Position Paper: Uniquely Identifying Native Context for Tag Names

Author: Ron Schuldt (ron.l.schuldt@lmco.com)

Date: 11 March 2002

Filename: draft-schuldt-native-context-position-paper-01.doc

1 Summary

Data that is used within a UBL document is typically derived from multiple back-office systems – particularly if the originating organization is very large. In addition, data obtained from a particular back-office system might fulfill one meaning within a particular UBL document but carry a different meaning when used within a different UBL document. In addition to briefly describing the basic problem, this paper provides a couple examples of the business process drivers and proposes a naming and structured UID approach for tagging data within its native context – i.e., assigning a rules based name and derived UID associated with the data's source system.

Note: Although this topic is perhaps outside of the scope of the UBL Naming Design Rules Sub-Committee, it is the author's view that the topic is within the scope of the UBL Library Committee.

2 Problem Description

Integrating legacy source back-office systems that feed and receive the data communicated in B2B presents a major barrier to the adoption of Internet and XML based e-business exchanges. In addition, many larger enterprises are faced with a daunting application-to-application (A2A) integration task. Across the globe there are perhaps millions of legacy systems that require a substantial effort before they can be integrated into the global e-business environment.

Traditionally, integration has been a point-to-point activity requiring substantial effort in the data analysis of the semantic meaning associated with the data that is to be shared between systems. Depending on the complexity of the interfaces, a given interface can cost from a few thousand to over a million U.S. dollars per interface. A significant portion of the cost is associated with the time and effort involved in the data analysis of the interfacing systems. A simple but often used measure for estimating the number of possible interfaces when the point-to-point approach is used between "n" systems is the expression "n(n-1)."

Currently, "system experts" typically support those systems are called upon over and over again any time another system needs to be interfaced. There is substantial job security for those who are the "system experts." The primary reasons they become indispensable is due to the fact that the systems are frequently poorly documented and the names assigned to the data are frequently character limited and quite cryptic. For example, "ANMLY" in a particular system carries the following definition – "A deviation from the specifications for a manufacturing product." Another example from the same system is "MFGR" which has the definition "Federal Supplier Code Number look up and verification table" which is a pointer to a look-up table.

To reduce the time required to analyze interfacing systems and to reduce dependency on the "system experts," a means to capture knowledge about the semantics of a given piece of data within a system would prove to be quite helpful and an opportunity to reduce costs. Ideally, the goal should be to reduce integration from an "n(n-1)" task to a "2n" task. The proposed approach uses a rules-based ISO 11179 compliant and ebXML naming convention compliant approach that has been adopted by the aerospace industry.

3 Two Illustrations of the Business Process Drivers and the Proposed Solution

3.1 Illustration 1: Enterprise Identifier

Within engineering and manufacturing intensive industries such as aerospace, the enterprise identifier is an essential piece of data. In addition to supporting typical e-business transactions such as purchase orders and invoices, it is a critical key within document and part identifiers. The enterprise identifier as a prefix to a document (such as engineering drawing) identifier or a part identifier allows global uniqueness to be possible. Global uniqueness is necessary to support systems that have life spans measured in many tens of years – frequently with many various configurations supporting many missions. Within the aerospace and defense industry, the government (Defense Logistics Agency) assigns a CAGE Code (identifier) to each business organization that wants to do business directly with the Department of Defense. However, many businesses are simply suppliers to other businesses that conduct business directly with the Department of Defense and therefore are not obligated to obtain a CAGE Code (identifier). Instead, they may have some other identifier such as that assigned by Dunn and Bradstreet – known as a DUNS number (identifier). **Therefore, it becomes necessary to keep track of the source (the assigner) of the identifier.**

Department of Defense systems such as the B-52 bomber have thousands of engineering drawings created in the 1950s that are still actively used and updated and all contain the CAGE Code that helps to uniquely identify the organization that designed the part shown on the drawing. In addition, the manufacturer of a part of a given aircraft may or may not be the same organization that designed or manufactured the original part. **Therefore, it becomes necessary to keep track of the role that an enterprise plays (design enterprise or manufacturer enterprise).**

3.2 Illustration 2: Product Identifier

Within engineering and manufacturing intensive industries there are basically two levels of product identifiers: 1) the identifier level that is visible to an external customer or user of the product for the purpose of repair or replacement and 2) the level of identifiers of the product and its component parts that is necessary to manage the configuration throughout all phases of the product's life. The first level is typically called the model number (a family of instances). The second level is typically called the serial, batch, or lot number (a specific instance of a product with a documented configuration). The second level becomes particularly important for product recalls as well as managing configurations to support making changes to individual products. **Therefore, it becomes necessary to keep track of the level and the type (batch, lot, serial, etc.) of product identifier.**

Department of Defense systems such as the B-52 bomber have been modified many times since the original production aircraft rolled off the assembly line. During the fifty plus years of the aircraft's existence, many configurations have been designed and installed to support a wide variety of missions. In addition, many original equipment designers and manufacturers have gone out of business since the original. As a result, new suppliers have manufactured replacement parts. The fact that a particular part is original equipment or replacement or remanufactured becomes important information in making decisions regarding maintenance actions. **Therefore, it becomes necessary to keep track of the role (e.g., original, replacement, remanufactured, etc.) that the product part plays relative to its associated identifier.**

3.3 Proposed Solution: The UDEF Naming Convention and its Intelligent UID

In the late 1980s and early 1990s, the CALS Industry Steering Group (ISG) tasked a committee to develop a means for integrating the data within the enterprise. Although the resultant Universal Data Element Framework (UDEF) went through several evolutionary iterations, it has remained relatively stable for the past 5-6 years. In January 2002, the Aerospace Industry Association's Electronic Enterprise Working Group Metadata Harmonization Project decided to adopt the UDEF naming convention and its associated intelligent UIDs as a suitable means for tagging data within its native context (e.g., its source system).

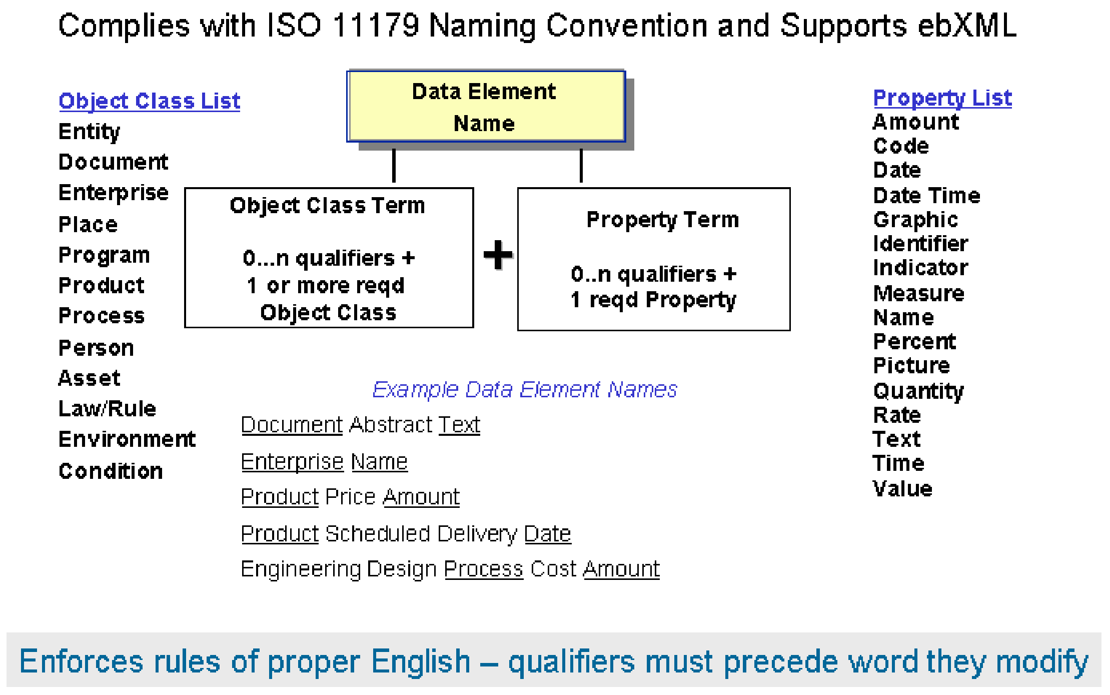
The UDEF naming convention fully complies with the ISO/IEC 11179-5 naming convention standard – since it uses object class terms and applicable qualifiers, property terms and applicable qualifiers, and representation terms (fully compliant with ebXML core components representation terms). The UDEF uses a fixed set of universally applicable object terms that correspond quite closely with the ebXML context categories. The UDEF object class terms and their definitions are as follows:

Entity - Any concrete or abstract thing of interest, including associations among things
Asset - Any data or information about any resource, other than human, which is used, consumed, or available for use/consumption by any process of an enterprise

Document - Any data or information about any collection of data or information, regardless of format, which has definable boundaries and is so designated for one or more purposes
Enterprise - Any data or information about any definable boundary collection of human and asset resources used to perform a collection of processes to create one or more products which are intended for use or consumption by outside entities
Environment - Any data or information about any natural or man-made surrounding that is relevant to the enterprise
Person - Any data or information about any person that is relevant to the enterprise
Law-Rule - Any data or information about laws (natural or man-made) or policies that govern any process of the enterprise
Place - Any data or information about any location that is relevant to the enterprise
Process - Any data or information about a definable course of events distinguishable by its purpose or by its effect, whether natural, manual, automated or machine supported and which is relevant to the enterprise
Product - Any data or information regarding something that is the result of a set of processes and which is intended to be used or consumed by activities outside of the enterprise

Program - Any data or information about any definable collection of enterprises bound by a common set of objectives
Condition - Any data or information that describes the state of something of interest to the enterprise

In addition to the above list, it is anticipated that the UDEF objects list will be expanded to include the universal objects applicable to the natural world – animal, plant, and mineral.

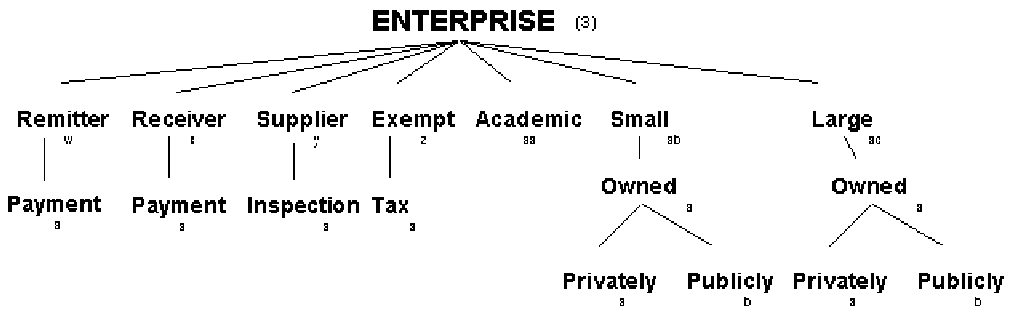


As illustrated in the above figure, the UDEF naming convention requires a relatively simple rule that is based on the rules of proper English. Specifically, a given qualifier must precede the word it is modifying. In addition, the last word in the UDEF compliant name is always the UDEF property word (same as the ebXML representation word). For example, in the above figure the last data element name example – "cost" modifies the property word "amount" and "design" modifies "process" and "engineering" modifies "design."

Within the UDEF, each object has a taxonomy of object roles and object types. For example, the "Enterprise" object includes roles such as manufacturer, design originator, buyer, payment remitter, payment receiver, etc. Also, the "Enterprise" object includes types such as government, commercial, academic, international, etc. Collectively, the UDEF object term establishes context for the UDEF property term that follows.

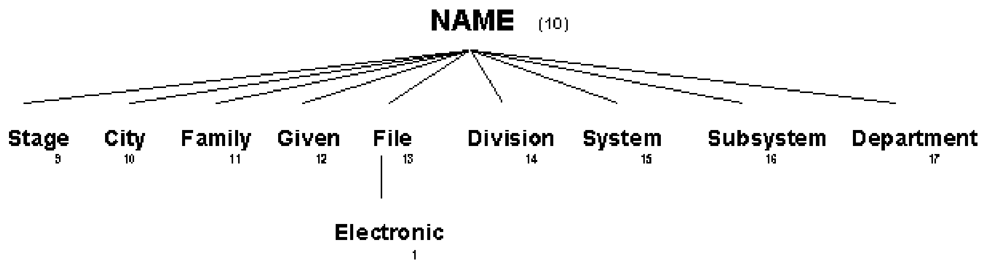
Similarly, each UDEF property (based on the ebXML representation words) term has a similar taxonomy. For example, the UDEF property “Amount” has a taxonomy of types of amount such as cost, price, tax, fee, balance, estimate, budget, etc.

Since each UDEF root level object and property has its own taxonomy, it provides its own foundation for an intelligent UID. The diagram below illustrates a portion of the UDEF “Enterprise” object taxonomy and a portion of the UDEF “Name” property taxonomy.



UDEF “Enterprise” Object Taxonomy Example

UDEF “Name” Property Taxonomy Example



The procedures for applying the UDEF to a native context source system and deriving a UID are illustrated in the following example.

1. **Identify the applicable UDEF property word that characterizes the dominant attribute (property) of the data element concept.** **For example, Name, Identifier, Date, etc.**
2. **Identify the dominant UDEF object word that the dominant property (selected in step 1) is describing.** **For example, Enterprise_Name, Product_Identifier, Document_Date, etc.**
3. **By reviewing the UDEF tree for the selected property identified in step 1, identify applicable qualifiers that are necessary to unambiguously describe the property word term.** **For example, Division Name**
4. **By reviewing the UDEF tree for the selected object identified in step 2, identify applicable qualifiers that are necessary to unambiguously describe the object word term.** **For example, Supplier Enterprise**
5. **Concatenate the object term and the property term to create a UDEF naming convention compliant name where it is recognized that the name may seem artificially long.** **For example, Supplier Enterprise Division Name**
6. **Derive an intelligent UID based on the UDEF taxonomy that carries the UDEF inherited indexing scheme.** **For example <SupplierEnterpriseDivisionName UID="y.3_14.10">**

By applying the UDEF derived intelligent UIDs to the source back-office systems, it would then be possible to let a relatively simple piece of software perform the analysis to align data from disparate systems. The following illustrates a realistic name conflict situation and how the UDEF based UIDs could provide a solution.

<ProductPartIdentifier UID="9_5.8">123-456-789</ProductPartIdentifier>

<ProductServiceID UID="9_5.8">123-456-789</ProductServiceID>

<PartNo UID="9_5.8">123-456-789</PartNo>

4 Some of the Benefits

The following is a listing of some of the possible benefits of using a UDEF based intelligent UID associated with the shareable data in source systems.

- **Built in indexing for all XML catalogs/repositories**
  - Find registered XML tags more rapidly within large catalogs/repositories
- **Enable faster alignment between disparate legacy systems – even for close matches**
  - Two hinge points (the object and the representation word)
- **Reduce costs associated with interfacing systems within the business**
- **Provide foundation for standardized global XML namespace categories**
  - PER:UID Person – all XML names with Person as the object
  - PRD:UID Product – all XML names with Product as the object
  - ENP:UID Enterprise – all XML names with Enterprise as the object
  - PRC:UID Process – all XML names with Process as the object
  - PLC:UID Place – all XML names with Place as the object
  - PRG:UID Program – all XML names with Program as the object
  - etc

5 Recommendation

The UBL Library Committee should consider adopting the UDEF naming convention and its rules as the foundation for establishing the complete (fully qualified) name for each business information entity. If adopted, then the Library Committee should also adopt the UDEF derived UID for each business information entity contained in the library.

Similarly, the UBL Naming Design Rules Sub-Committee should consider including the UDEF based UID as an optional attribute in the tag naming structure for leaf level (data containing) tags.