Abstract— This paper presents a graphical approach to model XML documents based on a Data Type Documentation called Graphical Notations-Data Type Documentation (GN-DTD). GN-DTD allows us to capture syntax and semantic of XML documents in a simple way but precise. Using various notations, the important features of XML documents such as elements, attributes, relationship, hierarchical structure, cardinality, sequence and disjunction between elements or attribute are visualize clearly at the schema level. We believe, by having GN-DTD as tool, helps the user to arrange the content of XML documents in order to give a better understanding of DTD structures, to improve XML design and normalization process. In this paper we presented also the transformation rules to convert from GN-DTD to DTD.

Keywords— XML Model, XML design, XML schema

I. INTRODUCTION AND MOTIVATION

Data modeling is an important part of database designs which dealing with structure, organization and effective use of the information. As pointed by Biskup [9], finding a unifying data model and extending the achievements of database design theory to advances database consist of complex object types such as XML is very challenging task. This is due to, first XML documents is a hierarchical structured, required to be conformed to its schema such as DTD [16] or XML schema [3]. Second, the expression of dependency constraint such as Functional dependency will be different with conventional model due to XML structure [2]. Third, mapping XML document into well-defined and highly-structured schemas, such as those in relational and object oriented models, often requires a lot of effort and frequent schema modification. These difficulties have prevented the use of relational or object oriented approaches to XML data modeling. Therefore an appropriate data model for XML documents has become important.

The current XML data models however do not pay sufficient attention to the problem of representing the structure of XML documents. We believe, in order to present more sophisticated forms of XML documents structure, the schema such as DTD or XML schema must taken into account since it is used to define and validate XML documents structure. In our work, we consider DTD, as it has been widely well accepted and expressive enough for a large variety of applications [2]. Furthermore DTD is an early standard for XML, and many legacy XML documents structures are defined by DTDs.

However, as shown in Fig. 1, DTD commonly represented as textual representation in a hierarchical structure which is difficult to be analysed and understand by end user or practioners. Normally even the design of simple DTD may cause difficulties, partly due to the textual form of the grammar itself. Furthermore, DTD is lack of clarity and readability, therefore may cause errors during the design process. The lack of current approaches in graphic notations of the DTD could cause difficulties in normalization process particularly. For example, normalization process proposed by Arenas and Libkin [1], Kolahi [10], Wang and Topor [17] is very difficult to understand and hard to be implemented practically due to many theoretical works defined.

In this paper, we proposed a graphical notation of DTD called GN-DTD to overcome the above limitations. The GN-DTD helps to arrange the content of XML documents in order to give a better understanding of DTD structures, to improve an XML design and normalization process as well. GN-DTD has richer syntax and structure which incorporate of attribute identity, simple data type, complex data type and relationship types between the elements. Furthermore, the semantic constraints that are important in XML documents are defined clearly and precisely to express the semantic expressiveness. We believe, GN-DTD can be used to represent and support XML structure and capture more semantic of XML documents in order to solve some difficult issues, for example, query processing, lossless information and normalization.

The rest of the paper is organized as follows. Section II provides a related work of XML model. Section III presents our example of XML document and DTD. Section IV proposed our notations of GN-DTD and Section V presents the transformation rules to show that GN-DTD can also be transformed back to DTD structure. We conclude the paper with our future work in Section VI.

II. RELATED WORK

Major current XML data models use directed edge labeled graphs to represent XML documents and their schemas. These models consist of nodes and directed edges which respectively, represent XML elements in the
document and relationship among the element. These existing XML models can be categorised into: XML model to represent instance of XML document, XML model to represent XML schema and XML model for representing both XML document and XML schema. Some common data models used to represent an instance of XML documents are DOM (Document Object Model) [14] and OEM (Object Exchange Model) [9]. On the other hand, S3-graph [11], semantic network [6], CM-Hyper graph [5], Extended Entity Relational (EER) model [13] and DataGuide [7] are models to represent the XML schema only while XML Tree [1] and ORA-SS [4] model can be used to model both instance and schema.

As a summary, data models such as OEM, DOM, DataGuide have been designed for the purpose of information or schema integration. The focus of these data models is on modelling the nested structure of semistructured data but not modelling the constraint that hold in the data. In constrast, data model such as S3-Graph, CM Hypergraph, EER, XML Trees and ORA-SS have been defined specifically for data management. Amongst these models, the notation of ORA-SS, semantic network model and EER notations are best to be adopted and applied in GN-DTD. Note that our notation is different from ORA-SS and Semantic Network since we have explicitly distinguished between complex element and simple element. We also made the ordering of sub element significant by treating them as a sequence.

III. EXAMPLES OF XML DOCUMENT AND DTD

Consider the DTD in Fig. 1 describes the XML documents in Fig. 2. The first line of DTD in Fig. 1 shows that department is the root of the DTD. While second line shows that department consists of subelement course. The semantic relationship between department and course is indicated by the symbol *; represents that department can consists of zero or many course for each department. The third line of the DTD shows that each element course has subelement title and element taken_by. Symbol “,” between them indicated that they must occur in sequence. The fourth line indicates that element course has an attribute cno. The keyword "#REQUIRED"represents that the attribute cno must appear in every course while “ID” indicates that the value of cno is unique within XML document. The fifth line of the DTD shows that the keyword “PCDATA” to despite that element title has no subelement and it is a leaf element and has a string value. The same semantic applied to DTD of line 6 until line 13 accordingly. The Fig. 2 shows the XML document conforming to the rules stated in DTD in Fig. 1.

```xml
<!DOCTYPE department[
<!ELEMENT department(course*)>
<!ELEMENT course(title,taken_by)>
<!ATTLIST course cno ID #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT taken_by (student*)>
<!ELEMENT student(firstname lastname?,teacher)>
<!ATTLIST student Sno ID #REQUIRED>
<!ATTLIST teacher tno ID #REQUIRED>
<!ELEMENT tname (#PCDATA)>
]>
</department>
</doctype>
```

Figure 1. DTD [12]

In GN-DTD, we differentiate between elements that contain subelements (e.g. department) and elements with no subelements (e.g. firstname). This is important for the normalization process because the element with subelement normally might cause data redundancy in XML documents. We called the former one is Complex Element and later is Simple Element respectively. Particularly Simple Element is element that associated with keyword "PCDATA" in DTD syntax. Most of the previous models do not distinguish between complex element and simple element precisely instead they defined simple elements similar to attributes. GN-DTD provides different notations for simple elements and attributes. For example, ATTLIST in line 8 is to define the attribute in DTD and used to define the identifier for Complex Element student, hence it must be is distinguished from simple element firstname or lastname.

```xml
<!DOCTYPE courses [ 
<!course>
<!course cno = "csc101">
<title>XML database</title>
<taken_by>
  <student>
    <student sno = "112344">
      <firstname>zurinahni</firstname>
      <lastname>zainol</lastname>
    <teacher>
      <teacher tno = "123">
        <tname>Bing</tname>
      </teacher>
    </student>
  </student>
<course cno = "csc102">
<title>Database Design</title>
<taken_by>
  <student sno = "112345">
    <firstname>Azli</firstname>
    <lastname>zainol</lastname>
  <teacher>
    <teacher tno = "123">
      <tname>Bing</tname>
    </teacher>
  </student>
</course>
```
IV. GRAPHICAL NOTATION OF GN-DTD

GN-DTD emphasizes the representation of semantic constraints between the complex elements, simple elements and attributes clearly. GN-DTD represents the structure and the semantic constraints of the XML document in a schema level. GN-DTD has six basic components:

- A set of complex element nodes representing the element that have subelements
- A set of simple element nodes representing the element that have no subelements
- A set of attribute nodes representing the attributes defined in ATTLIST
- A set relationship representing the semantic relationship between the complex elements, simple elements and attribute node
- A root node representing the first element in the DTD

A. Complex Element node

Complex element node is to represent a set of elements which has another sub element and attribute. Each complex element node has one and more labeled directed arrow from it going to another node. The complex element node is denoted as rectangle box. The label shows the name of complex element and it is written in rectangle box as tuple <name,level>, where name represent the name of node and level represent the depth of the node in the GN-DTD. The name is mandatory. Fig. 3 gives an example for complex element node labeled as <student,1> represents that this complex element node student located at level one in GN-DTD

Figure 3. Complex element node Student

B. Simple Element node

A simple element node is to represent the child of complex element that does not contain children element. The simple element node is represented by labeled rounded rectangle box. Each of simple element nodes is labelled inside with the form <name,level,type> where name is the name of simple element, level is the depth of the node in the three. A simple element has type #PCDATA or #CDATA.

Simple element node can be: Single value which has only one value, Multi value which can have a set of values, Required/mandatory, which must have a value for every instance, and Optional which may not have a value in some instances

In GN-DTD, all the above types of simple nodes will be represented using symbol ‘?’ , ‘+’ and ‘*’. The symbol is written in front of the tuple <name,level,type> to differentiate among them accordingly. All simple element nodes are assumed to be mandatory and single valued, unless the node contain an ? which shows that are single value and optional, or + which shows are they are multivalued and required, or an * which shows they are optional multivalued.

In GN-DTD, #PCDATA and #CDATA is represented as string ‘S’. Figure 4 shows the notation and semantic of each simple elements, for example, the simple element firstname is mandatory, simple element lastname is optional, tutor is multivalued and required, while address is multivalued and optional. All of them are located at level 2 and have type #CDATA.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;firstname,2,S&gt;</td>
<td>Mandatory, single value, CDATA</td>
</tr>
<tr>
<td>+ &lt;tutor,2,S&gt;</td>
<td>Required, multi value, CDATA</td>
</tr>
<tr>
<td>? &lt;lastname,2,S&gt;</td>
<td>Optional, single value, CDATA</td>
</tr>
<tr>
<td>* &lt;address,2,S&gt;</td>
<td>Optional, multi value, CDATA</td>
</tr>
</tbody>
</table>

Figure 4. Type of simple elements node

C. Attributes node

Attribute node is used to represent attributes, which describe the property for a complex element node. For attributes node, the attribute name, attribute type and attribute default must be presented clearly in the diagram. These criteria will be written as tuple <name,level,type>. However the attribute default is presented using different notations to differentiate among them. As shown in Fig. 5 type of node attributes is represented by various notations of labeled oval diagram correspond to attribute default.

An attribute node is an identifier for complex element node. It is represented as ID #REQUIRED which is unique and mandatory among the instance of complex element. The optional attribute is defined by the #IMPLIED keyword in the DTD. The attribute can be classified as single attribute and multi attributes. A single attribute has an atomic value only. A multi attributes for a complex elements such as IDREF(s) has special meaning value(s).
### Notations and Meaning

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>#REQUIRED</td>
<td>(mandatory)</td>
</tr>
<tr>
<td>#IMPLIED</td>
<td>(optional)</td>
</tr>
<tr>
<td>#FIXED</td>
<td>(optional)</td>
</tr>
<tr>
<td>#IDREF</td>
<td>(reference)</td>
</tr>
</tbody>
</table>

#### Figure 5. Type of node attributes

In addition, some attributes such as IDREFS contain one more data values. For example the following notation represents that attribute name ISBN, located at level 3 and required (unique) in DTD.

![ISBN,3,ID>](image)

The above notation is represented in DTD as follows;

```xml
<ATTLIST Book ISBN ID # REQUIRED>
```

**D. Root node**

A root node is a member of complex element node but the level of the root node always started at level 0. Root node notation similar to complex element notation where it is a special case of complex element node. The root node can be identified from DTD using the keyword `DOCTYPE`.

For example

```xml
<!DOCTYPE courses [ content..]>
```

The root node for the above example will be presented as follow:

```xml
<courses,0>
```

**E. Relationships between the nodes**

Link between nodes represent relationship which contains **type of cardinality constraint**. There are three types of link exist in this model: Inheritance link, Part- of Link and Referential Link.

1) **Inheritance Link**

The inheritance link is a relationship between complex element nodes and other complex element node. This link shows inheritance between complex elements and also a hierarchical relationship between parent nodes to child nodes or ancestor node to descendant node. Parent child relationship, is a binary relationship between parent and child where ancestor descendant is ternary relationship between ancestor and descendant where their parent must be from same path from the root. To reveal more semantic in their relationship, the cardinality constraint is associated with the link. In a DTD declaration, there are four possible cardinality relationships between the parents and children: This is illustrated as follows:

```xml
<! ELEMENT E (E1, E2+, E3*, E4?)>
```

The above segment of DTD shows that complex element E has four children E1, E2, E3 and E4. The cardinality of the constraint is described as follows:

- Only (default): An Element E must have one and only one child E1.
- Any (*): An element E can have zero or more child E3.
- Optional(??): An element E can have either zero or one child E4
- At least (+): An element E can have one or more child E2.

The inheritance link between complex element nodes is represented by a directed arrow(s). The semantic cardinality relationship between two complex element nodes is represented by `[cardinality constraint]` at the end of the arrow. The notation `constraint [0:N], [0:1] and [1:N]` is used to represent the operators *, ?, and + respectively. If no constraint is present, the cardinality constraint by default is assumed and omitted from the presentation. This relationship cardinality constraint is indicated using directed arrow as follows:

![Inheritance link between two Complex elements](image)

As shown in the above figure, complex element `student` has inheritance link with complex element `courses`. The semantic constraint between them is the `student` can take zero to many `courses` at one time. It shows inheritance link represents two complex elements must be conceptually located in different level between parent node and child node. In this example, complex element `student` is a parent node and complex element `courses` is a child node which located at level 1 and level 2 respectively.

2) **Part-of Link**

Part-of link is a relationship between complex element node and simple element node or attribute node. This is a composite relationship consists of some other parts. This relationship is illustrated using the following notation.

![Part-of Link](image)

For example Fig. 8 despites the part –of link between complex element node `BOOK`
and its attributes nodes. The attributes node ISBN, author, title and price are part-of complex element node book which are located at level 3. ISBN node is required since multiple books may be exist, thus ISBN may also appear many times but each must have a distinct value. The attribute author and title is optional with type CDATA but price is fixed with value £12.

The segment of DTD represented by the above GN-DTD is as follow;

```xml
<!ELEMENT Book (EMPTY)>
<!ATTLIST Book
  ISBN ID # REQUIRED
  Author CDATA # IMPLIED
  Title CDATA # IMPLIED
  Price CDATA # FIXED '£12'
```

3) Referential Link

Another relationship is to represent the internal reference relationship between complex element and another complex element referenced by the former element within document. Internal reference relationship means that the case the complex element with IDREF or IDREF(s) attribute refers another complex element within same document which has the corresponding ID attribute value. This relationship is represented as a dotted directed arrow.

![Referential Link between two attributes node](image)

4) Constraint between set of relationship

a) Sequence between set of child elements nodes

Normally each complex element node consist a single attribute node or multi attribute node. We emphasise in our notation those node must be located first in the sequence before include other simple or complex elements node. To illustrate this, we draw a directed curved up arrow and labelled with {sequence} across all the set of relationship involved. Consider the following segment of DTD and its GN-DTD where attribute Sno is located at first position in the sequence of child elements.

```xml
<!ELEMENT student (fname,lname,grade )>
<!ATTLIST student
  Sno ID #REQUIRED
  Name CDATA #REQUIRED>
```

b) Disjunction between the set of subelements

We have a set of subelements that are in an exclusive "OR" {XOR} relationship to represent notation "|" in DTD. For example, for the complex element node student, only one of its subelement which is fname or lname, to be appeared as its sublements in the XML document. To illustrate this, we draw a line and labelled with {XOR} across all the set of relationship involved. Follows is a real example of application taken from [15]. `<! ELEMENT chapter (page* citation* table*)* >` which is equivalent with `<! ELEMENT chapter (page*| citation*| table*)* >`. Fig. 11 shows the line with label {XOR} indicates more than two disjunction relationship. Subelements page, citation and table are simple elements with multivalued.

![Disjunction of several Simple elements](image)

Consider the following comprehensive example described in Fig 12 in order to fully understand the GN-DTD.

```xml
<!DOCTYPE school[
  <!ELEMENT school (course*|subject*)>
  <!ELEMENT course(students*)>
  <!ATTLIST course cno ID #REQUIRED>
  <!ELEMENT subject(students*)>
  <!ATTLIST subject
    sno ID #REQUIRED
    name CDATA #REQUIRED>
  <!ELEMENT students (student*)>
  <!ELEMENT student (tel?,address*,grade?)>
  <!ATTLIST student
    sno ID #REQUIRED
    name CDATA #REQUIRED>
  <!ELEMENT tel (#PCDATA)>
]>
```
Fig. 13 shows GN-DTD that represents the DTD structure in Fig. 12. This diagram shows that there is a disjunction relationship between complex element course and complex element subject. The complex element course consist sequence of attribute cno and complex element student. While complex element student consists of sequence of attribute sno, name, simple element tel, complex element address and simple element grade. Attribute sno and name are required for the complex element student while simple element tel and grade are optional. The relationship between complex element course and complex element student is zero to many. Complex element student can consist of zero to many addresses. Attribute code and city are required while attributes street is optional for complex element address. The complex element subject has the similar structure with complex element student as well.

V. Transformation from GN-DTD to DTD Structure

It is important to show that GN-DTD can be transformed back to DTD structure. The general rules and semantic rules are used to transform back the structure and semantic of GN-DTD model into DTD. General rule consisting of four aspects: Selected Root node, Complex Element node and Simple element node, Identify Attribute node, and identify Element subelement relationship. Whereas Semantic Rule involve of: semantic relationship, cardinality constraint, sequence rule and disjoint rule.

A. General Rule

1) Root node

The level of root node is represented by zero. The following graphical notation for root node will be transformed to the following syntax:

![Diagram](image)

Figure 13. GN-DTD
2) Complex element and simple element node

![Diagram of complex element and simple element node]

```xml
<!ELEMENT A(B,C,D)>
<!ELEMENT B #PCDATA>
<!ELEMENT C #PCDATA>
<!ELEMENT D #PCDATA>
```

3) Attribute node

![Diagram of attribute node]

```xml
<!ELEMENT A(C,D)>
<!ATTLIST A B ID #REQUIRED>
<!ELEMENT C #PCDATA>
<!ELEMENT D #PCDATA>
```

B. Semantic Rule

1) Sequence Relationship

![Diagram of sequence relationship]

```xml
<!ELEMENT A(B,C,D)>
<!ATTLIST A B ID #REQUIRED>
<!ELEMENT C #PCDATA>
<!ELEMENT D #PCDATA>
```

2) Disjoint Relationship

![Diagram of disjoint relationship]

```xml
<!ELEMENT A(B(C,D))>
<!ELEMENT B #PCDATA>
<!ELEMENT C #PCDATA>
<!ELEMENT D #PCDATA>
```

3) Cardinality Semantic

![Diagram of cardinality semantic]

```xml
<!ELEMENT A(C,D)>*
<!ATTLIST A B ID #REQUIRED>
<!ELEMENT C #PCDATA>
<!ELEMENT D (EMPTY)>
```

C. Transformation rules

Given the GN-DTD, the syntax and structure of DTD can be derived easily. This can be done by traversing all nodes by level to other level starting from root node until leaves nodes.

Step 1 Level 0, a root node is represented by

```
<!DOCTYPE root node name [element type definition] >
```

Step 2 Level 1, identity the subtree of GN-DTD, check the number of nodes, type of nodes and relationship type.

Step 3 If there is more than one node at level 1 and their relationship type between root and node(s) are a hierarchical link then generate

```
<!ELEMENT root node name (Ni) >
```

Where Ni is the list of subelements/child nodes.

3.1 Check the relationship set between parent nodes and child nodes,
3.1.1 If \{XOR\} means the relationship between node is a disjunction and will be represented using symbol ‘|’
Else
3.1.2 If \{sequence\} means the relationship is sequence and will be represented using symbol ‘,’

3.2 Check the semantic constraint between parent nodes and child nodes in each relationship set and map to following operator:
   3.2.1 if [0..N] map to operator *,
   3.2.2 if [1..N] map to operator +
   3.2.3 if [0..1] map to operator ?

Step 4 If the list of subelements (Ni) is not empty, using depth first traversal, for each node in list subelement Ni
4.1 repeat step 3.1 and 3.2
4.2 generate <!ELEMENT Ni (subelement Nj)>
4.3 for each complex element (Ni), find an attribute node and generate
   <!ATTLIST Ni attribute name attribute type>
4.4 For subelement Nj
   4.4.1 If Nj is a simple element has part of link with Ni then generate
      <!ELEMENT simple element name #PCDATA>
      (Repeat for all simple element nodes)
   4.4.2 If Nj is a complex element node has inheritance link with Ni
      Repeat step 4
   4.4.3 If Nj is a complex element node has part of link then generate
      <!ELEMENT Nj (EMPTY) >

Step 5 Go to next subtree GN/DTD and repeat step 4

VI. CONCLUSION
We have presented GN/DTD to describe XML documents based on DTD. Using GN/DTD, the syntax and semantic constraints of XML documents can be described clearly and precisely. To show the flexibility of the GN/DTD, the transformation rules to convert GN/DTD to DTD is provided. For the future work, we will define the operations and normal form of GN/DTD. Normalization algorithm will be developed as well to transform GN/DTD into its normal form to improve the quality of GN/DTD by reducing undesirable redundancies. Once the normal GN/DTD is derived, it will be transformed back to DTD and then generate a non redundancy XML document.

REFERENCES