OpenBridgeML Units

# Introduction

This document describes the OpenBridgeML Units schema design.

# Units of Measure

One of the first technical design tasks for OpenBridgeML is to establish a protocol to model data elements that represent a physical quantity. Such data elements might quantify Length, Force, Density or Time. These quantities might have units of measure of feet, Newton, kilogram per cubic meter, and days.

The development goals for the OpenBridgeML Units Schema are:

* OpenBridgeML instance documents must fully describe all physical quantities and their units of measure.
  + One of the most powerful aspects of XML is that of self-describing documents. OpenBridgeML compliant instance documents will be generated from a wide variety of sources. Each source may prefer the use of a different unit system. Extensions to the schema may require new units of measure and new physical quantities. The units of measure used in an instance document could be consistent (all lengths are measured in feet) or mixed (span length, alignment stations, and column heights are measured in feet while cross sectional dimensions are measured in inches).
* Eliminate the need for software applications to dictate requirements for representing physical quantities.
  + Every software application will prefer to store, retrieve, and manipulate physical quantities in different unit systems. A consistent and extensible method of modeling physical quantities and units of measure eliminates the need for application-specific requirements to be incorporated into instance documents.
* Must be easy to use.
  + A heavy burden should not be place upon application developers to specify units information in an instance document. A few global parameters should do the trick with the ability to be more specific when needed.
* Physical quantities and units of measure typically used in bridge and civil engineering applications need not be explicitly defined by instance documents.
  + There are physical quantities and units of measure that are common knowledge to all bridge engineers. Quantities like length, force, and moment can be implicitly defined as well as units of measure such as feet, kip, and kip-feet.
* Must be extensible whereby non-standard units of measure and physical quantities can be defined in instance documents.
  + The goal of OpenBridgeML is to be open and extensible. As such, the units system must support the definition of new physical quantities and units of measure. It would be short sighted to assume the modeling language could capture all possible units of measure for all possible physical quantities.

## Concepts

### Physical Quantity

A physical quantity represents the magnitude of something that can be measured in the physical world. Examples of physical quantities are length (distance), time, mass, force, torque, temperature, energy, and electrical charge. Some physical quantities are fundamental such as mass, length, and time where others are derived such as velocity, acceleration, and force.

Physical quantities are defined by their dimensionality. In the context of OpenBridgeML, a general physical quantity is represented by

M, L, T, K, and A represent the fundamental quantities of mass, length, time, temperature, and angle. The exponents m, l, t, k, and a express the dimensionality of the physical quantity.

The physical quantity mass would have a dimensional representation of . The physical quantity force would have a dimensional representation of .

??how is this used??

### Unit of Measure

A unit of measure defines the basis for measurement of a physical quantity. The most common bases for units of measure are the US/Imperial system of units and the SI system of units. Examples of units of measure for a physical quantity having a dimensionality of L are feet, inches, miles, meters, kilometers, and millimeters.

The magnitude of a physical quantity is invariant to the system of units in which it is expressed. That is to say, the length of an object does not differ whether is it measured in the US or SI unit systems. That fact leads to the notion of unit conversion.

### Unit Conversions

A unit conversion is the mathematical operation required to convert a measurement from one unit system to another. In the context of OpenBridgeML, unit conversions are accomplished by the following relationship:

V1 = the value to be converted

V2 = the converted value

C = conversion factor

Pa = pre-addition term

Pb = post-addition term

The conversion for a length from feet to meters would be

The conversion for a temperature in degrees Centigrade to degrees Fahrenheit would be

The conversion for a temperature in degrees Fahrenheit to degrees Centigrade would be

Units of measure are defined in terms of their parameters for the unit conversion equation.

????

### Fundamental Units

Fundamental physical quantities are the most basic physical quantities from which all other physical quantities can be derived. In the scope of OpenBridgeML, mass, length, time, temperature, and angle are fundamental physical quantities. As a comparison, force is a derived physical quantity defined by (Mass)(Length)/(Time)2.

OpenBridgeML specifies units of measure for these fundamental physical quantities. These units of measure form the basis for all unit conversions. This set of units of measure is known as the Fundamental Units. Staying true to the overall goal of OpenBridgeML being an international standard the Fundamental Units are the fundamental units of the SI unit system. The fundamental units for mass, length, time, temperature, and angle are kilogram, meter, second, centigrade, and radians, respectfully. The fundamental units are a special case of consistent units.

### Consistent Units

This can be explained a lot better

A data set is said to be in consistent units when the units of measure for all physical quantities are consistent with some set of fundamental units. As an example, consider a computer program subroutine with the variables M and A that represent mass and acceleration. If in some instances the length element of acceleration is in inches and in other instances it is in feet, when the product of M and A is computed the result unit of force is not readily known or in a convenient format (could be slug-in/sec2 or slug-ft/sec2 which is pound-force). If M and A are in consistent units, the units of force is always consistent with the units of M and A. This make is so that developers need not worry about units of measure. Add/subtract, multiply/divide, the units just work out without the overhead of dimensional analysis at compile time and unit conversion at runtime.

All physical quantities in OpenBridgeML instance documents shall be in a consistent set of units. If the declaration of consistent units is omitted from the instance document the physical quantities must be in Fundamental Units.

### Standard Units

All physical quantities and units of measure used in a conforming OpenBridgeML instance document must be fully described. To reduce the burden on software developers common physical quantities and units of measure are defined by specification rather than by explicit declaration. That is, common quantities and measures are “implied” in an instance document thereby satisfying the full description requirement. These common quantities and measures are described in the OpenBridgeML Specification document.

### Extended Units

Physical quantities and units of measure not defined in the OpenBridgeML Specification for Standard Units must be explicitly defined in an OpenBridgeML instance document.

The available units of measure for any of the standard physical units can be extended by defining new units of measure. The standard units of measure for area are square inch, square feet, square millimeter, and square meter. As an example, consider a software program that performs hydrological analysis on bridge decks. This program uses the bridge deck surface area in acres. When this program creates an OpenBridgeML instance document it can provide the bridge deck surface area in acre so long as it also defines the acre unit of measure and associates it with the physical quantity area. When another program loads this instance document and it encounters the bridge deck surface area in units of acre, this software will be able to convert the quantity to its own unit system. Extending units of measure, while rare, will be more common that extending physical quantity types.

OpenBridgeML is an extensible modeling language. OpenBridgeML schema extensions can add elements that represent physical quantities that are not defined in the OpenBridgeML Specification for Standard Units. Physical quantities, along with their associated units of measure, can be defined in instance documents.

## Design

Discuss design for units here

### Aliases

OpenBridgeML is primarily a modeling language to facilitate the exchange of information between computer systems. It would be extremely rare for an end user to view the content of an OpenBridgeML instance document. Unlike humans, computers are perfectly happy with a single method of representing a piece of information. Humans tend to prefer having a choice for representing the unit of measure “feet”. Common representations are “ft”, “FT”, “feet”, “FEET”, and “’”. Computer systems do not need these aliases. In fact, aliases create unnecessary overhead and processing requirements.

The OpenBridgeML does not support aliases for physical quantities and units of measure.

### Unit of Measure Representation

Discuss display units… OpenBridgeML includes data for the unit tags that can be used by computer systems to put the unit designation with a physical quantity.

### OpenBridgeML Example Mark Up

The schema for declaring unit types and units of measure is given in the OpenBridgeML\_Units.xsd schema instance document. This section describes, by way of example, how the mark up of physical quantities is accomplished in OpenBridgeML instance documents.

Physical quantities are represented with a rational number enclosed by element opening and closing tags. Physical quantity elements, at the schema designer’s option, can include an optional attribute for specifying the unit of measure. Consider an element for specifying span length. The markup would be

<SpanLength unit=”feet”>100.00</SpanLength>

or

<SpanLength>100.00</SpanLength>

The unit of measure for the first element is clearly feet as it is specified with the unit attribute. This markup is explicit. However, it will be an onerous task to provide and interpret these attributes on nearly every element in the instance document. The second markup by itself is seemingly ambiguous. When taken in the context of an OpenBridgeML instance document the unit of measure is defined by the <UnitDeclaration> element (or by a closer ancestor as will be shown later).

Physical quantities and units of measure are fully and completely defined in the <UnitsDeclaration> element in an OpenBridgeML instance document. An example of the<UnitsDeclaration> element

<OpenBridgeML>

<UnitsDeclaration>

<ConsistentUnits mass=”slug” length=”ft” time=”sec” temperature=”F” angle=”degree” />

</UnitsDeclaration>

…

</OpenBridgeML>

The <ConsistentUnits> element declares that all physical quantities, unless otherwise specified, have units of measure that are consistent with the specified units of measure for the fundamental units of mass, length, time, temperature, and angle. If this <UnitsDeclaration> were provided in the instance document with the second example of <SpanLength> given above, the quantity 100.00 would be interpreted as 100.00 feet.

The <ConsistentUnits> element is optional. When omitted, the implied consistent units are the fundamental units. Had the <ConsistentUnits> element been omitted from the <UnitsDeclaration> the quantity in the second example of <SpanLength> would be interpreted as 100.00 meter.

The easiest method for handling physical quantities is to provide and manipulate them in consistent units.

The OpenBridgeML Unit schema provides for the declaration of new physical quantities beyond the standard physical quantities defined by the OpenBridgeML Specification. As an example, imagine an application that creates an OpenBridgeML instance document and, because the schema is open and extensible, the instance document contains an element that describes velocity and a value for area that is described in acres. Since velocity and acre are not part of the Standard Units they must be defined. This is accomplished by providing the <ExtendedUnits> element.

The <ExtendedUnits> element contains definitions of physical quantities and units of measure that are not defined in the specifications. The example that follows defines the unit of measure “acre”, the physical quantity “velocity”, and several units of measure for velocity.

<UnitsDeclaration>

<ExtendedUnits>

<UnitOfMeasure name=”acre” type=”Length2”>

<ConversionFactor>4046.8564224</ConversionFactor>

<UnitSystem>unitsUS</UnitSystem>

<UnitTypes>

<UnitType name=”velocity” mass=”0.0” length=”1.0” time=”-1.0” temperature=”0.0” angle=”0.0”>

<UnitOfMeasure name=”kph”>

<ConversionFactor>0.2777778</ConversionFactor>

<UnitSystem>unitsSI</UnitSystem>

</UnitOfMeasure>

<UnitOfMeasure name=”fps”>

<ConversionFactor>0.3048</ConversionFactor>

<UnitSystem>unitsUS</UnitSystem>

</UnitOfMeasure>

<UnitOfMeasure name=”mph”>

<ConversionFactor>0.44704</ConversionFactor>

<UnitSystem>unitsUS</UnitSystem>

</UnitOfMeasure>

</UnitType>

</UnitTypes>

</ExtendedUnits>

<!—Optionally, the ConsistentUnits element can be used -->

</UnitsDeclaration>

### Suggested Mark Up for Sub-Schemas

Major sub-schema (need a better name) provide their own <UnitsDeclaration>. For example, if an OpenBridgeML instance document contains the description of several different bridges and the software program the generated each bridge model uses a different set of consistent units. Each <Bridge> branch in the instance document must declare its units so that the data the follows is correctly interpreted

<OpenBridgeML>  
 <UnitsDeclaration>  
 …  
 </UnitsDeclaration>  
 <Bridges>  
 <Bridge>  
 <UnitsDeclaration> …</UnitsDeclaration>  
 ….  
 </Bridge>  
 <Bridge>  
 <UnitsDeclaration> …</UnitsDeclaration>  
 ….  
 </Bridge>  
…

Minor sub-schema (need a better name) might provide a hierarchy for declaring units of measure. For example, the description of a pier could have mark-up similar to this

<Pier lengthUnit=”ft”>

<ColumnLength>60.0</ColumnLength> Length is in feet  
 <Circular>  
 <Diameter unit=”in”>60.0</Diameter>

</Circular>

<CapBeam>

<Width>5.0</Width>

<Height unit=”yard”>1.0</Height>

In this example, the lengthUnit attribute over-rides the consistent length unit declared for the instance document. All length-type quantities within the <Pier> branch are given in feet unless the unit of measure is over-ridden as in the Diameter and Height elements. ColumnLength and Width are in feet.

### Scalar Quantities

Some common engineering values are scalar quantities. These include slope (Length/Length) and fractional values (as in 50% of the span length – 0.5L). Scalar quantities shall be represented with a fractional value. Percentages are not used in OpenBridgeML. Use 0.5 for 50%, 1.0 for 100% and so on.

## Drawbacks and Alternatives

This design choice makes it impossible (based on my limited understanding of XML Schema) to validate the unit of measure used in xxxValueType elements, such as LengthValueType, using a schema validator. The validation must occur in the application. Application developers will have to ensure the unit of measure is one of the standard units of measure or one of the extended units of measure for the unit type in use.

Example:

<xs:element name=”SpanLength” type=”bxu:LengthValueType” />

<SpanLength unit=”bigFeet”>100.0</SpanLength>

Developers will have to validate “bigFeet” against the standard and extended units of measure for length.

An alternative to this design is to instead provide in every OpenBridgeML instance document a full definition of the unit types and units of measure used within the document. This, however, would require a lengthy prologue in every document that will reduce its read-ability and cause extra processing by computer applications.

## Reference Implementation

Describe the reference implementation in WBFL.

Provides the complete OpenBridgeML Units Model, units mode management, display units services, and unit conversion services.