Paper DH11

Addressing the Gap in Data Standards: Introducing the Aggregation Layer

Claudio Garutti, Oracle, Utrecht, The Netherlands
Marko Bomyer, Oracle, Barcelona, Spain
Jonah Brugger, Oracle, Amherst (MA), United States
Julie Smiley, Oracle, San Antonio (TX), United States

ABSTRACT

CDISC has improved the efficiency and quality of clinical research by developing standards across the clinical data life cycle including data collection, submission and analysis. However, there are still gaps. There is no standard for data review; and the data collection standard does not cover all data sources and is typically de-normalised, whereas submission datasets are normalised. As a result, getting from source to submission requires multiple programming iterations. Further, there is duplication of effort to produce different, but related outputs, such as tables for blinded versus un-blinded users, or moving from one version of a standard to the next. To address these challenges, we propose an intermediate state between data sources and downstream data outputs called the "aggregation layer". This presentation we will share our experience of implementing this approach with several top pharma organisations and will propose best practices for how to design your data architecture for maximum efficiency.

INTRODUCTION

The work described here is based on projects conducted at several large pharmaceutical organisations using a clinical data management platform solution to manage the clinical data flow from source through SDTM. However, the methodology being presented can also be applied to other solutions, such as SAS® or other flat-file based systems. This paper discusses three gaps in today's standards and provides suggestions on how to address them; introduces the key concept of data architecture and of aggregation layer, which is illustrated in a case study; and finally provides conclusions.

IDENTIFYING THE GAP

CDISC has improved the efficiency and quality of clinical research by developing standards across the clinical data lifecycle, including CDASH for data collection, SDTM for submission, and ADaM for analysis. These standards can be described as **data models**.

DATA MODELS

A very informal definition of a data model is "a collection of tables that describes the key features of a database". Common features within the model are the table definitions, column definitions including column formats, code lists, and allowed ranges, and table and field constraints (e.g., cardinality, uniqueness, keys, etc.). A data model describes the "structure" of the expected data. In terms of traditional clinical programming, a data model is the collection of dataset definitions for a specific consumer or use case (e.g., CDASH, SDTM, ADaM).

A few examples of data models referenced in this paper are:

- **Data collection model** – the design specifications for the forms and items (paper or electronic) for collection of the clinical data from the site. CDASH is the CDISC standard for this model.
- **EDC raw data model** – the set of tables coming from an EDC system, corresponding to the Case Report Forms, in the raw data extract format. This model typically varies by EDC solution with a set of rules for the structure and by clinical study. There is no CDISC standard (and typically no sponsor-defined standard) for this model, although many believe that CDASH is the standard for this. The lack of standard for this presents a gap in standards and therefore in the clinical data flow that must be addressed.
- **Lab data model** – the set of tables coming from one core lab. Each core lab can have its own data model. Note that the data coming from tests conducted in local labs (i.e. lab tests conducted in hospitals) are usually collected in the EDC system, and therefore part of the data collection model. However, there is no

standard for the lab data model, although there are standards for labs in the data collection standards, as well as the SDTM model for data tabulations. This also presents a gap in standards, as well as the clinical data flow that must be addressed.

- **Study Data Tabulation Model (SDTM)** – the tabulation datasets that are used for regulatory submission. The structure of the datasets in this data model are usually in normalised form ("long and skinny"). SDTM is the CDISC standard for this model and compliance to the standard is required by some regulatory agencies, including the FDA.

The EDC raw data model and the Lab data model are two examples of source data models. The **source layer** is the set of all source data models. Similarly, the **consumer layer** is the set of all target data models that are downstream from the source layer. The SDTM data model is a target data model that is mapped and transformed from one or more sources and is therefore in the consumer layer, but it is also the input model for analysis programming and could therefore also be in the source layer. As this paper is focused on data collection through data tabulations, SDTM is only included in the consumer layer in this context.

GAP #1: CDASH IS THE STANDARD FOR HOW DATA IS TO BE COLLECTED, NOT FOR RAW DATA OUTPUT

Usually, data collection standards like CDASH only address how data is to be collected in the UI. They do not address how data is extracted, which differs for every source solution. This leaves a major gap because the raw data format coming out of the data collection system is in a different structure than the data collection standard or specifications from which SDTM was mapped.

The best way to address this gap is to programmatically generate the raw data model by applying the EDC extract rules to the data collection standard, 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). Then SDTM can be mapped from the raw data model, rather than from the data collection standard. However, it is not necessary to manage the raw data structure as a model within the standards management system. 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.

GAP #2: LACK OF A STANDARD FOR DATA CLEANING

In most organisations, the group within R&D with the greatest responsibility for the integrity of the data is Data Management. Even in small organisations that heavily outsource the clinical conduct and operation of their trials, data managers are still needed to oversee the data delivered from the outsourcing partner. Data managers continuously review and clean the data during the conduct of a study and as a result represent most of the time spent with eyes in the listings of a study. The CDISC standard most closely aligned to data management is CDASH. However, CDASH was established for data collection, not data output (raw data model), and traceability of source into SDTM using highly normalised formats. Furthermore, the CDASH standard provides guidance for the creation of electronic Case Report Forms (CRFs), where it has been reported that as much as 70% of the data volume is coming from non-EDC data sources such as Clinical Outcome Assessments (eCOA), niche and large central laboratories, and even internet connected medical devices (IoT) (Nadolny, Zambas, Torche, & Bhardwaj, 2019). Therefore, as of today CDISC doesn't have a standard specific to the needs of data managers.

While CDISC doesn't have a standard for data cleaning/review, oftentimes sponsors do develop internal standards for data management. The name of this data model is different for each organisation, and it will be referenced as the Data Management Review Model (DMRM) throughout this paper. Occasionally, organisations prefer to develop separate models for different user groups (e.g. one model for data managers, one for medical reviewers), or for different purposes (e.g. analysis and reporting). Like all data standards, the additional data models like DMRM require additional work to initially define and subsequently maintain them. Furthermore, these data models do not exist in a vacuum; they require transformations to translate data from collection to the desired format. These transformations also need to be defined, version controlled, and maintained.

GAP #3: LACK OF A DELIBERATE DATA ARCHITECTURE

A common way of working in traditional programming is to define analysis datasets and mock TFLs as part of the Statistical Analysis Plan (SAP). Statisticians are responsible for the SAP, and clinical programmers are responsible for implementing it. This often translates into ad-hoc programming from the (variable) sources to the deliverables. The

issue is that the re-usability of these programs is often limited across trials. Macros are an attempt at increasing re-usability. However, the effectiveness of macros is limited by the variability of source structures. If the organisation has a standards team, most of their work is spent managing data collection and submission data models, but little goes into defining how “everything in between” looks like. In this scenario, transformations and data models are looked at as independent activities carried by different teams (programmers for transformations, standards team for data models), where in fact the two activities are interdependent.

A better approach is to look at the overall **data architecture**, which here is referred to as the definition of all the data models, and how they transform into each other. Within the data architecture framework, the traditional programming approach is equivalent to Figure 1 (left) and is referred to in this paper as **direct architecture**. In the direct architecture, data models in the consumer layer are derived directly from the sources. This approach has the advantage of requiring minimal preparation, as the only work up-front is the definition of the data models, while the transformation programs can be implemented later. With this architecture, it is not uncommon for an organisation to accept to have the programs ready six months after first patient first visit, or even later. However, there are several problems with this approach. First, the number of point-to-point transformations explodes when adding new sources, or new data models in the consumer layer. Second, the maintenance of these transformations is hard, because each change in one of the sources means re-writing all the transformations associated to that source. Third, since transformations are developed ad-hoc for each pair of data models, programming tends to be postponed six months or more into study conduct, which means that the study team has delayed access to derived datasets and is not able to catch data quality issues across sources until later.

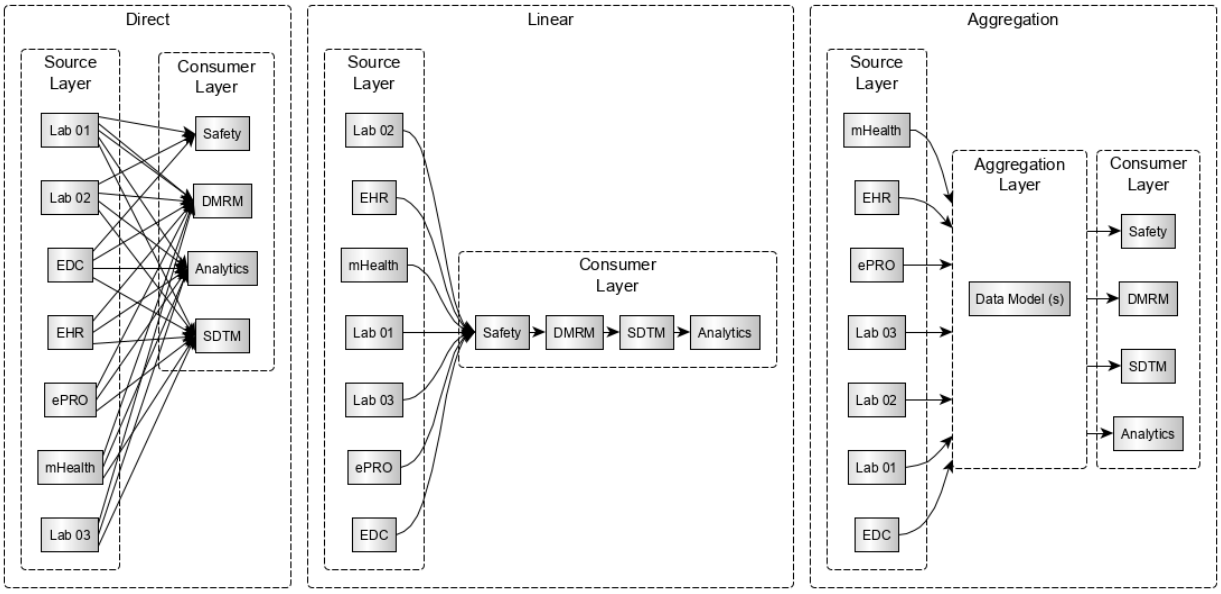

Figure 1. An example of direct architecture (left), linear architecture (center) and aggregation layer architecture.

Organisations that have experimented (and struggled) with the direct design occasionally go for a second approach, the **linear architecture**, as illustrated in Figure 1 (center). This architecture reduces the number of point-to-point transformations needed when adding a new source or target. Furthermore, it may seem logical that derivations would flow linearly from source to Analysis, going through any intermediate data model like SDTM. A third option is the **aggregation layer architecture** (Figure 1, right). The aggregation layer is a layer that segregates the source layer from the consumer layer. Source variability is addressed by transformations from source layer to aggregation layer, whereas the transformations from aggregation layer to consumer layer are standard and re-usable. The next section compares these two architectures in more detail.

CASE STUDY: MOVING FROM LINEAR TO AGGREGATION LAYER ARCHITECTURE

A top 10 Pharmaceutical organisation that conducts trials in different therapeutic areas had worked for decades using a direct architecture. Struggling with limited re-usability, they made a transition to a linear architecture.

They had defined the following four data models in the consumer layer:

- **Safety** for medical reviewers;
- **DMRM,** or Data Management Review Model, for data managers;
- **SDTM** for regulatory submission;
- **Analytics** for reporting, including Tables Figures and Listings (TFLs).

Their linear architecture is illustrated in Figure 2 (top).

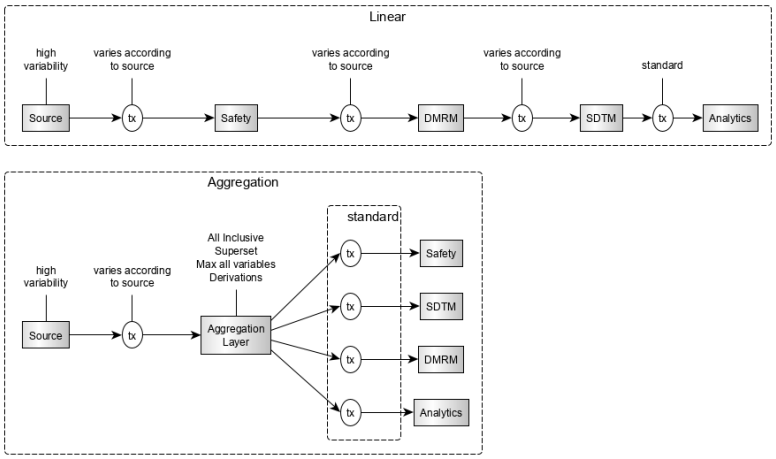

Figure 2. linear architecture on top, aggregation layer architecture at the bottom. The Source indicates the source layer, which includes EDC, labs, and any other source to the clinical trial. The white circles labeled as "tx" indicate the set of transformations.

The data models in their source layer changed frequently, as new data sources were added to new trials, and as Case Report Forms differed from trial to trail. Therefore, the transformations from the source layer to the Safety data model would change from study to study. The variability in transformations would also mean variability in transformations from Safety to DMRM, and from DMRM to SDTM. Only the set of transformations from SDTM to Analytics was stable, as this organisation had their own SDTM IG and a standard set of TFLs for reporting.

Eventually, they decided to move away from the linear architecture and they settled on the aggregation layer architecture, as illustrated at the bottom of Figure 2. Their aggregation layer is a superset of all the tables of the source data models, and it includes all variables, as well as any derivation needed for the downstream data models (Safety, SDTM, DMRM and Analytics). The study-by-study variability is now managed between the source layer and the aggregation layer, and within the data models in the aggregation layer. However, the downstream transformations are standard, reducing the need for programming.

LIMITING THE IMPACT OF ERRORS

One frequent problem they had with a linear architecture was that any issue in a data model propagated immediately to any downstream data model, as illustrated in Figure 3.

In our example, if something went wrong in the Safety data model, it would affect the downstream transformations, and prevent users of DMRM, SDTM and Analytics from working. Even worse, if the users of DMRM, SDTM and Analytics were unaware that any problem occurred at all, they would continue working on stale data, leading to a proliferation of queries out of sync, or reports on old data. In contrast, the aggregation layer segregates the data models in the consumer layer. An error on the Safety data model doesn't affect DMRM, SDTM or Analytics, and therefore their users can continue working.

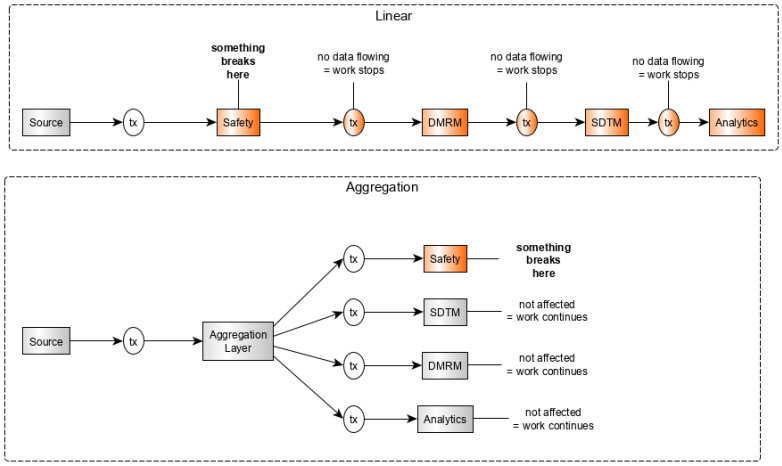

Figure 3. An example of issue to an early data model. In the linear architecture (top), an error in Safety affects all the work downstream. The aggregation layer (bottom) segregates the downstream data models, and therefore when an error occurs in Safety, work can continue elsewhere.

REDUCING REDUNDANCY TO SIMPLIFY MAINTENANCE

In the linear architecture, variables that were needed downstream had to be available in earlier data models, as in Figure 4. For this organisation, this meant that the Safety model not only had tables and variables for the medical reviewers, but also tables and variables for data management, SDTM and analytics. In turn, their DMRM included tables and variables for SDTM and analytics, and the SDTM data model included analytics.

This duplication was making maintenance hard. For example, an error occurring in DMRM would affect SDTM and Analytics down the line. Fixing the error in a table or variable required the identification of all the downstream models where that information was duplicated. On the other hand, with the aggregation layer architecture, all variables are present in the data models of the aggregation layer, and only the relevant variables are included in each of the data models to the right, which simplifies maintenance.

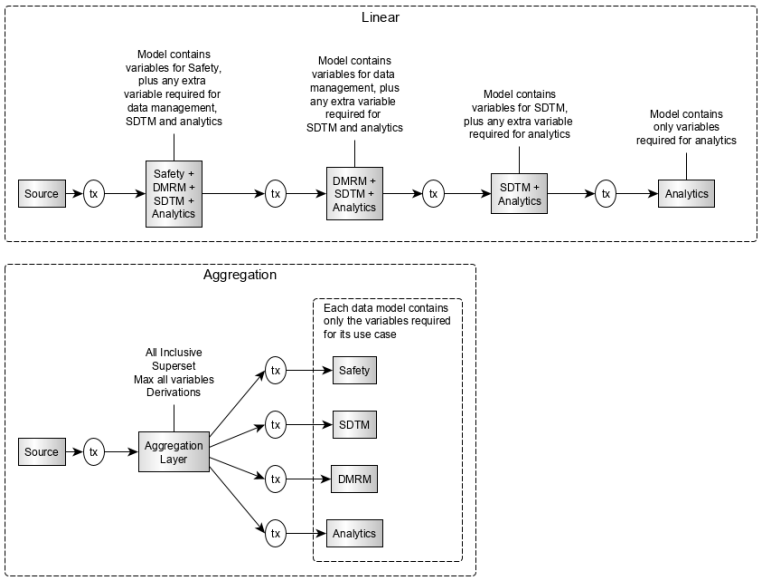

Figure 4. In a linear architecture (top), each model must contain sufficient data for the calculation of variables required by each downstream model. In the aggregation layer architecture (bottom), each downstream model contains only the relevant variables.

AVOIDING UNNECESSARY TRANSFORMATIONS BETWEEN NORMALISED AND DENORMALISED DATA

No single data model is the best structure for all uses. All data models have strengths and weaknesses depending on how data are stored and queried. There is no single answer to the question of which data model is better, other than it depends on your use case.

For example, SDTM is likely the most used CDISC standard, and it is the required or preferred format for data tabulations by most regulatory agencies. That is because normalised data (i.e. "long and skinny" or vertical tables) are well suited for programmable queries, like SQL `select` statements, reporting functions such as SAS® `proc freq`, and analysis algorithms. On the other hand, humans prefer to review de-normalised data (i.e. "short and fat" or horizontal tables). Figure 5 illustrates these differences.

Normalized

STY1A	CTR1N	SBJ1D	VIS1N	ETSTNM	ETSTRES
ACB123D456	25067	26557	1	HR	87
ACB123D456	25067	26571	1	HR	58
ACB123D456	25067	26585	1	HR	98
ACB123D456	25067	26599	1	HR	56
ACB123D456	25067	26613	1	HR	78
ACB123D456	25067	26627	1	HR	75
ACB123D456	25067	26641	1	HR	76
ACB123D456	25067	26655	1	HR	68
ACB123D456	25067	26669	1	HR	98
ACB123D456	25067	25914	1	SYSBP	120
ACB123D456	25067	25997	1	SYSBP	112
ACB123D456	25067	26499	1	SYSBP	115
ACB123D456	25067	26513	1	SYSBP	124
ACB123D456	25067	26528	1	SYSBP	145
ACB123D456	25067	26543	1	SYSBP	123
ACB123D456	25067	26557	1	SYSBP	154
ACB123D456	25067	26571	1	SYSBP	133

De-normalized

STY1A	CTR1N	SBJID	VIS1N	HR	SYSBP	DIABP	HT	WT
ACB123D456	25067	26585	1	98	154	98	174	78
ACB123D456	25067	26513	1		124	67	177	56
ACB123D456	25067	26599	1	56	122	56	182	91
ACB123D456	25067	26641	1	76	130	76	192	46
ACB123D456	25067	26655	1	68	111	68	177	87
ACB123D456	25067	26669	1	98	143	98	188	84
ACB123D456	25067	26499	1		115	55	166	55
ACB123D456	25067	26627	1	75	127	75	188	56
ACB123D456	25067	26683	1	56	132	56	184	92
ACB123D456	25067	26697	1	67	128	67	156	91
ACB123D456	25067	26711	1	89	137	89	182	83
ACB123D456	25067	25914	1		120	87	185	87
ACB123D456	25067	25997	1		112	93	156	63
ACB123D456	25067	26528	1		145	87	189	82
ACB123D456	25067	26557	1	87	154	87	167	78
ACB123D456	25067	26613	1	78	114	78	166	64
ACB123D456	25067	26571	1	58	133	58	177	66

Figure 5. Two extracts of the same data set in normalised (left) and de-normalised tables. For a data manager, the table to the right is more readable. It is obvious quickly that there are 5 subjects who are missing a heart rate value. This would be difficult to identify in the normalised structure. However, to identify outliers across the entire set of values for a specific test, it would require much more programming to do this against the de-normalised structure.

This organisation had standardized most of their data collection on CDASH, which includes normalised tables, also present in SDTM. However, their medical reviewers, data managers and analytics & reporting team had a preference for dealing with de-normalised data. This situation is illustrated in Figure 6.

In the linear architecture, that meant going through multiple rounds of de-normalization of normalised tables, and normalization of de-normalised tables. This translated in a series of pivot/unpivot transformations in an SQL world, or `proc transpose` in a SAS® world. This step seemed unnecessary and it was error prone. The aggregation layer architecture took away this "flip flopping" between normalised and de-normalised data, as each data model in the consumer layer is now mapped directly from the aggregation layer.

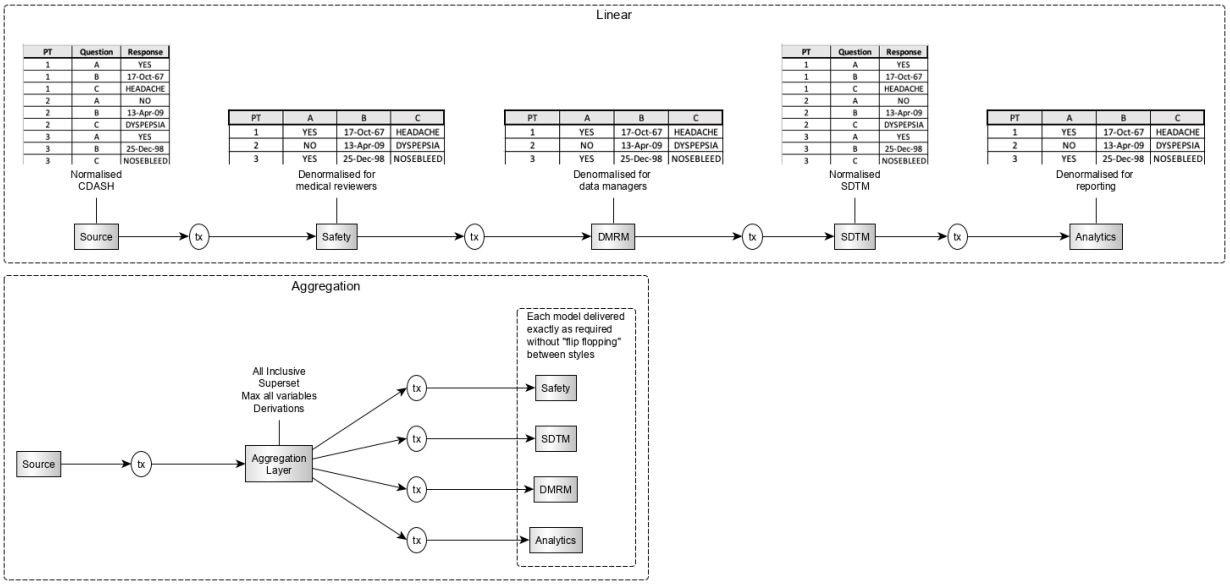

Figure 6. In the linear architecture (top), user requirements dictate different shapes for the tables in the data models, which results in a cascade of unnecessary transpositions ("long-and-skinny" to "short-and-fat" and vice versa).

KEEPING CONTROL OF BLINDING

The term “masking” is sometimes used interchangeably with “blinding”, although there is some disagreement on which term should be used, as exemplified by BMJ’s “Masking is better than blinding” (Morris, Fraser, & Wormald, 2007) and subsequent letter to the editor “Blinding is better than masking” (Schulz, Altman, & Moher, 2007). In this paper, the term “blinding” indicates any intentional masking of information contained in a table, columns, rows or cells (see Figure 7) to facilitate trial designs that required blinding of the data to one or more individuals. Masking is just the method to blind or hide the actual values of the data.

STY1A	CTR1N	SBJ1N	VIS1N	SMPCOL1D	PARNAM1C	LABRSL1A	LABRSL1N	PARUNT1C
ACB123D459	10	26357	9	23-jul-07	SGOT	XXXXXX	9999	U/L
ACB123D459	10	26357	9	23-jul-07	SGPT	XXXXXX	9999	U/L
ACB123D459	10	26357	9	23-jul-07	SOD	137	137	mmol/L
ACB123D459	10	26357	9	23-jul-07	TBIL	0.8	0.8	mg/dL
ACB123D459	10	26357	9	23-jul-07	TCHOL	185	185	mg/dL

Figure 7. An example of cell-level blinding. Columns LABRSL1A and LABRSL1N represent the same value, in string and numeric respectively. For this patient, the results of the SGOT and SGPT tests are masked with ‘XXXXXX’ (string variable) and ‘9999’ (numeric variable) to hide (or blind) the actual lab values for this patient.

At this Pharmaceutical organisation, a blinded study could result in different blinded schemas for different uses, as illustrated in the hypothetical example of Figure 8. Implementing different blinding schemas in the linear architecture required careful programming, as any mistake in one blinding schema could propagate to other data models downstream and unintentionally break the blind.

Furthermore, each interim analysis required the study team to interrupt the data flow, temporary unblind the tables needed by the Data and Safety Monitoring Board, restore the blinding and re-start the data flow. During this time, the access to the data would be suspended for all users.

In contrast, the aggregation layer is treated as a black box, and the blinding implemented separately for each data model in the Consumer layer. This takes away the need for any special workflow at interim analysis, and reduces the risk of unintentionally breaking the blind.

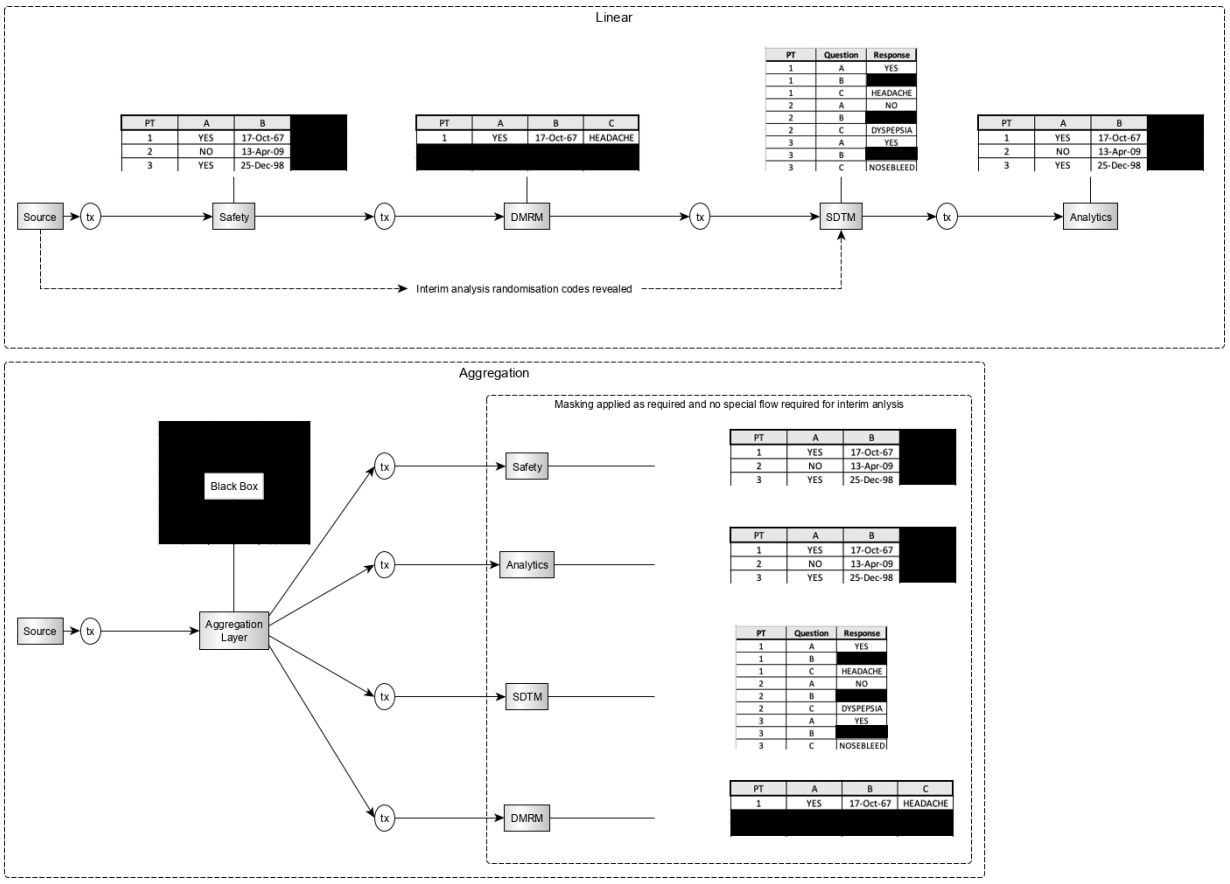

Figure 8. An example of different blinding schemas for different data models. Tables are blinded at the level of the columns (Safety and Analytics, hiding entire columns), of the rows (DMRM, hiding entire patient visits) and of the cells (SDTM, some values for some visits).

SDTM "PLUG-N-PLAY"

Occasionally, this organisation wanted to add a new SDTM version to a study, or a set of studies in the clinical development program of a compound, as illustrated in Figure 9. This could happen both for live studies and for completed studies.

In the linear architecture, this required a new set of programs to transform DMRM to the new SDTM version, and then a new set of programs to transform the new SDTM to Analytics. In a live trial, both sets of transformations would have to be maintained.

In contrast, the aggregation layer architecture requires only the set of programs that transform from Aggregation to the new SDTM. Since SDTM is a global standard, and the aggregation layer is maintained as an internal standard, the organisation looks at adding SDTM versions as a plug'n'play mechanism.

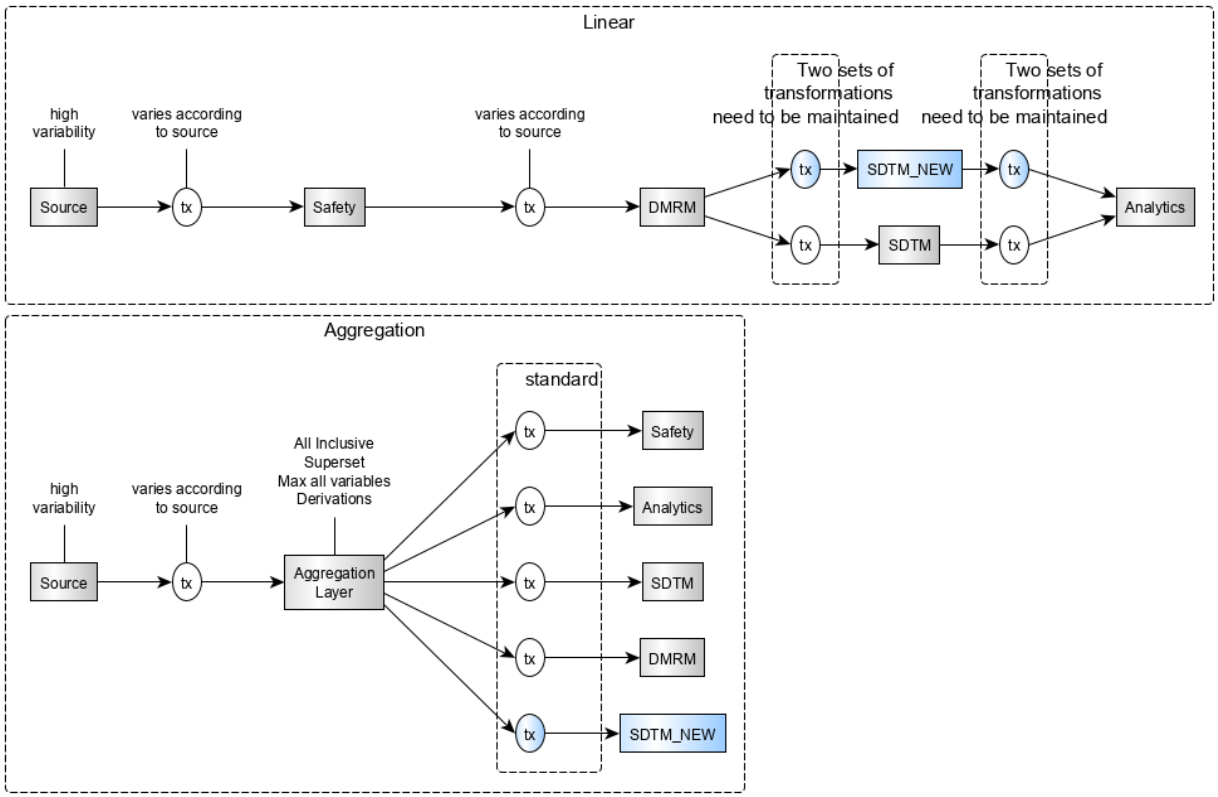

Figure 9. An example of adding a new SDTM version to an existing study.

BEST PRACTICES IN AN AGGREGATION LAYER ARCHITECTURE

Here are three practical tips on how to implement the aggregation layer itself.

First, this layer should be hidden to end-users (e.g. data managers, medical reviewers), as its only purpose is to serve data to multiple data models that are relevant for the end-users. One way to do it is to restrict access to users who build data models and transformations, but not to end users such as data managers, medical reviewers, or statisticians. Another way is to blind every table of the aggregation layer, although this is harder to do if you are using file-based technology.

Second, if a derivation is needed by more than one model in the consumer layer, then it should be derived in the aggregation layer. For example, if Gender is used by both DMRM and SDTM, then it should be derived in one of the tables in the aggregation layer and exposed to both DMRM and SDTM through simple transformations. This reduces programming and increases robustness. For the same reason, one should also consider performing all complex calculations in the aggregation layer and keep downstream transformations as simple as possible.

Third, many of the benefits of the aggregation layer are realized when its structure is preserved across studies. To this extent, the standards team should consider adding the data models in the aggregation layer to their internal standards.

CONCLUSION

This work discussed three gaps between the CDISC standards and the way clinical data science is conducted today. The first gap is that CDASH is a standard for how data is to be collected in the UI, not for raw data output that is extracted from the source solution. This can be addressed by applying the EDC extract rules to the data collection standard, 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). The second gap is the lack of a standard that could be used by Data Management for data cleaning. Organisations often address this by developing internal standards for data management, such as the Data Management Review Model (DMRM). The third gap is the lack of a deliberate data architecture, derived from a separation of transformations programming and of data model definitions, which are looked at as independent activities and conducted by separate groups. To this extent, the paper introduced three architectures. Traditional programming is equivalent to a direct architecture, which has severe limitations in re-usability of transformations across studies. A linear architecture is the alternative approach of transforming all sources to one data model, and then having a linear chain of transformations to derive the next data models. It can help to reduce the number of transformations, but it lacks in robustness. The aggregation layer is a third approach that adds a third layer to segregate source and consumer models. In a case study that compared linear and aggregation layer architectures, the aggregation layer was more robust to errors; reduced the number of unnecessary transformations between normalised and de-normalised structures; led to easier implementation of blinding; and enabled an SDTM plug'n'play approach.

Although the message is intended to be agnostic to the technology used to run clinical trials, the work described in this paper was based on projects conducted using a clinical data management platform solution, powered by a data warehouse. More work with flat-file based systems is needed to validate the message. Additionally, the example described in this paper is with large Pharmaceutical organisations managing development portfolios with more than a hundred concurrently running studies. These organisations have internal standards teams. Smaller organisations may lack the resources for an internal standards team, and therefore may be confined to use the Direct architecture that has limited reusability across studies or be far more reliant on external organisations to define and update standards

REFERENCES

- Morris, D., Fraser, S., & Wormald, R. (2007). Masking is better than blinding. *BMJ*, 334(7597), pp.799-799.
- Nadolny, P., Zambas, D., Torche, F., & Bhardwaj, S. (2019). The Evolution of Clinical Data Management to Clinical Data Science. *Society for Clinical Data Management*.
- Schulz, K. F., Altman, D. G., & Moher, D. (2007). Blinding is better than masking. *BMJ*, 334(7600), pp.918-918.

CONTACT INFORMATION

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

Claudio Garutti
Oracle Netherlands
Hertogswetering 163-167, 3543 AS Utrecht, Netherlands
Email: claudio.garutti@oracle.com

Marko Bomyer
Oracle Spain
Avinguda Diagonal, 615, 08028 Barcelona
Email: marko.bomyer@oracle.com

Jonah Brugger
Oracle America
10 Van de Graaff Drive Burlington MA 01803, United States
Email: jonah.brugger@oracle.com

Julie Smiley
Oracle America
10 Van de Graaff Drive Burlington MA 01803, United States
Email: julie.s.smiley@oracle.com