OpenBridgeML Overview

# Random Thoughts

* The data we create will outlast the method of creating it, outlast XML, outlast our current processes
* This data is really important, otherwise we would not be going through such extraordinary efforts to preserve it
* Most of this data belongs to the citizens of the world. It needs to be in open data formats designed for longevity.
* Open Bridge Modeling Language (OpenBridgeML) is a specification
* Everything follows the right hand rule!

# Documents

Overview – This document. Gives a broad overview of the various concepts in OpenBridgeML

Schema Guide – Gives schema coding and naming guidelines

Specification – Gives details specifications of the Open Bridge Modeling Language and its associated schema.

# Introduction

This document provides an overview of the Open Bridge Modeling Language (OpenBridgeML). OpenBridgeML is an open and extensible modeling standard for bridge structures. It is designed to be an international standard for modeling bridge structures.

Previous attempts at defining a bridge modeling language have focused on highway bridge structures. The fact that a bridge carries highway traffic has little to do with the structure apart from its loading. OpenBridgeML is applicable to all types of roadway/highway bridge structures, pedestrian bridges, bridges that carry utilities, aqueducts, runway/taxiway bridges that are found at airports, railway bridges, and others.

We live in an interconnected global society with an intercontinental transportation infrastructure. With the many available mapping software and web sites, a continuous transportation route can be found between Alaska and Argentina, Seattle to Miami, Lima, Peru to Rio de Janeiro, Brazil, London to Cape Town, South Africa, Oslo, Norway to Chabarovsk, Russia, and many other transcontinental and intercontinental routes. There are bridges along all of these routes. Bridges are an international thing and this is way OpenBridgeML is designed to be an international standard.

Need to design for future unforeseen uses of the data. Data will outlive its current intended use, the software intended to interrogate and manipulate it, and the computer languages and hardware used to create and execute these software programs. Technologies exist today that, when advanced, could result it completely automated bridge design and construction projects without any human intervention whatsoever. We have ITS to monitor traffic demands and trends, autonomous cars driving around test tracks in the deserts of Arizona, space ships that can fly themselves to Mars, deposit a vehicle, and the vehicle drives itself, uses excavation equation, and analyzes the soil. Advances are being made in robotics – Asimov’s mining robots on Mercury is a real possibility. We could have autonomous construction vehicles and robots. We could have the “construction controller” computer send purchase orders for materials to suppliers where their robots prepare the order, load it onto trucks that drive themselves to the job site where other robots unload the materials and do the construction. While this may all be science fiction today, we have to keep an open mind to this kind of automated future and design and develop our XML data models accordingly.

A schema is a formal specification of a vocabulary that defines the names of elements and attributes, their content, and the structural relationship between them.

# Why Open – See Learning XML book Introduction

???

# Goals/Requirements

Global standard for the modeling and exchange of bridge related information

Covers all bridge disciplines (road, train, pedestrian, etc)(not limited to highway bridges)

An open standard that is well defined and visible to all interested parties.

An extensible standard that can evolve and change as needs dictate

Extensibility – must have incremental approach to developing full schema. New technology will require new information. New uses of data will require new information. Extensible by 3rd parties for general or specific use.

Modeling detail – accommodate abstract to high-fidelity full product model.

Flexibility – many options for describing physical characteristics of bridges.

# Problems to be solved

Data storage and management – where is it stored? Who can access it? Who can change it? From where can it be accessed?

Integration of existing data sources – lots of bridge data, for lots of different purposes (design, analysis, plans, specs, inspections, ratings, permitting, etc) already exist. How to bring all this together into a cohesive representation? – This is why we need simple to full data descriptions.

Volatility of data – Some data about a bridge needs to exist from “cradle to grave” and some has a finite lifetime (e.g. construction management Gant chart)

Purpose of data – some data is general purpose and can be used in different contexts other data is specific to an application.

Ownership of data – who owns the data?

Software – how to get existing software to use and create the data? How to keep application-specific and proprietary data separate from general data? How to merge it back together when needed?

<Project>

<Units>

<LayoutGeometry>

<Alignments>

<Structures>

<Bridge>

# Schema Realms

Physical/Product Modeling

??? Modeling (stuff like code LRFD parameters independent of any application, probably required by all applications, don’t describe the physical structure)

Application Specific Information

This can be viewed like the W3C XML products (XML, XML Schema, XInclude, XPath, XQuery, XPointer, XSL Tranform, etc) – Cohesive family of schema that address specific issues.

# Schema Versioning and Namespaces

It is clear that the problem that is trying to be solved is difficult and it hasn’t been solved yet (because if it had, there would be no need for this work). After decades of using computers in transportation and bridges, nothing we have come up with works all that well. It would be foolish to think that these early attempts at devising a solution based on XML technology are going to succeed either.

The schemas developed are going to change. They will be modified to correct short comings and they will be extended and enhanced to permit more robust modeling (even entirely new schemas in the “family”). Need to adopt clear rules for versioning instance documents and schemas based on best practices to ensure the longevity of instance documents and schemas.

The data will outlive XML, its applications, and originally intended use.

## Types of Schema Changes

Case 1 – The revised schema changes the interpretation of some element. Examples of this type of change include removing an element, renaming an element, adding child elements to an element, or removing a value from an enumerated list. This type of change results in a construct that was valid and meaningful for the previous schema does not validate against the new schema.

Case 2 – The new schema extends the namespace but does not invalidated previously valid documents. Examples of this type of change include adding new elements, attributes, extensions to an enumerated list, or relaxing a constraint.

## Schema Version Convention

Significantly changed (case 1) – increment the major version number (e.g. 1.0 to 2.0)

Extensions only (case 2) – increment the minor version number (e.g. 1.2 to 1.3)

## Dealing with Schema Changes

1. Capture the schema version in the XML Schema
2. Identify in instance documents what schema version(s) the document is compatible with
3. Provide XSLT transforms for changing instance documents between versions (when it makes sense. Can be uni- or bi-directional transformations)
4. Continuously maintain availability of previously published schema.
5. When a schema is only extended (new elements, attributes, extensions to en enumerated list, etc.) strive to not invalidate existing instance documents if an XSLT is not provided. To the extent possible, make new elements optional.
   1. Change the schema version number within the schema document.
   2. Make a record of the changes in the schema change history
   3. Make the new and previous schemas available (which necessitates a change in filename and/or location)
   4. Create an XSLT transform to convert from the previous to the new schema and optionally from the new to the previous schema.
6. When a schema changes the interpretation of some elements, change the target namespace. The changes in 5 above are all applicable plus
   1. Change the target namespace in the schema file

Instance documents will also have to be updated in this case.

1. Update the instances to reflect the new namespace
2. Confirm that there is no compatibly issues with the new schema
3. Change the attribute that identifies the version/versions of the schema with which the instance is valid
4. Update the schema name/location if applicable.

# Families of Schemas

A one size fits all solution is not likely to succeed for this endeavor. Many times of bridge need to be modeled. There are many ways to model each type of bridge. There is an unlimited number of data models to be created for bridge-related modeling (stuff that doesn’t model the physical attributes of a bridge). The outcome of this project will be families of schemas that can be mixed, matched, and extended to solve specific problems.

These schemas will work better if things like versioning, namespaces, and units are handled consistently.

# APIs and Software Libraries/Tools

All of this data, no matter how well defined, isn’t much use without software tools to act upon it. Need APIs and software libraries to make development easier.

BridgeLink Extensible Application Environment + OpenBridgeConnect + Washington Bridge Foundation Libraries (WBFL) – Quickly build applications.