	VISIT	CHAR		VISIT	

=[@FORM] & "." & [@ITEM]

Figure 6: Add in FORM.ITEM name to DC Specs

2. From the raw data extract metadata (data dictionary), programmatically generated the DC Form.Item Name based on the data collection system rules, as illustrated in Figure 7.

Raw Data Extract Metadata:

TABLE	VARIABLE	DATA TYPE
LB_NR	SUBJID	CHAR
LB_NR	VISIT	CHAR
LB_NR	LBDAT	CHAR
LB_NR	LBTIM	CHAR
LB	SUBJID	CHAR
LB	VISIT	CHAR
LB	LBTEST	CHAR
LB	LBTEST_C	CHAR
LB	LBRES	CHAR
LB	LBUNIT	CHAR
LB	LBUNIT_C	CHAR

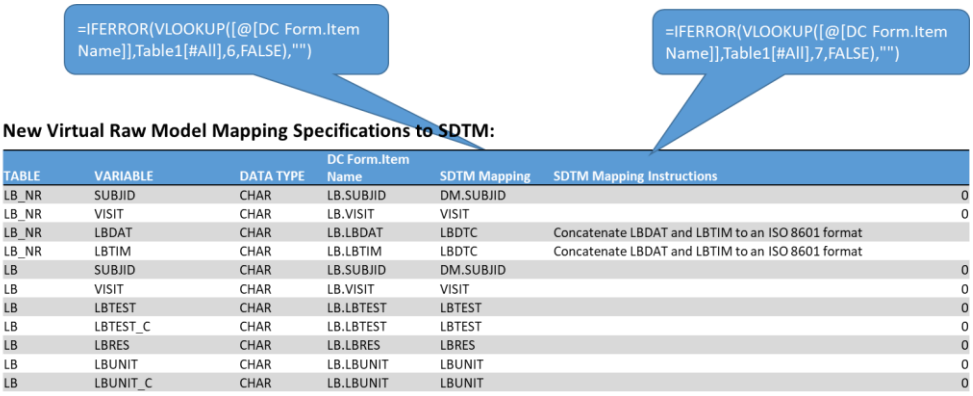
=IFERROR(LEFT([@TABLE],FIND("_",[@TABLE])-1),[@TABLE]) & "." &
IFERROR(LEFT([@VARIABLE],FIND("_",[@VARIABLE])-1),[@VARIABLE])

New Virtual Raw Model Mapping Specifications to SDTM:

TABLE	VARIABLE	DATA TYPE	DC Form.Item Name	SDTM Mapping	SDTM Mapping Instructions	
LB_NR	SUBJID	CHAR	LB.SUBJID	DM.SUBJID		0
LB_NR	VISIT	CHAR	LB.VISIT	VISIT		0
LB_NR	LBDAT	CHAR	LB.LBDAT	LBDTC	Concatenate LBDAT and LBTIM to an ISO 8601 format	
LB_NR	LBTIM	CHAR	LB.LBTIM	LBDTC	Concatenate LBDAT and LBTIM to an ISO 8601 format	
LB	SUBJID	CHAR	LB.SUBJID	DM.SUBJID		0
LB	VISIT	CHAR	LB.VISIT	VISIT		0
LB	LBTEST	CHAR	LB.LBTEST	LBTEST		0
LB	LBTEST_C	CHAR	LB.LBTEST	LBTEST		0
LB	LBRES	CHAR	LB.LBRES	LBRES		0
LB	LBUNIT	CHAR	LB.LBUNIT	LBUNIT		0
LB	LBUNIT_C	CHAR	LB.LBUNIT	LBUNIT		0

Figure 7: Programmatically generate DC Form.Item Name from raw data model

3. Used the new DC Form.Item Name to programmatically look up and add the SDTM Mapping and SDTM Mapping Instructions from the Lab Data Collection Specifications table (Table1), as seen in Figure 8.



New Virtual Raw Model Mapping Specifications to SDTM:

TABLE	VARIABLE	DATA TYPE	DC Form.Item Name	SDTM Mapping	SDTM Mapping Instructions	
LB_NR	SUBJID	CHAR	LB.SUBJID	DM.SUBJID		0
LB_NR	VISIT	CHAR	LB.VISIT	VISIT		0
LB_NR	LBDAT	CHAR	LB.LBDAT	LBDTC	Concatenate LBDAT and LBTIM to an ISO 8601 format	
LB_NR	LBTIM	CHAR	LB.LBTIM	LBDTC	Concatenate LBDAT and LBTIM to an ISO 8601 format	
LB	SUBJID	CHAR	LB.SUBJID	DM.SUBJID		0
LB	VISIT	CHAR	LB.VISIT	VISIT		0
LB	LBTEST	CHAR	LB.LBTEST	LBTEST		0
LB	LBTEST_C	CHAR	LB.LBTEST	LBTEST		0
LB	LBRES	CHAR	LB.LBRES	LBRES		0
LB	LBUNIT	CHAR	LB.LBUNIT	LBUNIT		0
LB	LBUNIT_C	CHAR	LB.LBUNIT	LBUNIT		0

Figure 8: Programmatically add mapping specifications

Although this is only one example of how to programmatically generate the raw data model and mappings specifications to SDTM, it shows how even simple programming in Excel can help achieve this. However, the more rules that are involved, the more complex the programming will be, which will call for a more in-depth, manageable programming language, such as PL/SQL or SAS to fully build this out.

CONCLUSION

This paper describes the gap between data collection standards and SDTM, the raw data model, and its impact on the overall clinical data flow. It provides several examples of how the gap occurs, as well as multiple options for how to bridge it, including a recommended approach for virtualizing the raw data model and its mappings to SDTM using the data collection system extract rules against the raw extract metadata (or data dictionary). This approach minimizes the manual governance effort to create and maintain operational metadata in addition to standards, while also limiting the manual programming effort to convert the raw data output for each study to the data collection standard. In addition to this being a more efficient method overall, it helps to accelerate SDTM delivery and reduces overall resources. It is an automated solution that is easily scalable since it requires no study specific programming effort and it can be achieved with existing tools, such as Excel, SAS, or SQL. Overall, understanding this gap and addressing it can help organizations achieve data flow efficiency and a higher return on their investment in clinical data standards management and governance programs.

CONTACT INFORMATION

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

Julie Smiley
Oracle
613 NW Loop 410, Suite1000
San Antonio, TX 78216
Work Phone: 210-536-9152
Email: julie.s.smiley@oracle.com
Web: www.oracle.com