Michigan Questionnaire Documentation System, Version 3 (MQDS-V3)

Karl Dinkelmann, Nicole Kirgis, Gina-Qian Cheung, University of Michigan

1. Introduction

The Michigan Questionnaire Documentation System (MQDS) was designed to extract comprehensive metadata from Blaise survey instruments and render it as an eXtended Markup Language (XML) document using the Data Documentation Initiative (DDI) standard. The DDI standard is an international effort establishing metadata markup standards for expressing social science data. In April 2008, the DDI Alliance released the third version of specifications, which added rich support for documenting Computer Assisted Interviewing (CAI) instruments. The current version of MQDS has been updated to exploit many newly introduced features by changing the method of processing Blaise instruments in to a relational database. The process of generating documentation from a Blaise datamodel in MQDS V3 will be explained in this paper.

2. Background

Several papers have detailed the involvement of the Survey Research Operations (SRO), a unit within the University of Michigan's Institute for Social Research's Survey Research Center, in the development and creation of the MQDS application (Sparks and Liu 2004, Guyer and Cheung 2007). Sparks and Liu (2004) describe early development of MQDS and what was referred to as "BlaiseDoc" in the early stages of developments (or MQDS Version 1). Guyer and Cheung (2007) further explained development activities that lead to MQDS Version 2 and 2.5, rewriting the program in .NET, and described additional utilities. Ironically, in most cases, the new utilities were added to alleviate limitations experienced by users. For example, users had difficultly processing large-complex instruments, thus a utility named *XML Merge* was added allowing users to break processing into two or more steps by selecting sections or blocks within an instrument. The *XML Merge* utility then concatenated the multiple XML files into one XML file theoretically representing the output that MQDS would have been generated if it could have processed large-complex instruments.

The creation of the most recent version of MQDS, Version 3, has been fueled by several development initiatives as well as the limitations observed by users of previous versions of MQDS. The remainder of this section will focus on outlining three fundamental aspects that assist in setting the stage for the eventual creation of MQDS V3. These three areas include the fundamental limitations of previous MQDS versions, the newly released Data Documentation Initiative (DDI) version 3 metadata standard, and a collaborative project between the Interuniversity Consortium for Political and Social Research (ICPSR), a unit within the University of Michigan's Institute for Social Research, and SRO to create a database to store, share, and disseminate DDI metadata. ICPSR is the world's largest archive of digital social science data (http://www.icpsr.umich.edu/).

2.1 Limitations of Previous MQDS Versions

Although MQDS V2 offered many new, previously unavailable data documentation tools, the fundamental limitations found in previous version of MQDS were related to limited usability. The main limitations users experienced with MQDS V2 included:

- Many users with large-complex instruments found processing difficult (if not impossible) due to memory limitations. This was due to MQDS storing and processing redimensioned arrays in memory causing performance degradation over time as the memory usage continued to grow. Some found that to process large and complex instruments one would have to experiment by selecting fewer options (i.e. without goto's, without fills, etc) or breaking instruments into smaller manageable pieces (as described earlier with the XML Merge utility). However, this experimentation did not always result in positive outcomes, and for those users, MQDS remained unusable.
- Another drawback for the user was that they were required to select all the items they wanted to see in their output at runtime. If the user wanted to add additional items or remove some of them, they had to return and rerun MQDS to produce the desired set of output files.
- A small set of external clients, found it difficult to implement supporting tools around the core MQDS application. For example, tools that would allow one to repurpose the MQDS output or to polish it in the preparation of producing instrument documentation that would eventually be published in an online archive.

- The final output did not validate to the DDI V2 DTD. Document Type Definition (DTD) defines the set of elements (XML tags) and attributes (attached to a given tag) permitted in a given XML document. Wikipedia states that DTD's, "…explains precisely which elements and entities may appear where in the document and what the elements' contents and attributes are." (2009). Even though the MQDS V2 output contained 80 – 90 % of what would be considered a valid DDI V2 file; it remained noncompliant as many items needed by MQDS had not yet been added to the DDI specification and were not declared in its DTD.

2.2 The introduction of the DDI V3 Standard

As mentioned earlier, The DDI Version 2 standard lacked essential elements needed to accurately describe the Computer Assisted Interviewing (CAI) instrumentation process. The scope of previous versions of the DDI standard had traditionally focused on the archival process of data documentation and the creation of codebooks (Thomas et. al., 2008). However, the DDI Alliance's release of its third version of the DDI metadata standard in April 2008 modestly attempts to alleviate these issues. In fact, DDI V3 represents a dramatic shift towards documenting the overall survey life cycle; allowing for "a more coherent and richer description of the questions, their survey instruments, and the means of data collection" (Thomas et. al., 2008). Substantial work went into extending DDI V3 to encompass expanded elements describing questionnaires and the CAI instrumentation process (Dinkelmann, 2006). This work was the amalgamation of a work group consisting of individuals from more than 10 organizations who met via conference calls for over two years creating the proposed additions to the DDI V3 standard.

The extensions to DDI V3 come at a cost. Essentially, unless endless resources are at hand, the notion of marking up a DDI XML document manually has become extremely difficult. Therefore, for an organization to implement the DDI V3 into their processes it has become essential to utilize a mechanism for automating the creation of a DDI instance. A DDI instance could be thought of as a single, self contained XML document that validates to the DDI V3 XSD. Some extensions to the latest version include, but are not limited to, an extended set of metadata elements allowing the capturing of survey instrumentation logic (cf. the *ControlConstructScheme* element found in the Data Collection module in DDI V3). The Control Construct consists of a series of elements that are used to describe the sequence and flow of questions and supporting information within a data collection instrument (DDI 3.0 XML Schema Documentation, 2008). These elements include the following: *IfThenElse*, *RepeatUntil*, *RepeatWhile*, *Loop*, *Sequence*, *ComputationItem*, *StatementItem*, and *QuestionConstruct*. This new structure allows individuals who wish to document the survey instrument new opportunities. For example, a procedure is available in MQDS that reads the Blaise meta information file (or .bmi) to create Goto's for representing skip patterns in a given instrument. Previous DDI versions did not accommodate for documenting Goto's. However, the current DDI version allows for the storage of Goto information in the *StatementItem* element and references the *StatementItem* in the *IfThenElse* element.

DDI V3 moves from using DTD to XML schema language (XSD), often considered the successor of the DTD. Wikipedia describes XSD (2009) as follows:

> "…can be used to express a set of rules to which an XML document must conform in order to be considered 'valid' according to that schema. However, unlike most other schema languages, XSD was also designed with the intent that determination of a document's validity would produce a collection of information adhering to specific data types. Such a post-validation *infoset* can be useful in the development of XML document processing software…"

Therefore, outside of the rules for validating a DDI V3 XML instance, it also moves us a step closer to ensuring interoperability between XML instances. Wikipedia goes on to state the following about the "namespace" feature offered in the XSD standard,

> "The most obvious features offered in XSD that are not available in XML's native Document Type Definitions (DTDs) are namespace awareness, and datatypes, that is, the ability to define element and attribute content as containing values such as integers and dates rather than arbitrary text."

DDI V3 uses XML namespaces for the first time allowing users to modularize its content, offering an architecture where users can select modules within the scope of their given documentation projects (Thomas et. al., 2008). Furthermore, users can capture elements when and where they happen in the survey life cycle, aiding in the ability to pass metadata forward in the life cycle to those not directly involved in creating survey instrument and the data collection process thereby offering an unidentifiable number of ways to of reusing and repurposing of the metadata. To understand more about XML namespaces, we turn again to Wikipedia (2009):

> "XML namespaces are used for providing uniquely named elements and attributes in an XML instance. They are defined by a W3C recommendation called Namespaces in XML. An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a namespace then the ambiguity between identically named elements or attributes can be resolved."

Other significant advances in the DDI V3 standard that MQDS will take advantage of include the explicit use of the XHTML standard and documenting content once and referencing it elsewhere. Additionally, others that might be incorporated into MQDS in the future include the ability to document version changes within survey instruments, extending the DDI V3 schemas by adding user-defined or MQDS namespace (for elements and attributes outside the scope of DDI), and a mechanism for grouping and comparing related survey instruments to document similarities and differences that exists.

2.3 Data Documentation and Dissemination Project

Beginning in late 2007, members of the MQDS development team collaborated with a team from the ICPSR to begin work on the data dissemination and documentation project. The goal of this project was to create a common database design for storing survey metadata that was based on the newly released DDI V3 standard. The common database would allow for the transfer of survey data and metadata between SRO and ICPSR seamlessly. An initial SRO/ICPSR joint vision of this project is shown below in figure 1.

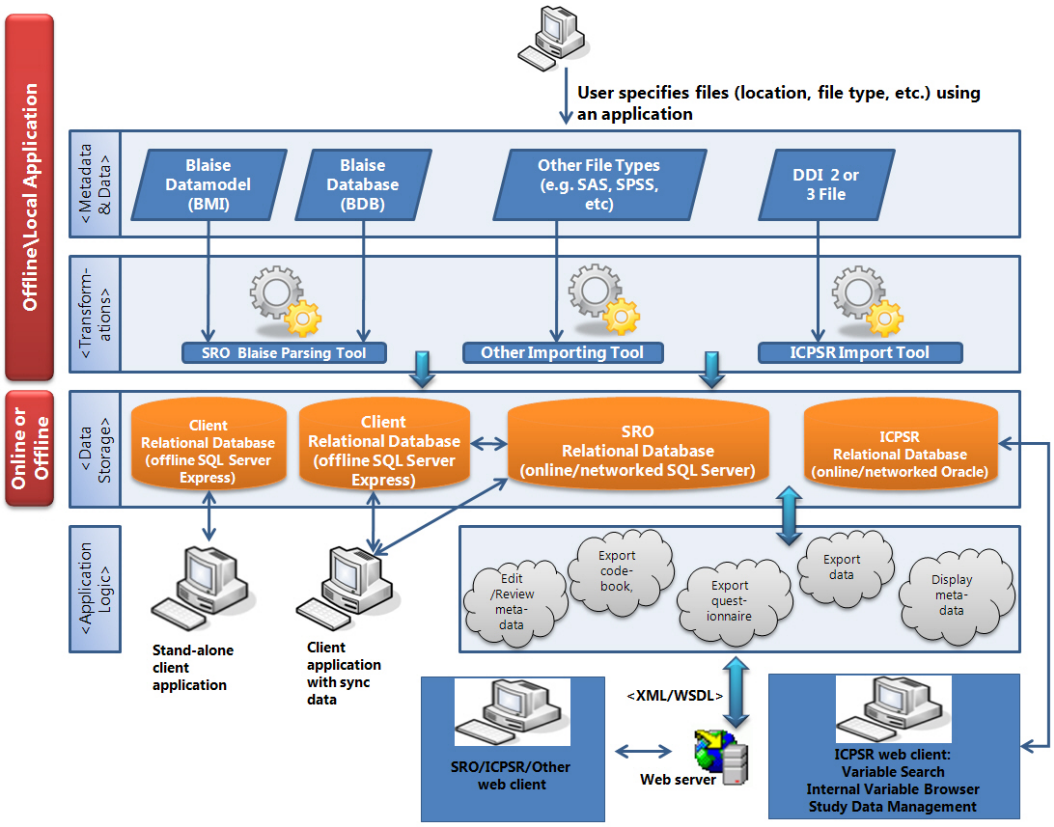

Figure 1: Initial SRO/ICPSR Joint Vision

The main outcome of this project was a database datamodel specification that has become the foundation of the new MQDS V3 database.

3. MQDS V3

The guiding principles in creating MQDS V3 were as follows:

- Address limitations identified in the previous versions of MQDS;
- Exploit new and expanded elements of the DDI V3 standard and to validate to its XSD;
- Move from the "in memory" processing model to streaming to a relational database model;
- Allow users to extend the MQDS datamodel to meet the needs of their project teams;
- Provide a tool that is closely aligned with the data collection and archival community's needs;
- The ability to process Blaise instruments of any size.

The next several sections will discuss MQDS V3 as we envision it as the development efforts for V3 had not finished as of writing this paper.

3.1 Import-Export-Transform Process Model

One of the primary differences in MQDS V3 is the use of a database design, modeled after DDI V3, to ensure alignment between MQDS and the latest DDI standard. The basis of the MQDS V3 design can be explained using the Import-Export-Transform (IET) process models. This process model divides MQDS into three different core components. These components are illustrated below in Figure 2.

Figure 2: MQDS V3 – IET Process Model

The import component takes the Blaise BMI (and associated data and metadata elements) and loads them into the MQDS V3 relational database. The export component extracts the database content to a DDI V3 compliant XML file. The transform component then renders the XML file in an output format of the user's choice (for example html representation of a questionnaire). The next three sections will describe these components.

3.1.1 The Import Process

The import process reads instrument data (summary statistics and frequencies) and metadata (*.bmi and *profile* data; profiles are explained in the next paragraph) into the MQDS V3 database. In redesigning the user interface for V3, the method of creating documentation was streamlined. Previous MQDS versions offered three different paths based on the type of output the user desired (i.e. questionnaire documentation, codebook from bdb, or codebook from SAS). The current version combines these into one operation.

The configuration file (used in previous versions) is replaced with *Profiles*, permitting the user to capture information and save it for future use, much like the previous configuration. However, the content captured and stored in the profiles is specific to items introduced in the DDI V3 standard. Two types of profiles have been added: the *Instance Profile* stores information about the current study and import session and the *User Profile* captures user information less likely to change overtime. The *User Profile* also store information about the database connection settings. The many of the items captured in the profiles have a direct mapping to DDI V3 elements. The use of profiles has been added to allow the user to overwrite some of the information that would typically be pulled directly from the *.bmi to allow for an XML instance that is more self contained. Additionally, some of the items captured in the profiles are mandatory for the user to enter before they continue, the mandatory elements are set forth from elements that are deemed as mandatory within the DDI V3 schema definitions and are not always captured in a given *.bmi file. Figure 3 provides an overview of the import process into the MQDS relational database.

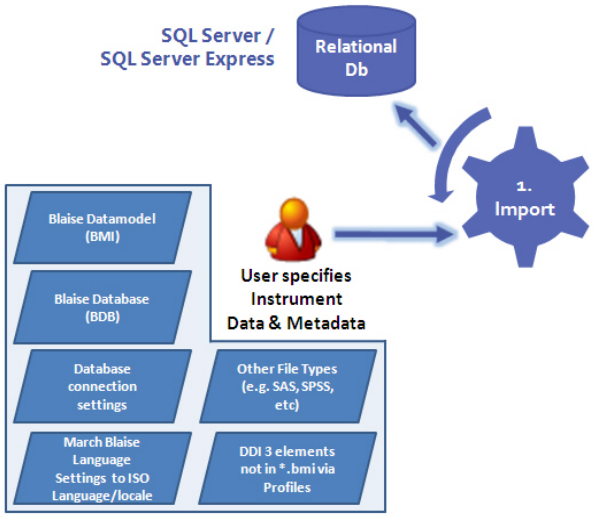

Figure 3: Import Instrument Data & Metadata

The MQDS V3 database design offers the ability to utilize database methods for queries, comparisons, and merging of data. MQDS V3 writes to either the freely available Microsoft SQL Server Express (2005 and 2008), typically residing locally on the user's computer, or to a remote SQL Server instance residing on the user's network. The remote SQL Server instance is a licensed version of Microsoft SQL Server. Nevertheless, we have focused the majority of our development on the SQL Server Express version.

Once the user has imported the instrument data and metadata into the database, the next step is to export data to DDI V3 XML instance discussed in the next section. However, this is also a point where the user could extend MQDS to meet their needs (described in section 3.3 Extending MQDS).

3.1.2 The Export Process

The export process creates a 'valid' DDI V3 XML Instance. During the export process, the user is asked to select the languages to export and to match ISO language pairs (i.e. Language-locale) that accurately reflect the language settings found in the *.bmi. The user can also select to export only the question fill names or the fill representation based on the Blaise rules and if the exported text should be formatted (based on the Blaise syntax presentation) or unformatted They can elect to export formatted or unformatted text. Figure 4 presents an overview of the Export process for creating a DDI V3 XML Instance.

Figure 4: Export DDI V3 XML Instance

3.1.3 The Transform Process

The transform process begins by taking the DDI V3 XML Instance generated in the export process and allows the user to create understandable output. Figure 5 (shown below) offers a visual representation of the transformation process from a DDI V3 XML instance process. While working with this process, the user walks through a series of steps identifying the following options: the output location of the transformed content, the transformation format (HTML, RTF, or PDF), default stylesheet settings or a modified stylesheet. If the user selected to modify the stylesheet, a new window is presented allowing the user to select the elements to include

in their output (similar to the stylesheet section window in previous versions of MQDS). A new output feature has been added to MQDS at the request of users offering the option of creating a question/variable index listed with appropriate links to its output location facilitating faster searches of the output content. This is listed at the beginning of the output, on a separate page and presents the questions by blocks or sections thus facilitating faster searching of the output content.

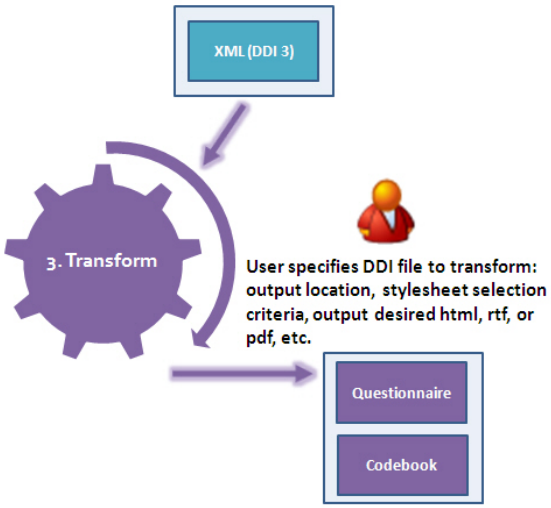

Figure 5: Transform DDI V3 XML Instance

3.2 The IET Process Model & MQDS

Figure 6 shown below illustrates how the three parts of the MQDS V3 IET process model work together.

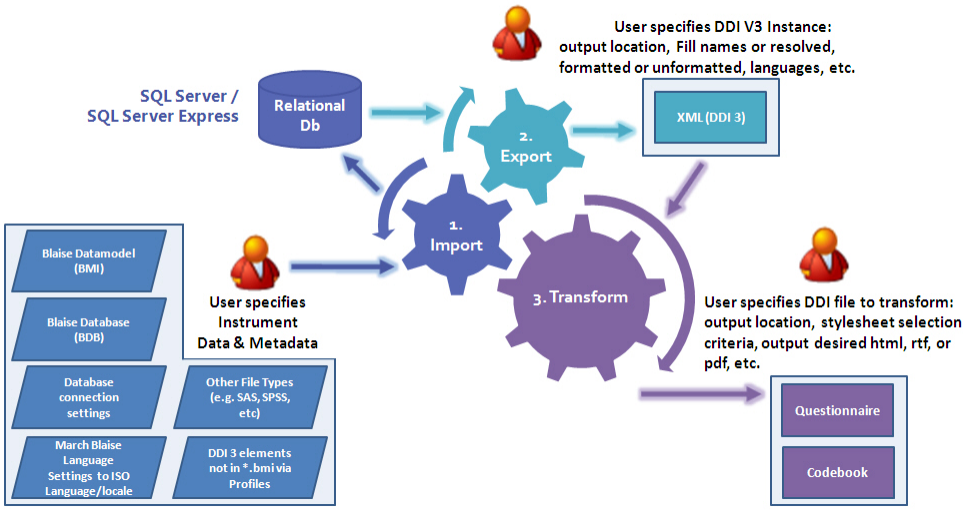

Figure 6: The MQDS IET Process Model

Despite the simple nature of the development of MQDS V3 and its use of the DDI V3 standard, development was quite complex. The initial development plan called for reusing much of the code used in Version 2 whereas full sections were rewritten from the ground-up. MQDS V3 development began before the final version of the DDI V3 standard was released. Much of the xml that MQDS needed to create had never been created before. As there were no examples to follow, new examples were manually created.

3.3 Extending MQDS

While designing MQDS V3 and attempting to address the limitations experienced in previous versions of MQDS, the development team attempted to generate use cases for the possible ways users would want to extend the application. The image below in figure 7 illustrates only a fraction of the notion of extending MQDS, as its new database structure offers the user with many other opportunities that will not be discussed here. The arrows pointing up in the image are intended to demonstrate possible points where users could extend MQDS by creating supplementary tools.

Figure 7: Extending MQDS

For example, the user could extend the import component by creating a database-editing tool allows direct access to the MQDS output allowing for direct modification of the metadata. Possible areas where the user might want to modify the output include modifying field descriptions (or SAS labels) for a more readable format for data out

purposes or adding human-readable universe descriptions (as MQDS currently exports machine-readable universes descriptions).

4. Known limitations of MQDS V3

Despite efforts to combat the known limitations in previous versions MQDS, some not so obvious limitations still exist within the output generated from MQDS V3. These limitations can be classified into two separate categories, one for documenting instruments programmed in Blaise and those imposed by DDI V3. MQDS V3's limitations in documenting instruments programmed in Blaise center on the expanding uses of the Blaise language setting. MQDS V3 currently makes the assumption that the use of languages within Blaise applications is used for the purpose of expressing different languages (i.e. English, Dutch, Spanish, etc). The *Blaise 4.8 Online Assistant* defines "Languages (setting)" as, "…the form languages available in a multilingual data model." (2009). However, the use of the language feature in Blaise currently extends past the semantic interpretation of language. For example, a user can use the language feature to define different modes in a mixed-mode survey, add help and metadata content, multimedia content, or different instrument versions within the same study. Unfortunately, unless the Blaise software were to force language identifiers, such as HLP for help and MDL for metadata language, it becomes difficult to determine that these are not real languages. The only way to tell currently is by looking to see if the given language identified in the data model is set as a spoken and unspoken language. This means that the methods for assessing how the language function is being used within a Blaise application are limited. Therefore, they are processed as though they are actual languages until a more elegant solution is devised. The interesting aspect here is that MQDS is able to create DDI V3 instance representations that are syntactically valid yet they remain semantically incorrect. It is important to note the paradoxical nature of this limitation, that is, even though it poses a problem in creating systems that support instrument documentation, we could develop equal counterpoints to explain how this could be considered one of the systems most valuable features. Nevertheless, we must attempt to devise a solution that suits the majority of users.

Most of the limitations brought forth as a result of integrating the DDI V3 standard are still unknown, as the MQDS's implementation of its most recent version has only scratched the surface. There are several known issues related to Blaise. For example, Blaise does not currently offer an explicate way of documenting *Alien Procedures* (calls to external routines) and *Alien Routers* (calls to external programs). Additionally, DDI currently does not offer ways of documenting paradata associated with the instrument. However, the DDI standard is an on-going process with mechanisms in place for submitting proposals for additions to the current standards. Additionally, the ability to extend the schema by adding additional namespaces exists. For example, we could add a MQDS or Blaise namespace to accomplish documenting these areas.

5. Conclusion

Despite the lengthy development time, many of the limitations experienced with the previous version of MQDS are now resolved leading to improved usability. Although limitations will be identified with MQDS V3, the many gains of the latest version should outweigh the limitations.

6. Acknowledgements

The authors would like to thank the following individuals from SRO for their contributions towards the development and creation of MQDS V3: Sheila Deskins, Nathaniel Doran, David Padot, Ruth Philippou, Peter Sparks, and Isabella Yeh. Thank you to the ICPSR team for their collaborative efforts in creating the shared datamodel (a.k.a. the "data dissemination and documentation project").

7. References

Blaise 4.8 Online Assistant (Version 4.8.1.1339). (2009).

Data Documentation Initiative (DDI). http://www.ddialliance.org/, Retrieved Apr. 28, 2009

DDI 3.0 XML Schema Documentation (2008-04-28). (2008) Windows Help file. http://www.icpsr.umich.edu/DDI/ddi3/DDI_3_0_Documentation_WindowsHelp.zip, Retrieved Apr. 28, 2009

Dinkelmann, K. (2006). DDI Version 3 and Instrument Documentation. *The proceedings of the 32nd International Association for Social Science Information Services & Technology (IASSIST).* Ann Arbor, Michigan.

Document Type Definition. (2009, May 7). In *Wikipedia, the free encyclopedia*. Retrieved April 28, 2009, from http://en.wikipedia.org/wiki/Document_Type_Definition

Guyer, H. & Cheung, G. (2007). Michigan Questionnaire Documentation System (MQDS): A User's Perspective. *The proceedings of the 11th International Blaise Users Conference (IBUC).* Annapolis, Maryland.

Microsoft SQL Server 2005 Express Edition. http://www.microsoft.com/Sqlserver/2005/en/us/express.aspx.

Microsoft SQL Server 2008 Express. http://www.microsoft.com/express/sql/

Microsoft SQL Server 2008. http://www.microsoft.com/sqlserver/2008/en/us/default.aspx

Sparks, P. & Liu, Y. (2004). Blaise Documentation System. *The proceedings of the 9th International Blaise Users Conference (IBUC).* Gatineau, Québec.

Thomas, W., Gregory, A., Gager, J., Kuo, I., Wackerow, A., and Nelson, C. (2008) Data Documentation Initiative (DDI) Technical Specification Part I: Overview (Version 3.0) (document version 7). Copyright DDI Alliance 2008.

XML Namespace. (2009). In *Wikipedia, the free encyclopedia*. Retrieved April 28, 2009, from http://en.wikipedia.org/wiki/XML_Namespace

XML Schema (W3C). (2009). In *Wikipedia, the free encyclopedia*. Retrieved April 28, 2009, from http://en.wikipedia.org/wiki/XML_Schema_Definition