Customization Discussion

Tuesday, 08 May 2007

Customizing UBL – toward a unified theory: Tim McGrath

draft-mcgrath-customization-2-1.odt (or .doc)

UBL 2.0 customizations, extensions, versions, validation and interchange: G. Ken Holman

gkholman-ubl-modeling-0.5.html

Definitions: Customization

- Customization: To alter something in order to better fit actual requirements.
  - The goal is to maximize interoperability so that all parties understand the meaning of information in the documents being exchanged.
- UBL Customization: The description of XML instances, or XML-based applications acting on those instances, that are somehow based on or derived from the UBL 2.0 specification.

Definitions: Conformance

- UBL-conformant instances: An XML instance is considered UBL conformant if there are no constraint violations when validating the instance against the published UBL schema.
- UBL-conformant systems: Accepts UBL-conformant instances even if its business rules choose to reject a particular UBL-conformant instance if some optional UBL information required by the system is absent, some extension element required by the system is absent, or any other system-related business rules are violated.
- UBL-open systems: All UBL instances can be processed without consideration of receiving-system business rules.
  - Further out-of-band exchanges may be necessary

Definitions: Compatibility

- Compatibility: To be consistent or in keeping with the principles behind UBL's models and/or their development. For example, re-using UBL Information Entities to create new document models.
  - While we cannot expect automatic conformance and interoperability of these customized documents we an expect some degree of familiarity through the re-use of common patterns.

When to Customize

- Decision is whether to use UBL as-is or modify to satisfy additional requirements
- Historically, businesses have not been given a satisfactory off-the-shelf solution.
  - Although, for many, UBL will suit that need, there will still be those who have additional requirements that are not met.
- The decision is driven by real world needs balanced against perceived economic benefits

Context Methodology

- “Context”, as conceived in the ebXML Core Component work, was intended to categorize collections of business rules and constraints that affect the information entities involved in document exchanges.
- “Context methodology” is based on the premise that by using formalized classification taxonomies (context drivers) will allow automatic identification of context.
  - This is part of ongoing work and as yet we have not developed the necessary methodology to achieve this level of automation.

Regional:*
Official Constraint:*
Business Process:*
Product:*
Structure: (a result, I think)
Industry:*
Role:

Temporal:
Information Structural Context:
Application Processing:
Service Level:
Business Purpose:
Virtual Marketplace:
Contractual:

How to Customize

- Consistent customization of UBL involves a common approach to:
  - Design – knowing what models to change and how
  - Specification – communicating these new models (both to humans and applications)
  - Validation – ensuring documents conform to the required changes.

Designing for Customization

- Designing a customization involves applying business rules that apply to the given context of use or removing those that do not apply. Some rules may require:
  - Refining the meaning of information entities
  - Modifying their value space
  - Combining (or recombining) and assembling information entities into aggregations or documents.
- It is a combination of these rules that form the requirements of our context of use.
- If designers want to customize an aggregate business information entity we must define a new aggregate business information entity.

Designing for Customization: Reusing UBL Patterns

- Chin's Atomic Rule of information entities: "*All UBL aggregate business information entities must be treated as if each is a single, indivisible entity, conveying its unique structure, assigned meanings and identity as described by its schema. This applies recursively down through each and every constituent elements and types used within the aggregate business information entity.*" Chin Chee-Kai, SoftML Pte Ltd; Tapping Standards from Nature - Atomic Model for Creating Electronic Message Schema.
- If designers want to customize an aggregate business information entity they must define a new aggregate business information entity.
- All customization activities are performed external to any existing UBL patterns
  - This preserves the information entities' semantics by forbidding any modification of existing aggregations.

Designing for Customization: Qualification

- The ebXML Core Component Technical Specification supports the idea of qualifying the name of information entities as a means of indicating a context of use.
  - For example, a designer can qualify the name of “Address” when used for the delivery of goods to be called “DeliveryAddress”
  - The actual structure (or type) of DeliveryAddress would be the same as an Address.
- If a designer does not wish to change the definition of an information entity but simply re-use it in another context she can simply define a new association to an existing information entity.
- The qualifying terms themselves should describe the role of the re-used information entity in the association.

Designing for Customization: Restriction/Subsetting

- If all information entities within a UBL Order were enumerated, we would find approximately 50,000 elements.
- Actual implementations may not wish to process this massive structure.
- Subsetting involves the designer removing any **optional** information entities that are not within his business requirements.
- Example: The UBL Small Business Subset (SBS) v1.0
  - An SBS 1.0 document instance is always compliant to a UBL 1.0 schema.
    - A UBL 1.0 compliant document instance need not conform to the SBS subset.
  - Applications dealing with the subset simply ignore extraneous information entities.

Designing for Customization: Extension

- There is a recognized requirement that some documents will need additional information not covered by the UBL library, thus requiring either:
  - New aggregations
  - New information entities in existing aggregations, or
  - New associations between aggregations (both new and existing)
- Any new information entities should follow the same design rules as UBL information entities.

Designing for Customization: Value Constraints

- Some customizations involve additional constraints on the value space of information entities.
  - "The Total Value of an Order cannot exceed $100,000“
  - "The Currency Code should be expressed using ISO 4217 codes".
- Code lists or enumerated lists of possible values are a common form of value constraints.
- There may also be rules about dependencies between values of components
  - "The Shipping Address must be the same as the Billing Address“
  - "The Start Date must be earlier than the End Date"

Specifying Customizations: Versions, Subsets and Profiles

- The following information entities at the root aggregation for each document allow instances to identify their precise customization:
  - 'UBLVersionID' an identifier reserved for UBL version identification.
    - Not strictly a customizable value but necessary to understand which version of UBL is being customized.
  - 'UBLCustomizationID' an identifier (such as a namespace) for a user defined customization of UBL.
  - 'UBLProfileID' an identifier (such as a namespace) for a user defined profile of the customization being used.
    - Profiles are further refinements of customizations that enable 'families' of customizations to be implemented.
    - For example, “Stand Alone Invoicing” may be a profile for the “Northern European Subset” customization. Meaning that these IDs refer to the namespaces for schema defining the requirements for stand alone invoicing in the northern European context.

Specifying Customizations: The UBL Subset Scheme

- The UBL Small Business Subset is one example of how a subset may be implemented.
  - The schema are not restricted but spreadsheet models indicate the usable information entities and any constraints on their possible values.
  - This ensures UBL compliance of any subset instance and does not involve additional schema.
- Another approach would be to create new schema that only describe the usable information entities.
- The approach chosen affects the validation method to be used.

Specifying Customizations: The UBL Extensions Scheme

- UBL provides a special information entity known as ‘UBLExtensions’.
  - It is a container for extensions
  - Provided as a means of specifying (at a schema level) extensions to the UBL library without affecting UBL conformance.
  - A placeholder for user defined business semantics.
  - No inherent constraints
- ‘ExtensionContent’ should contain a collection of the extended information entities required for the context of use.
  - New information entities
  - New assemblies of existing UBL information entities.
- ‘UBLExtensions’ should be used with caution
  - Injudicious use of ‘UBLExtensions’ will have damaging consequences for interoperability of documents.
  - Extensions should never be used for information that may properly be conveyed in standard UBL patterns elsewhere in the document.

Specifying Customizations: XSD Schema – Extension Schema

- Ensures backwards compatibility
- Relatively straightforward
- Supports Inheritance (and other OO goodies)
- Maintains semantic lineage
  - my:NewParty extends cac:Party
  - my:NewParty is a cac:Party

Specifying Customizations: XSD Schema – Extension Schema

Issues:

- Unable to declare derivatives of the extension point
- Unable to express different enumeration restrictions based on context
- Unable to express co-occurrence constraints
- Unable to elide optional elements through derivation MJG3
- Unable to maintain UBL conventions using XSD extension MJG4

Slide 18

MJG3 Need clarification. Can state 'this was available in original but not here' in comments; actual restriction is by exclusion.
Michael Grimley, 07 May 2007

MJG4 Not sure this is necessary.
Michael Grimley, 07 May 2007

Specifying Customizations: XSD Schema - Standalone

- Two ways one could create a standalone UBL XSD customization schema:
  - Edit an existing UBL schema expression or UBL customization schema expression either by hand or by a program (Crane Softwrights Ltd. makes "Simplified UBL schema customization" software available [Crane Resources] for this task)
  - Edit a model representation of what you want in the schema and use a tool to translate the model into a schema expression; two available tools that do this for UBL 1.0 and are anticipated to have versions available for the use with UBL 2.0 are:
    - GEFEG.FX [GEFEG.FX] by GEFEG mbH in Germany
    - UBLer [UBLer] by Invinet Systems in Spain
    - UBLish [UBLish] (UBL inter-schema helper) by SoftML in Singapore

Validating Customizations: XPath Files

- The XPath files express, in a programmatically processed form, all of the possible combinations of XML hierarchy for the information items described by a document model, schema or instance. They can be produced from:
  - XSD schemas
  - XML instances
  - UBL spreadsheets (or any spreadsheets describing content nesting and definition)
- XPath files can be utilized to check for the presence of a given information item described by some other means
- An exhaustive proof of conformance and compatibility can be implemented using XPath files
  - The containment of an edited or synthesized XPath file can be rigorously tested as being a proper subset of full UBL by programmatically checking that all of the XPath file entries of the smaller are found in the larger file, and that none of the mandatory items in the larger file are missing from the smaller.