

## *From Domain Analysis Model to Common Data Model - the trials and tribulations of implementing BRIDG in an Information Technology Environment*

By Terry Hardin and Isabelle deZegher

### **Overview**

Over the past several years there has been much discussion regarding BRIDG (Biomedical Research Integrated Domain Group) within CDISC and HL7 as a Domain Analysis Model (DAM). These discussions center on ways that BRIDG can be used as semantic “glue” across evolving and established standards. However, there has been little work in moving BRIDG from “glue” to a common data model (CDM) that could be implemented within an IT infrastructure. In this new role BRIDG 3.0.1 would be used to move from a conceptual model with abstract data types to an implementable data model with simple data types that, in combination with business process rules, would allow for flexible system integration.

### **The Challenge**

The most common method used throughout the industry for system integration is point-to-point. An Enterprise Service Bus (ESB), generally presented as the solution for integration, works at the syntactic (programming) level while the semantic (meaning) integration remains point-to-point as long as the data models from the underlying applications are not harmonized.

With point-to-point mapping the approach is to build a set of mapping files between each pair of systems. This does provide a “quick fix” to system integration challenges, but it also raises a significant resource issue as the number of mappings increases with the number of systems to be supported. In FIGURE 1 each arrow represents a set of mapping files between systems. This makes point-to-point mappings both difficult and costly to maintain, limits re-use, and the data lacks consistency from one system integration to the next. The problem with a point-to-point interface becomes obvious when one (or more) system(s) require a release update: the interfaces need to be thoroughly re-tested and re-validated as internal system variables may have changed.

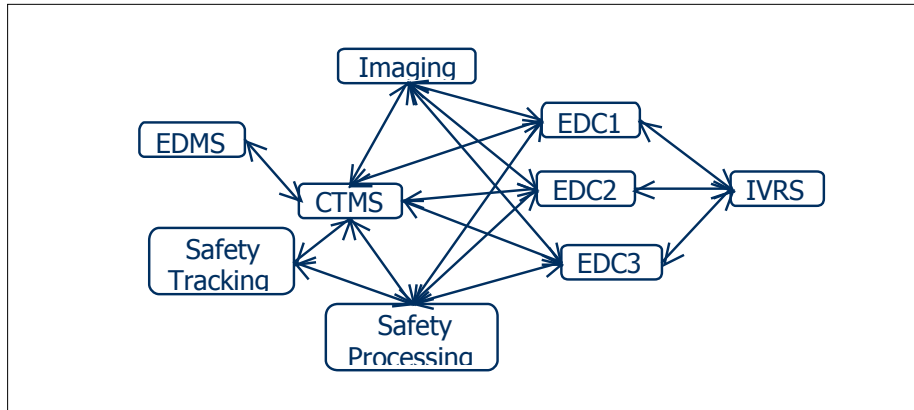


FIGURE 1 – Point-to-Point Interface Example.  
 In a CRO environment, several EDC packages need to be supported, increasing the complexity of integration.

Since BRIDG is a universally accepted standard as a Domain Analysis Model (DAM), PAREXEL looked at how we could extend this model for practical implementation to move from a *point-to-point* interface paradigm to a *multi-system* paradigm with business process rules. In our approach, BRIDG becomes the foundation for a common data model to be used for *multi-system* integration. FIGURE 2 illustrates where each system is mapped to the common data model (PAREXEL Physical Data Model or PDM) based on BRIDG. In this configuration each system has a mapping to/from BRIDG through the PDM. If a specific system is updated then only the mapping to/from the PDM needs to be tested, validated and potentially changed – not all the mappings for all related single interfaces as would be necessary in a point-to-point configuration.

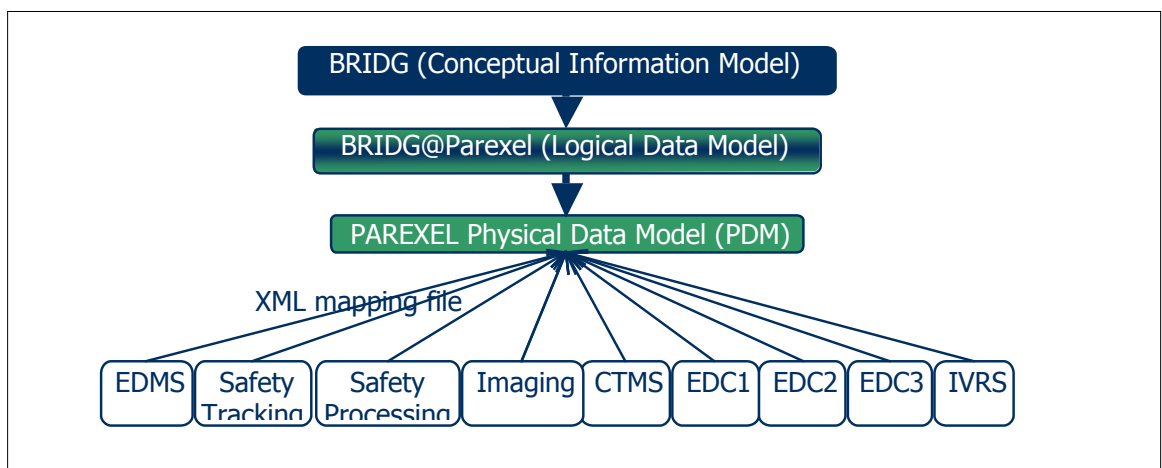


FIGURE 2 – Common Data Model for Multi System Integration

This is somewhat confusing, so we would like to examine the Safety Processing System as an example. In FIGURE 1 there are five *point-to-point* interfaces for this system: one for each of the EDC’s in place at PAREXEL, one for the investigator Safety Tracking System and one for the Clinical Trial Management System. In FIGURE 2 there is just one single interface (and associated mapping) to the

PDM. If the Safety Processing System is updated only the Safety/BRIDG mapping needs to be re-tested and validated and potentially changed – not each of the five mappings outlined in FIGURE 1.

## The Value Proposition

Before we could move forward with this project it was essential that we defined the value proposition of a common data model. Our analysis indicated the following three key values that could be obtained.

1. Build a global, implementable data model with precise and standard data element definitions that could be used across technology platforms and applications. This would allow the various systems in our enterprise to “understand each other” through mapping across all the different systems that are in use and that required interfaces. Thus, we could have true **computer semantic interoperability** on top of **technology integration**.

2. Developing an organization specific implementable data model from an externally vetted conceptual model that allows for a phased and efficient approach.

BRIDG is a reasonably stable standard after 6 years of work within the clinical research community. Even if we start with a few use cases and a subset of the model, we can be confident that the model will remain stable when adding new use cases. In addition, using BRIDG allows us to speed along internal consensus and provides for mapping to the HL7 Reference Information Model (RIM), while introducing ISO 21090 standards on Abstract Data Types.

3. Facilitate consistent set-up of systems by having programmers consult the data element definitions provided with the common data model. Most systems used to support clinical trials require study specific set-up, executed by different teams across the organization. Enforcing common data element definition ensures consistency across systems and correct behavior of the interfaces.

After discussing these value propositions with several of the internal teams the pilot project was approved to move forward.

Next we will examine the process undertaken at PAREXEL to move from BRIDG at a conceptual level to a BRIDG common data model implementation in our organization.

## The Process

Once we made the decision to use BRIDG as the basis for integration mappings we were faced with the task of scaling the full BRIDG model down to specific classes to implement. We determined that four specific use cases applied regularly in studies independently of the system – patient, visit, site and user – and these would be a good place to start to map to the appropriate BRIDG classes.

### *Step 1 – SCOPE: Move from BRIDG to Logical Data Model (LDM)*

For this work the challenge was to take the larger BRIDG DAM and scale the sub-domains down further to just the classes and attributes needed for the four specific use cases centered around visit, site, patient, and user. Using a UML modeling tool we started with building the basic LDM as a starting point to see how these four use cases would be modeled. An overview of this work is outlined below.

### Classes and Attributes

- Identify concepts of interest in the interface
- Identify relevant BRIDG classes (including related classes supporting the full concept)
- Copy BRIDG classes within LDM
- Look at class attributes and choose which ones are needed; keep the ones not needed in grey (as they may be needed for future interfaces)
- For each attribute,
  - If this is a simple data type – keep as such
  - If this is an Abstract Data Type (ADT) – follow the conformance rule, as described below, for transformation of the ADT into simple data types.

#### *Step 2 – from Abstract Data Types to Simple Data Types*

The Abstract Data Type (ADT) as defined in ISO 21090, like coded data types, identification and location data types, person, address data types are complex data types represented by an object where the attributes are in turn defined either by an ADT or by a simple data type like boolean, integer, char, .... Our next step was consequently to define strict conformance rules defining the transformation of a ADT into a set of simple data types while ensuring consistency across the LDM. An overview of this work is outlined below.

Approach for conformance rules from ADT to simple data types were defined

- For each ADT identify
  - Which attributes need to be kept in the logical model
  - Which attributes should not be kept in the logical model but for which a value should be assumed as implicit (and can be retrieved from the conformance rules afterward in the case of mapping)
  - Which attributes should not be kept at all
- If there is more than one attribute to be kept to express the ADT, define a class
- In the logical model replace the attribute with a link to the class (typically abstract data will become a class in the logical data model)

Additionally, we defined a PAREXEL specific sub-domain that contained classes to define simple data types. A snap-shot of this sub-domain is illustrated in FIGURE 3 below.



### *Step 3 – LDM to PAREXEL Physical Data Model*

The work here was similar to that in Step 2 with relation to conformance rules. The main difference here is there was now the denormalization of some of the attributes, and we worked to define code lists used for the coded data types to ensure value binding.

### *Step 4 – Generation of XML*

Here we used the modeling tool to generate the XML schema that would be used by the internal mapping application, together with the business rules, to integrate system to system. Our first set of work was done based on CTMS to EDC systems as this is one of the sets of systems most commonly deployed in our business.

### *Step 5 – The CRUD Matrix (Create, Read, Update and Delete)*

Once the mapping work was completed the challenge was to identify “who owns the data”. We worked on developing a CRUD matrix that could be used to determine which systems were “masters” and which systems were “slaves” in relation to the data. Where two applications share a Create (C) and/or a Update (U), there is a potential data reconciliation issue that may need to be solved either through a process change or through manual intervention. For this step the following provides an overview.

## **Data Reconciliation**

- Need to define the authoritative data sources
- Understand the impact on existing business processes – and agree on change. For instance when two different teams have a process describing how to enter site information in, respectively, the EDC and the CTMS systems we must decide which system will be the master, which team will keep entering data, and which team will have to change their process.
- Agree on the data reconciliation process in the case where data is still entered in a “slave” system
- Evaluate the possibility of implementation, and potentially identify alternatives
  - other technologies or other processes
  - or delay and discuss with vendors on an approach when technology does not allow for resolution (API, Web Services, etc.)
- Develop and deploy the solution as part of the interfaces

There were a few “bumps in the road” as we moved through this pilot, but support from BRIDG experts in the industry helped us overcome these. Additionally, we found some “bugs” in BRIDG. Some were as simple as the relationships in BRIDG being defined in the wrong direction, while others involved classes that we believe should be in BRIDG, are in line with what has already been defined in BRIDG, and are currently missing. We will be pulling these findings together to report to the BRIDG SCC so they can determine how best to proceed.

## Conclusion: The BRIDG CDM Revolution

In an era of pumping out interface updates every time a system revision occurs, implementing BRIDG as a common data model breathes new life into the system integration process. By employing the BRIDG CDM as a revolutionary tool to provide for a common communication infrastructure it provides a mechanism for enforcement of data standards, and to ensure compliance with standards – helping PAREXEL from system set-up to system integration.

In the future, a BRIDG CDM can provide for better system integration within PAREXEL, as well as with our external customer systems. Once the BRIDG CDM infrastructure is built, it can be rolled out to agencies and industry around the world allowing for the use of standard data sets – enabling faster study set-up, easier communication of issues, and better data mining.

---

© Copyright CDISC October 2011 [cdisc.org](http://cdisc.org)