An XML Schema Naming Assister for Elements and Types

Puja Goyal

Computer Scientist
Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory

An XML Schema Naming Assister for Elements and Types

Puja Goyal

Computer Scientist
Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory

July 2004

Abstract

Providing a consistent naming convention for elements and types is essential in the creation, development, and maintenance of Extensible Markup Language (XML) schemas. It improves schema readability and consistency, consequently speeding up future schema adoptions and implementations. The Naming Assister focuses on mapping terms used to assemble element or type names against a table of allowable terms, and checking the construction of compound names against the Automated Equipment Exchange (AEX) Testbed's extension to the International Standardization Organization (ISO) -11179 recommended naming convention. This tool was originally written to determine naming inconsistencies within the AEX Testbed's XML schemas, and to assist in the establishment of a table of standard terms.

.

Introduction

The Naming Assister facilitates the process of establishing consistency when naming elements and types in Extensible Markup Language (XML) schemas [1]. These element and type names found in XML schemas are formed by concatenating terms (such as "facility", and "location") to form a compound name ("facilityLocation"). The compound name in turn, corresponds to the tags in an associated XML instance file. The Naming Assister parses the names found in a schema into their constituent terms, and checks these terms against a list of allowable terms (or a table of standard terms) provided by the user.

Aside from individual term checking, the tool also verifies the structure of the entire compound name. The International Organization for Standardization (ISO) [2] established a naming convention in ISO-11179 Part 5 [3], that recommends names should consist of terms categorized into four (4) usages including <object class>, <qualifier>, <property>, and <representation>. The qualifier may qualify the <object class>, <property>, and/or <representation>. These usages signify locations of terms in a compound name. The Automating Equipment Information Exchange (AEX) Testbed [4] further extended ISO-11179 by adding the three (3) usages <prefix>, <quantity>, and <suffix>. It also restricts the qualifier usage to the property. Based on this restriction and extension, terms according to their usages must be compounded in the following order <prefix><object class><qualifier><property><representation><quantity> <suffix>. The Naming Assister checks the order of terms according to their usages to verify the compound name complies with this suggested naming convention, and suggests a rearranged name if it does not comply.

This tool can drastically improve the time it takes developers to locate and fix mistakes in their schemas. If a term is not found in the table, this could signal a spelling error in the schema, i.e., a term is used in the schema and is not allowed, or the term is a new term that should be added to the table. If a compound name does not follow the AEX Testbed naming convention, this could signal an incorrect arrangement of terms. This type of name and term checking can assist developers to use terms consistently throughout their schemas, and institute naming guidelines for future XML schemas.

The rest of this document details the Naming Assistor's system requirements, programming environment, input files, output files, tool logic, user interface (the form), and a guide to running the naming assister using the form or from the command line. Each of this is described in its own subsection.

System Requirements Environment

The software has been tested to work on Windows 2000 and XP. The latest version of the .NET framework must be installed [5].

Programming Environment

The Naming Assister is developed using the Microsoft Visual Basic.NET environment.

Input Files

The following input files are required from the user:

· A table of terms spreadsheet that contains a list of vocabulary allowed when naming types and elements. The format and information required in the table are described in Table 1.

· XML schema files. These files contain the names that will be verified by the Naming Assister

· Optional – A file containing a list of line-separated XML schemas should the user wish to parse multiple schemas in a single run as shown in Figure 1.

Abbr	Acrn	Term	Expansion	Usage	2nd Usage	Explanation
A	Yes	3D	threeDimensional	Qualifier		Three-dimensional
A	No	E	enumeration	Prefix		Enumeration
F	No	algorithm	algorithm	representation		Algorithm

Table 1: Example of a table of terms

1) The **Abbr** column indicates if a term is an abbreviation. The value of "A" signals the term is an abbreviation, while the value "F" signals the term is not.
2) The **Acrn** column indicates if the term is an acronym. The value is either "yes" or "no", where "yes" indicates that the term is an acronym.
3) The **Term** column contains the term itself. A value in this column is *required*.
4) The **Expansion** Column expands the term to its full spelling. An example in Table 1 expands the term "3D" to its full expansion "threeDimensional."
5) The **Usage** column indicates the term's usage. The value must be one of seven possible choices as defined by the AEX Testbed naming convention either prefix, class object, qualifier, property, representation, quantity, or suffix. The term's usage is *required*.
6) The **2nd Usage** column specifies the term's alternate usage. The value of this column has the same selections as those of the fifth column.
7) The **Explanation** column contains a short explanation or meaning of the term.

*Note: These headings (Abbr, Acrn, etc) in Table 1 are not required; they are only used to illustrate the information contained in each column. Additionally, the *values* used above to indicate the term is an abbreviation ("A", "F") or an acronym ("Yes", "No") are also not required – you may use any value to illustrate this behavior. However, the *order* in which this data is stored in each row needs to be maintained. In other words, the third column must contain the term, the fourth its usage, etc.

```
ctx.xsd
ctxU.xsd
abcd.xsd
xyz.xsd
```

Figure 1: Example of a list of XML schemas

Figure 1 illustrates an optional file where "ctx.xsd", "ctxU.xsd", "abcd.xsd", and "xyz.xsd" are XML schema files the user wishes the Naming Assister to use.

Output Files

The output contains the following information when the program has finished.

1) The XML Schema file name
2) The line number which points to the location where the compound name was found
3) The term itself, extracted when the compound name was broken down
4) The term's full declaration/compound name

These columns indicate a possible inconsistency associated with the term/compound name, and outputs the following:
5) Displays the term if it is not found in the table
6) Displays the compound name if it is greater than 25 characters
7) Displays the original compound name and the suggested rearranged name if it does not follow the naming convention mentioned earlier.

Schema Name	Line No	Term	Full Declaration /Compound name	Term NOT found in table	Compound Name > 25 Characters	Suggested Name
ctx.xsd	514	custom	\<xsd:element name="customLocation AndGeographicArea" type="ext:Custom" minOccurs="0"\>		customLocati onAndGeogr aphic Area	
ctx.xsd	514	location	\<xsd:element name="customLocation AndGeographicArea" type="ext:Custom" minOccurs="0"\>	location	customLocati onAndGeogr aphic Area	

ctx.xsd	140	organiz ation	<xsd:element name="customOrganizat ion" type="ext:Custom" minOccurs="0">			ORIGINAL NAME: customOrganization SHOULD THE NAME BE THE FOLLOWING?: OrganizationCustom

Table 2: Example output

Tool Logic

The Naming Assister parses through one or more schemas, finds the compound names, and breaks the compound name into their component terms. These terms can either be acronyms, full words, or abbreviations (numbers are also broken down as terms, and unless numbers are specified in the table of terms, they have no significance in this tool). The program then checks for the following situations: 1) Is the term located in the table? 2) Is the entire compound name greater than 25 characters? (Note that the 25 characters length used here is an arbitrary value, this number is typically derived from the restriction of the database field name), and 3) Does the construction of terms in the compound name follow the AEX Testbed naming convention of <prefix>< object class><qualifier><property><representation><quantity><suffix>? The Naming Assister works as follows.

First the program opens the table of terms, and converts it to a **.csv** (comma delimited) file, and renames it "Table_Temp.**txt**." This text file is used as a lookup table for the program to check against XML schema names. As the program opens a schema file, it searches for the string "name=" in the XML tagging to determine names associated with elements and types. The set of characters in the double quote encountered immediately to the right is the compound name. The program begins the breakdown process by reading the first character of the name to determine if it's 1) a capital or 2) a lowercase character from its ASCII value. 1) If it is lowercase, it reads the rest of the characters until an uppercase character is found. All the characters read are stored as a term. 2) If the first character is uppercase, it then checks if the second character is A) a capital or B) lowercase. In case A where the second character is also an upper case, further characters are read until a lowercase character is encountered. These characters are also joined to form one term, and since they contain all capital - it is an acronym. In case B where the second character is a lowercase, it's similar to the first case in that it keeps reading until an uppercase character is found.

The diagram below illustrates an example of the terms produced from the original name declaration following this breakdown process:

Searches for "name="
in XML tagging to
locate the name
declaration

<xsd:element
name="customLocation
AndGeographicArea....>

Cursor starts at the
beginning of
compound word

customLocationAndGeo
graphicArea

custom, Location, And,
Geographic, Area

Locates next
"name="

Breaks down
compound word
into terms
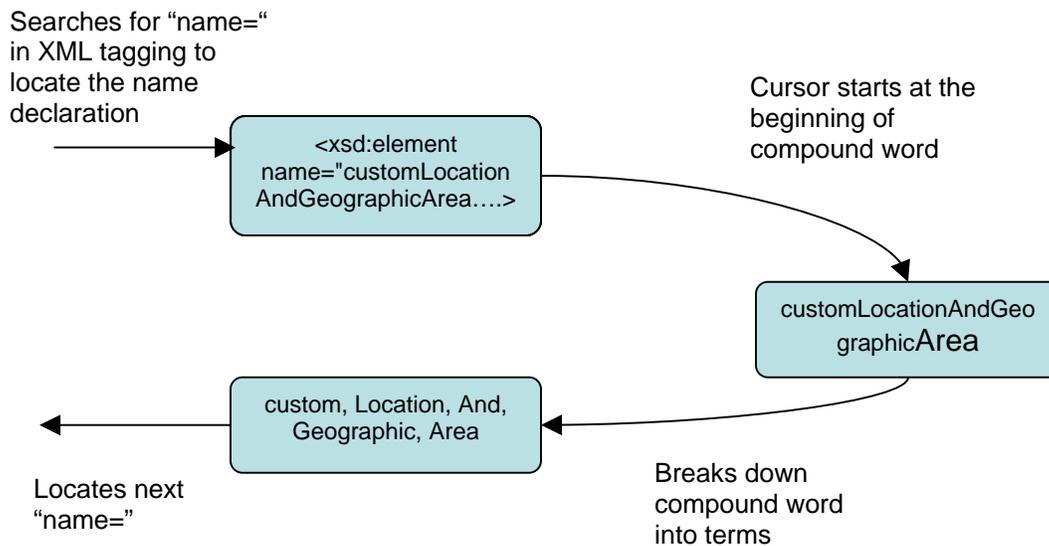
Figure 2: Breaking down compound names into its constituent terms

As each term is retrieved:
  1) The program verifies if the term is allowed by looking up the TERM column of the user's table and retrieves its corresponding usage in the USAGE column of the same table. The term is written to the output file if it was not found.

The program continues the breakdown process until the entire compound name is read. After the compound name is retrieved, the following occurs:
  2) The program determines if the length of the compound name is greater than 25 characters. The compound name is written to the output file if it is greater than 25 characters.

  3) Next the program verifies if the compound name follows the naming convention: <prefix><class object><qualifier><property><representation><quantity><suffix>. If not, the program outputs the original compound name, and suggests a rearranged name to the output file. Note that if one of the terms does not appear in the table of terms, then the term will not appear in the suggested rearranged name.

This process continues until the program has located and broken down all the names in the schema. If more than one schema is specified, it repeats this process on the rest of the schemas.

User Interface

If **no** command line arguments are given, a form will be displayed upon opening the Naming Assister as shown in Figure 3.
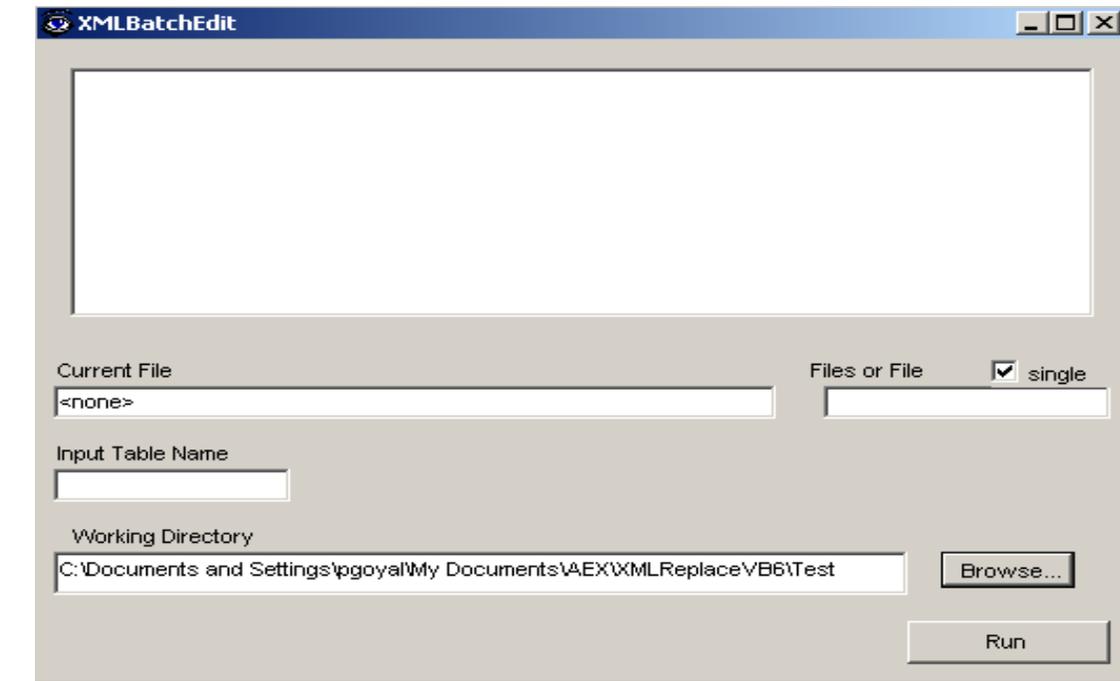
Figure 3: User Interface

On the form, there are four text fields including *CurrentFile, File or Files, Input Table Name,* and *Working Directory*, one check box labeled *single*, and two buttons labeled *Browse* and *Run*. Each one is explained below.

The *CurrentFile* text field displays the schema file in process by the Naming Assister

The *Table Name* text field is where the user must enter in the file name of his/her table of standard terms

The *Working Directory* text field contains the path to the location of schema files and table. The user must change this according to the location of schema files and table of terms on his/her own machine using the *Browse* button.

The *File or Files* text field indicates if the program should search through one file, or multiple files. If the user prefers one file, the user must click on the checkbox "*single*" and then type the name of the schema file, for example, "ctx.xsd" in the input box directly below. Otherwise, if the user wants to run through all the files, the checkbox "*single*" must be unchecked, and the user input into the text field the name of the file that contains a list of all the files to parse.

The *Run* button starts the program

Guide to running the Naming Assister

1. Create and save a table of terms as an Excel Spreadsheet.

Using the Form

2. Save the XML schema file(s) in *the same location as the table*.

3a. For parsing multiple schemas:
   - Create a text file containing the name of all the schema files to be checked in the same location as the table and schema files.

   On the user interface form:
   - Enter the file name of the table of terms under the *Input Table Name*
   - Ensure that the checkbox "single" is unchecked.
   - Type the file name just created to contain schema files on the form under the *File or Files* text box.

 3b. For parsing a single schema:
   On the Form:
   - Enter the file name of the table of terms under the *Input Table* Name.
   - Check the checkbox marked "single".
   - Type the name of the schema file on the form under the *File or Files* text box.

4. Next to the *Current Working Directory* text box, click the *Browse* button to specify the location of the table and schema files.

5. Click *Run* for the Naming Assister to begin.

When the message box "Completed Operations!" appears, the program has concluded and an "Output" folder is created in the working directory to store all the output files – "DetailParseLog.xls," and "DetailParseLog.txt" (comma delimited version).

From the Command Line

2. Save the XML schema file(s).

3a. Change to the directory where you installed the Naming Assister (This is where the .exe file should be)

4a. Run the following command:
NamingAssister.exe <*inputTable*.xls> <*inputSchema*.xsd | *listOfFiles*.txt> <*outputFile*.xls>

<*inputTable*.xls> - absolute location of the table of terms
<*inputSchema*.xsd> - absolute location of the schema you want to parse OR
<*listOfFiles*.txt> - absolute location of the text file containing list of schemas. These should be listed one schema name per line.
<*outpuFfile*.xls> - absolute location of the file containing results of the program

*Note if the user specifies a text file (.txt) as the second argument, the Naming Assister will assume the user wants to run multiple schemas and will treat this text file as a list of files as

described earlier. However, if the argument contains an .xsd extension, the tool will assume a single schema.

The results from the tool are written to the output file, and a message box "Completed Operations!" appears when the program has concluded

References

1. World Wide Web Consortium (May 2001). *XML Schema Part 1: Structure W3C Recommendation*. Available online via http://www.w3.org/TR/xmlschema-1
2. International Organization for Standardization. Information about this organization is available online via http://www.iso.org. Accessed May 2004.
3. Metadata Standards Organization. *ISO/IEC 11179, Information Technology – Metatdata Registries (MDR)*. Available online via http://metadata-stds.org/. Accessed May 2004.
4. Fiatech. *Automating Equipment Information Exchange (AEX)*. Information about the AEX project is available online via http://www.fiatech.org/projects/idim/aex.htm. Accessed May 2004.
5. Microsoft. *.NET Framework 1.1*. Available online via http://msdn.microsoft.com/netframework/technologyinfo/howtoget/. Accessed May 2004

DISCLAIMER

Acknowledgements

Status

The Naming Assister is currently a work in progress. Enhancements are being discussed and designed such as converting this application to a web-based tool, and providing an interactive look-up of terms in addition to schema parsing.