PharmaSUG 2013 - Paper AD10

A Generic Concept to Handle SDTM (and other CDISC) Data Sets

Peter Schaefer, BASS, LLC, Raleigh, NC

ABSTRACT

CDISC-defined standards – like SEND, SDTM, or ADaM – are specified to be compatible with the SAS® Transport Version 5 file format (".xpt-files"). Of course, this compatibility – and the widespread use of .SAS throughout the pharmaceutical industry – very often makes SAS the tool of choice when it comes to reading and writing study data in any of the CDISC-defined formats. In addition, other tools include SAS transport file access as an extension to their use of native data file formats – like the widely used open source package R with the 'SASxport' package. However, working with a CDISC data set requires more than reading or writing the specific .xpt-file for each domain; for example, standard compliance requires that certain relations between data are maintained or created or that controlled terminology is used throughout the data set.

This short paper presents the concept called CDISC- Helper of a data format independent software package to handle datasets that are formatted following the data model used for SDTM and other CDISC-defined standards. A suggested implementation of this concept – in part as a Web Service demonstrates how programmers can focus on data analysis and workflow issues in their programs instead of data management. In addition, the additional level of abstraction supports other file formats than SAS transport files as long as CDISC metadata requirements are met. With this kind of approach, programmers can take advantage of the many choices when it comes to select the appropriate software technology for a program. They are no longer limited by one aspect – file access – when other aspects might call for using a different more modern and appropriate technology.

INTRODUCTION

With the recent authorization of the Prescription Drug User Fee Act (known as PDUFA V [1]) a deadline for the mandated use of data standards in submissions to the FDA is coming closer. Consequently, pharmaceutical and biotech companies are increasingly investing into processes and tools to handle data sets that are formatted according to CDISC-defined standards – specifically, SDTM, SEND, and ADaM. The natural and evolutionary path forward for most companies is the use of SAS for handling CDISC data sets because the standards are defined in a way that assures compatibility with the SAS Transport Version 5 file format. However, software tool need to do much more than reading and writing data files. Other features are as critical for great tools, such as an easy-to-use user interface, fast access to one or multiple databases, or efficient handling of metadata. For software vendors which are driven to add handling of CDISC data sets to their existing tools or to develop new tools, a restriction to use SAS is often not even a choice because other programming environments provide more flexibility, productivity and stronger programming support. In addition, the weaker support of the general software developer community for the more specialized SAS environment could make it harder to meet quality, time and cost objectives compared to using a more commonly used IDE (Integrated Development Environment).

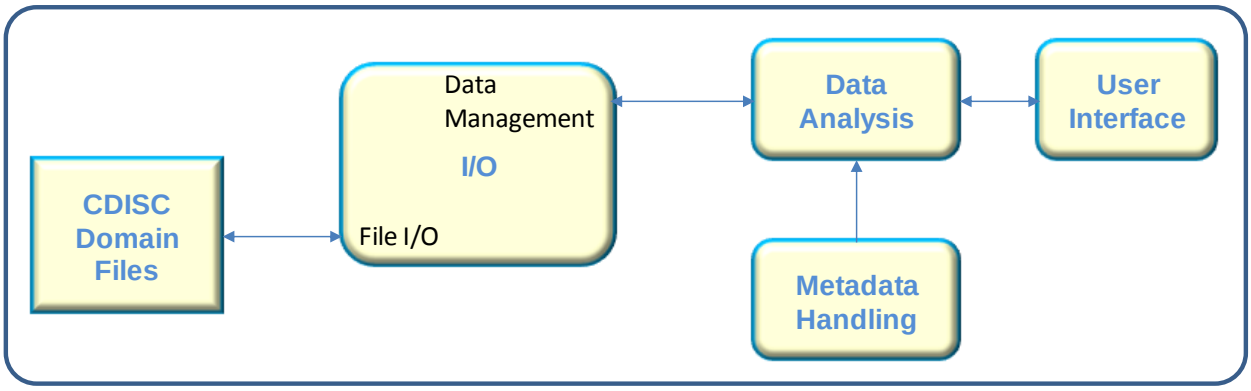

Figure 1. Generic Components of a Software Tool

When looking closer at the generic structure of a software tool (see figure 1 above), it also becomes clear that the interface between data processing and the data files is actually not a simple one-step-process in the case of CDISC formatted data sets. Just take a simple data analysis like NCA (Non-compartmental Analysis): NCA will calculate pharmacokinetics parameters from time-concentration profiles and, in addition, any NCA tool will calculate some simple summary statistical values across the patient population using some covariates such as patient demographics

data (like age, race) or baseline vital signs (like weight, height). From a tool perspective, this means that the data are not only read from the data file (like the PC, DM, and VS domain) but also need to somehow be merged together across multiple domains in a way that the resulting data set is – what we call – 'analysis-ready'.

A very similar two-step data preparation concept can be found in the architecture of the Janus Clinical Trial Repository (Janus CTR, see for example in [2]). The Janus CTR architecture includes the so-called 'Extended SDTM Views' in the SDTM Analysis Database which represents clinical trial data in the repository that are ready for analysis through JMP, JReview, or SAS. Note that this should not be confused with what some companies call SDTM+ or similar as an attempt to express that what they are using is 'kind of SDTM' but has certain additional elements that are not covered by the SDTM standard.

Finally, it's worth mentioning the point that maybe SAS Transport is not the final answer: Multiple initiatives and discussions seem to indicate that the community is looking at alternatives for the proprietary SAS Transport format. For example, a public meeting called by the FDA in November 2012 (see [3]) brought together industry experts for a brainstorming about "the advantages and disadvantages of current and emerging open, consensus-based standards for the exchange of regulated study data". Along a similar line of thinking, the CDISC organization is looking into the possibilities of using of the XML-based ODM standard as the basis of a file format for SDTM domain data. Of course, any replacement of transport files in clinical data standards would be a slow process and clear benefits of a replacement format still need to be seen, a forward looking strategy for handling CDISC data sets would probably be well advised to develop a system architecture that can account for possible changes of the underlying data format.

THE CONCEPT OF CDISC-HELPER

This paper presents a generic concept that separates the lower level access to the data files for the SDTM domains from the re-organization of the data that is required to create an analysis-ready data set. Note that this concept can be applied for a relatively simple case like the creation of tables, listings, and figures across domains as well as for data set preparation for straightforward NCA methods or for more complex pharmacometrics models. The main components of the concept have been successfully implemented in a proof-of-concept application using C#, R, and Microsoft's Visual Studio IDE. However, the purpose of the paper is not to promote a specific implementation or library but to generate a discussion about best practices to access CDISC data sets that provide efficient flexibility and powerful abstraction between the data (i.e. reading and writing CDISC data sets) and the rest of a tool (UI and data processing). Throughout the paper, SDTM is used as an example and the proof-of-concept application only handles a subset of all SDTM domains. However, it should be clear without any additional information, how this concept is generalized to handle additional SDTM domains or SEND and ADaM data sets.

SDTM DOMAIN FILE STRUCTURE

Regardless of the underlying file format, all SDTM domain files are pretty much organized in the same way: A programmer would look at it as a database table, i.e. as a set of data elements organized in vertical columns (identified by a column name) and horizontal rows. In addition to the Variable Name as the column name, the SDTM standard provides for additional metadata, for example, the Variable Description, the data Type, and controlled terminology for many columns. These metadata are typically required to process data correctly. And as mentioned above, any tool will typically need to re-organize the data and combine data from different domains to create a data set according to the specific tool requirements. The proposed two-step concept for accessing SDTM domain data separates the three components – file access, metadata access, and data management – and provides a well-structured modular approach to handling SDTM data sets.

CDISC-HELPER ARCHITECTURE

The figure below illustrates the structure of the CDISC-Helper architecture and the main components.

- The **File Handler** classes implement the transfer of data between domain files on the file level and structured data in the so-called Data Staging Area. By providing methods for different file formats with the same call signature, the File Handler can easily be implemented to work with a variety of underlying file formats other than SAS Transport.

  The one required method for reading SDTM transfers a domain file into a section of the staging area and add metadata. However, to provide more flexibility a set of access method such as explicitly opening and closing a domain file, transferring multiple files at once, and reading the first record or the next record are proposed in the concept as well. The methods for writing the content of the staging area into SDTM domain files are using domain-specific mapping files which define which the relationship between the columns of the staging area and the variables in the SDTM domains. The concept allows for describing relationships between data which result in the creation of additional RELREC records, for example, for the relation between a PK parameter and the time-concentration profiles that are used to calculate the PK parameter value.

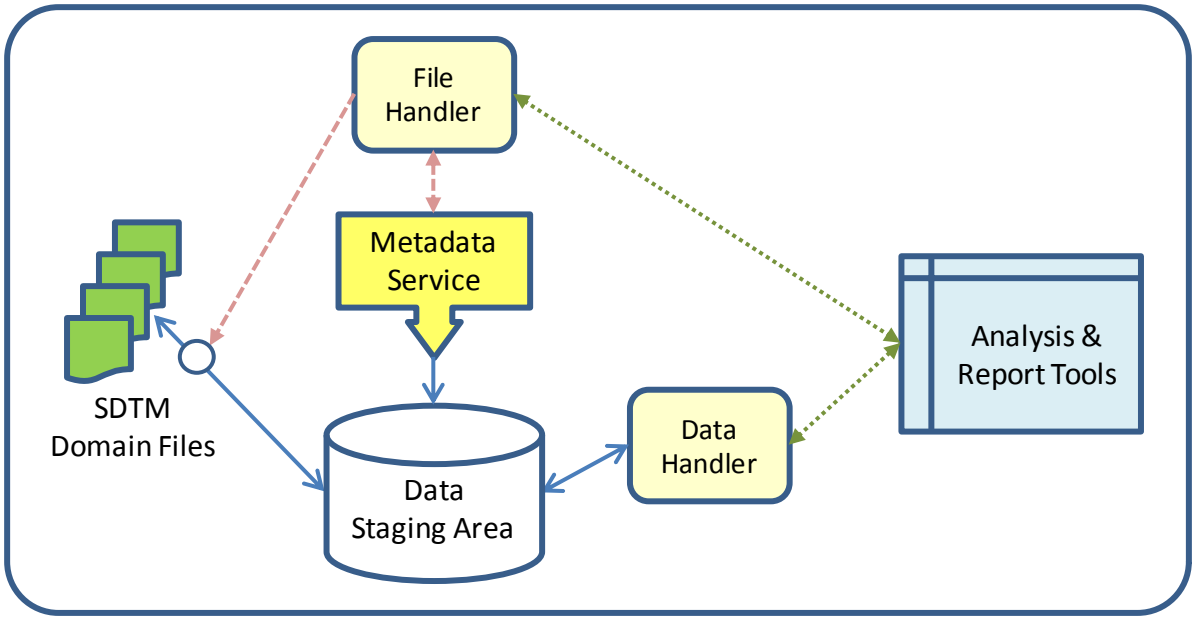

Figure 2. Structure and Components of CDISC-Helper

- The **Data Handler** class provides standardized methods to access the content of the Data Staging Area. These are the methods that are used when implementing a software tool that uses the CDISC-Helper concept to access the staging area and to extract analysis-ready data sets or to write result data to the staging area.
- The **Data Staging Area** is an intermediate container for the data that is independent of file format of the original SDTM domain files and supports efficient access to the data from software tools. From an architectural point of view, the Data Staging Area can be viewed as a black box that responds to a generic query language (used by the Data Handler methods) to read and write data. This can be implemented in different ways, using a database system, xml files, or appropriate data structures.
- The **Metadata Service** provides the domain and column level metadata of SDTM through a set of methods and is used by the File Handler. Whether metadata are required or not depends on the specific requirements of a concrete implementation of the concept. The use of metadata would enable certain validation check, for example, with respect to required variables, the use of controlled terminology, or compliance with data type specification. In general, it seems that using metadata is more critical when writing SDTM domain files. When data are read and prepared for analysis or reporting, it seems be possible to not incorporate metadata into the process. Note that for the proof-of-concept implementation, a generic metadata service for CDISC data was implemented as a Web service. This allows updating of the metadata without re-compiling the prototype application as long as the call signature of the Web service is maintained.

PROOF OF CONCEPT IMPLEMENTATION

This section describes some details of a proof of concept implementation of the CDISC-Helper using Microsoft technologies (C#, .NET, for example [6]), and some other tools, specifically, R [4]. The prototype is implemented using mainly basic data structures like lists, some of the newer concepts like dictionaries, are not used in the current implementation. Also, the prototype does not include all features of the CDISC-Helper concept but focuses on reading SDTM domain files and extracting data from the staging area for some simple analysis and reporting. The described implementation is work-in-progress and currently, features are added or modified on a regular basis to improve performance and ease-of-use of the resulting libraries.

INTERMEDIATE STORAGE OF DATA – DATA STAGING AREA

As mentioned above, the Data Staging Area is a data storage that responds to queries to read and write data. In the proof of concept, the staging area is implemented using C# data structures which support LINQ (Language Integrated Query, see for example,[5] or [6] for details about LINQ, but note that there are plenty of other books available) to access the data.

The implementation of the Data Staging Area is very much based in the List<T> data type that is available in .NET since version 2.0. The following describes the details from bottom up.

The basic element of the data stage is a column which is defined as follows:

```
/// <summary>
/// data structure for a single column
/// </summary>
public struct SingleColumn
{
    public headerElement colHeader;
    public List<object> colValues;
}
```

The values are stored as generic objects, which is very flexible but requires casting using the correct data type when processing the data. The column header is a generic structure that allows storing the variable name and other related metadata, specifically the data type that is used for casting the data values. In the prototype the header is implemented using the following structure and enumerations.

```
/// <summary>
/// Generalized column header
/// </summary>
public struct headerElement
{
    public string ColumnName;   // Variable Name in SDTM
    public string ColumnLabel;  // Variable Label from SDTM
    public TypeValue ColumnType;// Type of the values in the column
    public CoreUsage ColumnCore;// Value of Core column in SDTM
    public bool Controlled;     // True, for controlled terminology
}

public enum TypeValue
{
    Num,
    Char,
    ISO_DateTime
}

public enum CoreUsage
{
    Required,
    Expected,
    Permissible
}
```

Based on the column data type, the construction of a data set and of the whole data stage is rather straight forward: A data set is a list of columns with the appropriate metadata – at a minimum the name of the domain, i.e. a `dataSet` will be defined as follows:

```
/// <summary>
/// data structure for a whole dataset
/// </summary>
public struct dataSet
{
    public string domainName;
    public List<SingleColumn> dataMatrix;
}
```

In a similar way, the Data Storage Area is created by using a list of data sets and adding the required metadata – at a minimum the applicable standard.

```
/// <summary>
/// CDISC-Helper Data Staging Area class definition
/// </summary>
public struct DSA_Handle
{
    public string standardName;
    public List<dataSet> DataStage;
}
```

The data type `DSA_Handle` will be available to programmers through the Data Handler class, though from an architecture point of view it belongs to the Data Staging Area. When using the CDISC-Helper, a programmer needs to create an object of type `DSA_Handle` and use the methods that are provided by the Data Handler class.

READING SDTM DOMAIN FILES – FILE HANDLER

At this point, the transfer of SDTM domain files into the staging area is the most developed part of the concept. The prototype implementation uses the interface concept of .NET for the support of different file formats of the SDTM domain files. Currently, support for SAS Transport files (through the use of R and the SASXport package) and Excel or Comma Separated Value files (through use of a 3rd part library [7]) is implemented in the prototype.

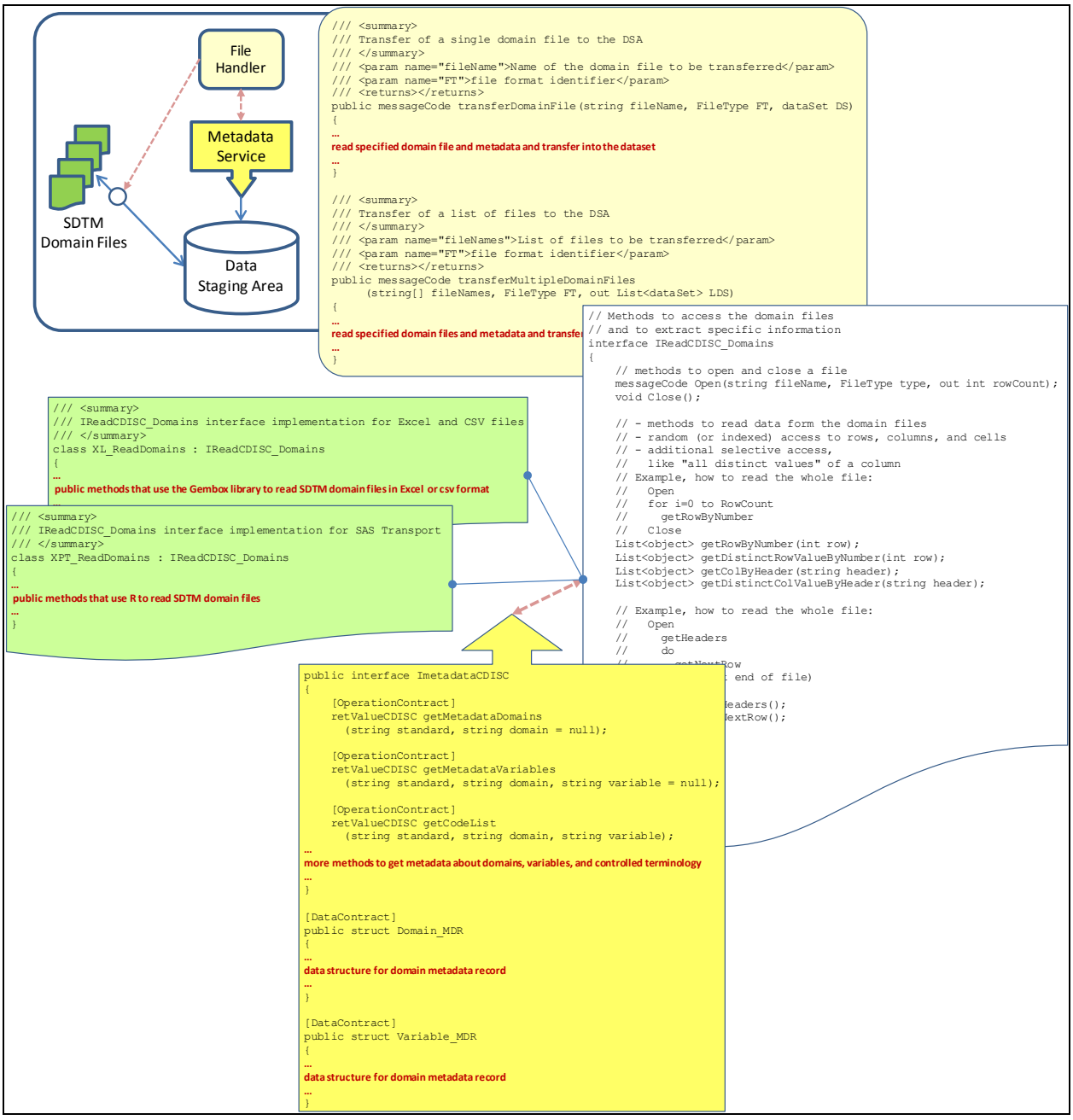

Figure 3. Implementation of File Handler using .NET 'Interfaces' and a CDISC Metadata Web Service

The classes `XL_ReadDomains` and `XPT_ReadDomains` (in green) are implementations of the methods defined in the `IReadCDISC_Domains` (in white) interfaces and by providing additional implementations other file formats, for

example XML, can be easily supported without changes to the File Handler methods. In a similar way, a set of methods to write domain files could be implemented by defining an interface and providing the appropriate implementations.

The metadata Web Service uses a SQL Server database to store the metadata about domains, variables, and controlled terminology. The methods of the Web Service are published through a standard SOAP interface that is consumed by the File Handler.

EXTRACTING DATA – DATA HANDLER

The foundation of the implementation of the Data Handler is the definition of an 'analysis-ready' data set. For the proof-of-concept, the assumption was that the time-value profile observed for a subject is the base structure that is required for analysis or reporting. Based on this structure, the prototype implementation also supports a group of profiles that is identified by a specified value for so-called group variables. This could be a treatment common to a group of subjects, or an analyte observed in the matrix.

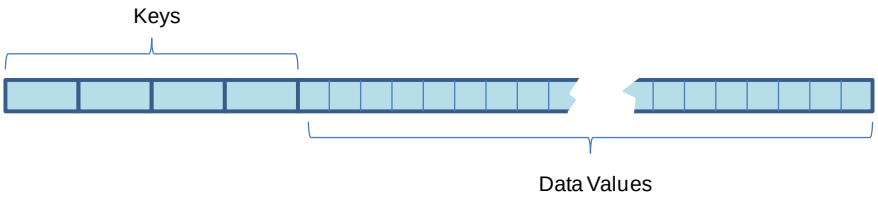

Figure 4. Structure of a data structure for a time-value profile

This structure is implemented as a list of keys and of values using a specific data types for keys and values.

```
/// <summary>
/// Basic structure for a time-value profile
/// the profileKeys and profileData variables will each
/// only have one element in the list of values
/// </summary>
public struct profile
{
    public List<keyElement> profileKeys;
    public List<valuePair> profileData;
}

/// <summary>
/// Basic data structure of an <ID, value> pair
/// used as key in data sets
/// ID is a string identifier, essentially the Key
/// </summary>
public struct keyElement
{
    public string ID;
    public List<object> Values;
}
/// <summary>
/// Basic data structure of an <ID, value> pair
/// used for values in a profile
/// </summary>
public struct valuePair
{
    public object ID;
    public List<object> Values;
}
```

Note that for a single profile the list of Values in each element of the `profileKeys` and `profileData` variables will have just one element, i.e. the use of a list is not exactly required. The data types are defined as lists, because the specification is supposed to support also groups of profiles as depicted in the figure below. The corresponding specification of a grouped data set with profiles looks like this:

```
/// <summary>
/// structure for a group of profiles
/// the groupKeys variables will only have one element in the list of values
/// </summary>
public struct group
{
    public List<keyElement> groupKeys;
    public List<keyElement> groupProfileKeys;
    public List<valuePair> groupProfileData;
}
```

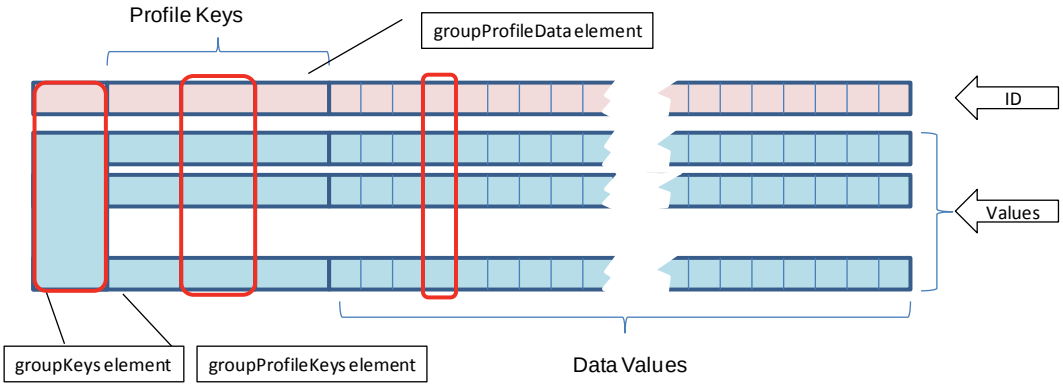

Figure 5. Structure of a data structure for a group of time-value profiles

The same basic structure can be applied to define a study, which again can be considered to be a collection of groups identified by some key. For the prototype implementation of the CDISC-Helper only profiles and groups of profiles are currently implemented.

Finally, a brief look at LINQ. LINQ is Microsoft's technology to separate access to and processing of data from the underlying data structure. LINQ was first released with .NET Framework 3.5 and is essentially a query technology similar to SQL. With LINQ, a programmer can change the underlying structure of the data being queried without needing to change the change the code that actually performs he queries. For the proof-of-concept implementation of the CDISC-Helper, LINQ was used to assure that the Data Handler is pretty much independent of the implementation of the Data Staging Area. From a programmer perspective, LINQ delivers means of very flexible transformation of data from one data structure into another – under the premise that both data structures implement the `IEnumerable` interface. A typical LINQ sequence for the proof-of-concept will include the definition of a query expression using the data structures of the Data Staging Area as the data source and the execution of the query where the data are placed into the profile or group data structures as defined above. To prove the flexibility of the CDISC-Helper concept the report implementation allow the user to select certain parameters for creating an output object (like a table or graphs) and the LINQ expression is formed dynamically.

CONCLUSION

This paper presents a concepts how to handle SDTM study data in way that easily supports different underlying file formats and the creation of data sets that can be directly consumed by analysis and report tools. The concept separates reading of the files and extracting data sets into two steps by introducing the a data staging area, a storage area that can be queried like a database, therefore providing the flexibility to extract data sets according to the requirements of any tool that needs to consume the study data.

Some features of the concept were implemented in a proof-of-concept implementation using C#, .NET, and R. The experience with the implementation shows that the concept works as expected and provides an easy-to-use API (Application Programming Interface) to handle CDISC domain data files. The prototype program reads domain files formatted as CSV and SAS Transport files. It extracts the profile of subjects in a structure that can be used to create time concentration tables or graphs or to calculate pharmacokinetics parameters using a simple NCA algorithm.

REFERENCES

1. "PDUFA V: Fiscal Years 2013 – 2017". Last Updated: 11/26/2012. Available at http://www.fda.gov/ForIndustry/UserFees/PrescriptionDrugUserFee/ucm272170.htm
2. Rosario, Lilliam. "CDER's Computational Science Center. Modernizing The Review Process Through Innovation". Presentation at FDA/PhUSE Computational Science Symposium, March 18th – 19th, 2013, Silver Spring, MD, USA.
3. Meeting Report "Information from the November 5, 2012 Solutions for Study Data Exchange Standards Meeting". Last updated: 12/20/2012. Available at http://www.fda.gov/Drugs/DevelopmentApprovalProcess/FormsSubmissionRequirements/ElectronicSubmissions/ucm332003.htm
4. "The R Project for Statistical Computing". http://www.r-project.org/
5. Pialorsi, P., Russo, M. 2010. "Programming Microsoft® LINQ in Microsoft .NET Framework 4". Redmond, WA. Microsoft Corporation.
6. Freeman, A. 2010. "Introducing Visual C# 2010 (Expert's Voice in .NET)". New York City, NY. Apress Media LLC.
7. Gembox Software Company Website. http://www.gemboxsoftware.com/

CONTACT INFORMATION

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

Peter Schaefer
Business Analysis Software & Services, LLC
184 Locke Woods Rd.
Raleigh, NC 28603
(512) 431-9785
pschaefernet@yahoo.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.