18th INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2016
SÃO PAULO, BRAZIL, AUGUST 29 - 30, 2016

DSMDE: A Data Exchange Format for Design Structure Models

Lazima Ansari[1], Shahadat Hossain[2], Ahamad Imtiaz Khan[3]
Department of Mathematics and Computer Science, University of Lethbridge

Abstract: We propose **D**esign **S**tructure **M**odel **D**ata **E**xchange file format as a common file format to promote reliable and efficient exchange of Design Structure Model (DSM) data. The DSMDE is an extension of the Matrix Market (MM) file format, a widely used file format for the exchange of dense and sparse matrix data arising in numerous scientific applications. At present there does not exist a common standard for sharing DSM data. We believe that a standardized exchange format will greatly facilitate research and development of DSM modelling techniques by making data widely available than currently possible. The DSMDE is expected to be a standard way to share DSM data among researchers, practitioners, and on different programming environments.

Keywords: Sparse Matrix, Complex Network, EBNF Grammar, Portable Exchange, DSM, MDM

1 Introduction

The DSM is a modelling tool to capture, display, and analyze interactions between constituent elements of a complex system. As elucidated in (Eppinger and Browning, 2012), a DSM *M* is an $n \times n$ matrix where element i of the system is associated with column and row i such that the entry $M(i,j), i \neq j, i,j \in \{1,2,\dots n\}$ represents the interaction between the elements i and j. Depending on the underlying system a pair of elements may interact in multiple ways. In general, interactions between elements i and j can be represented by a small set of attributes some of which may be symbolic. However, in almost all cases attributes can be easily mapped to numerical values. Hence, for all practical purposes, we can view a DSM *M* as a $n \times n$ matrix where each interaction is encoded by a vector of dimension $d \geq 1$ (i.e. in $\Re^d$), where d is the number of attributes associated with an interaction. For a comprehensive review of the DSM methods we refer to (Browning, 2016).

2 Rationale and Design Philosophy

Many real world examples of DSM matrices remain scattered in the literature and most of these examples are not in an easily retrievable digital form. Recently, the book by Eppinger and Browning (Eppinger and Browning, 2012) has compiled 44 DSM

examples from diverse areas. As DSM techniques continue to find its applications in new and emerging areas (e.g., complex networks), it is conceivable that exchange of data will play a crucial role in future development of techniques and algorithms for DSM data analytics challenges. The purpose of this work is to suggest a common framework for exchange of model data. Fundamental to our proposal is the exploitation of duality between sparse matrices and graphs (as complex networks). For recent overviews on graph and matrix file formats we refer to (Roughan and Tuke, 2015) and (Bodlaj, 2014). The following basic features (in no particular order) are considered desirable in a file format.

1. **Portability**. The data in the file should be easily transferable between hardware and operating systems efficiently.
2. **Simplicity**. By simplicity we mean ease of reading and writing data. We require that the file can be viewed with general purpose text editor programs such as vi(m), emacs, TextEdit, NotePad, etc. The stored data should be structured such that it facilitates easy input and parsing.
3. **Extensibility**. The format should be flexible enough to allow adaptation and extension of the base format without requiring too much effort. For example, it is reasonable to envision applications in which interactions involve more than two elements. One way to represent such information is by extending the 2-dimensional matrix framework to higher-dimensional tensors.

The simplicity and portability requirements allowed us to rule out binary (non text) files from consideration. Our design emphasizes simplicity of representation such that the proposed format is to be independent of specific software toolkit for display and manipulation of data. Consequently, we only consider ASCII (and UNICODE where applicable) text files.

The Harwell-Boeing (HB) sparse matrix collection (Duff et al., 1989) is one of the earliest efforts to compile and maintain a standard set of sparse matrix test problems arising in a wide variety of scientific and engineering disciplines. Unfortunately, HB format is not easily extensible. Further, because of HB format's heavy reliance on FORTRAN specific input/output constructs, it is somewhat complex to comprehend the data. Graph and Matrix Format (GAMFF) (Zien et al., 1995) is a closely related (to HB) format which is more flexible in that it permits additional information specific to graphs and hypergraphs. One difficulty with using compressed column (or compressed row) is the potential for overflow of indices.

The Matrix Market (MM) exchange format (Boisvert et al., 1996) is a simple but extensible file format for storing and exchanging sparse and dense matrix data stored in an ASCII text file. The MM format enables extensibility by allowing format specialization in the form of qualifier attributes and structured documentation. The information about the matrix is organized in three syntactic sections: Header, Comment, and Data, in that order. The header section encodes metadata such as numerical field, structure, data format etc. The comment section consists of free-format lines of text and can be used to provide specific information about the data. The last section, the data section, contains the numerical values. Recently, Yzelman and Bisseling (Yzelman and Bisseling, 2010) proposed Extended-Matrix-Market-Format (EMM) suitable for storing sparse matrices and vectors. The new features of EMM enable sparse matrices and vectors to be used in a distributed computing environment for performing sparse matrix

operations. Our proposal, the DSMDE, exploits MM format's extensibility while maintaining its generality and is independent of specific computing environments.
The remainder of the paper is structured in the following way. Section 3 contains elaborate description of the proposed DSMDE exchange format. The section concludes with an Extended Backus-Naur Form (EBNF) specification of the grammar for DSMDE format. We note that the original MM exchange format specification does not include an EBNF description. In Section 4 we provide an example DSM taken from (Eppinger and Browning, 2012) and show its representation in DSMDE exchange format. Finally, the paper is concluded in Section 5 with a discussion on directions for future development.

3 The Extended File Format for DSM and MDM Data

It is nearly impossible to come up with "the best" format since some of the required properties may be conflicting. In this section we provide the general specification of our proposed format DSMDE as an extension of MM exchange format. As in MM format the contents of a DSMDE file are organized into three main sections: Header, Comments, and Data, in that order.

1. **Header**. Our choice of MM exchange format to form the basis for DSMDE has largely been influenced by MM format's simplicity and extensibility. The header of MM format has the following structure:

   Banner ObjectType FormatType Qualifiers

   Banner is the literal string %%MatrixMarket of 15 ASCII characters. The DSMDE exchange format views design structure models as matrices (and in general, higher-order tensors). Extensibility of the base MM format can be realized in a number of ways. It is not necessary to change the banner string of the base MM format since the additional features needed to represent DSM, MDM, and DMM objects can be incorporated in the remaining fields of the header section. The header is extended to include the objects DSM, MDM, and DMM under *ObjectType* field. That is, in addition to Matrix we allow DSM, MDM, and DMM as type of mathematical objects that can be represented. The existing data layout schemes Coordinate and Array of MM are adequate for the new object types. A matrix can be full (in which case each matrix entry is explicitly stored with Array specification) or sparse (only the nonzero entries are stored with Coordinate specification). The third component of MM header enables us to specify a list of qualifiers. DSMDE takes advantage of this field to provide properties that are specific to DSM, MDM, and DMM data. The two MM qualifiers *Numeric-Field* and *Structure* are retained. We introduce the following additional qualifiers.

   a. *Orientation.* This qualifier (*Orientn*) encodes information about the DSM orientation convention for off-diagonal marks. The two variants are: Input in Row and output in column (IR) and Input in Column and output in row (IC). With IR, in a process DSM, "feedback" is indicated by a mark above the main diagonal (FAD) while with IC a mark below the main diagonal (FBD) indicates "feedback". Accordingly, we use the codes IR and IC to represent orientation.

   b. *Interaction Attributes*. While for "simpler" system models, a scalar value would suffice to represent system interaction, many real-life models have a more elaborate interaction structure. Eppinger and Pimler (See Example 3.1 in (Eppinger and Browning, 2012)) studied the climate control systems of cars and trucks produced by Ford Motor Company. They have identified four types of interaction among the system components: spatial, energy, information, and materials. Interactions may also differ with respect to the source they originate from. The product architecture example "Building Schools for the Future" (Example 3.8 in (Eppinger and Browning, 2012)), uses three interaction

sources: explicit, inferred, and perceived. In the DSM model of software library CSparse (Hossain et al., 2015), dependencies (between code files) can be due to function calls or object references, for example. There are DSM models that use colors to depict specific interactions. In the Helicopter Change Propagation DSM model (Example 3.6 in (Eppinger and Browning, 2012)), red, amber, and green shadings represent "significant-", "lower-", and "small-risk" of change propagation, respectively. For simplicity, DSMDE treats interaction varieties as attributes. The number of interaction attributes is recorded in the header with the qualifier *NIattribute*. A mapping between the attributes and the integers $1,2,\ldots,n_a$, where n_a denotes the number of interaction attributes, can be provided in the comments section. The same qualifier can also be used to represent a composite DSM (composition of different instances of the same model). In the product architecture model "Johnson & Johnson Clinical Chemistry Analyzer", the Expert DSM (See Example 3.7, Figure 3.7.2 in (Eppinger and Browning, 2012)) model displays interactions recorded at two different dates.

As noted earlier, an attribute may assume a numerical value (integer, real, complex) or a symbolic name (e.g., color red, color green, etc.). For symbolic names, the DSMDE requires a mapping between the names and the integers $1,2,\ldots,n_s$ to be specified; n_s denotes the number of symbolic names that can be attribute values. The mapping can be documented under the Comments section of the DSMDE file. For a pattern DSM (*Structure* = pattern) $n_a = 0$ since the type of interaction is binary.

The qualifier *NumericField* for a DSM or a DMM object has *NIattribute* (n_a) components. This is due to the fact that for each attribute its *NumericField* has to be specified. An MDM is treated as a collection of DSMs and DMMs such that the header field for an MDM has a simpler structure.

c. *Domain*. To incorporate MDM models in DSMDE, we record the number of domains $n_d \geq 1$. For DSMs and DMMs we have $n_d = 1$, for MDMs, $n_d > 1$. An MDM model can be viewed as a block triangular matrix as shown below.

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n_d} \\ 0 & A_{22} & \cdots & A_{2n_d} \\ \vdots & \ddots & \cdots & \vdots \\ 0 & 0 & \cdots & A_{n_dn_d} \end{bmatrix}$$

The diagonal blocks $A_{ii}, i = 1,\ldots,n_d$ correspond to the DSMs. The off diagonal blocks $A_{ij},\ i < j \text{ and } i,j = 1,\ldots,n_d$ represent interactions between elements in domains $i \neq j$ and are known as domain mapping matrix (DMM). In an n_d-domain MDM, there are n_d DSMs and $\sum_{i=1}^{n_d-1} i = \frac{n_d(n_d-1)}{2} \equiv n_{dmm}$ DMMs. The header section is modified accordingly. There are $1 + n_d + \frac{n_d(n_d-1)}{2}$ header lines where the first line consists of the banner string, MDM as object type and the number of domains; each of the next n_d lines must have a value for each of *FormatType*, *Numerical Field*, *Structure*, *Orientn*, *NIattribute* for the DSMs contained in the MDM. Each of the $\frac{n_d(n_d-1)}{2}$ lines must have a value for each of *FormatType*, *NumericField*, *Structure*, *NIattribute*, for the DMMs. For a DMM object, orientation information is not needed. The relevant data for DMMs are stored according to "block row-major" order. The block row- major order is to consider the off-diagonal blocks in the order $A_{12},\ldots,A_{1n_d}, A_{23},\ldots,A_{2n_d},\ldots,A_{n_{d-1}n_d}$.

The overall DSMDE header section is depicted in Table 1. Note that when *ObjectType* is DSM or DMM the header section consists of only one line. As in the MM format it is to be emphasized that not all header filed combinations are meaningful. In general, context-

free grammars are not powerful enough to express context-sensitive requirements. Consequently, header field combinations are validated informally.

2. **Comments**. As we have already observed, the header section of the DSMDE format provides a high-level specification of the DSM, MDM, and DMM data contained in the file. Information such as name of design elements, source and type of interactions etc. are essential components of the underlying models. The comments section provides a convenient

Table 1. Header section components and their values in DSMDE

Fields	*Banner*	*Object Type*	Qualifier (*Ndomain*)	Qualifier (*Format Type*)	*Qualifier (Structure)*	Qualifier (*NIattribute*)	*Qualifier (Numeric Field)*	Qualifier (*Orientn*)
Values	%%MarixMarket	Matrix, DSM, MDM, DMM	n_d	Coordinate, Array	General, Symmetric, Skew-symmetric, Hermitian	n_{a_1} n_{a_2} n_{a_3} . . . $n_{a_{n_d}}$ $n_{a_{n_d+1}}$ . . . $n_{a_{n_d+n_{md}n}}$	Integer, Real, Complex, Pattern . . .	IC, IR IC, IR IC, IR IC, IR IC, IR

way to record such information about the data. To enable automatic (machine) parsing of the information, the DSMDE comments section enforces specific syntactic rules on the text that appear here. The comments section consists of two parts: a required section and an optional section. The optional section is similar to the comments in MM format - no specific syntactic structure is enforced. The required section consists of three ordered subsections as described below.

   a. **Domain**. For a DSM $(n_d = 1)$, this is a one-line description of the model. Each such string can be used to provide a brief description of the corresponding DSM model. The subsection is enclosed in the pair of literal strings beginDomain endDomain.
   b. **Model Element**. For a DSM $(n_d = 1)$ model, the element names are provided in the file as a list of n character strings, one per line. The subsection is enclosed in the pair of literal strings beginModElement endModElement.
   c. **Interaction Attributes**. For a DSM $(n_d = 1)$, this is a list of n_a character strings describing the interaction attributes. A mapping between the n_a names (of attributes) and the set $\{1,2,\dots,n_a\}$ must be provided. The subsection is enclosed in the pair of literal strings beginAttribute endAttribute. For a MDM $(n_d > 1)$, the above documentation is repeated for each DSM (the diagonal blocks of the block upper triangular representation of MDM). This is followed by the documentation for each DMM in block row major order (the off-diagonal blocks of the block upper triangular representation of MDM). An optional subsection of the comments section can be used to provide additional information about the model.

3. **Data**. As in the base MM exchange format, the data section records the numerical data. Intuitively, each matrix/DSM/MDM/DMM data point (i.e., interaction) represents an instance

of a relation defined on interaction attributes. An element (or a data point) of a matrix is uniquely identified by its location. For a two-dimensional matrix object the location information is provided as an ordered pair (i, j), where i denotes the row index and j denotes the column index. Each element of the matrix possesses certain attributes depending on the type of the object. Consider the product architecture DSM example 3.8 "Building Schools for the Future" (Eppinger and Browning, 2012). There are three sources of interactions: Explicit (1), Inferred (2), Perceived (3), and three types of interactions: Structural (1), Spatial (2), and Service (3). For the purpose of data exchange, we just need to identify each interaction attribute with a unique integer from the set $\{1,2,3\}$ as discussed in the preceding section. With *FormatType* = Coordinate and *NumericField* an ordered pair (Integer, Integer) where the first component is associated with the attribute "interaction source" and the second associated with the attribute "interaction type", a dependency mark can now be specified with an ordered 4-tuple (i, j, s_{ij}, t_{ij}) where $i, j \in \{1, \dots, n\}$, indicates the row index and the column index, respectively; $s_{ij} \in \{1,2,3\}$ indicates interaction source, and $t_{ij} \in \{1,2,3\}$ indicates interaction type associated with the mark at location (*i,j*). This information is recorded in DSMDE exchange format as:

$i \quad j \quad s_{ij} \quad t_{ij}$

in a line in the file. Formally, a tuple of the form (i, j, s_{ij}, t_{ij}) is an element of the set produced by the Cartesian product $\Im \times \Im \times \Im \times \Im$ where,

$$\Im = \Im = \{1,2, \dots, n\}, \ \Im = \Im = \{1,2,3\}$$

for this particular example. Note also that the location of an interaction in a DSM or DMM object (in Coordinate format) is a k-tuple; $k = 2$ implies a matrix and $k > 2$ implies a higher-dimensional tensor. For a MDM object, ordering of the data is as below.

a. **DSM** data. $A_{11}, A_{12}, \dots, A_{n_d n_d}$

b. **DMM** data. $A_{11}, \dots, A_{1n_d}, A_{21}, \dots, A_{2n_d}, \dots, A_{n_{d-1}n_d}$

3.1 DSMDE Grammar

In this section we provide the syntax specification of DSMDE exchange format using EBNF (Extended Backus-Naur Form) notation. Unfortunately, notation to describe grammar rules in BNF/EBNF has not been standardized (Zaytsev, 2012). Hence, we describe the meaning of symbols used and the syntactic conventions adopted.

1. **Start nonterminal**. The start nonterminal of the EBNF grammar for DSMDE format is denoted by the string DSMDEFORMAT.
2. **Reserved Words**. Text strings written in `teletype` font have special meaning in DSDME format. They appear as string literals in DSDME formatted files. They are: `%%MatrixMarket,Matrix,DSM,MDM,DMM,Coordinate,Array,Integer,Real,Complex,Pattern,General,Symmetric,SkewSymmetric,Hermitian,IC,IR,%beginDomain,%endDomain,%beginModElement,%endModElement,%beginAttribute,%endAttribute`,
3. **Nonterminal**. Words with first character in uppercase denote nonterminal symbols.
4. **Terminal**. Words with first character in lowercase denote terminal symbols. A terminal symbol or token describes a lexical pattern of strings defined over the set of ASCII printable characters (ASCII code 33,…,126). For example, the token named "Integer" matches strings defined over ASCII characters {0,1,2,3,4,5,6,7,8,9}. Thus, the string 311 is an "Integer" while the string 102a is not. The literal string `%%MatrixMarket` matches the token named "banner" and this is the only such string. In the EBNF syntax description, the name "charSymbol" denotes a printable ASCII symbol. Additionally, we use the following ASCII symbols.

a. newline (ASCII LF (in some computing environment); value 10) to start a new line in the file
b. space (ASCII value 32), to separate adjacent tokens appearing in the file
c. tab (ASCII HT; value 9), to separate adjacent tokens or to format text strings.

We remark that adjacent tokens in a DSMDE file must be separated by at least one separator symbol.

5. EBNF Meta Symbols

a. **Repetition**. S* implies zero or more occurrences of grammar symbol S; S+ implies one or more occurrences of grammar symbol S; [S] implies zero or one occurrence of grammar symbol S.
b. **Option**. Options are indicated as R|S, meaning either R or S but not both.
c. **Scope**. Parentheses are used to group together grammar symbols to indicate scope.

Table 2. EBNF Grammar for DSMDE Exchange Format

DSMDEFORMAT	::=	Header+ Comments Data
Header	::=	banner [objectType] [Qualifiers]
banner	::=	`%%MatrixMarket`
objectType	::=	`Matrix | DSM | MDM | DMM`
Qualifiers	::=	[NDomain] QualList
QualList	::=	(formatType [structure] [NIAttribute] [numericType] [orientn] newline)+
formatType	::=	`Coordinate | Array`
numericType	::=	`(Integer | Real | Complex | Pattern)+`
structure	::=	`General | Symmetric | Skew-Symmetric | Hermitian`
NDomain	::=	Integer
NIAttribute	::=	Integer
orientn	::=	`IC | IR`
Comments	::=	TextLine* Documentation+ TextLine*
TextLine	::=	%charSymbol* newline
Documentation	::=	[DomainNames][ModElementNames] [InteractAttributeNames]
DomainNames	::=	beginD newline TextLine+ endD newline
ModElementNames	::=	beginME newline TextLine+ endME newline
InteractAttributeNames	::=	beginIA newline TextLine+ endIA newline
beginD	::=	`%beginDomain`
endD	::=	`%endDomain`
beginME	::=	`%beginModElement`
endME	::=	`%endModElement`
beginIA	::=	`%beginAttribute`
endIA	::=	`%endAttribute`
Data	::=	CoordData | ArrayData
CoordData	::=	NRows NCols Nnz newline CoordDataLine+
ArrayData	::=	NRows NCols newline ArrayDataLine+
CoordDataLine	::=	RowIndex ColIndex Values newline
NRows	::=	Integer
NCols	::=	Integer
Nnz	::=	Integer
RowIndex	::=	Integer
ColIndex	::=	Integer

ArrayDataLine	::=	Values newline
Values	::=	IAttribute*
IAttribute	::=	Integer | Real | Complex
Integer	::=	[sign] digit+
Real	::=	[sign] digit* . digit* [Mantissa]
Sign	::=	+ | -
Mantissa	::=	E [sign] digit+
Complex	::=	Real Real
digit	::=	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Separator	::=	space | tab

4 Example

In this section we illustrate the DSMDE exchange format with a real-life design structure model.

```
%%MatrixMarket DSM 1 Array General 4 Real Real Real Integer IC
%
% Product Architecture DSM Model of AW101 Change Propagation
% Example Fig. 3.6.3;[Steven D Eppinger and Tyson R Browning;
% Design structure matrix methods and applications; MIT press, 2012].
% Number of domains:  1
% Number of attributes : 4
% Input convention: : Input in column (IC)
%
%beginDomain
% Product Architecture DSM
%endDomain
%beginModElement
% 1 = air conditioning, 2 = auxillary electronics, 3 = avionics,
% 4 = bare fuselage, 5 = cabling and piping , 6 = engines,
% 7 = engine auxillaries, 8 = equipment and furnishings,
% 9 = fire protection, 10 = flight control systems, 11 = fuel,
% 12 = fuselage additional items, 13 = hydraulics,
% 14 = ice and rain protection, 15 = main rotor blades, 16 = main rotor
head,
% 17 = tail rotor, 18 = transmission, 19 = weapons and defensive systems
%endModElement
%
%beginAttribute
% 1 = Impact(height), Real; 2 = Likelihood(width), Real; 3 = Risk(height*
% width), Real; 4 = Change Propagation(shade), Integer (Red = 3, Amber =
% 2, Green = 1)
%endAttribute
19 19
0 0 0 0
0.4 0.8 0.32 2
0.7 0.8 0.56 3
```

Figure 1. Product Architecture DSM Model AW101 (Eppinger and Browning, 2012) in DSMDE Format.

4.1 Product Architecture DSM Example (Fig. 3.6.3 of (Eppinger and Browning, 2012))

Figure 1 displays a product architecture DSM model in DSMDE exchange format. The header line

```
%%MatrixMarket DSM 1 Array General 4 Real Real Real Integer IC
```

indicates that the file contains a DSM object (*ObjectType* = `DSM`, *NDomain* $n_d = 1$) stored in array format (*FormatType* = `Array`), contains no special structure (*Structure* = `general`), uses 4-attribute interactions (*NIattribute* $n_a = 4$) of type real, real, real, integer, and that it uses input-in-column (*Orientn* = `IC`) convention. Recall that array format stores all n^2 entries of the DSM in column-major order. The next 8 lines after the header line provide information on the DSM model. This part is optional. The three-part structured documentation section provides the name of the domain (DSM), enumerates the model elements and their integer mapping, followed by the attribute names and their integer mapping information. These three subsections are enclosed in their respective "begin" and "end" format tags. For brevity, the mapping of model elements and attributes are not shown in the required syntactical format (they must occur one per line). The first line of the data section,

```
19      19
```

indicates that the DSM model consist of 19 rows and 19 columns. The next $19 \times 19 = 261$ lines contain interaction data, one interaction per line. The second line of the data section,

```
0       0       0       0
```

corresponds to the interaction object located at row 1 and column 1. The four zeroes indicate that the diagonal element does not have any useful information. The next line (line 3),

```
0.4     0.8     0.32    2
```

corresponds to DSM cell at row 2, column 1. The four numerical values represent Impact (height) $= 0.4$, Likelihood (width) $= 0.8$, Risk (height*width) $= 0.4 \times 0.8 = 0.32$, and Change Propagation (shade) encoding value Amber $= 2$. Figure 1 displays only the first 2 interaction values of the DSM.

5 Concluding Remarks

A DSM is much more than an adjacency matrix representation of a complex network. As has been articulated in (Browning, 2009) the characteristics of a complex system may not be fully comprehensible from a single viewpoint. A DSM provides an important "view" of such a system. The design and analysis of complex engineered systems (Eppinger and Browning, 2012) can be greatly aided by techniques and tools that can capture, organize, and represent nontrivial interactions among systems' elements. The new exchange format is an extension of the widely used Matrix Market data exchange format. As such, it retains the simplicity and flexibility of the base MM format and now facilitates the exchange of DSM, MDM, and DMM model data. The structured comments section can be used to record important system information about the models stored in the file. For the purpose of data exchange an interaction is viewed as a k-tuple (conceptually) consisting of two parts: the address or location and the value. Although the DSMDE exchange format does not require any special software to read or write model data, in practice, some software support is typically expected to perform input and output. We have implemented a syntax-directed interpreter in JAVA programming

language for input and output. We are currently developing a software tool that will have support for task/activity sequencing and data visualization in a user-friendly manner.

Acknowledgement

This research was partially supported by the Natural Sciences and Engineering Research Council of Canada under a Discovery Grant Individual.

References

Bodlaj, J., 2014. Network Data File Formats, in: Encyclopedia of Social Network Analysis and Mining. Springer New York, NY, pp. 1076–1091.

Boisvert, R.F., Pozo, R., Remington, K.A., 1996. The Matrix Market Exchange Format: Initial Design. NISTIR, 1996. math.nist.gov/MatrixMarket/reports/MMformat.ps.gz (accessed May05, 2016)

Browning, Tyson R., 2009. The many views of a process: Toward a process architecture framework for product development processes. Systems Engineering, 12(1), pp. 69-90.

Browning Tyson R., 2016. Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities. IEEE Transaction on Engineering Management, 63(1), pp. 27-52.

Duff, I.S., Grimes, R.G., Lewis, J.G., 1989. Sparse matrix test problems. ACM Trans. Math. Softw. 15, 1–14.

Eppinger Steven D., Browning Tyson R., 2012. Design Structure Matrix Methods and Applications, MIT Press.

Hossain, S., Khan, S.F., Quashem, R., 2015. On Ranking Components in Scientific Software. in the Proceedings of 17th International DSM Conference, pp. 245-254.

Roughan, M., Tuke, J., 2015. The Hitchhikers Guide to Sharing Graph Data. 3rd International Conference on Future Internet of Things and Cloud (FiCloud), pp. 435-442.

Yzelman, A.N., Bisseling, R.H., 2010. Extended matrix-market file format – an extension to the standard schme http://www.staff.science.uu.nl/~bisse101/Mondriaan/Docs/(accessed May05, 2016)

Zaytsev, V., 2012. BNF was here: what have we done about the unnecessary diversity of notation for syntactic definitions. Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12. ACM Press, New York, USA, pp. 1910-1915.

Zien, J.Y., Simon, H.D., Woo, A.C., 1995. GAMFF - A Graph and Matrix File Format. http://www.nas.nasa.gov/Software/GAMFF/ (accessed May 05, 2016)

Contact: Shahadat Hossain, University of Lethbridge, Department of Mathematics and Computer Science, 4401 University Dr W, Lethbridge, AB T1K 3M4, Alberta, Canada, phone: (403) 329 2475, Fax: (403) 329 2519, Email: shahadat.hossain@uleth.ca.

About the Authors:

Lazima Ansari received BSc in CSE from MIST, Bangladesh in 2010. Currently she is a MSc student in computer science at the University of Lethbridge, Alberta, Canada.

Shahadat Hossain is Professor of Computer Science at the University of Lethbridge, Alberta, Canada. He received the degree of Doctor Scientiarum in Computer Science from the University of Bergen, Norway in 1998.

Ahamad Imtiaz Khan received BSc in CSE from MIST in 2010 and MPhil in Biomedical Physics and Technology from the University of Dhaka, Bangladesh. Currently he is a MSc student in the Department of Mathematics and Computer Science of the University of Lethbridge, Alberta, Canada.