DSMDE: A Data Exchange Format for Design Structure Models

Shahadat Hossain

Abstract We propose **D**esign **S**tructure **M**odel **D**ata **E**change 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 computing environments.

1 Introduction

The DSM is a modelling tool to capture, display, and analyze interactions between constituent elements of a complex system. As elucidated in ([1], p. 6), a DSM M is a $n \times n$ matrix where element i of the system is associated with column and row i such that the entry $M(i,j), i \neq j, i,j \in \{1,2,\ldots,n\}$ 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 the 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.

Shahadat Hossain
University of Lethbridge, Canada e-mail: shahadat.hossain@uleth.ca

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 easily retrievable digital form. Recently, the book by Eppinger and Browning [1] 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 innovative techniques and algorithms for DSM data analytics challenges. One of the main goals of this work is to create a publicly accessible collection of interesting DSM examples. We believe that the properties as outlined in Figure 1 could form the basis for a common file format to facilitate the exchange of DSM data. There exists a myriad of different file formats of varying

1. Supports a common and unambiguous format for the storage of DSM that is both human and machine readable.
2. Provides adequate documentation on the included test problems to be useful to researchers and practitioners.
3. Allows extension of the base format without unreasonable efforts to meet the requirements for new attributes or properties.

Fig. 1 Desirable Properties in a DSM Exchange Format

degree of sophistication (in terms of properties such as data compression, error-correction, special attributes for subsets of elements etc.) for the exchange of graph and matrix data [3, 4]. We have decided to not invent a yet another "new" format. Instead, our proposal is to identify a "suitable" existing exchange format and extend it to incorporate DSM specific information. More specifically, the following basic features (in no particular order) are considered desirable in a file format.

1. **Protability.** The data in the file should be easily transferable between hardware and operating systems. As an indicator, we require that the file can be viewed with general purpose text editor programs such as vi(m) , emacs, TextEdit, NotePad, etc.
2. **Simplicity.** By simplicity we mean ease of reading and writing data. The data in the file is needed to be in a human readable form and can be input and edited without the aid of special-purpose software tools. In other words, the syntactic structure of the format is simple enough to not impede with human readability.
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. Some contemporary graph and matrix file formats use XML encoding (see [3, 4]). We believe that the extra layer of syntactic structure of XML is an impediment to human readability. Consequently, we only consider ASCII (and extension to UNICODE where applicable) text files.

The Harwell-Boeing (HB) sparse matrix collection [5] 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. The file format for HB matrices is influenced by FORTRAN programming language array data type and input/output constructs. Only the nonzero elements and associated auxiliary data structures (to locate the nonzero elements in the matrix) are explicitly stored. The matrix is stored as a sequence of "compressed" columns. 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) [6] is a closely related (to HB) format which is more flexible in that it permits additional information specific to graphs and hypergraphs. The matrix is stored as compressed columns as in HB. One difficulty with using compressed column (or compressed row) is the potential for overflow of indices. This can be best explained with an example. Consider a $n \times n$ sparse matrix with $n = 2^{30}$ and assume that on average each column contains 4 nonzero entries. Using compressed column approach as in HB and GAMFF, the largest index in the auxiliary array needs to be $nnz + 1 = 2^2 \times 2^{30} + 1$. Most present day computer systems employ a 32-bit encoding scheme for signed integers such that the largest positive integer that can be represented is $2^{31} - 1$. The problem becomes even more serious if one considers higher dimensional tensors. On the other hand, each nonzero entry can simply be identified by a pair (i, j) where i, j represent the row index and column index, respectively. Since $n = 2^{30}$ the indices can be encoded with 32-bit representation of integers without any overflow danger.

The Matrix Market (MM) exchange format [7] is a simple but extensible file format for storing and exchanging sparse and dense matrix data. The data is stored in an ASCII text file. For general sparse matrices each nonzero is explicitly stored together with its row index and column index. The MM format enables extensibility by allowing format specialization in the form of qualifier attributes and structured documentation. Recently, Yzelman and Bisseling [8] 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. EMM, however, is a rather specialized format and intended for specialized computing environment and scientific calculations.

The remainder of the paper is structured in the following way. In Section 3 we describe the base MM file format. Section 4 contains elaborate description of the proposed DSMDE exchange format. The section concludes with a Extended Backus-Naur Form (EBNF) specification of the grammar for DSMDE format. We note that the original MM exchange format specification does not include a BNF description. In Section 5 we include several example DSMs and MDMs from [1] given in DSMDE exchange format. Finally, the paper is concluded in Section 6 with a discussion on directions for future development.

3 Matrix Market File Format

In MM exchange format matrix data are stored in a ASCII text file. The information about the matrix is organized in three syntactic sections: Header, Comment, and Data, in that order. The header section encodes meta data 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.

1. **Header.** The header has the following structure:

   Banner ObjectType FormatType Qualifiers

   where *Banner* is the literal string `%%MatrixMarket` of 15 ASCII characters, *ObjectType* indicates the type of object (matrix) stored in the file, *FormatType* indicates the format (coordinate, array) in which the object is stored, and *Qualifiers* indicate special structural property (e.g., general, symmetric, skew-symmetric, Hermitian) and numerical field (e.g, integer, real, complex, pattern). An example header in MM exchange format may appear as below.

   ```
   %%MatrixMarket Matrix Coordinate Pattern General
   ```

   The header implies that the file contains a pattern (i.e., $0/1$) matrix stored in general coordinate format where the designation “General” indicates that the format does not exploit any special structure (e.g., symmetry) that may be present.
2. **Comment.** A line in which symbol `%` is the first non whitespace (tab, blank) character is a comment. There can be zero or more lines of comments. Comment lines can be used to provide meta data and elaborate documentation on the matrix object. For example, the following text is a comment line.

   ```
   % This is a comment and is not part of the matrix data.
   ```

3. **Data.** The content of the file after the comment section is designated for numerical data. As a convention, each line contains data for one and only one matrix element. An example data section is depicted below.

   ```
   4  4  5
   1  1
   2  2
   3  3
   1  4
   4  4
   ```

   The first line indicates that the matrix is 4×4 (has 4 rows and 4 columns), and contains 5 nonzero elements. In each of the next 5 lines, the two integers represent the row index and the column index, respectively, of a nonzero element. In the data section, no values are provided for the nonzero entries. This is an example of a pattern matrix.

The following 4×4 matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

stored in MM exchange format may appear as below.

```
%%MatrixMarket  Matrix  Coordinate  Pattern  General
% This is a comment and is not part of the data.
%
% A 4x4 sparse pattern matrix with 5 nonzero entries.
4  4  5
1  1
2  2
3  3
1  4
4  4
```

The MM format does not impose any restriction on how the values of different fields in the header can be combined. However, not all header field combinations are meaningful. The header fields and the values that are currently defined are summarized in Table 1. The literal text strings provided in `teletype` font under the label **Values** represent the values of the corresponding header field.

Table 1 Header fields and their values

Fields	*Banner*	*ObjectType*	*FormatType*	*Qualifier (NumericField)*	*Qualifier (Structure)*
Values	`%%MatrixMarket`	`Matrix`	`Coordinate, Array`	`Pattern, Integer, Real, Complex`	`General, Symmetric, Skew-symmetric, Hermitian`

3.1 Specification Detail

The banner is a fixed string of 15 ASCII characters `%%MatrixMarket` which must begin the header line in any MM file; leading whitespace characters are ignored. As displayed in Table 1 matrix is the only object type that is supported in base MM format. *FormatType* of a matrix can be one of

1. `Coordinate`. The coordinate format is intended for sparse matrices where only the nonzero entries and their locations are explicitly stored. For each nonzero

entry a_{ij} of $m \times n$ matrix A, its coordinate location is indicated by the row index i and the column index j. The first line of the Data section contains three integer values: the number of rows (m), the number of columns (n), and the number of nonzero entries (nnz). The indices $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$ and the value $a_{ij} \neq 0$ (if the *NumericField* is other than `pattern`) is stored, one per line, after the first line. The nonzero entries can be provided in any order.

2. `Array`. The array format is intended for dense matrices where all matrix entries (including zero) are explicitly stored in *column-major order*. Unlike the coordinate storage, the location information is not required. The first line of the data section contains two integer values: the number of rows (m) and the number of columns (n). After the first line, the mn matrix entries $a_{ij}, i = 1, 2, \ldots, m; \; j = 1, 2, \ldots, n$ are stored, one entry per line in the following order $a_{11}, a_{21}, \ldots, a_{m1}, a_{12}, a_{22} \ldots, a_{m2}, \ldots, \ldots, a_{1n}, a_{2n}, \ldots, a_{mn}$, which is also known as the *column-major order*.

There are two qualifiers: *NumericField* indicates the type of numerical values and *Structure* indicates the existence of special structure. The four types of numerical values are `Real`, `Integer`, `Complex`, and $0-1$ or `Pattern`. The four structure values are `General` (no special structure), `Symmetric`, `Skew-symmetric`, and `Hermitian`. Special property such as symmetry, when present in a matrix, enables savings in terms of storage (file size). A symmetric matrix A has the property $a_{ij} = a_{ji}$. For a symmetric matrix only the entries in the triangular part (entries $a_{ij}, i = 1, \ldots, n; j = 1 \ldots, i$ are stored. The full matrix can be reconstructed by using symmetry. For a skew-symmetric matrix A, we have $a_{ij} = -a_{ji}$. Only the elements below the main diagonal are stored since $a_{ii} = 0$. A Hermitian matrix satisfies the property $a_{ij} = \overline{a_{ji}}$, where $\overline{(.)}$ indicates complex conjugate. Similar to symmetric matrices only the entries in the triangular part are stored.

Listed below are additional syntax rules that apply to MM format.

1. **Separator.** ASCII space or blank character (ASCII code 32) is a separator. Numerical data on a line in a MM file must be separated by at least one space character. Leading and trailing spaces are ignored.
2. **Line Buffer.** A line is a sequence of ASCII printable characters, tab, or space; the newline character (some implementations use ASCII linefeed character LF whose ASCII code is 10) ends the line. Each line may contain at most 1024 characters. Blank lines can appear anywhere after the first line.
3. **Numerical Data.** Real and complex values must be given in floating point decimal format with optional exponential notation as in programming languages `C`, `C++`, `Fortran`.
4. **Character Data.** The character strings are not case-sensitive. For example in the header line, `matrix`, `Matrix`, `MaTriX` etc. all mean the same string.

4 The Extended File Format for DSM and MDM Data

It is almost impossible to come up with "the best" format since some of the required properties may be contradictory. In this section we provide the general specification of our proposed format DSMDE which is an extension of the popular MM exchange format.

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 DSMDE exchange format views design structure models as matrices (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 line. 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` 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 part of MM format 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 MDM data. The two existing qualifiers *Numeric-Field* and *Structure* are retained as they apply to new object types. We introduce the following additional qualifiers.
   a. *Orientation.* This qualifier (*Orientn*) encodes the information about the DSM orientation convention for off-diagonal marks. The two variants are: `I`nput in `R`ow and output in column (`IR`) and `I`nput in `C`olumn 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 is sufficient to represent system interactions, many real-life models may require a more elaborate interaction structure. Eppinger and Pimler (See Example 3.1 in [1]) studied the climate control systems of cars and trucks produced by Ford Motor Company. They have identified four types of interactions 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 [1]), uses three interaction sources: explicit, inferred, and perceived. In the DSM model of software library CSparse [9], 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 [1]), 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 are recorded in the header with the qualifier *NIattribute*. A simple 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. This 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 [1]) model displays interactions recorded at two different dates.
As has been noted, 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 documentation 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. A MDM is treated as a collection of DSMs and DMMs such that the header field for a 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, $n_d > 1$. A MDM model can be viewed as a block triangular matrix as shown below.

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

The diagonal blocks $A_{ii}, i = 1,\ldots,n_d$ correspond to the DSMs. The off-diagonal blocks $A_{ij}, i < j, i, j = 1,\ldots,n_d$ represent the interactions between elements in domains $i \neq j$ and are known as domain mapping matrix (DMM). In a 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 next $\frac{n_d(n_d-1)}{2}$ lines must have a value for each of *FormatType*, *Numerical Field*, *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 2. 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 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.

Table 2 Header fields and their values in DSMDE

Fields	`Banner`	*ObjectType*	*Qualifier (Ndomain)*	*Qualifier (FormatType)*	*Qualifier (Structure)*	*Qualifier (NIattribute)*	*Qualifier (NumericalField)*	*Qualifier (Orientn)*
Values	`%%MatrixMarket`	`Matrix,`	n_d	`Coordinate,`	`General,`	n_{a_1}	`Integer,`	`IC, IR`
		`DSM,`		`Array`	`Symmetric,`	n_{a_2}	`Real,`	`IC, IR`
		`MDM,`			`Skew-symmetric,`	n_{a_3}	`Complex,`	`IC, IR`
		`DMM`			`Hermitian`	n_{a_4}	`Pattern`	`IC, IR`
						$\vdots$	$\vdots$	
						$n_{a_{n_d}}$		`IC, IR`
						$n_{a_{n_d+1}}$		
						$\vdots$		
						$n_{a_{n_d+n_{mdm}}}$		

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 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, such that they correspond to the row and column indices $i, j = 1, \dots, n$ of the DSM matrix provided in the data section. 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).
The 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 possess certain attributes depending on the type of the object. Consider the product architecture DSM example 3.8 "Building Schools for the Future" [1]. There are three sources of interactions: *Explicit* (1), *Inferred* (2), *Perceived* (3), and three types of interactions: *Structural* (1), *Spatial* (2), *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 $i, j, \in \{1, \ldots, n\}$, indicate 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 follows:

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

on 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 $\mathscr{I} \times \mathscr{J} \times \mathscr{S} \times \mathscr{T}$ where

$$\mathscr{I} = \mathscr{J} = \{1, 2, \ldots n\}, \quad \mathscr{S} = \mathscr{T} = \{1, 2, 3\},$$

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. $A_{11}, A_{22}, \ldots, A_{n_d, n_d}$
b. **DMM** data. $A_{12}, \ldots, A_{1n_d}, A_{23}, \ldots, A_{2n_d}, \ldots, A_{n_{d-1}n_d}$

4.1 DSMDE Grammar

In this section we provide the syntax specification of DSMDE exchange format using EBNF (Extended Backus-Naur Form) notation. We remark here that MM exchange format does not provide a EBNF specification.

Unfortunately, notation to describe grammar rules in BNF/EBNF has not been standardized [10]. Hence, we briefly describe the meaning of symbols used together with 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 upper-case denote nonterminal symbols.
4. **Terminal.** Words with first character in lower-case denote terminal symbols. A terminal symbol or token describes a lexical pattern of strings defined over the set of ASCII printable characters (ASCII code $33,\ldots,126$). 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[1]; 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 in the documentation section

   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.

[1] In some computing environment.

Table 3 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`
Seperator	::=	space | tab

5 Examples

In this section we illustrate the DSMDE exchange format with three real-life design structure models. The first is a product architecture DSM, the second a process architecture DSM, and the third a 2-domain MDM.

5.1 Product Architecture DSM Example (Fig. 3.6.3 of [1])

Figure 2 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`) orientation. 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.

The fist 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 2 displays only the first 10 interaction values of the DSM.

5.2 Process Architecture DSM Example (Fig. 6.3 of [1])

Figure 3 depicts a process architecture DSM model in DSMDE exchange format. The header line

```
%%MatrixMarket DSM 1 Coordinate General 0 Pattern  IR
```

indicates that the object stored in the DSMDE file is a DSM model, single domain, given in `Coordinate` format with `General` structure. The DSM is a pattern matrix (*NumericField* = `Pattern`) implying that the interactions are 0-attribute

```
%%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
0.4  0.9  0.36 2
0.3  0.9  0.27 2
0.2  0.1  0.02 1
0.8  0.1  0.08 2
0.2  0.9  0.18 2
0.1  0.8  0.08 1
0.6  0.7  0.42 2
```

Fig. 2 Product Architecture DSM Model AW101 [1] in DSMDE Format.

objects. Consequently, the documentation subsection “InteractAttributeNames” is empty. The first line of data section

```
12  12  52
```

indicates that the object is a 12×12 matrix containing 52 (nonzero) interaction marks. Each of the next 52 lines contains the location (row index, column index) of a mark. Line 2 of the data section

```
1     2
```

```
%%MatrixMarket DSM 1 Coordinate General 0 Pattern  IR
% Unmanned Combat Aerial Vehicle (UCAV) Conceptual Design Process DSM
% Example 6.3;[Steven D Eppinger and Tyson R Browning;
% Design structure matrix methods and applications; MIT press, 2012]
% Number of domain:  1
% Number of attributes: 0
% Input convention: Input in row (IR)
%
%beginDomain
% Process Architecture DSM Unmanned Combat Aerial Vehicle (UCAV)
%endDomain
%
%beginModElement
% 1 = prepare UCAV conceptual DR&O, 2 = create configuration concepts,
% 3 = prepare3-view drawing & geometry data,
% 4 = perform propulation analysis & evaluation,
% 5 = perform aerodynamics analyses&evaluation,
% 6 = perform mechanical&electrical analysis & evaluation,
% 7 = perform weights analysis & evaluation
% 8 = perform S&C characteristics analysis & evaluation,
% 9 = perform performance analysis & evaluation ,
% 10 = perform multidisciplinary analysis & evaluation ,
% 11 = make concept assessment and variant decisions,
% 12 = prepare&distribute choice config. Data set
%endModElement
%
%beginAttribute
%endAttribute
%
12  12  52
1  2
1  11
2  1
2  11
3  1
3  2
3  8
4  1
4  2
4  3
4  5
4  8
5  1
5  2
5  3
5  7
```

Fig. 3 Process Architecture DSM Model UCAV [1] in DSMDE Format.

tells that there is a nonzero mark at row 1 and column 2.

5.3 Multidomain Architecture MDM Example (Fig. 9.1.1 of [1])

Figure 4 displays the BMW Hybrid Vehicles Architecture Concepts MDM described in [2]. The MDM consisted of two DSM models:

1. The Product Functions DSM is a 19×19 general pattern matrix containing some empty rows and columns. In the displayed matrix if a system element does not interact with any other system element, the corresponding row and column is not shown. For example, element number 9 does not have any interaction with other elements. Thus, row and column 9 is omitted in the displayed matrix.
2. The Components DSM is a 27×27 symmetric pattern matrix also containing a few empty rows and column; they are omitted in the displayed matrix and data.

The only DMM in the MDM model defines a mapping between Product Functions and Components DSMs. The resulting rectangular matrix is a pattern matrix. Since the MDM defines a relation between two separate domains, the underlying network is undirected. Figure 4 displays only a part of the interactions in the data section. We recall that for an MDM with n_d domains, there are $n_d + \frac{n_d(n_d-1)}{2} + 1$ header lines. The first header line in Figure 4

```
%%MatrixMarket MDM  2
```

states that the DSMDE file stores an MDM object consisting of 2 domains. Of the next $n_d + \frac{n_d(n_d-1)}{2}$ header lines, first n_d lines contain information about the individual DSMs and the following $\frac{n_d(n_d-1)}{2}$ lines are for MDMs. With $n_d = 2$, header lines 2 and 3 contain header information for the 2 DSMs (block diagonals A_{11} and A_{22}). The following $\frac{n_d(n_d-1)}{2} = 2(2-1)/2 = 1$ header line is for the only DMM object (off-diagonal block A_{12}) defined by the MDM. The DMM specification does not have a value for input convention (`IC, IR`) since such information is not meaningful for the DMM. The comments sections and data sections are laid out in the same order as that of the header lines. The first three nonempty lines of the data section correspond to DSM 1 data and the next three nonempty lines of the data section correspond to DSM 2. Finally, the DMM data is provided in the next three lines. Note that Figure 4 does not include all DSM and DMM interaction data.

6 Concluding Remarks

A DSM is much more than an adjacency matrix representation of a network. The design and analysis of complex engineered systems [1] can be greatly aided by techniques and tools that can capture, organize, and represent nontrivial interactions among systems' elements. The DSM is a tool that can be used to represent a system's design structure in an intuitive and visually appealing manner. The DSM research over the last 3 decades have produced novel analysis techniques that have been successfully applied to a variety of industrial projects. These practical DSM examples from different application areas constitute an important database of scientific knowledge. The DSMDE data exchange format proposed in the paper is an attempt to facilitate sharing of this knowledge base among practitioners and researchers.

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. Structured comments section which is machine parseable 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. The address or location is a k_a tuple of integers indicating the coordinate location of the entry; $k_a = 2$ implies a matrix while $k_a > 2$ implies a higher-dimensional tensor. The value is a k_v-tuple over integer, real, or complex numbers representing the attributes of the interaction. Thus, each piece of interaction data can be viewed simply as a $k = k_a + k_v$-tuple object defined by the Cartesian product over k domains of values. This type of simplification of data representation is achieved with the help of structured comments where the necessary mapping between the actual symbolic attribute names to the number fields can be concisely specified.

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 are currently developing code in programming languages C and MATLAB that can perform simple input and output as well as can retrieve and display documentation in a readable manner. A powerful feature of a DSM is its ability to represent features of the underlying model in a visually appealing way. For example, after partitioning a process DSM, the strongly connected components of the underlying directed network can be concisely displayed and visually recognized as upper triangular blocks in the matrix form of the model. User-centric visualization tools that can display complex DSM data will certainly be helpful to managers and practitioners alike. This is an under-researched but promising area for future developments.

References

1. Eppinger, Steven D., and Tyson R. Browning. :Design structure matrix methods and applications. MIT press, 2012.
2. Carlos, G.: Vehicle Architecture and Lifecycle Cost Analysis In a New Age of Architectural Competition, PhD thesis, 2011, TECHNISCHE UNIVERSITÄT MÜNCHEN Lehrstuhl für Produktentwicklung, MÜNCHEN, Germany ,
3. Roughan, M., Tuke J.,: The Hitchhikers Guide to Sharing Graph Data, IEEE International Conference on Open and Big Data, Rome, Italy, August 2015.
4. Bodlaj, J.: Network Data File Formats. Encyclopedia of Social Network Analysis and Mining 2014: 1076-1091
5. Duff, I. S., Grimes R.G., Lewis J.G.,Sparse matrix test problems. ACM Trans. Math. Softw. 15, 1 (March 1989), 1-14.
6. Zien, J.Y., Simon, H.D., Woo, A.C.: GAMFF - A Graph and Matrix File Format. `http://www.nas.nasa.gov/Software/GAMFF` (Accessed Aug 17,2015).

7. Boisvert, R.F., Pozo, R., Remington, K.A.,: The Matrix Market Exchange Formats: Initial Design. NISTIR 5935, 1996.
`http://math.nist.gov/MatrixMarket/reports/`(Accessed Aug 17, 2015)
8. Yzelman, A.N., Bisseling, R.H.: Extended Matrix Market File Format – An Extension to the Standard Scheme.
`http://www.staff.science.uu.nl/bisse101/Mondriaan/Docs/extendedMM.pdf`
(Accessed Aug 17, 2015)
9. Hossain, S., Khan, S.F., Quashem R.,: On Ranking Components in Scientific Software, The 17th International DSM Conference: Modeling and Managing Complex Systems, Neely School of Business, Texas Christian University, Texas, USA, November 4 – 6, 2015.
10. Zaytsev, V.: BNF was here: what have we done about the unnecessary diversity of notation for syntactic definitions. In Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC '12).

```
%%MatrixMarket MDM  2
%%MatrixMarket DSM  1 Coordinate General     0 Pattern IR
%%MatrixMarket DSM  1 Coordinate Symmetric   0 Pattern IR
%%MatrixMarket DMM  1 Coordinate General     0 Pattern
%
% MDM model of BMW's Hybrid Vehicle Architecture Concepts
%    Example 8.1 ; Steven D Eppinger and Tyson R Browning;
% Design structure matrix methods and applications; MIT press, 2012]
% Number of domain:  2
%
% This model is a MDM (Multi-Domain Matrix) consisting of two DSM domains and
% One DMM (Domain Mapping Matrix) relating two DSMs.
% DMM maps the domain of one DSM to the domain of another DSM.
% The First DSM is a 19X19 matrix; The Second DSM is a 27X27 matrix;
% The resulting DMM is a 19X27 matrix in block row-major order.
%
%    A = [ A11 A12;0 A22]
%           A11: DSM  1
%           A22: DSM  2
%           A12: DMM  1
%
% Domain 1 (DSM 1):
% Number of attributes: 0
% Input convention: Input in row (IR)
%
%beginDomain
% Product Functions DSM
%endDomain
%
%beginModElement
%
% 1 = Store Fuel, 2 = Store Electric Energy, 3 = Convert Fuel into Mechanical Energy,
% 4 = Convert Mechanical into Energy, 5 = Convert Electrical into Mechanical Energy,
% 6 = Deliver (Recover) torque to (from) wheels, 7 = Convert Moment transferred,
% 8 = Equate Rotation, 10 = Couple/Uncouple Moment, 11 = Release Energy as Heat to the Environment,
% 12 = Transfer Heat (to cooling system), 13 = Transfer Moment to (from) the Road,
% 14 = Slow or Stop Vehicle(recovering energy), 15 = Slow or Stop Vehicle (using friction),
% 16 = Control Energy Flow, 18 = Consume EI. Energy for Auto Accessory OPS,
% 19 = Consume Mech. Energy for Engine Accessory
%
%endModElement
%
%beginAttribute
%endAttribute
% Domain 2 (DSM 2):
% Number of Attribute: 1
% Input convention: Input in row (IR)
%beginDomain
% Components DSM
%endDomain
%
%beginModElement
% 1 = Fuel Tank, 2 = High Voltage Battery, 4 = Internal Combustion Engine,
% 5 = E-Motor/Generator1, 11 = Transmission, 13 = Differential Gear,
% 18 = Clutch Direct Coupling1, 21 = Cooling System, 22 = Wheels,
% 23 = Brake-system, 24 = Power Electronics/Inverter,
% 26 = Additional Electric Accessories, 27 = Mechanical Accessories
%endModElement
%beginAttribute
%endAttribute
%beginDomain
% Product Functions-Components DMM
%endDomain
%beginModElement
% DMM 1 represents mapping between DSM 1 and DSM 2
% DMM 1 has 19 Rows and 27 Columns.
%endModElement
%
%beginAttribute
%endAttribute
19  19  42
1  3
2  12

27  27  15
1  4
2  21


19  27  24
1  1
2  2
```

Fig. 4 Multidomain Architecture DSM Model of Integrated Starter Generator [1] in DSMDE Format.