Bridge Data File Protocols for Interoperability and Life Cycle Management

Volume III: Model View Definition Elements for Highway Bridge Interoperable Data Protocols

FHWA Cooperative Agreement DTFH61-11-H-0027

Advancing Steel and Concrete Bridge Technology
to Improve Infrastructure Performance

August 2013

FHWA-HIF-16-003

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle Bridge Data File Protocols for Interoperability and Life Cycle Management Volume III - Model View Definition Elements for Highway Bridge Interoperable Data Protocols	5. Report Date January 2014 6. Performing Organization Code
7. Author(s) Chen, S. S. ; Hu, H. Hu; Ali, N. Ali; Srikonda, R.	8. Performing Organization Report No.
9. Performing Organization Name and Address The Research Foundation for the State University of New York on behalf of Civil, Structural and Environmental Engineering, University at Buffalo, State University of New York Ketter Hall Buffalo, NY 14260	10. Work Unit No. 11. Contract or Grant No. DTFH61-11-H-00027
12. Sponsoring Agency Name and Address Federal Highway Administration Office of Infrastructure – Bridges and Structures 1200 New Jersey Ave., SE Washington, DC 20590	13. Type of Report and Period Covered 14. Sponsoring Agency Code HIBS-10

15. Supplementary Notes
Work funded by Cooperative Agreement "Advancing Steel and Concrete Bridge Technology to Improve Infrastructure Performance" between FHWA and Lehigh University.

16. Abstract
This report and its accompanying appendices contain Model View Definition (MVD) elements for structured digital data exchanged during the lifecycle of typical steel and concrete bridges. These define what information is to be included in the data exchanges, how that information is organized, and to what level of detail that information is defined. In contrast to traditional drawings, structured data can be leveraged for re-use in many different ways, but to structure that data requires a 'schema' or template for such structure.

17. Key Words eXtensible, mapping, ISM, IFC, OpenBrIM Schema, data schemas, roadway map, bridge control information	18. Distribution Statement No restrictions. This document is available to the public online and through the National Technical Information Service, Springfield, VA 22161.

9. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No of Pages 482	22. Price

(This page is intentionally left blank)

SI* (MODERN METRIC) CONVERSION FACTORS

APPROXIMATE CONVERSIONS TO SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
		LENGTH		
in	inches	25.4	millimeters	mm
ft	feet	0.305	meters	m
yd	yards	0.914	meters	m
mi	miles	1.61	kilometers	km
		AREA		
in^2	square inches	645.2	square millimeters	mm^2
ft^2	square feet	0.093	square meters	m^2
yd^2	square yard	0.836	square meters	m^2
ac	acres	0.405	hectares	ha
mi^2	square miles	2.59	square kilometers	km^2
		VOLUME		
fl oz	fluid ounces	29.57	milliliters	mL
gal	gallons	3.785	liters	L
ft^3	cubic feet	0.028	cubic meters	m^3
yd^3	cubic yards	0.765	cubic meters	m^3
		NOTE: volumes greater than 1000 L shall be shown in m^3		
		MASS		
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams (or "metric ton")	Mg (or "t")
		TEMPERATURE (exact degrees)		
°F	Fahrenheit	5 (F-32)/9 or (F-32)/1.8	Celsius	°C
		ILLUMINATION		
fc	foot-candles	10.76	lux	lx
fl	foot-Lamberts	3.426	candela/m^2	cd/m^2
		FORCE and PRESSURE or STRESS		
lbf	poundforce	4.45	newtons	N
lbf/in^2	poundforce per square inch	6.89	kilopascals	kPa

APPROXIMATE CONVERSIONS FROM SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
		LENGTH		
mm	millimeters	0.039	inches	in
m	meters	3.28	feet	ft
m	meters	1.09	yards	yd
km	kilometers	0.621	miles	mi
		AREA		
mm^2	square millimeters	0.0016	square inches	in^2
m^2	square meters	10.764	square feet	ft^2
m^2	square meters	1.195	square yards	yd^2
ha	hectares	2.47	acres	ac
km^2	square kilometers	0.386	square miles	mi^2
		VOLUME		
mL	milliliters	0.034	fluid ounces	fl oz
L	liters	0.264	gallons	gal
m^3	cubic meters	35.314	cubic feet	ft^3
m^3	cubic meters	1.307	cubic yards	yd^3
		MASS		
g	grams	0.035	ounces	oz
kg	kilograms	2.202	pounds	lb
Mg (or "t")	megagrams (or "metric ton")	1.103	short tons (2000 lb)	T
		TEMPERATURE (exact degrees)		
°C	Celsius	1.8C+32	Fahrenheit	°F
		ILLUMINATION		
lx	lux	0.0929	foot-candles	fc
cd/m^2	candela/m^2	0.2919	foot-Lamberts	fl
		FORCE and PRESSURE or STRESS		
N	newtons	0.225	poundforce	lbf
kPa	kilopascals	0.145	poundforce per square inch	lbf/in^2

*SI is the symbol for the International System of Units. Appropriate rounding should be made to comply with Section 4 of ASTM E380.
(Revised March 2003)

(This page is intentionally left blank)

TABLE OF CONTENTS

TABLE OF CONTENTS

SECTION **TITLE** **PAGE**

TABLE OF CONTENTS

(This Page Intentionally Left Blank)

LIST OF FIGURES

FIGURE **TITLE** **PAGE**

LIST OF FIGURES

FIGURE **TITLE** **PAGE**

LIST OF FIGURES

LIST OF FIGURES

FIGURE **TITLE** **PAGE**

LIST OF TABLES

(This Page is Intentionally Left Blank)

1 EXECUTIVE SUMMARY

This report and its accompanying appendices contain Model View Definition (MVD) elements for structured digital data exchanged during the lifecycle of typical steel and concrete bridges. These define what information is to be included in the data exchanges, how that information is organized, and to what level of detail that information is defined. In contrast to traditional drawings, structured data can be leveraged for re-use in many different ways, but to structure that data requires a 'schema' or template for such structure.

A Model View Definition formally defines a subset of schemas needed to realize a specified digital data exchange scenario and expresses it in a form intended for computer software implementers.

The purpose and role of each such data exchange scenario of interest is contained in an Information Delivery Manual (IDM), which serves as a functional specification for relevant digital data exchanges. The companion (IDM Elements) report contains substantial elements of that specification expressed in the language of bridge lifecycle stakeholders.

The language(s) in which the MVD expresses these data exchange schemas reflect the conventions of the organization(s) that shepherd the development and promulgation of that data exchange standard.

For the purposes of this document, three distinct such potential languages (also know as, language bindings) are illustrated:

- eXtensible Markup Language or XML [36] (As advocated and used by such organizations as AASHTO.) .

- Industry Foundation Classes or IFC (buildingSMART Alliance) [7].

- Integrated Structural Model, or ISM (several software developers led by Bentley Systems [4]).

Thus, the MVD, in contrast to the IDM, is intended for software (translator) implementers. It is recognized that individual software solution providers (SSPs, also known as "vendors") need not implement the complete MVD.

Companion Appendices contain the following:

- Side-by-side comparison of candidate roadway alignment models (schemas) under consideration by related data exchange standards development efforts (see Appendix A, Comparison of Roadway Geometry Models).

- Listing of bridge analysis and design data elements under consideration by related data exchange standards development effort (see Appendix B, Structural Design Model).

- Concrete and steel case study bridges examples as described by the openBrIM XML schemas presented herein and rendered by the openBrIM Viewer developed by Red Equation Corp. [17] in conjunction with this work (see Appendix C, Case Study Bridges and OpenBrIM Models).

- Programmers' manual - like openBrIM schema description (see Appendix D, Describing Schema Elements).

It should be noted that a particular data exchange (e.g., Highway Design to Bridge Structural Design) will depend on just a subset of the overall model data and that a different data exchange will depend on, in general, a distinct but overlapping subset. Thus, in eventual deployed form, these subsets (called "Exchange Models" in the companion IDM Elements report) will be used selectively in multiple distinct contexts. The bridge case study appendix to this MVD Elements report exemplifies guidelines for how to use the schema elements correctly and further illustrates their use by viewer software (developed by Red Equation Corp.) [17] to provide prospective software users and software implementers with a visual check on model geometry and topology as described in the XML version of the schema described herein.

These two (IDM and MVD) reports and their accompanying appendices thus provide a substantive initial version of bridge digital data exchange standards for the lifecycle. Depending on which organization(s) further shepherd the development/refinement, promulgation, extension, and maintenance of bridge lifecycle data exchange standards, applicable portions of these two reports are recommended as the starting content for those eventual shepherding processes. Depending on when that occurs, suitable consolidation of ongoing openBrIM schema/viewer development, related concrete and steel structure data exchange standards development work, and oversight of standards terminology use; e.g., via the International Framework for Dictionaries (IFD), are also recommended.

The authors thank colleagues involved in various aspects of the openBrIM development effort, including Mr. A. M. Shirolé and Dr. S. G. Karaman. The authors also thank colleagues in the AASHTO/NSBA Steel Bridge Collaboration and the National Concrete Bridge Council and the buildingSMART Alliance along with the New York State Department of Transportation (NYSDOT) and other Departments of Transportation (DOTs) for their review comments on earlier portions of this report contents and access to information about their bridge design and construction process, bridge data, and construction sites. In addition, support and assistance from the following are gratefully acknowledged: NWFP University of Engineering and Technology, the University at Buffalo, Bentley Systems, Red Equation Corp. and other software solution providers, the Association for Bridge Construction and Design-Western NY Chapter, and the Federal Highway Administration under Cooperative Agreement No. DTFH61-11-RA-00010 administered by Advanced Technology for Large Structural Systems (ATLSS) at Lehigh University. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the Federal Highway Administration.

2 ADVISORY COMMITTEE WORKING TEAMS

For the companion IDM (upon which the present document is based) to reflect current practices within the bridge industry, a number of stakeholders have been involved in this effort to make this openBrIM development effort robust enough to constitute a substantive initial proposed data exchange standard for workhorse steel and concrete bridge concepts, design and construction.

The advisor group was formed to participate in, review and comment on the development of data exchange protocols for bridges, and the associated viewer software presented herein. This group included members from National Steel Bridge Alliance (NSBA), State Department of Transportation, consulting firms, fabrication firms, contractors, and software companies. Additional participation involved individuals involved in BIM Committees (PCI, ACI), AASHTO (HSCOBS and Steel Bridge Collaboration), the buildingSMART Alliance, various BIM/BrIM workshop participants, etc.

The group has held numerous physical meetings and conference calls. Several meetings were to define and review the contents of the Information Delivery Manual. Multiple meetings and conference calls were held to discuss the development of the Model View Definition, data schema and viewer software. Earlier draft Information Delivery Manual and Model View Definition contents were posted on the NSBA Google website for review and comment.

The recent proliferation of digital data exchange standards development efforts in concrete and steel bridge related industries and standards bodies has significantly complicated the prospect of shepherding the process of leveraging and complementing those efforts in the best interests of bridge industry stakeholders. What would be the appropriate shepherding organization(s)? It is recommended that an entity such as PCI, ACI, AISC, NSBA, etc. not be the sole organization chosen, considering each has particular sector biases, though each sector should have their interests represented. Then what driving organization should be chosen, buildingSMART? If buildingSMART is selected, then what kind of memorandum of understanding (MoU) should be developed with AASHTO? If only AASHTO is chosen, should AASHTO's ASIS (Subcommittee on Information Systems) branch, the Construction branch, or the Highway Subcommittee on Bridges and Structures (HSCOBS) branch be chosen? These are not simple questions to resolve.

In planning, it is recommended that the formal meta-level processes and corresponding Working Group(s) be instituted with appropriate bridge industry, stakeholder representation to facilitate the further development/refinement, piloting, promulgation, extension, and maintenance of the openBrIM bridge lifecycle data exchange standards presented herein. Recommendations, with additional details, are contained in Volume I, entitled "Implementation Roadmap for Bridge Information Modeling (BrIM) Data Exchange Protocols," for this type of an implementation plan.

(This Page is Intentionally Left Blank)

3 MVD OVERVIEW

3.1 Introductions to Candidate Data Schemas

After development of the Information Delivery Manual (IDM) [1], which captures the use cases and the project data to be exchanged using the terminology of domain experts, IDM content will be translated into Model View Definition (MVD, subsets of a data schema) for software development, based on a suitable data schema selected. For BIM standard projects, such as the PCI NBIMS project, the data schema selected to map the IDM is to be the Industry Foundation Classes (IFC) [7], which has been implemented on over a hundred software applications. For BrIM related standard projects, such as NCHRP project 20-64 that developed TransXML, Geography Markup Language (GML) [9] was chosen as the basis for schema development. In 2013, an open and neutral Markup Language, OpenBrIM XML [16] was proposed for BrIM projects. It is believed to be a viable alternative to IFC for software vendors because of its unique Object- Oriented features.

In this section, IFC, GML and OpenBrIM XML are carefully investigated and compared to determine which data schema should be selected as the basis for mapping the companion BrIM IDM. Sections 3.1.1 to 3.1.3 introduce the three schemas and their pros and cons. Section 3.2 shows an experiment conducted for comparison.

In this paper, the terms “data schema” and “project data file” are used with the following meaning:

- A *data schema* is a collection of entities, attributes, and relationships between entities. It defines the templates by which populations of these entities, attributes and relationships shall be represented [20], [21], [24].

- A project data file is a population of a data schema, following the templates defined. It contains the instances of the entities, attributes and relationships [20], [21], [24].

3.1.1 Industry Foundation Classes

The Industry Foundation Classes (IFC) is a neutral data schema for Building Information Modeling data that is exchanged throughout the AEC/FM project life cycle, among disciplines and across software applications [7]. IFC was registered with the International Standardization Organization (ISO) as ISO 16739:2013. It is developed and maintained by the buildingSMART International, which is formally known as International Alliance for Interoperability (IAI). IFC consists of a master EXPRESS STEP based IFC definition, an accompanying XML schema (XSD) definition called ifcXML, and the HTML based online documentation of the EXPRESS schema.

Development of the Industry Foundation Classes (IFC) was undertaken to develop a set of consistent data representations of building information for exchange between Architecture, Engineering, and Construction (AEC) software applications. It relies on the ISO-STEP EXPRESS language and concepts, with a few minor variations. While many of the ISO-STEP efforts focused on detailed exchanges between different engineering software applications, it was thought that such a piecemeal effort would lead to a set of incompatible standards in the AEC. Instead, IFC was designed as an extensible framework model. That is, its initial developers intended it to provide broad general definitions of objects and data from which more detailed and task-specific models could be defined. In this regard, IFC has been directed to address all building information, over the whole building lifecycle, from feasibility and planning, through design (including analysis and simulation), construction, to occupancy and operation. Recently, IFC data schema has been adopted as the starting point for development of National Building Information Modeling

Standard (NBIMS) for precast/prestressed concrete and by American Concrete Institute (ACI) for cast-in-place concrete data schemas.

IFC is a data schema designed specifically to handle the exchanges between project stakeholders in the buildings industry. The modeling of a structure starts with laying out a grid system, which is commonly used in the buildings industry to define the location of members in space, but it is completely different from the approach used in the bridge industry. In the bridge industry, the highway alignment is used as the main reference point to define the location of members (structural and non-structural) in space. Failure to provide a roadway alignment based coordinate system is one of the shortcomings of this data schema for bridges.

Superstructure bridge properties are also different to a certain degree. Bridge girders can be larger and different in cross-sectional shape than the ones in buildings, and there needs to be a library that will include the cross-sections used in bridges. Gap analysis of IFC data schema shows that such a library that should be a part of a data schema is missing from IFC. Items such as diaphragm, curb, haunch, etc. are specific to bridges and cannot be found in IFC data schema either. Thus, there are significant gaps in the schema regarding usage of the schema for bridges.

Similarly, the loading information needed for design and analysis of bridges is different from that of buildings. Thus, the IFC data schema lacks not only the geometric aspects of modeling the bridge but also the loading information required to design bridges.

A major part of the bridge life cycle includes the maintenance, analysis, and rehabilitation of bridges. Due to the lack of various bridge related items in the IFC data schema, that are either related to geometry, modeling or design and analysis, IFC data schema cannot be used for maintenance of a bridge.

A tally of ACI's approach, as of this writing, towards important items for cast-in-place concrete data schema and BrIM Group data schema is shown in Table 3-1.

Table 3-1. BrIM Group Data Schema Vs ACI Data Schema

Section Number	Important Items	ACI handling	BrIM Group Data Schema
1	overlapping concrete	Not needed	Not Present
2	common rebars	Not needed	Not Present
3	beam bars in slab	should be attached to beam	Present
4	mat foundations	ifcfooting	Present
5	slab on grade	not clear yet	Not Present
6	finish	ifcCovering	Not Present
7	Tolerances	considered as "finish"	Not Present
8	standard tolerances	covered outside the model	Not Present
9	Non-standard tolerances	should be in model	Not Present
10	stirrups	handled with other rebar	Present
11	Chairs	ifcDiscreteAccessory	Not Present
12	support bars	ifcDiscreteAccessory	Not Present

Section Number	Important Items	ACI handling	BrIM Group Data Schema
13	standees	ifcDiscreteAccessory	Not Present
14	Anchorage slip	Not needed	Not Present
15	elongation	needed for tendons	Not Present
16	couplet effect on bar lengths	needed	Not Present
17	studded shear reinforcement	ifcDiscreteAccessory	Not Present
18	Splice type/info	Needed	Present
19	tag bar size and spacing	IfcRebar PSET	Present
20	Design intent	Design Intent PSET	Not Present
21	Welds on embeds	Not needed	Not Present
22	bent bars	Placing radius shown	Present
23	site issues	design model	Not Present
24	sequence of pour	Needed	Not Present
25	Schedule	Not needed	Not Present
26	Pour break face finish	standard case	Not Present
27	Pour break face finish	Exceptional case	Not Present
28	split items in multiple pours	member geometry required	Not Present
29	pour break	ifcPourBreak not needed (use ifcpour)	Not Present
30	bar placement sequence	Not needed	Not Present
31	Finish surfaces	IfcCovering	Not Present
32	Default surface finish	Needed	Not Present
33	sequence of bar placement	Low priority	Not Present
34	caissons	not supported in IFC, IfcPile can be used	Present
35	caissons	Special caisson PSET (?)	Present
36	caissons bar length	Needed	Present
37	Grids needed	not sure yet	Not Present
38	Location reference point	Grids intersection can be used	Present
39	Bar Splice	Project standards	Present
40	Bar Splice	use override in PSET for special case	Present
41	Keyways	PSET (similar to surface finish)	Not Present
42	Concrete on steel deck	describe with PSET (?)	Not Present
43	deck profile	needs to be noted	Present
44	beam bar cutoffs	Needed	Not Present

Section Number	Important Items	ACI handling	BrIM Group Data Schema
45	metal deck bar cutoffs	Needed	Not Present
46	slab bar cutoffs	Needed	Not Present
47	set of bars	IfcRebar should be used	Present
48	Geometry of bars	Needed	Present
49	Bend number of bars	Needed	Present
50	Parameters of bars	Needed	Present
51	Bar Shapes	BVBS shapes (library can be used)	Present
52	Properties of bars after fabrication	Needed	Not Present
53	Bar Standards	Not sure yet	Not Present
54	NRC details	Not needed	Not Present
55	Coating	property of bar	Present
56	Non-metallic bars	Needed	Not Present
57	Mesh Sheet type	Needed	Not Present
58	Mesh Engineered sheets	Very low priority	Not Present
59	MeshEngineered rolls	Very low priority	Not Present
60	Post-tensioning	Low priority (postpone till rebar is done)	Present
61	studded shear reinforcement	Do it parametrically	Not Present
62	Composite columns	Not sure yet	Not Present
63	Openings in structural elements	Design phase	Not Present

3.1.1.1 EXPRESS STEP based IFC definition

The Standard for the Exchange of Product (STEP) data schema was registered with the ISO as ISO 10303. Its development has been undertaken by ISO to address the electronic exchange of product data between computer-based product life-cycle systems. STEP is widely used in computer-aided design (CAD) and product data/lifecycle management (PDM/PLM) systems. Major aerospace, automotive, and shipbuilding companies have proven the value of STEP through production implementations resulting in actual savings of $150M per year in the US (and potential savings of $928M per year) [27].

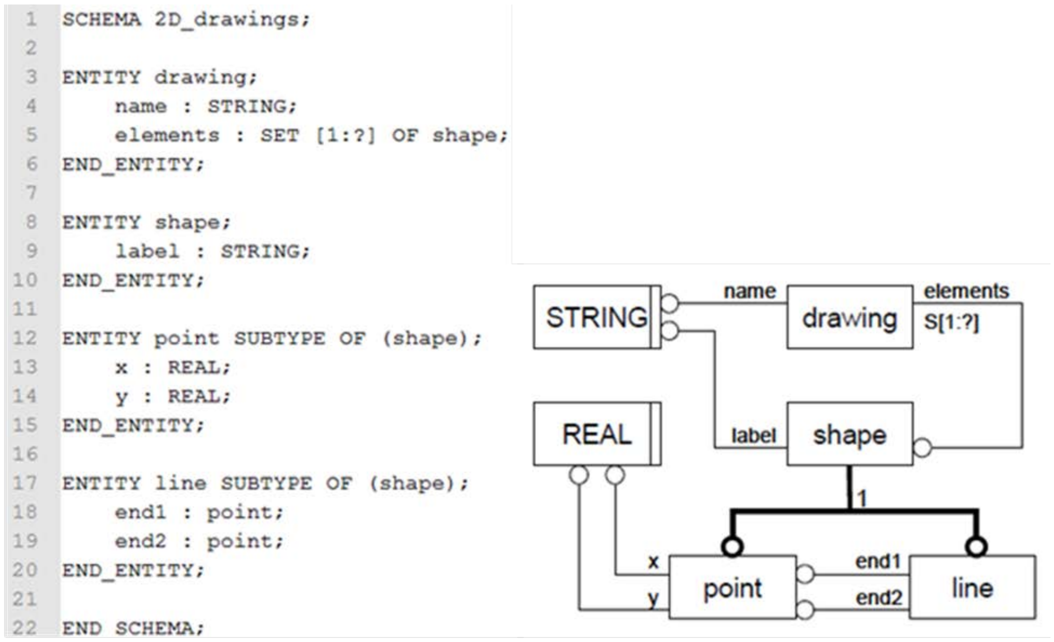
STEP is specified in a data modeling language called EXPRESS, which is defined in ISO 10303-11. EXPRESS language combines concepts from entity-attribute-relationship modeling language and ideas from Object-Oriented Programming (OOP). It can define a data schema in two ways, textually and graphically. The textual representation is written in an ASCII STEP file (ISO 10303-21). It is used for data input and product model verification. The graphical representation, called EXPRESS-G, is used as explanation of product models for domain experts.

The following example is a schema written in EXPRESS that represents 2D drawings. It demonstrates some EXPRESS language features (Figure 3-1):

- EXPRESS contains simple data types, such as STRING and REAL.

- EXPRESS contains multiple ways of repetition of entities, such as SET, BAG, and LIST.

- EXPRESS has several OOP features, such as inheritance. In this example schema, point and line are both inheritances of shape.



```
SCHEMA 2D_drawings;

ENTITY drawing;
    name : STRING;
    elements : SET [1:?] OF shape;
END_ENTITY;

ENTITY shape;
    label : STRING;
END_ENTITY;

ENTITY point SUBTYPE OF (shape);
    x : REAL;
    y : REAL;
END_ENTITY;

ENTITY line SUBTYPE OF (shape);
    end1 : point;
    end2 : point;
END_ENTITY;

END SCHEMA;
```

Figure 3-1. EXPRESS and EXPRESS-G Diagram for 2D_drawings Schema [7]

Similar to SQL and other database languages, EXPRESS is able to define various constraints, such as WHERE constraint, as shown in Code 3-1.

```
ENTITY point_on_hyperbola SUBTYPE OF (point);
    WHERE
        hyperbola : y = 1 / x;
END_ENTITY
```

Code 3-1 EXPRESS `WHERE` Constraint

Other than in numerical equations, the "WHERE constraint" can define the complex control flow, including logical judgments and value judgments.

The master IFC definition, as an application of STEP and EXPRESS, is a data schema specified in the EXPRESS language. It has the file extension *.exp. IFC definition provides templates for populating an instance file, the IFC project data file.

IFC project data files are text files following the ASCII STEP file format (ISO 10303-21). Such a file has the file extension *.ifc. Each IFC project data file can be divided into a header section and a data section. The header section contains the following information:

- The IFC version.
- The application that exported this file.

- The date and time when the export was done.
- The name, company and authorizing person of this file.

Code 3-2 shows an example of a header section in an IFC project data file.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION (('ViewDefinition [CoordinationView,
QuantityTakeOffAddOnView]'), '2;1');
FILE_NAME ('example.ifc',
            '2013-11-24T15:52:37',
            ('Architect'),
            ('Building Designer Office'),
            'IFC Engine DLL version 1.02 beta',
            'IFC Engine DLL version 1.02 beta', 'The authorising
            person');
FILE_SCHEMA (('IFC2X3'));
ENDSEC;
```

Code 3-2 Header Section in IFC Project Data File

The data section contains instances of the entities, attributes and relationships that the *.exp file defines. The instances have a unique id (hash number), an entity name and a list of attributes, as shown in Code 3-3.

```
DATA;
#1 = IFCPROJECT('194HFvjwvDlu99NKlj7HuT', #2, 'Default Project',
'Description of Default Project', $, $, $, (#20), #7);
#2 = IFCOWNERHISTORY(#3, #6, $, .ADDED., $, $, $, 1385326357);
#3 = IFCPERSONANDORGANIZATION(#4, #5, $);
#4 = IFCPERSON('ID001', 'Bonsma', 'Peter', $, $, $, $, $);
#5 = IFCORGANIZATION($, 'TNO', 'TNO Building Innovation', $, $);
#6 = IFCAPPLICATION(#5, '0.10', 'Test Application', 'TA 1001');
#7 = IFCUNITASSIGNMENT((#8, #9, #10, #11, #15, #16, #17, #18, #19));
#8 = IFCSIUNIT(*, .LENGTHUNIT., .MILLI., .METRE.);
#9 = IFCSIUNIT(*, .AREAUNIT., $, .SQUARE_METRE.);
#10 = IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
ENDSEC;
```

Code 3-3 Data Section in IFC Project Data File

Although STEP is widely used in CAD and PDM/PLM systems, some drawbacks hinder its application. Firstly, EXPRESS language is not well known by most programmers, because it is a relatively “old” language. Secondly, the syntax of EXPRESS language lacks extensibility and is difficult for humans to read. Lastly, only a few software applications support EXPRESS [27].

3.1.1.2 ifcXML

In order to widen the application of IFC data schemas, buildingSMART International developed and maintains an eXtensible Markup Language (XML) based IFC definition. This is because XML has a larger range of supporting utilities and databases, and is the basis for web services [24]. Besides, XML can be converted to human readable file by applying eXtensible Stylesheet Language Transformations (XSLT) or Document Type Definitions (DTD).

XML schema based IFC definition (*.xsd), also known as ifcXML schema, was then developed as an alternative to the EXPRESS STEP based IFC definition (*.exp). An ifcXML file (*.xml), which is an instance file of the ifcXML schema, can be created as an alternative to the IFC project data file (*.ifc). Figure 3-2 shows the relationships between them.

Besides ifcXML schema and ifcXML file, the content of ifcXML also includes an ifcXML methodology, which is used to convert the *.exp documents to *.xsd document, or convert *.ifc file to *.xml file, as shown as a notched right arrows in Figure 3-2.

Although ifcXML might be of interest to a wider community, its file size hinders its applications. An ifcXML file usually is 2 to 10 times larger than the corresponding *.ifc file. For a large project, the size of *.ifc can reach dozens of megabytes. Thus, the size of the corresponding ifcXML file can exceed a hundred megabytes. This is unacceptable for software vendors and end users. Therefore, buildingSMART International tends to limit the use of ifcXML to document based representations (e.g., quantity takeoff), message based representations (e.g., Requests-For- Information), and communication with XML based domains (e.g., GIS models).

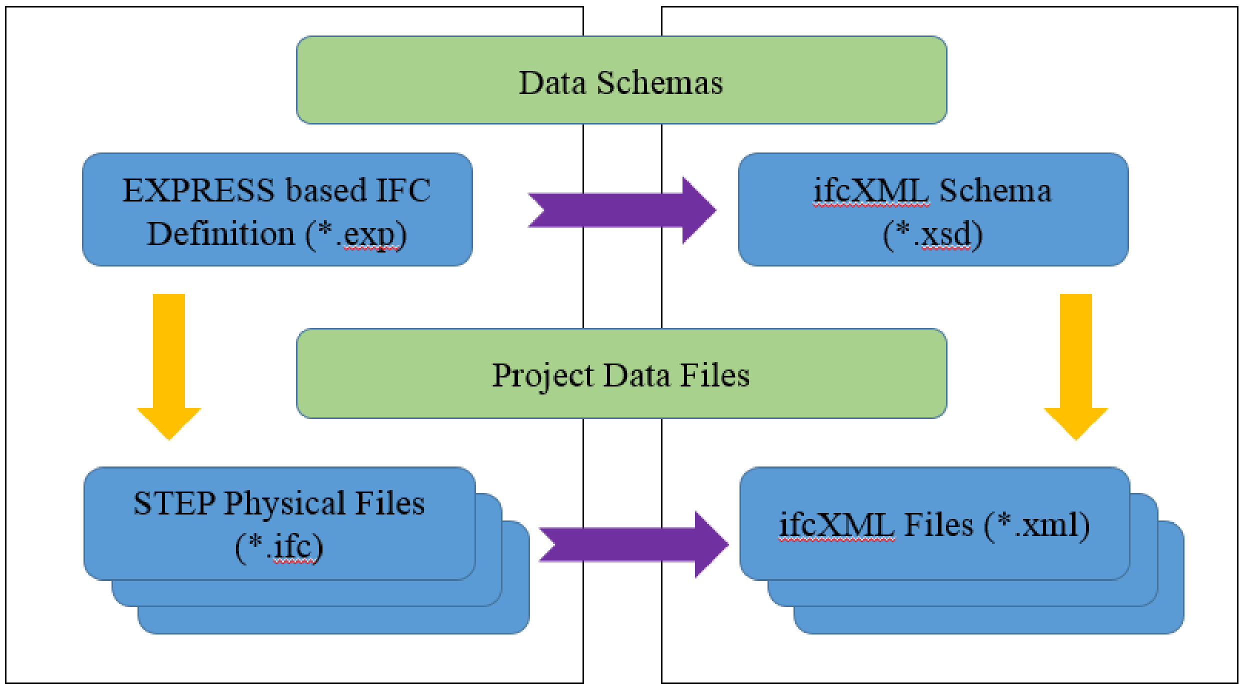


Figure 3-2. Data schema and Project Data File Differentiation for EXPRESS and XML

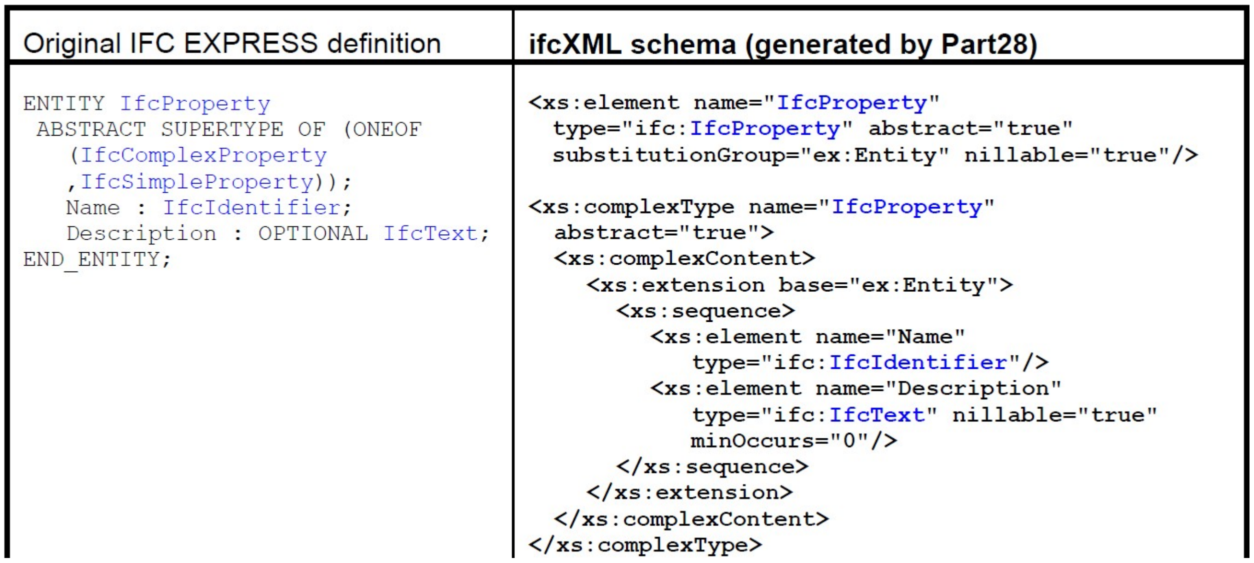
3.1.1.3 Relationship between EXPRESS STEP based IFC and ifcXML

The method to convert EXPRESS based IFC definitions to ifcXML schema is developed under ISO TC184/SC4 as Part 28 edition 2 [24]. The rules of conversion are:

- Semantics of ifcXML schema should conform to the original EXPRESS based IFC definition.

- Syntax of the ifcXML schema can be adjusted accordingly.

Figure 3-3 shows an example conversion of an IFC EXPRESS definition.



Original IFC EXPRESS definition

```
ENTITY IfcProperty
 ABSTRACT SUPERTYPE OF (ONEOF
   (IfcComplexProperty
   ,IfcSimpleProperty));
   Name : IfcIdentifier;
   Description : OPTIONAL IfcText;
END_ENTITY;
```

ifcXML schema (generated by Part28)

```
<xs:element name="IfcProperty"
  type="ifc:IfcProperty" abstract="true"
  substitutionGroup="ex:Entity" nillable="true"/>

<xs:complexType name="IfcProperty"
  abstract="true">
  <xs:complexContent>
    <xs:extension base="ex:Entity">
      <xs:sequence>
        <xs:element name="Name"
          type="ifc:IfcIdentifier"/>
        <xs:element name="Description"
          type="ifc:IfcText" nillable="true"
          minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figure 3-3. Example Conversion of an IFC EXPRESS definition

Table 3-2 shows a detailed mapping between IFC EXPRESS definition and ifcXML schema based on Figure 3-3.

Table 3-2. Detailed Mapping between IFC EXPRESS Definition and ifcXML Schema

IFC EXPRESS Definition	ifcXML Schema
ENTITY	xs:element
ABSTRACT	Abstract=”true”
Name : IfcIdentifier;	<xs:element name=”Name” type=”ifc:IfcIdentifier”/>
Description : IfcText;	<xs:element name=”Description” type=”ifc:IfcText”
OPTIONAL	Nillable=”true” minOccurs=”0”

3.1.2 Geography Markup Language (GML)

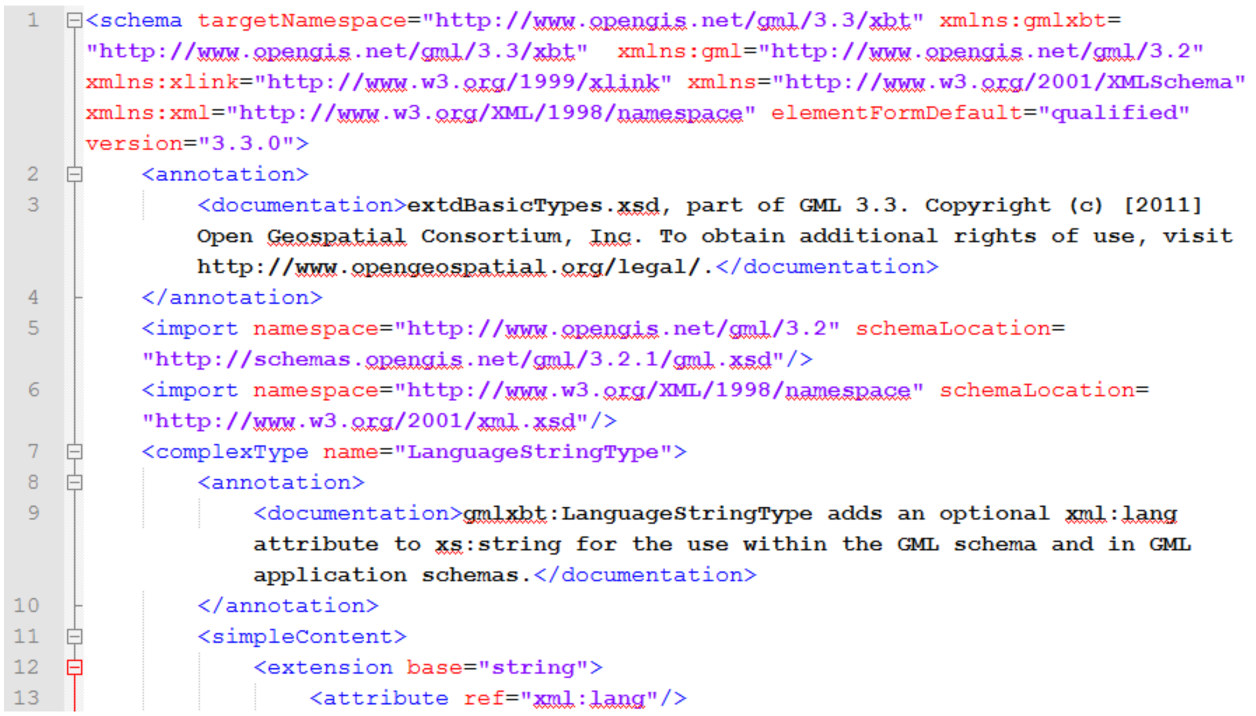
3.1.2.1 Introduction to GML

The Geography Markup Language (GML) is an XML encoding for the exchange and storage of geographic information and the spatial and non-spatial properties of geographic features [26]. It is developed and maintained by the Open Geospatial Consortium (OGC). GML was initially used for users and developers to describe generic geographic data sets that contain points, lines and polygons. However, the developers of GML envision communities extending its use to roads, highways and bridges [26]. For instance, GML can be used to provide multiple representations of a roadway by describing its functional classification, its location in space and along a longer route, and its horizontal and vertical alignments – all of which are, of course, relevant to bridges.

Similar to other XML based data schemas, GML is composed of two parts – the GML schema document (*.xsd) and GML instance files (*.xml). The latest version of the GML schema is version 3.3, which was published in 2012. This schema consists of nine schema (XSD) documents:

- Extended basic types: language, code, date, etc. Extended encoding rule.
- Geometry compact: simple polygon, simple rectangle, simple triangle, simple arc string, simple circle, etc.
- Linear reference: position expression, linear element type, etc.
- Linear reference offset: lateral offset distance expression, vertical offset distance expression, etc.
- Linear reference offset vector: vector offset distance expression, vector offset expression, etc. Linear reference towards referent: dual along referent type, dual along referent, etc.
- Referenceable grid: referenceable grid property, referenceable grid by array, referenceable grid by vector, etc.
- TIN: trangulated surface, TIN element, etc.

Figure 3-4 shows a portion of the GML schema document.



```
<schema targetNamespace="http://www.opengis.net/gml/3.3/xbt" xmlns:gmlxbt=
"http://www.opengis.net/gml/3.3/xbt"  xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xml="http://www.w3.org/XML/1998/namespace" elementFormDefault="qualified"
version="3.3.0">
    <annotation>
        <documentation>extdBasicTypes.xsd, part of GML 3.3. Copyright (c) [2011]
        Open Geospatial Consortium, Inc. To obtain additional rights of use, visit
        http://www.opengeospatial.org/legal/.</documentation>
    </annotation>
    <import namespace="http://www.opengis.net/gml/3.2" schemaLocation=
    "http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>
    <import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation=
    "http://www.w3.org/2001/xml.xsd"/>
    <complexType name="LanguageStringType">
        <annotation>
            <documentation>gmlxbt:LanguageStringType adds an optional xml:lang
            attribute to xs:string for the use within the GML schema and in GML
            application schemas.</documentation>
        </annotation>
        <simpleContent>
            <extension base="string">
                <attribute ref="xml:lang"/>
```

Figure 3-4. Portion of GML Schema Document

3.1.2.2 Data Schemas Developed Based on GML

GML was selected as the basis for developing data schemas including TransXML [23] and Steel Bridge Construction Design Model (SBCDM) [19]. The following guidelines were followed [23]:

- The data schemas must conform to the W3C XML Schema Recommendations [36].
- The data schemas try to avoid duplication of existing elements of related schema that are widely used.
- The most appropriate and restrictive data types are used for entity types and attributes.
- Names of elements and attributes are self-explanatory and meaningful to domain experts.
- Naming conventions – Upper Camel Case is used for schema elements and data types, while lower Camel Case is used for properties. No abbreviation is used in names of entities and attributes unless the abbreviations are widely understood. Schema component names do not contain Underscores (_), Periods (.), and Dashes (-).

During the development of TransXML and SBCDM, because GML did not provide schema for tangible features (e.g. bridge), several bridge-oriented data schemas were used, such as AASHTO Opis/Virtis Database [1].

3.1.3 OpenBrIM

3.1.3.1 Introduction to OpenBrIM

The OpenBrIM is intended to be a community driven Bridge Information Modeling (BrIM) product. It includes an XML encoding schema which describes BrIM information and the information related to bridge projects, and a non-commercial 3D viewer software application, which visually renders a BrIM model defined in OpenBrIM XML files. The OpenBrIM is developed and maintained by Red Equation Corp. [16]. The version described herein is version 2, which was released in 2013.

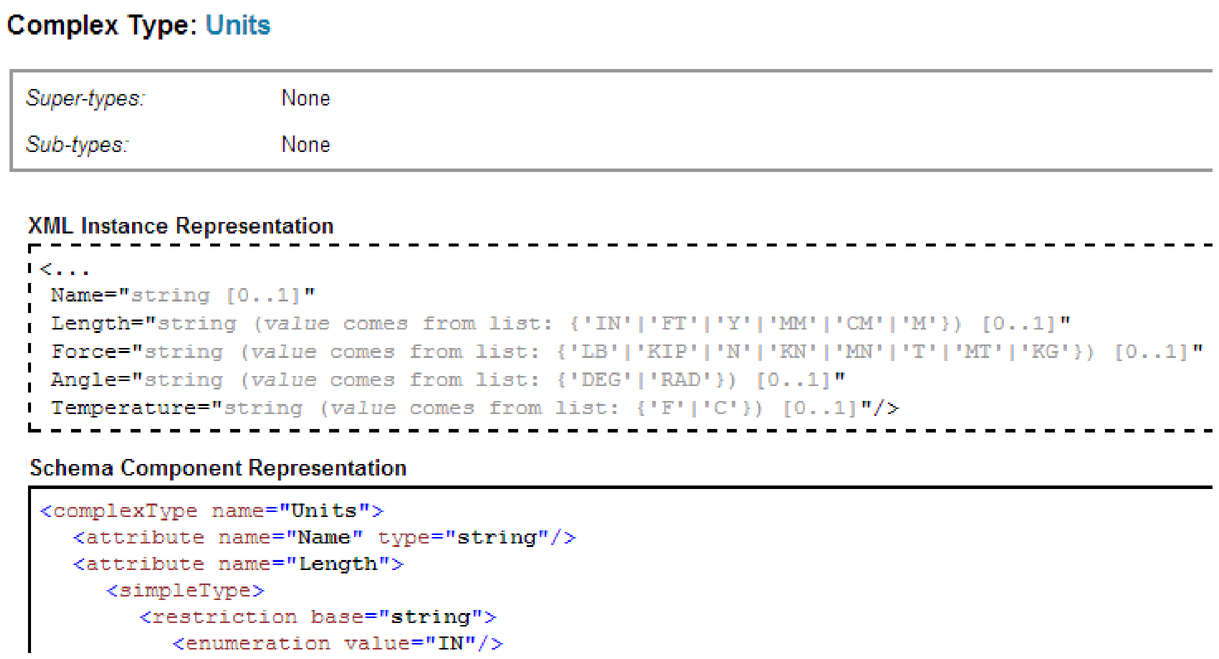
The OpenBrIM schema contains the following schema documents (XSD):

- Units: length, force, angle, temperature, etc.
- Roadway geometry: horizontal alignment, etc.
- Shapes: point, line, surface, circle, and volume
- Other: obj(ect), parameter and repeat.

The OpenBrIM schema is relatively flexible because any entity or attribute other than units, roadway geometry and basic shapes can be defined by obj(ect) and parameter. Figure 3-5 shows a portion of the OpenBrIM schema document.

In order to aid the schema development and testing of BrIM standards implementation for case study bridges, a non-commercial 3D model viewer (OpenBrIM Viewer) was developed. The basic function of OpenBrIM Viewer is to parse the bridge project XML file conforming to the BrIM standards and visually render the BrIM objects contained in the XML file.

The OpenBrIM Viewer uses OpenGL for rendering computer graphics. The graphics and geometric engine of this software has been developed to render BrIM objects as extrusion solids and B-rep solids, and implement horizontal and vertical curves of various types.



Complex Type: Units

Super-types:	None
Sub-types:	None

XML Instance Representation

```
<...
 Name="string [0..1]"
 Length="string (value comes from list: {'IN'|'FT'|'Y'|'MM'|'CM'|'M'}) [0..1]"
 Force="string (value comes from list: {'LB'|'KIP'|'N'|'KN'|'MN'|'T'|'MT'|'KG'}) [0..1]"
 Angle="string (value comes from list: {'DEG'|'RAD'}) [0..1]"
 Temperature="string (value comes from list: {'F'|'C'}) [0..1]"/>
```

Schema Component Representation

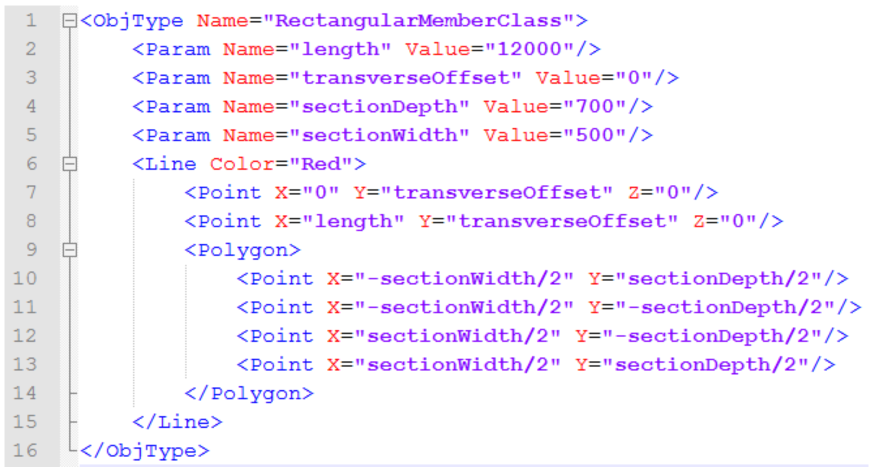
```
<complexType name="Units">
  <attribute name="Name" type="string"/>
  <attribute name="Length">
    <simpleType>
      <restriction base="string">
        <enumeration value="IN"/>
```

Figure 3-5. Portion of the OpenBrIM Schema Document

3.1.3.2 Object-Oriented Programming Features of OpenBrIM Schema

The OpenBrIM schema was developed based on the concept of Object-Oriented Programming (OOP). It treats an object as a fundamental building block of OpenBrIM XML files. Object is an instance of class. It usually has data fields (e.g., attributes that describe the object) and associated methods.

Class is represented by ObjType node in the OpenBrIM schema, as shown in Code 3-4.



```
<ObjType Name="RectangularMemberClass">
    <Param Name="length" Value="12000"/>
    <Param Name="transverseOffset" Value="0"/>
    <Param Name="sectionDepth" Value="700"/>
    <Param Name="sectionWidth" Value="500"/>
    <Line Color="Red">
        <Point X="0" Y="transverseOffset" Z="0"/>
        <Point X="length" Y="transverseOffset" Z="0"/>
        <Polygon>
            <Point X="-sectionWidth/2" Y="sectionDepth/2"/>
            <Point X="-sectionWidth/2" Y="-sectionDepth/2"/>
            <Point X="sectionWidth/2" Y="-sectionDepth/2"/>
            <Point X="sectionWidth/2" Y="sectionDepth/2"/>
        </Polygon>
    </Line>
</ObjType>
```

Code 3-4 Definition of Class in OpenBrIM Schema

In this code, RectangularMemberClass defines a collection of rectangular members by defining their attributes (e.g., length, section width, etc.) and a method (i.e., extrude rectangular section along the length). The attributes have their default values, which can be used by the objects of this class.

Object, as an instance of a class, is represented by the Obj node in the OpenBrIM schema, as shown in Code 3-5.

```
<Obj Name="RectObj1" RefObj="RectangularMemberClass">
    <Param Name="length" Value="13500"/>
    <Param Name="transverseOffset" Value="500"/>
    <Param Name="sectionDepth" Value="800"/>
    <Param Name="sectionWidth" Value="600"/>
</Obj>
```

Code 3-5 Definition of Object in OpenBrIM Schema

In the above code, RefObj=”RectangularMemberClass” means object RectObj1 is an instance of class RectangularMemberClass. User can use default values for the attributes, or can set different values to overwrite the default ones.

Object-Oriented Programing has the following three characteristics. They are well represented in the OpenBrIM schema:

- Encapsulation. The attributes and method of class RectangularMemberClass are encapsulated in one ObjType node. They can only be used by objects of that class or its subclasses.

- Inheritance. A new class that inherits attributes and method from its superclass can be defined, as shown in Code 3-6. In the code, class TrapezoidMemberClass inherits the attributes of its superclass RectangularMemberClass. It has its own attributes sectionTopWidth and sectionBotWidth. In addition, its method is rewritten.

- Polymorphism. In Code 3-6, class TrapezoidMemberClass has a method defined in the Line node. In Code 3-5, the superclass of TrapezoidMemberClass, class RectangularMemberClass also has a method defined in the Line node. However, these two methods are different. Thus, when an object of either of these two classes calls the method, only the corresponding method will respond.

Objects form the core of the BrIM data schema. Any component of a bridge can be defined using objects. Objects consist of various parameters that contain the data to define the object. Model view of objects is defined using points, lines, surfaces, extruded solids and volumes (breps).

```
<ObjType Name="TrapezoidMemberClass" RefObj="RectangularMemberClass">
    <Param Name="sectionTopWidth" Value="500"/>
    <Param Name="sectionBotWidth" Value="800"/>
    <Line Color="Red">
        <Point X="0" Y="transverseOffset" Z="0"/>
        <Point X="length" Y="transverseOffset" Z="0"/>
        <Polygon>
            <Point X="-sectionTopWidth/2" Y="sectionDepth/2"/>
            <Point X="-sectionBotWidth/2" Y="-sectionDepth/2"/>
            <Point X="sectionBotWidth/2" Y="-sectionDepth/2"/>
            <Point X="sectionTopWidth/2" Y="sectionDepth/2"/>
        </Polygon>
    </Line>
</ObjType>
```

Code 3-6 Definition of a Subclass in OpenBrIM Schema

Fundamental concepts of BrIM data schema are as follows:

- Instantiation. Objects shall be instantiated from templates. This allows for sharing and reuse of model view definitions of various elements in the bridge across different projects.

- Hierarchy. Objects can be defined inside other objects and follow the hierarchy. Child objects can access the parameters of parent objects directly. Data access between two independent objects is possible using the dot operator.

- Inheritance. New templates can inherit definitions from existing templates to build upon. This allows reusing the already created templates by making necessary additions.

BrIM data schema components include the following:

- Obj.
- ObjType.
- Units.
- Param.
- Point.
- Line.
- Surface.
- Polygon.
- Circle.
- Volume.
- Repeat.
- RoadwayGeometry.
- HorizontalAlignment.
- Curve.
- Straight.
- VerticalProfile.
- ProfilePoint.

BrIM data exchange standards consist of two files:

- Project data file.
- Project template file.

Project Data File:

- A project data file contains the bridge project data structured as information and object nodes. Node hierarchy is shown in Figure 3-6a Project File Definition Diagram

- Information nodes contain entities such as project name, roadway geometry and global project parameters

- “Roadway Geometry” node defines the horizontal alignment, vertical profile and super-elevation

- Global parameters such as stations, elevations, etc. can be defined under the topmost object node that defines the project name.

- Object nodes define the physical entities such as deck, girder, pier, etc. Object node contains the definition of the physical entities as a collection of placement location, rotation, parameters and a reference "Object template."

Project Template File:

- A project template file contains the template object definitions that are referenced in project data file by "Object" instances.
- Every object instance in the project data file must refer to a corresponding type of template object in the project template file.
- Object templates are used to define parametric 3D extrusion and BREP solids. Node hierarchy of object template files is shown in Figure 3-6b Object Template Definition Diagram.

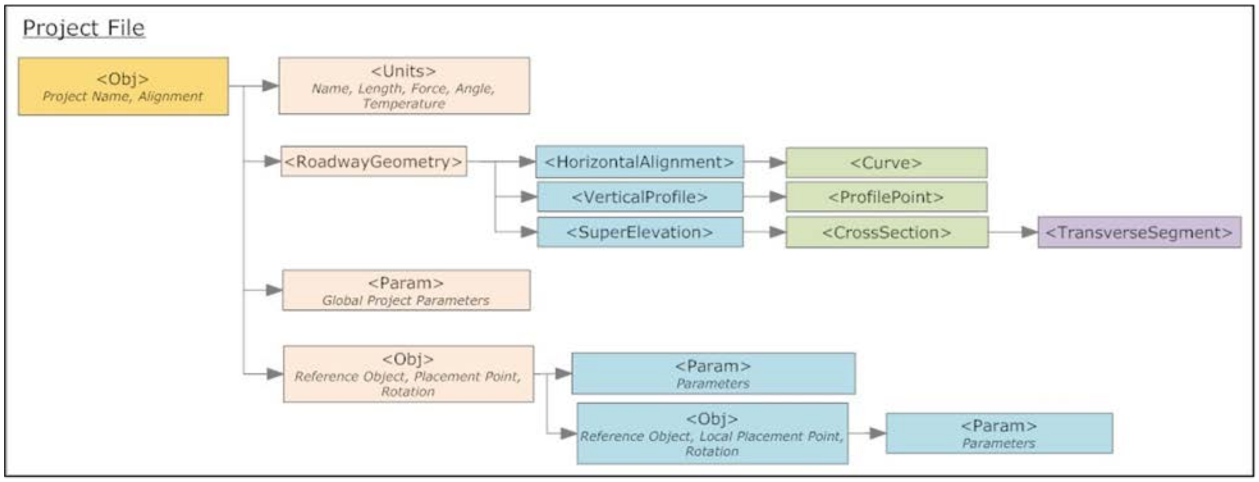

Project File
<Obj> Project Name, Alignment
<Units> Name, Length, Force, Angle, Temperature
<RoadwayGeometry>
<HorizontalAlignment>
<VerticalProfile>
<SuperElevation>
<Curve>
<ProfilePoint>
<CrossSection>
<TransverseSegment>
<Param> Global Project Parameters
<Obj> Reference Object, Placement Point, Rotation
<Param> Parameters
<Obj> Reference Object, Local Placement Point, Rotation
<Param> Parameters

Figure 3-6a. Project File Definition Diagram

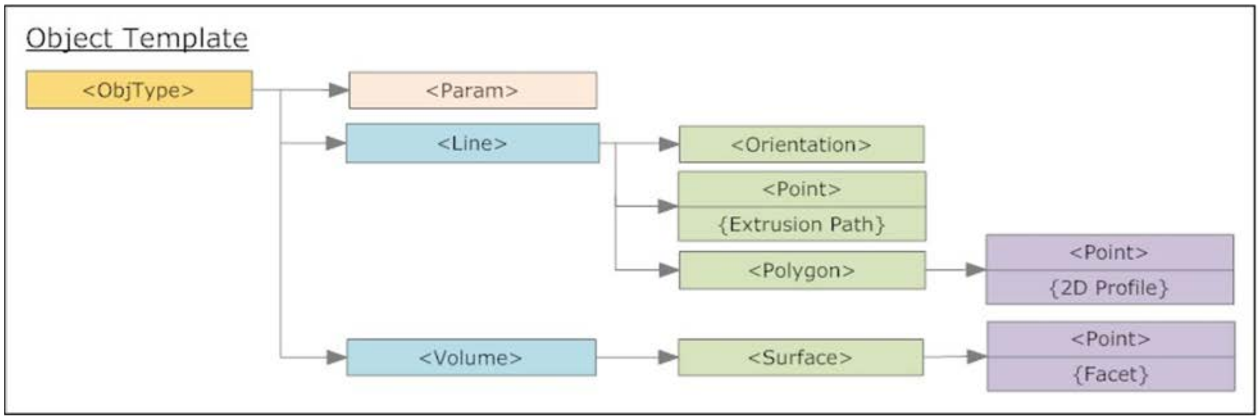


Figure 3-6b. Object Template Definition Diagram

3.2 Data Schema Experiments

In order to select the most suitable data schema for mapping IDM to MVD, two experiments were conducted to compare the three data schemas introduced in the previous section, i.e., IFC, GML and OpenBrIM XML. Contents of these experiments include the following aspects:

- Effort – the difference in effort required for developing BrIM MVD by using the three product models. The part of experiment focused on how many predefined entities, attributes and relations of the three models can be directly borrowed and used.

- File size – the difference in size of the instance project data files conforming the three data schemas. The part of experiment focused on whether the difference is large, and if so, what reasons would be.

- Clarity – the difference in clarity of the instance product data files conforming the three data schemas. The part of experiment focused on how complex these files were.

- Parsing – the difference in effort required to parse the instance product data files.

- Applicability – Whether there is an existing application that can parse the instance product data files and view the model in the files.

3.2.1 Experiment 1 – IFC vs. OpenBrIM XML

The first experiment was conducted to compare IFC and OpenBrIM XML. The structural member modeled was a wall with an opening. The thickness, height, and width of the wall are 300 mm, 2300 mm, and 5000 mm, respectively. The height and width of the opening are 1400 mm and 750 mm, respectively. The offset along the wall width is 900 mm, measured from the bottom left corner of the wall. The offset along the wall height is 250 mm, also measured from the bottom left corner of the wall.

In this experiment, IFC and OpenBrIM instance data files, which contain model of the wall were created separately. Figure 3-7 shows a portion of the IFC instance data file for the wall. Figure 3-8 shows a portion of the OpenBrIM XML instance data file for the wall.

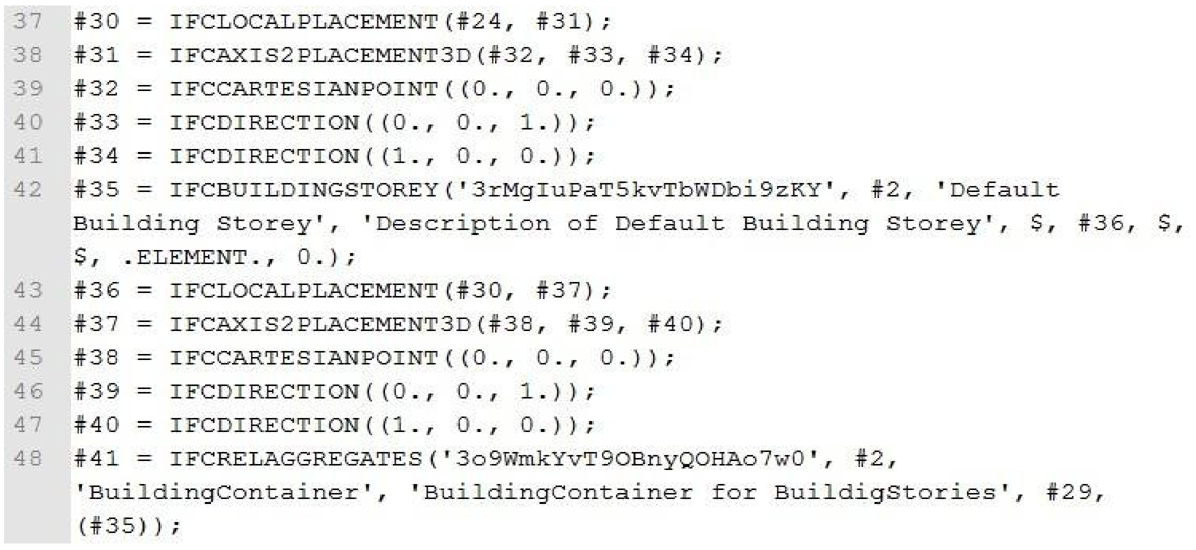
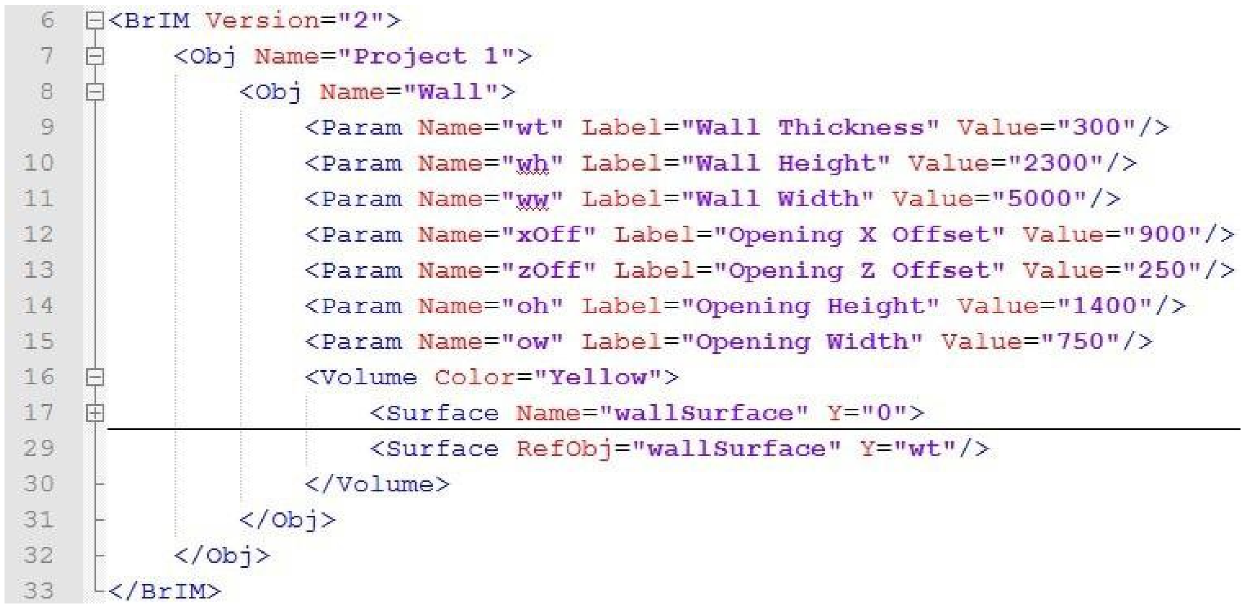

```
#30 = IFCLOCALPLACEMENT(#24, #31);
#31 = IFCAXIS2PLACEMENT3D(#32, #33, #34);
#32 = IFCCARTESIANPOINT((0., 0., 0.));
#33 = IFCDIRECTION((0., 0., 1.));
#34 = IFCDIRECTION((1., 0., 0.));
#35 = IFCBUILDINGSTOREY('3rMgIuPaT5kvTbWDbi9zKY', #2, 'Default
Building Storey', 'Description of Default Building Storey', $, #36, $,
$, .ELEMENT., 0.);
#36 = IFCLOCALPLACEMENT(#30, #37);
#37 = IFCAXIS2PLACEMENT3D(#38, #39, #40);
#38 = IFCCARTESIANPOINT((0., 0., 0.));
#39 = IFCDIRECTION((0., 0., 1.));
#40 = IFCDIRECTION((1., 0., 0.));
#41 = IFCRELAGGREGATES('3o9WmkYvT9OBnyQOHAo7w0', #2,
'BuildingContainer', 'BuildingContainer for BuildigStories', #29,
(#35));
```

Figure 3-7. Portion of the IFC Instance Data File for the Wall



```
<BrIM Version="2">
    <Obj Name="Project 1">
        <Obj Name="Wall">
            <Param Name="wt" Label="Wall Thickness" Value="300"/>
            <Param Name="wh" Label="Wall Height" Value="2300"/>
            <Param Name="ww" Label="Wall Width" Value="5000"/>
            <Param Name="xOff" Label="Opening X Offset" Value="900"/>
            <Param Name="zOff" Label="Opening Z Offset" Value="250"/>
            <Param Name="oh" Label="Opening Height" Value="1400"/>
            <Param Name="ow" Label="Opening Width" Value="750"/>
            <Volume Color="Yellow">
                <Surface Name="wallSurface" Y="0">
                <Surface RefObj="wallSurface" Y="wt"/>
            </Volume>
        </Obj>
    </Obj>
</BrIM>
```

Figure 3-8. Portion of the OpenBrIM XML Instance Data File for the Wall

The IFC instance data file was parsed and rendered by an IFC viewer called Solibri. The IFC model of the wall is shown in Figure 3-9.

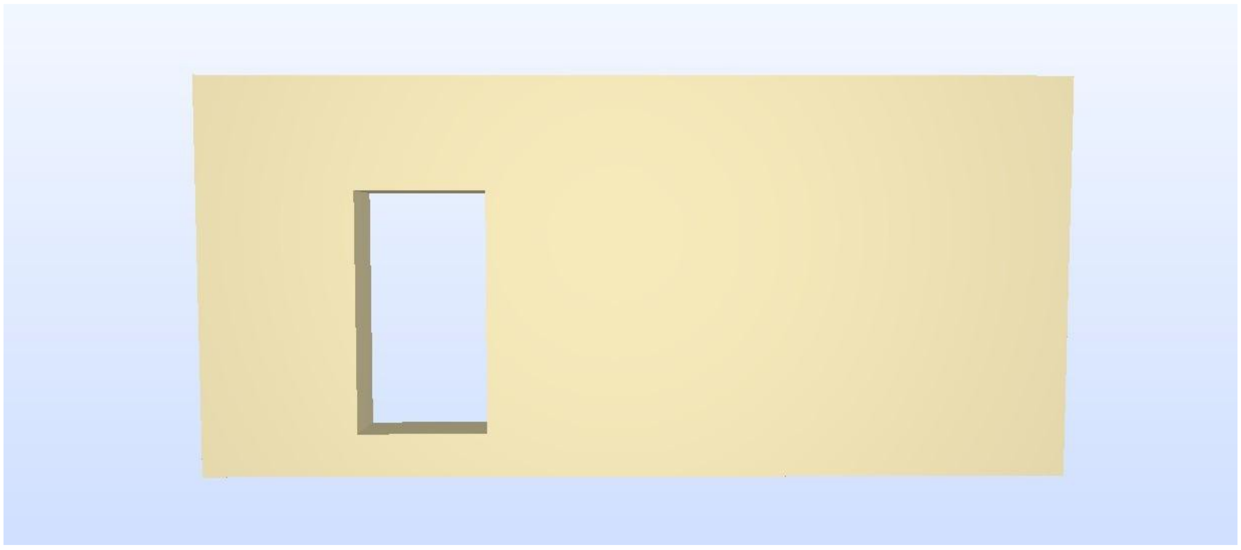


Figure 3-9. IFC Model of the Wall

The OpenBrIM XML instance data file was parsed and rendered by the OpenBrIM Viewer. The OpenBrIM model of the wall is shown in Figure 3-10.

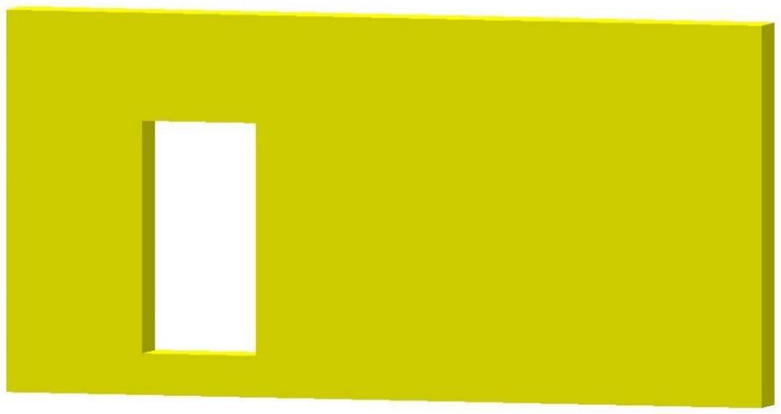
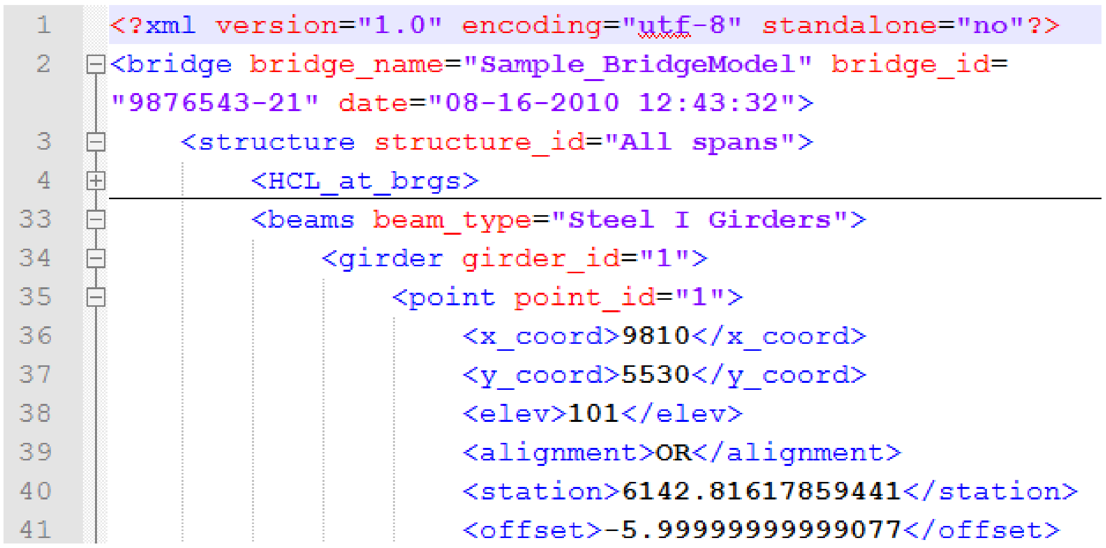


Figure 3-10. OpenBrIM Model of the Wall

3.2.2 Experiment 2 – GML vs. OpenBrIM XML

Another experiment was conducted to compare GML and OpenBrIM XML. The structural members modeled were six steel I girders. The thickness and depth of the girder web are 14 mm and 1100 mm, respectively. The thickness and width of the girder top flange are 20 mm and 450 mm, respectively. The thickness and width of the girder bottom flange are 25 mm and 500 mm, respectively.

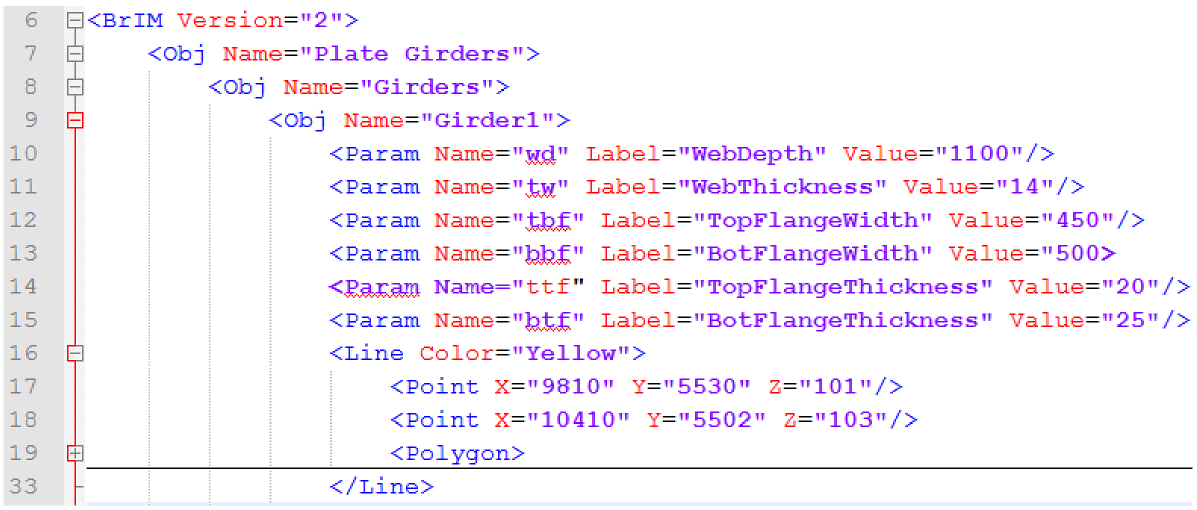
In this experiment, GML and OpenBrIM instance data files which contain models of the six steel I girders were created separately. Figure 3-11 shows a portion of the GML instance data file for the girders.



```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<bridge bridge_name="Sample_BridgeModel" bridge_id=
"9876543-21" date="08-16-2010 12:43:32">
    <structure structure_id="All spans">
        <HCL_at_brgs>
        <beams beam_type="Steel I Girders">
            <girder girder_id="1">
                <point point_id="1">
                    <x_coord>9810</x_coord>
                    <y_coord>5530</y_coord>
                    <elev>101</elev>
                    <alignment>OR</alignment>
                    <station>6142.81617859441</station>
                    <offset>-5.99999999999077</offset>
```

Figure 3-11. Portion of the GML Instance Data File for the Girders

Figure 3-12 shows a portion of the OpenBrIM XML instance data file for the girders.



```
<BrIM Version="2">
    <Obj Name="Plate Girders">
        <Obj Name="Girders">
            <Obj Name="Girder1">
                <Param Name="wd" Label="WebDepth" Value="1100"/>
                <Param Name="tw" Label="WebThickness" Value="14"/>
                <Param Name="tbf" Label="TopFlangeWidth" Value="450"/>
                <Param Name="bbf" Label="BotFlangeWidth" Value="500>
                <Param Name="ttf" Label="TopFlangeThickness" Value="20"/>
                <Param Name="btf" Label="BotFlangeThickness" Value="25"/>
                <Line Color="Yellow">
                    <Point X="9810" Y="5530" Z="101"/>
                    <Point X="10410" Y="5502" Z="103"/>
                    <Polygon>
                </Line>
```

Figure 3-12. Portion of the OpenBrIM XML Instance Data File for the Girders

The GML instance data file was parsed by a user-programed application, and rendered by SDS/2 [17]. The GML model of the girders are shown in Figure 3-13.

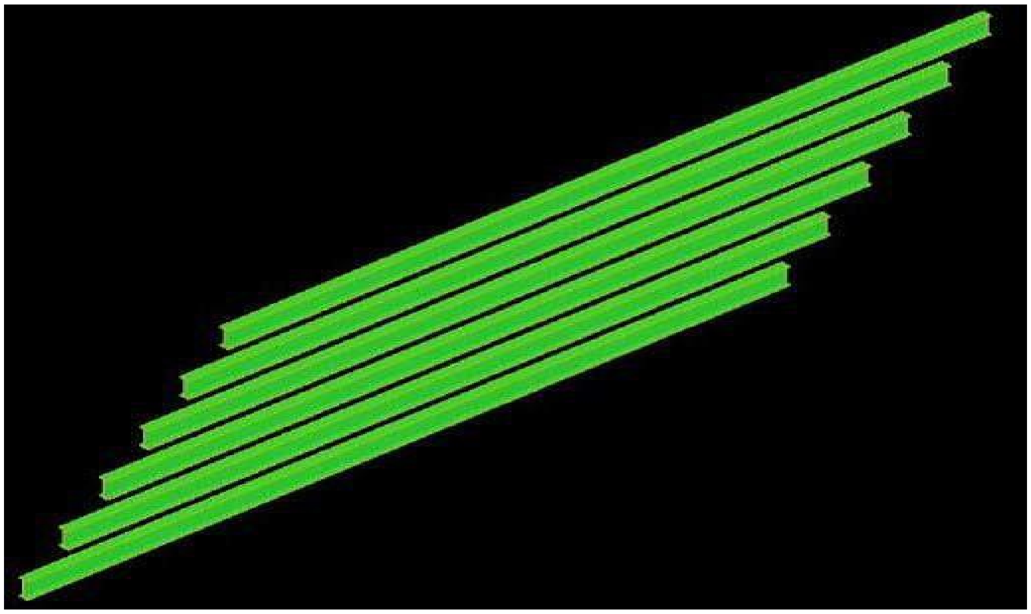


Figure 3-13. GML Model of the Girders

The OpenBrIM XML instance data file was parsed and rendered by the OpenBrIM Viewer. Figure 3-14 shows the OpenBrIM model of the girders.

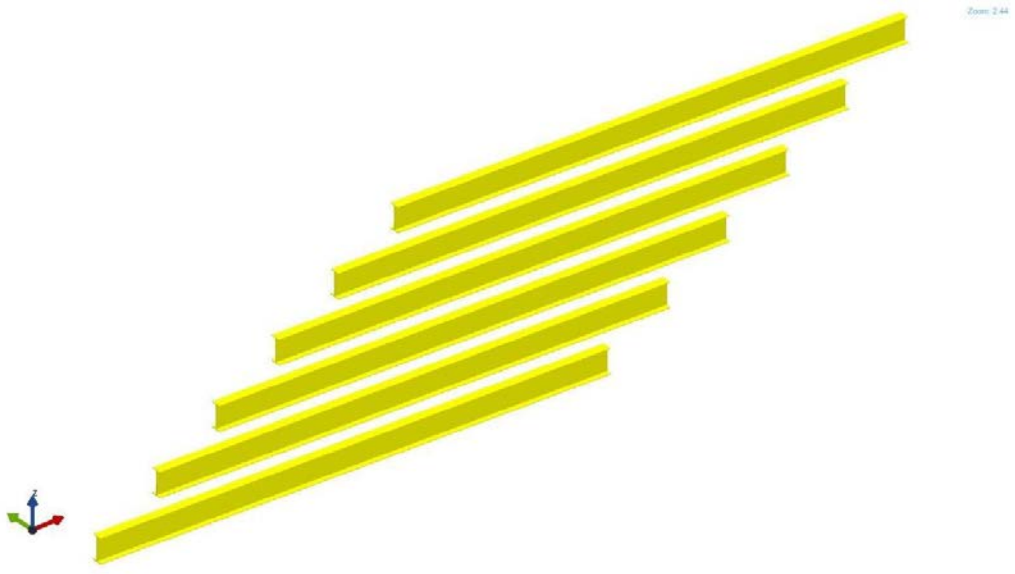


Figure 3-14. OpenBrIM Model of the Girders

3.2.3 Conclusions

From the two experiments, the following conclusions were drawn:

- Effort - IFC, GML and OpenBrIM required comparable effort in developing instance data files, because the entities, attributes and relationships that are predefined in these product models can be directly borrowed and used.

- File size - the size of the IFC instance file for the wall is three times larger than the size of the OpenBrIM instance file. This is because IFC, as a building-oriented data schema, saves many places for information about building story, which does not apply for bridges. Another reason is that the IFC instance file does not have a hierarchical structure. The relations between entities are reflected by referring ID numbers. Therefore, this type of expression is much less efficient than the one the OpenBrIM uses.

  Similarly, the size of the GML instance file for the girders is three times larger than the size of the OpenBrIM instance file. This is because OpenBrIM instance file uses an Object-Oriented way to define girders. Therefore, the OpenBrIM file does not include duplicated and redundant information.

- Clarity- the GML and OpenBrIM instance files are easier to read than the IFC file. This is because the IFC file does not use a hierarchical structure, and it utilizes ID numbers to reflect relations between entities. Therefore, the IFC file is less human readable than the other two files.

- Parsing - the effort required in parsing the IFC, GML and OpenBrIM instance files is comparable. This is because all three files are conforming to the basic XML schema, which is recommended by the World Wide Web Consortium (W3C) [36].

- Applicability - the applicability of IFC and OpenBrIM is better than GML in terms of viewer application. Because both IFC and OpenBrIM have their respective viewers, which can parse and render the models in the instance files, this aid largely alleviates the programmer’s work burden and facilitates the MVD development.

Table 3-3 lists the results of the comparisons.

Table 3-3. Data Schema Comparison Results

Comparison Items	**Experiment 1 IFC**	**Experiment 1 OpenBrIM**	**Experiment 2 GML**	**Experiment 2 OpenBrIM**
Effort	Comparable	Comparable	Comparable	Comparable
File size	OpenBrIM is better	OpenBrIM is better	OpenBrIM is better	OpenBrIM is better
Clarity	OpenBrIM is better	OpenBrIM is better	Comparable	Comparable
Parsing	Comparable	Comparable	Comparable	Comparable
Applicability	Comparable	Comparable	OpenBrIM is better	OpenBrIM is better

Based on these experiments, OpenBrIM data schema is selected as the basis for mapping IDM to create MVD and is emphasized in the following (Section 4) presentation and corresponding openBrIM viewer development.

4 MVD EXAMPLES

The references used to develop the BrIM MVD include data schemas (e.g., CIS/2 [9], ISM [4], LandXML [18], etc.), software products (e.g., AASHTOWare [1], etc.), NSBA manuals, and various contract plans [25], [28].

4.1 Roadway Geometry and Bridge Control Information

4.1.1 Alignment Based Coordinate System

Geometry definitions of structural members require setting a coordinate system. Currently, the types of coordinate systems supported by BIM data exchange models, such as Industry Foundation Classes (IFC) [7], CIMsteel Integration Standards Release 2 (CIS/2) [9] and Integrated Structural Modeling (ISM) [4], include Cartesian, polar, cylindrical, and spherical. Although these coordinate systems are sufficient for describing conventional or sophisticated geometries of building members, they are awkward or insufficient to define bridge geometry since bridge geometry depends on the alignment of the roadway that the bridge carries. The alignment can involve complex curves, such as clothoid spiral, parabola vertical profiles, etc. Although such complex curves can be approximately described in the Cartesian coordinate systems, doing so brings two major shortcomings:

- Approximation of coordinates will accumulate error and carry it throughout the life-cycle of bridge project; thus, accurate detailing of bridge members cannot be achieved.

- Approximation compromises the relationship between bridge and roadway, which loses the intent of bridge design.

Therefore, a robust yet concise definition of roadway alignment based coordinate system is needed as a basis for defining bridge geometry.

A key approach for the alignment-based coordinate system is the linear referencing method (LRM), which is well known in the field of transportation. An LRM describes a specific location in terms of a measurement along (and optionally offset from) a one-dimensional linear element (e.g., representing a road), from a defined starting point [26]. Location expressions for linear referencing consist of the linear referencing method, the linear element along which the distance is measured, and a distance expression. There are more than 12 types of LRMs used in the transportation industry [30]. They can be categorized as absolute distance LRMs, relative distance LRMs, and others. Linear elements include alignment, route, links, anchor sections, etc. The distance expression is dictated by which LRM is selected [29].

An alignment-based coordinate system (ABCS) can be built based on the linear referencing method. An ABCS has three coordinate axes: the longitudinal axis is along a roadway alignment, the transverse axis is perpendicular to the longitudinal axis, and the vertical axis is for measurement of elevation. Table 4-1 shows a comparison of the Cartesian coordinate system and the alignment based coordinate system.

Bridge geometry is typically described in a manner that depends on an ABCS. Roadway alignment is defined first as a reference line. Geometries of major bridge components such as girders are then described as offsets from the reference line. Geometries of bridge components that connect to the girders are defined according to girder geometry by using parametric modeling method and the relative locations.

Table 4-1 Cartesian Coordinate System vs. Alignment Based Coordinate System

Cartesian Coordinate System (used by buildings)	**Alignment Based Coordinate System (used by bridges)**
X	Longitudinal (L, along alignment)
Y	Transverse (T, perpendicular to alignment)
Z	Vertical (V, Measurement of elevation)

Parametric modeling uses dimensional parameters to build a flexible geometric model. The dependency between model components is reflected by the numeric relationships between parameters of the components using mathematical formulas and geometric constraints. As a result, the whole model will be automatically updated when the value of some parameters change. Parametric modeling not only is able to grasp the design intent, but also reduces needless data duplication [11], [12], [13], [14].

Several widely implemented data exchange schema such as CIS/2 and IFC that do not support parametric modeling are based on EXPRESS language. The eXtensible Markup Language (XML) [36] is better for creation of the alignment based data exchange schema that supports parametric modeling. XML data structure is also sufficiently flexible to enable object-oriented inheritance hierarchies. In contrast to EXPRESS, XML is supported by a broader range of software applications and can be easily implemented for web services. It was adopted as the basis of data exchange schemas for transportation such as LandXML and TransXML [18], [23]. The buildingSMART alliance was attracted by the power of XML and developed guidelines for using XML to implement IFC standard, known as ifcXML [2], [7], [13], [20], [21].

The following sections present and illustrate development of alignment based data schema for exchange of important geometric information for bridges for an actual bridge project in New York State. Data from the actual contract plans is used to create the XML codes based on the proposed schema. The geometry represented by these codes can be viewed by using a public-domain viewer software application called OpenBrIM viewer available at openbrim.org. Eventually, of course, software applications would write translators to generate and parse such XML codes.

4.1.2 Introduction to Roadway Geometry

Elements of a horizontal alignment include straight line, circular curve, and transition curve (e.g., clothoid spiral). Elements of a vertical profile include straight line, parabolic curve (symmetric and asymmetric) and sometimes circular curve. Elements of a cross section include side slope, transverse offset and elevation. Figure 4-1 and Figure 4-2 show the plan view of one of the case study bridges used in this research and details of the horizontal alignment, respectively.

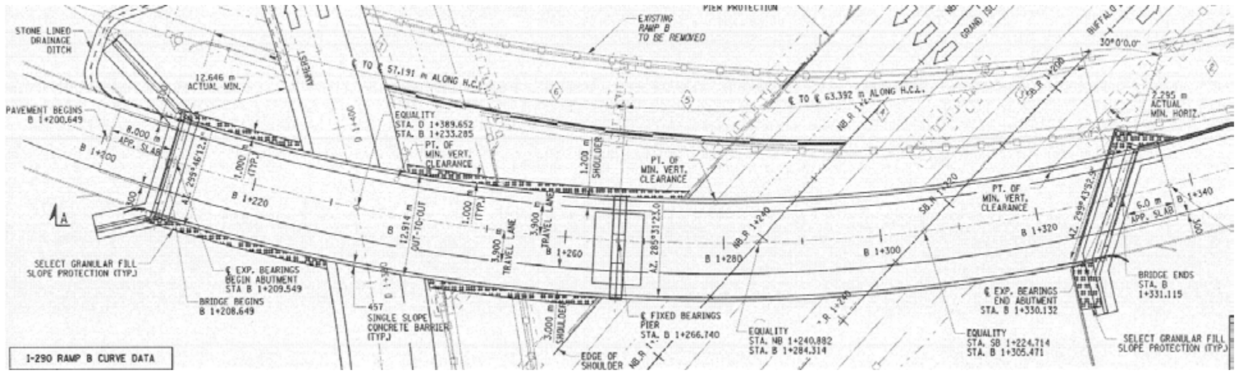
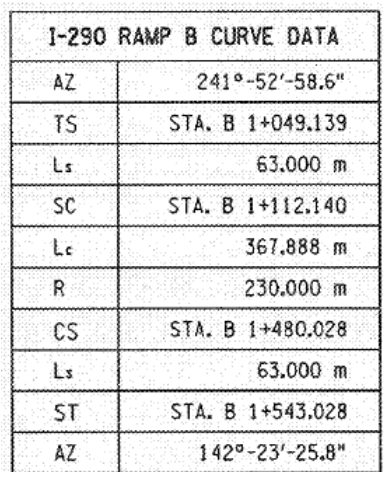

STONE LINED DRAINAGE DITCH
PAVEMENT BEGINS B 1+200.649
12.646 m ACTUAL MIN.
EQUALITY STA. D 1+389.652 STA. B 1+233.285
PT. OF MIN. VERT. CLEARANCE
EXISTING RAMP B TO BE REMOVED
SELECT GRANULAR FILL SLOPE PROTECTION (TYP.)
BRIDGE BEGINS B 1+208.649
EXP. BEARINGS BEGIN ABUTMENT STA B 1+209.549
457 SINGLE SLOPE CONCRETE BARRIER (TYP.)
FIXED BEARINGS PIER STA. B 1+266.740
EQUALITY STA. NB 1+240.882 STA. B 1+284.314
EXP. BEARINGS END ABUTMENT STA. B 1+330.132
EQUALITY STA. SB 1+224.714 STA. B 1+305.471
BRIDGE ENDS STA. B 1+331.115
SELECT GRANULAR FILL SLOPE PROTECTION (TYP.)
7.295 m ACTUAL MIN. HORIZ.
I-290 RAMP B CURVE DATA

Figure 4-1. Horizontal Alignment and Bridge Plan View [25]



I-290 RAMP B CURVE DATA	
AZ	241°-52′-58.6"
TS	STA. B 1+049.139
Ls	63.000 m
SC	STA. B 1+112.140
Lc	367.888 m
R	230.000 m
CS	STA. B 1+480.028
Ls	63.000 m
ST	STA. B 1+543.028
AZ	142°-23′-25.8"

Figure 4-2. Details of Horizontal Alignment [25]

Figure 4-3 shows the vertical profile of the case study bridge and its details.

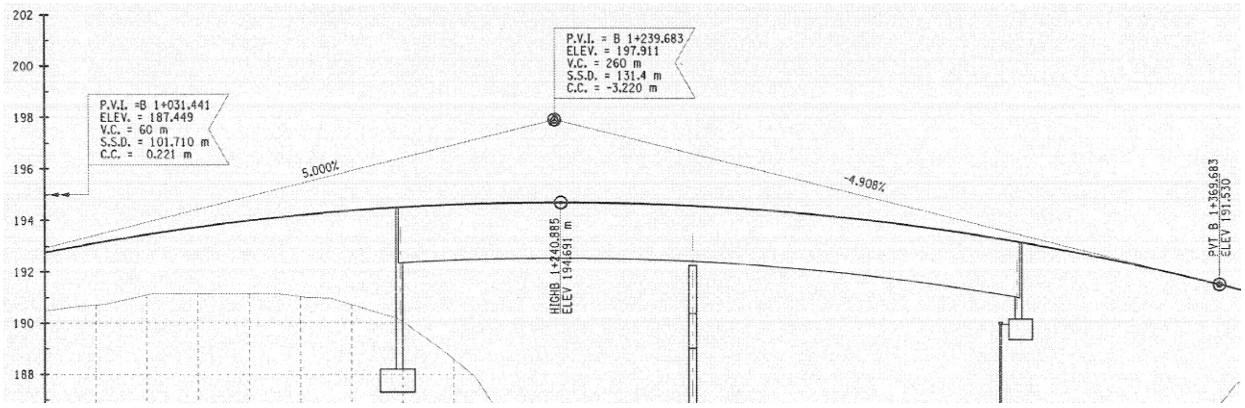


Figure 4-3. Vertical Profile and Bridge Elevation View [25]

Figure 4-4 shows the cross section of the case study bridges.

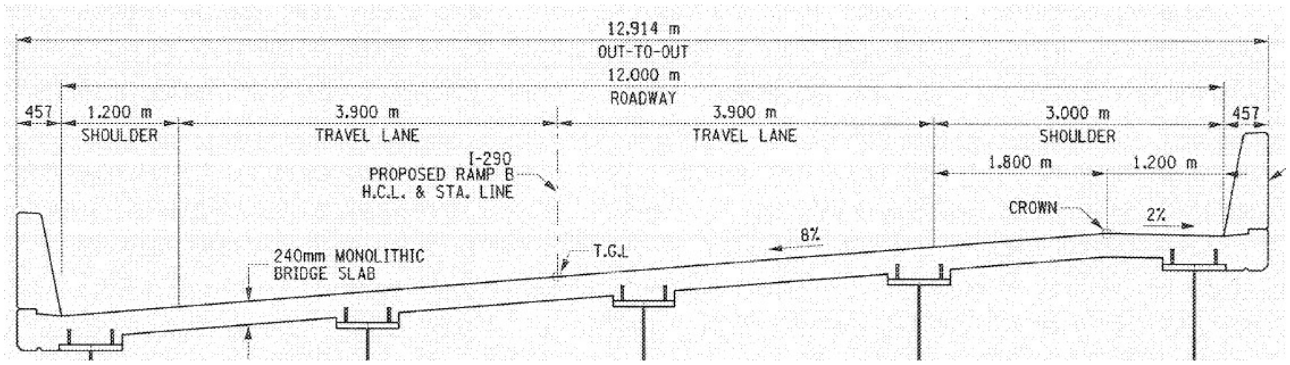


Figure 4-4. Bridge Cross Section [25]

4.1.3 Mapping Roadway Geometry to Data Schemas

Roadway alignment is the basis of defining bridge geometry. Currently, there are several data exchange schemas implemented or proposed for describing alignment information suitable for data interoperability.

- LandXML [18] has been implemented in many software applications. However, it is not considered a sufficiently concise schema because it contains more data than necessary to create roadway alignments.

- buildSMART European chapters proposed a IFC-based data model for bridges, known as IFC-Bridge [19]. It aims to expand IFC for infrastructures, but it has some drawbacks, e.g., it does not have a definition for parabola which is an important type of vertical curve.

- Liebich, et al. 2009 [21], make some improvements on IFC-Bridge, but it does not provide an explicit definition of cross section.

Table 4-2(a-c) shows a comparison of the roadway geometry definitions from LandXML, a data model proposed by buildingSMART, German Chapter, the contract plans, and the OpenBrIM schema proposed in this report. Appendix A shows a more detailed comparison. "*Derived data"* in the table, means the parameter can be derived from other parameters using a mathematical formula. For example, end station is equal to the StartSta (start station) plus Length. From the comparison, it is evident that the OpenBrIM schema using parametric modeling requires less data to achieve unambiguous accurate alignment geometry in support of interoperability.

Table 4-2a. **Comparison of Roadway Geometry Models - Horizontal Alignment**

LandXML[18]	**IFC Infrastructure Alignment Representation**[6]	**Bridge Contract Drawings**[25]	**OpenBrIM**[8]
Line	Line	(Line)	Straight (Line)
start station	Not Applicable	ST, TS	StartStation
length	Length	Not Applicable	Length
dir(ection)	Start direction, end direction	AZ(imuth)	StartAzimuth
(start, end) easting, northing	(start, end) easting, northing	Not Applicable	Not Applicable
Spiral (clothoid)	Transition curve (clothoid)	(Spiral)	(Spiral) Curve
start station	Not Applicable	TS, CS, SC, ST	StartStation
length	length	Ls	Length
Not Applicable	start direction, end direction	AZ(imuth)	StartAzimuth
radiusStart, radiusEnd	start radius, end radius	(Infinity), R	Not Applicable
theta, totallY, totalX, tanLong, tanShort	parameter	Not Applicable	Not Applicable
(start, PI, end) easting, northing	(start, end) easting, northing	Not Applicable	Not Applicable
rot(ation)	Not Applicable	Not Applicable	Direction
Curve (arc)	Arc	(Circular curve)	(Circular) Curve
start station	Not Applicable	CS, SC	StartStation
length	length	Lc	Length
radius	start radius, end radius	R	Radius
dirStart, dirEnd	start direction, end direction	Not Applicable	StartAzimuth
chord, delta, external, midOrd, tangent	Not Applicable	Not Applicable	Not Applicable
(start, center, end and PI) easting, northing	(start, end) easting, northing	Not Applicable	Not Applicable
rot(ation)	Not Applicable	Not Applicable	Direction

Table 4-2b. Comparison of Roadway Geometry Models – Vertical Alignment/Profile

LandXML[18]	**IFC Infrastructure Alignment Representation**[6]	**Bridge Contract Drawings**[25]	**OpenBrIM**[8]
Line	(Straight line)	(Line)	(Line)
(start, end) station	Point vertical intersection (PVI)	station	station
(start, end) elevation	Point vertical intersection (PVI)	elevation	elevation
ParaCurve (symmetric)	(Ifc)VerticalAlignmentParabola	(Parabolic curve)	(Parabolic curve)
PVI station	Point vertical intersection (PVI)	(PVI, PVC, PVT) station	(PVC, PVT) station
PVI elevation	Point vertical intersection (PVI)	(PVI, PVC, PVT) elevation	(PVC, PVT) elevation
length	Intersection point distance	V.C.	Not Applicable
Not Applicable	Not Applicable	G1, G2	Grade (1, 2)
Not Applicable	Not Applicable	S.S.D., C.C.	Not Applicable
ParaCurve (Unsymmetric)	Not Applicable	Not Applicable	Not Applicable
PVI station	Not Applicable	Not Applicable	Not Applicable
PVI elevation	Not Applicable	Not Applicable	Not Applicable
Length in	Not Applicable	Not Applicable	Not Applicable
Length out	Not Applicable	Not Applicable	Not Applicable
Circle	Not Applicable	Not Applicable	Not Applicable
PVI station	Not Applicable	Not Applicable	Not Applicable
PVI elevation	Not Applicable	Not Applicable	Not Applicable
length	Not Applicable	Not Applicable	Not Applicable
radius	Not Applicable	Not Applicable	Not Applicable

Table 4-2c. Comparison of Roadway Geometry Models – Cross Section

LandXML[18]	IFC Infrastructure Alignment Representation[6] (Not Applicable)	Bridge Contract Drawings[25]	OpenBrIM[8]
station	Not Applicable	station	Station
Not Applicable	Not Applicable	Not Applicable	LeftEdgeToHCL
offset	Not Applicable	offset	Not Applicable
elevation	Not Applicable	Not Applicable	Not Applicable
slope	Not Applicable	slope	Slope
distance	Not Applicable	width	Width

4.1.3.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIM schema is shown as follows. Elements of vertical profile include station, elevation, and grade.

```
<RoadwayGeometry Name="I-290 Ramp B Alg">
<HorizontalAlignment StartStation="1049139" StartAzimuth="4.2195">
<Curve Type="Spiral" Direction="Left" Length="63000"/>
<Curve Type="Circular" Direction="Left" Length="367888"
Radius="230000"/>
<Curve Type="Spiral" Direction="Left" Length="63000"/>
</HorizontalAlignment>
<VerticalProfile>
<ProfilePoint Name="PVC" Station="1139000"
Elevation="192700" Grade="0.05"/>
<ProfilePoint Name="PVT" Station="1369683"
Elevation="191530" Grade="-0.04908"/>
</VerticalProfile>
<SuperElevation>
<CrossSection Station="1200649" LeftEdgeToHCL="-5557"
ElevationAtHCL="0">
<TransverseSegment Slope="-0.08" Width="11257"/>
<TransverseSegment Slope="0.02" Width="1657"/>
</CrossSection>
</SuperElevation>
</RoadwayGeometry>
```

Figure 4-5 illustrates the viewer-generated model of the bridge deck which is compliant with the horizontal alignment, vertical profile and cross section defined in the XML code.

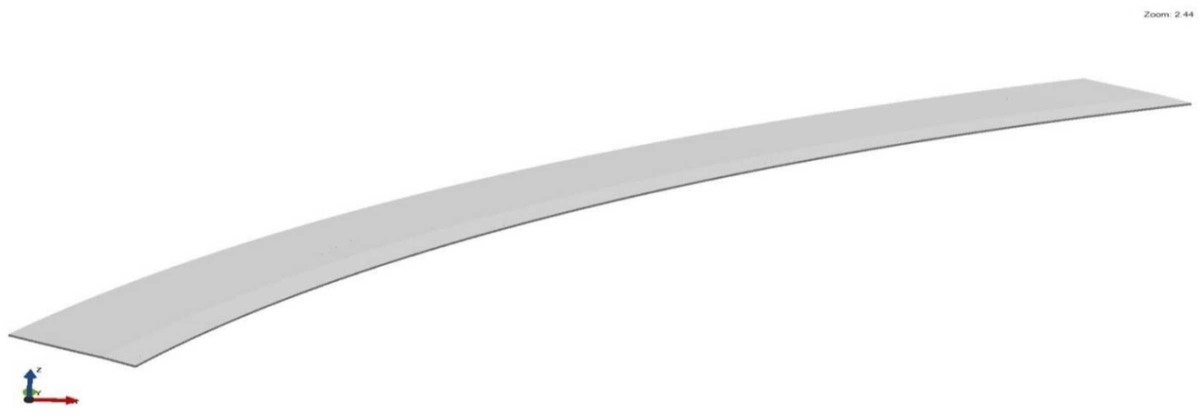


Figure 4-5. Model of Bridge Deck

4.1.3.2 Mapping to ISM

To define the roadway geometry, an entity IsmRoadway was created as a complement to the current ISM schema. Table 4-3 shows a comparison of the horizontal alignment parameters used in LEAP Geomath, the contract plans, and the IsmRoadway schema. From the comparison, it is evident that the current IsmRoadway is lack of many parameters for defining roadway geometry.

Table 4-3. Comparison of Three Sets of Data for Horizontal Alignment

LEAP Geomath	The Contract Plans	IIsmRoadway
Alignment	(Horizontal alignment)	IIsmCurve2D
Tangent	(Line)	IIsmCurve2D
Starting sta, Ending sta	ST (spiral to tangent), TS (tangent to spiral)	StartStation, EndStation (IIsmPoint2D)
Start Dir	AZ(imuth)	StartAzimuth
(PI) easting, northing	Not applicable	IIsmPoint2D
Spiral	(Spiral)	IIsmCurve2D
N/A	TS, CS (curve to spiral), SC (spiral to curve), ST	StartStation, EndStation (IIsmPoint2D)
N/A	Ls	Not applicable
N/A	AZ(imuth)	Not applicable
Spiral-In, Spiral-Out	Infinite, R	Not applicable
(PI) easting, northing	Not applicable	IIsmPoint2D
Arc	(Circular curve)	IIsmCurve2D
Starting Sta, Ending Sta	CS, SC	StartStation, EndStation (IIsmPoint2D)
N/A	Lc	Not applicable
Radius	R	Not applicable
Start Dir	Not applicable	Not applicable
(PI) easting, northing	Not applicable	IIsmPoint2D

4.1.3.3 Mapping to IFC

Roadway alignment driven data schema heavily depends on parametric dependencies. As pointed out in Section 4.1.1, EXPRESS language on which IFC is based on, lacks the ability to implement parametric dependencies. A crucial part of the bridge design is the alignment curve. For this, the authors introduce an *IfcReferenceCurveAlignment2D* element, which references a horizontal and a vertical alignment curve. For both the horizontal and vertical alignment, an *IfcCurve* element is used. Since *IfcCurve* is very general, it allows many different types of descriptions of curves, which makes it hard for software application implementers to handle all types and combinations of curve types [4].

4.1.4 Introduction to Bridge Control Information

After roadway alignment has been defined, bridge configuration will be specified on the alignment by using two sets of parameters, station and skew angle. Because the roadway does not need to know where the bridge is placed, these two sets of parameters are independent of the definition of roadway (Figure 4-6).

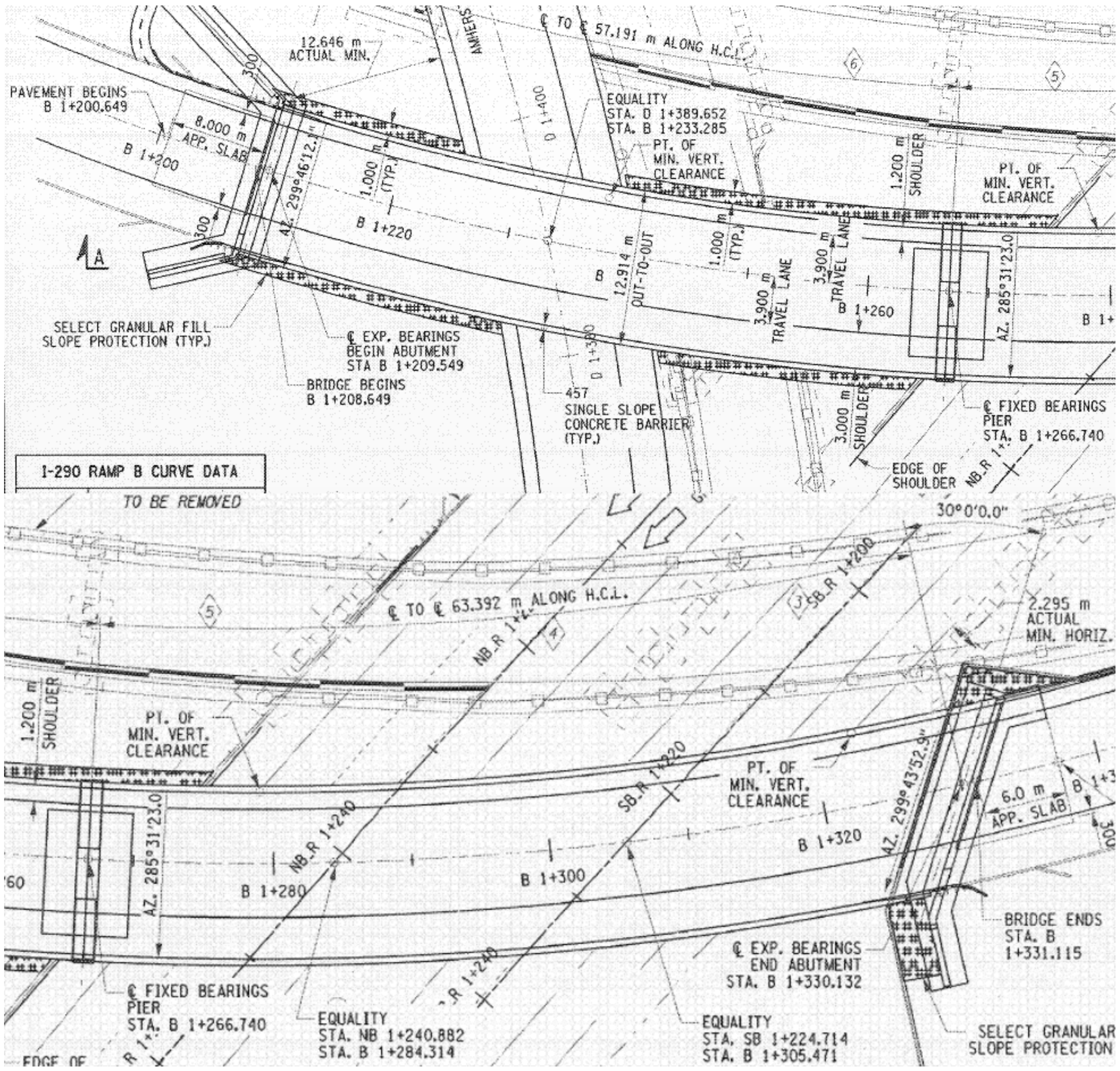

PAVEMENT BEGINS B 1+200.649
12.646 m ACTUAL MIN.
8.000 m APP. SLAB
B 1+200
AZ. 299°46'12."
1.000 (TYP.)
B 1+220
SELECT GRANULAR FILL SLOPE PROTECTION (TYP.)
℄ EXP. BEARINGS BEGIN ABUTMENT STA B 1+209.549
BRIDGE BEGINS B 1+208.649
℄ TO ℄ 57.191 m ALONG H.C.L.
EQUALITY STA. D 1+389.652 STA. B 1+233.285
PT. OF MIN. VERT. CLEARANCE
12.914 m OUT-TO-OUT
3.900 m TRAVEL LANE
457 SINGLE SLOPE CONCRETE BARRIER (TYP.)
1.200 m SHOULDER
3.000 m SHOULDER
AZ. 285°31'23.0"
B 1+260
℄ FIXED BEARINGS PIER STA. B 1+266.740
EDGE OF SHOULDER
I-290 RAMP B CURVE DATA
TO BE REMOVED
℄ TO ℄ 63.392 m ALONG H.C.L.
30°0'0.0"
2.295 m ACTUAL MIN. HORIZ.
B 1+280
B 1+300
B 1+320
AZ. 299°43'52.9"
6.0 m APP. SLAB
BRIDGE ENDS STA. B 1+331.115
℄ EXP. BEARINGS END ABUTMENT STA. B 1+330.132
EQUALITY STA. NB 1+240.882 STA. B 1+284.314
EQUALITY STA. SB 1+224.714 STA. B 1+305.471
SELECT GRANULAR SLOPE PROTECTION

Figure 4-6. Bridge Control Information [25]

4.1.5 Mapping Bridge Control Information to Data Schemas

4.1.5.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIM schema is shown as follows:

```
<Param Name="PavementBeginsSta" Value="1200649"/>
<Param Name="BridgeBeginsSta" Value="1208649"/>
<Param Name="BeginAbutmentSta" Value="1209549"/>
<Param Name="PierSta" Value="1266740"/>
<Param Name="EndAbutmentSta" Value="1330132"/>
```

```
<Param Name="BridgeEndsSta" Value="1331115"/>
<Param Name="PavementEndsSta" Value="1337115"/>

<Param Name="BeginAbutmentSkewAngle" Value="0/180*pi"/>
<Param Name="PierSkewAngle" Value="0/180*pi"/>
<Param Name="EndAbutmentSkewAngle" Value="30/180*pi"/>
```

In the above XML code, name “PavementBeginsSta” indicates that this parameter contains information about station at pavement begins. The value “1200649” means the station is 1+200.649. The station parameter other than the parameters defined above can be added as requested by schema users. . The skew angle is defined as the angle between the centerline of bearing and the normal to the horizontal control line. It is positive in counterclockwise direction. The value of skew angle is in radian.

4.1.5.2 Mapping to ISM

In ISM, station can be defined by using IIsmPoint3D interface. There is no explicit way to define skew angle.

4.1.5.3 Mapping to IFC

As, Bridge Control Information is the data specific to bridges, IFC at this moment cannot handle the information such as roadway geometry and important features of bridges such as abutment skew angles, pier skew angles, and start and end stations of bridge.

4.2 Decomposition

4.2.1 Introduction to Decomposition and Composition

Development of Model View Definition requires both the decomposition and composition mechanisms. Decomposition demonstrates how the “work-horse” concrete and steel bridges are broken down for the definition of bridge members and structural connections, while its reverse counterpart, composition describes how these member assemblies are combined to form the whole bridge, by using joint systems and parametric dependency.

A bridge can be divided into superstructure and substructure, and they can be further divided into multiple member assemblies. Thus, the combination of conceptual representation of member assemblies is the conceptual representation of the whole bridge. Therefore, specifying member assemblies is critical for the development of Model View Definition.

4.2.2 Specifying Bridge Members

Specifying member assemblies can be done by using various bridge models and contract drawings. Bridge models have been created using several software applications for several purposes. These models are considered representative of those that could be involved in various stages of the workflow requiring interoperability functionality that is the ultimate aim of the present investigation. Figure 4-7 shows the bridge model created using SAP2000 Bridge Module (now CSiBridge) [10] for conventional and seismic analysis. The model only contains structural components such as slab, girder, pier cap beams and pier columns. It ignores all the non-structural components, haunches, diaphragms, and piles at the abutments. It also does not include the 2% side slope on the deck, the abutments and the reinforcement in all the concrete members.

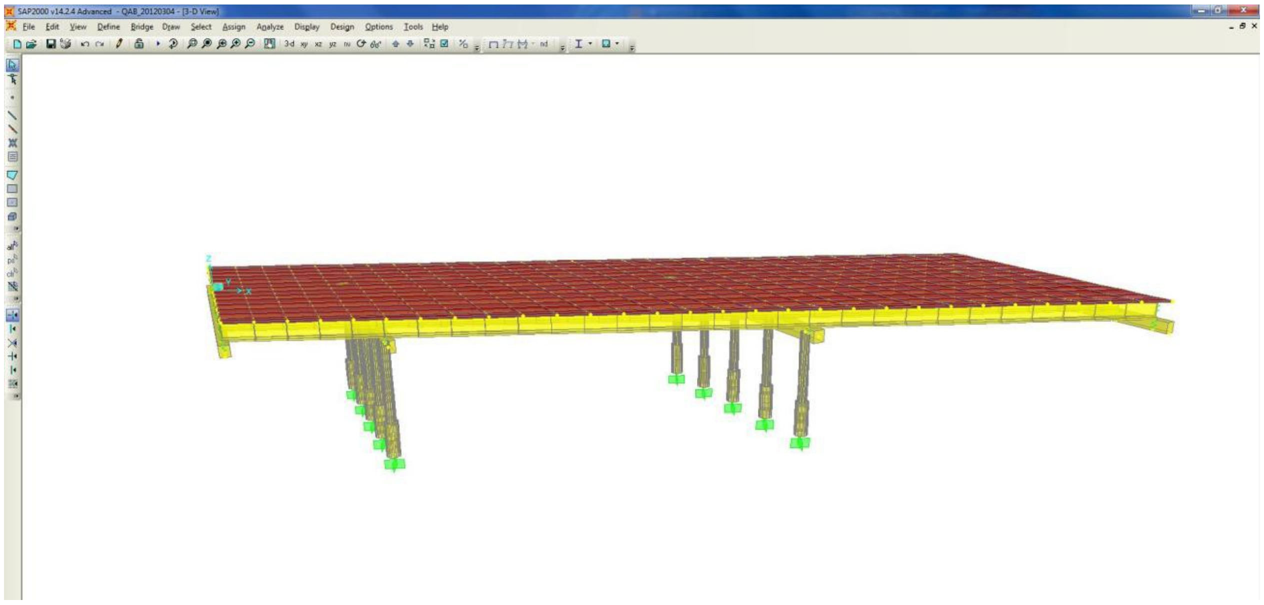


Figure 4-7. SAP2000 Model of Quincy Avenue over I-25 and LRT

Figures 4-8 and 4-9 show additional models of the superstructure and substructure of the case study bridge, respectively. They have been created using LEAP Conspan and RC-Pier [5] for analysis and design. These two models contain all the necessary bridge components except diaphragms. They include the side slope, but neglect the reinforcement.

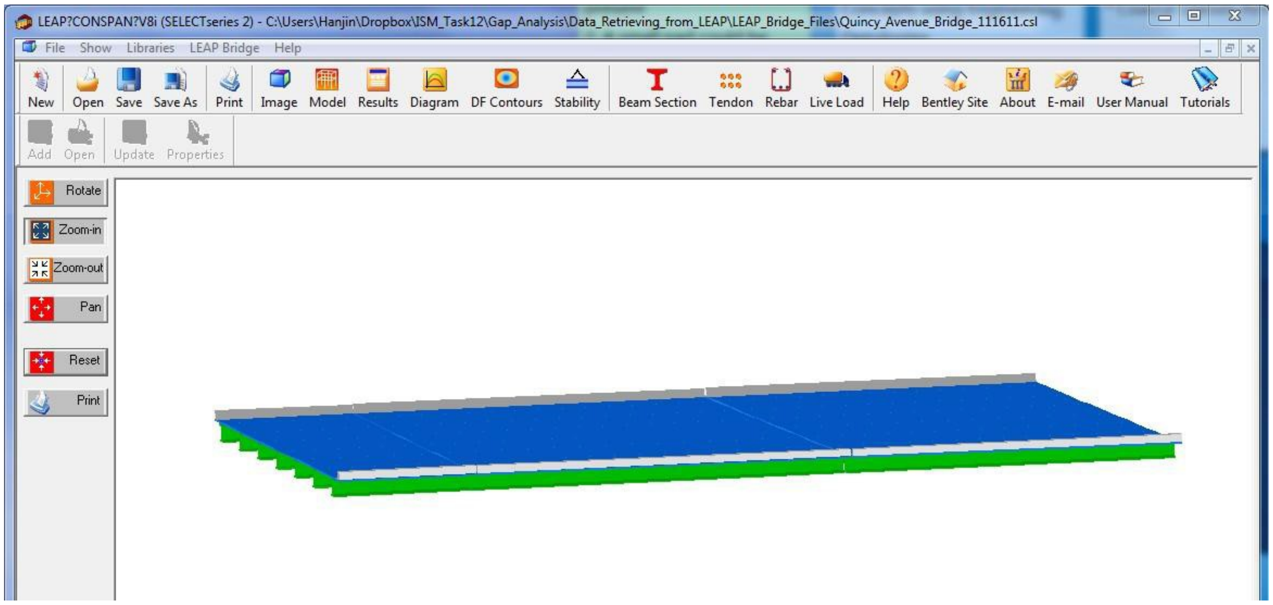

LEAP?CONSPAN?V8i (SELECTseries 2) - C:\Users\Hanjin\Dropbox\ISM_Task12\Gap_Analysis\Data_Retrieving_from_LEAP\LEAP_Bridge_Files\Quincy_Avenue_Bridge_111611.csl
File Show Libraries LEAP Bridge Help
New Open Save Save As Print Image Model Results Diagram DF Contours Stability Beam Section Tendon Rebar Live Load Help Bentley Site About E-mail User Manual Tutorials
Add Open Update Properties
Rotate
Zoom-in
Zoom-out
Pan
Reset
Print

Figure 4-8. Superstructure Model of Quincy Avenue over I-25 and LRT in LEAP Conspan

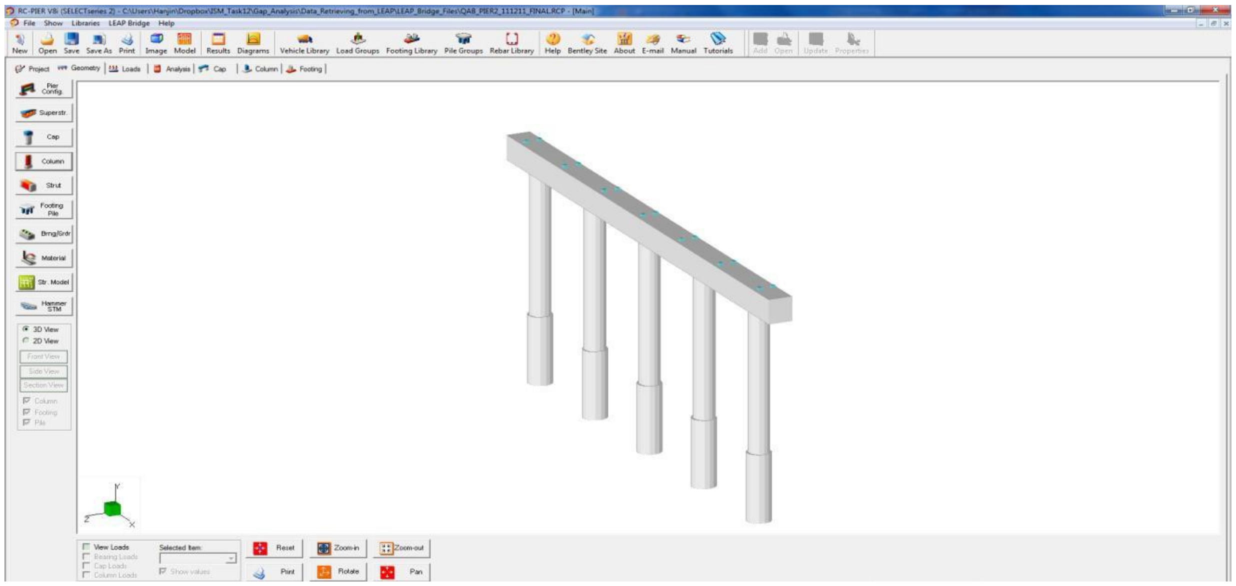


Figure 4-9. Substructure Model of Quincy Avenue over I-25 and LRT in LEAP RC-Pier

Figures 4-10 and 4-11 show bridge models created using Tekla Structures [32] for detailing and construction. The models contain all the bridge components, both structural and non-structural. It also includes the side slope. In addition, the reinforcement is embedded in the concrete members.

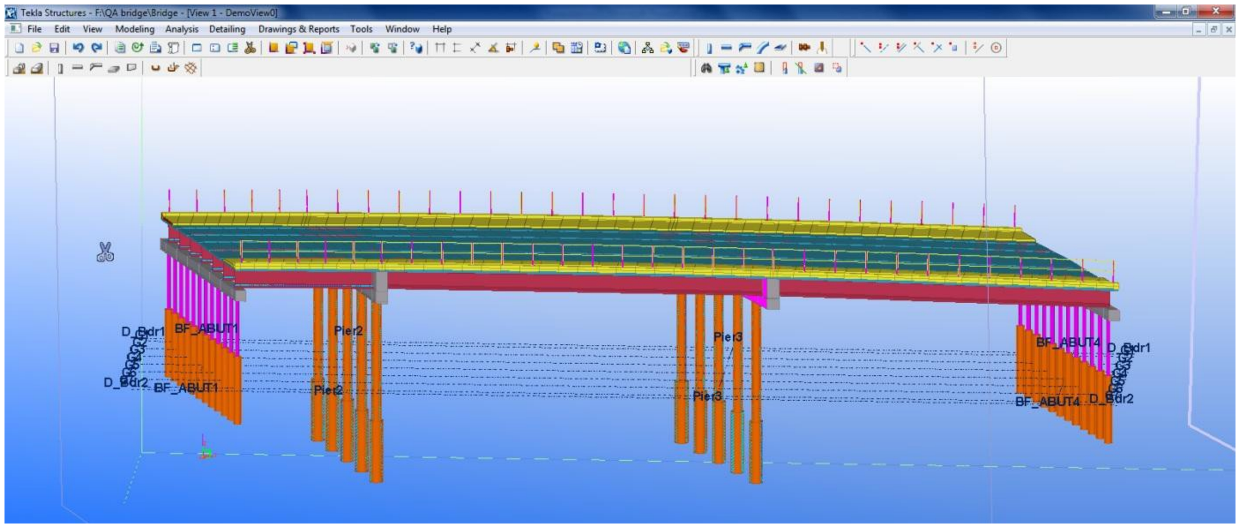


Figure 4-10. Tekla Model of Quincy Avenue over I-25 and LRT

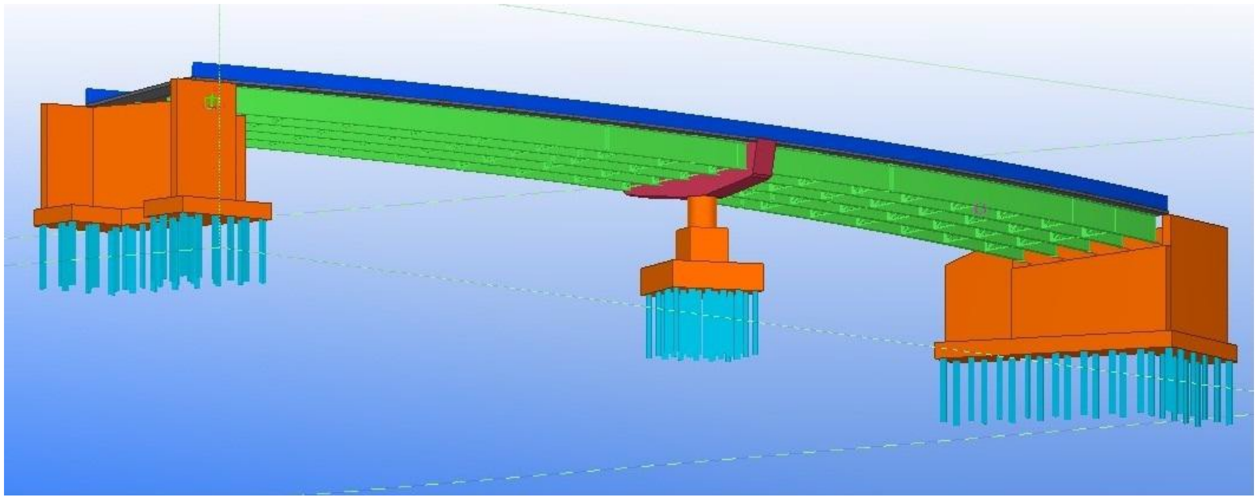


Figure 4-11. Tekla Model of I-290 Ramp D over I-190

Figure 4-12 shows the Industry Foundation Classes (IFC) [7] model of I-290 Ramp D over I-190. IFC is an open and international data model standard developed by buildingSMART Alliance for exchanging BIM data. IFC is supported by more than 150 software applications. CSiBridge is one of them. This IFC model of the case study bridge was exported from CSiBridge and was used for data exchange.

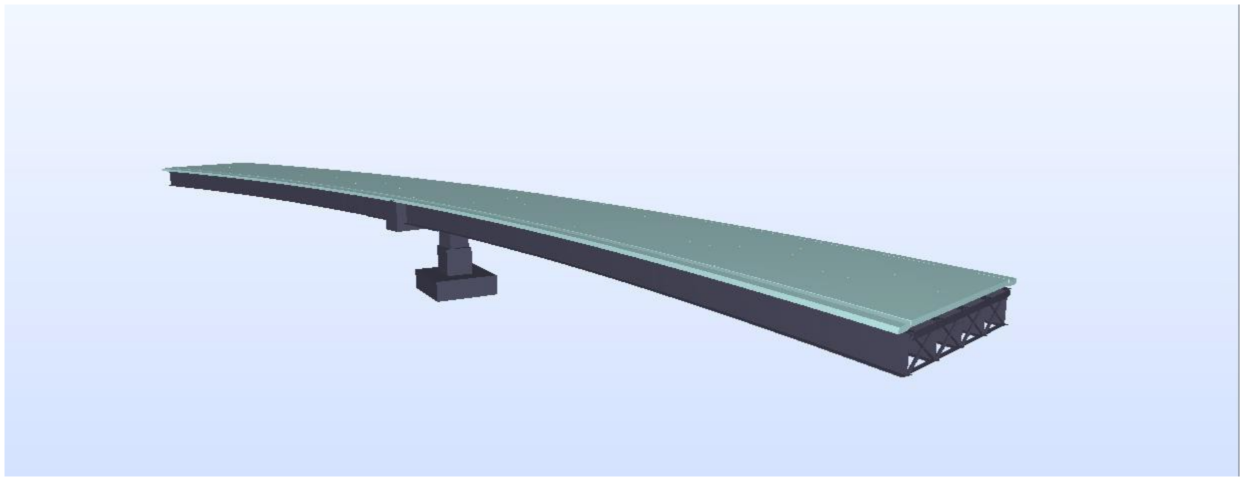


Figure 4-12. IFC Model of I-290 Ramp D over I-190

Table 4-4 lists the components of one of the concrete case study bridges, Quincy Avenue over I- 25 and LRT defined in these bridge models.

Table 4-4. Components of Bridge Models of Quincy Avenue over I-25 and LRT

Type	Component	Tekla	SAP2000	LEAP Bridge
Superstructure	Railing	Y	N	N
Superstructure	Parapet	Y	N	Y
Superstructure	Sidewalk	Y	N	N
Superstructure	Slab	Y	Y	Y
Superstructure	Haunch	Y	N	Y
Superstructure	Girder	Y	Y	Y
Superstructure	Bearing	N	N	Y
Substructure	Diaphragm	Y	N	N
Substructure	Cap Beam	Y	Y	Y
Substructure	Pier Column	Y	Y	Y
Substructure	Drilled Shaft	Y	Y	Y
Substructure	Abutment Beam	Y	Y	Y
Substructure	Pile	Y	N	Y
Substructure	Shaft	Y	N	Y

Table 4-5 lists the components of one of the steel case study bridges, I-290 Ramp D over I-190 defined in these bridge models.

Table 4-5. Components of Bridge Models of I-290 Ramp D over I-190

Type	Component	Tekla	CSiBridge	IFC
Superstructure	Parapet	Y	N	N
Superstructure	Slab	Y	Y	Y
Superstructure	Girder	Y	Y	Y
Superstructure	Stiffener	Y	N	N
Superstructure	Cross frame	Y	Y	Y
Superstructure	Bottom lateral brace	Y	N	N
Superstructure	Shear stud	Y	N	N
Substructure	Cap Beam	Y	Y	Y
Substructure	Pier Column	Y	Y	Y
Substructure	Pile cap	Y	Y	Y
Substructure	Pile	Y	N	N
Substructure	Bearing	Y	N	N
Substructure	Pedestal	Y	N	N
Substructure	Abutment walls	Y	N	N

Table 4-6 shows the structural components of all the concrete case study bridges used in this research.

Table 4-6. Structural Components of Concrete Case Study Bridges

Bridge Members	Quincy Avenue over I-25 and LRT	Glenridge Road over Alplaus Kill	Staten Island Expressway over Slosson Avenue (westbound)	Otay River Bridge
Girder	Bulb Tee	Box beam	NEXT beam	CA box beam
Deck slab	Y	Y	Y	N
Haunch	Y	N	N	N
Diaphragm	Y	Y	Y	Y
Bearing	Y	Y	Y	Y
Cap beam	Y	N	N	N
Column	Y	N	N	Y
Footing	N	N	Y	N
Pile cap	N	Y	N	Y
Pile / drilled shaft	Y	Y	N	Y
Abutment walls	Y	Y	Y	Y

Table 4-7 shows the structural components of all the steel case study bridges used in this research.

Table 4-7. Structural Components of Steel Case Study Bridges

Bridge Members	I-190 over CSX Railroad (Southbound)	I-290 Ramp B over I-290 Ramp D and I-190	I-290 Ramp D over I-190
Steel plate girder	Y	Y	Y
Deck slab	Y	Y	Y
Haunch	Y	Y	Y
Shear stud	Y	Y	Y
Stiffener	Y	Y	Y
Cross frame	Y	Y	Y
Bottom lateral brace	Y	Y	Y
Bearing	Y	Y	Y
Pedestal	Y	Y	Y
Cap Beam	N	N	Y
Hammer head	N	Y	N
Pier wall	Y	N	N
Pier column	N	N	Y

Bridge Members	I-190 over CSX Railroad (Southbound)	I-290 Ramp B over I-290 Ramp D and I-190	I-290 Ramp D over I-190
Pile cap	Y	Y	Y
Pile	Y	Y	Y
Abutment walls	Y	Y	Y
Bolt	Y	Y	Y
Weld	Y	Y	Y

Based on Table 4-6 and Table 4-7, the basic bridge structural components of all the concrete and steel case study bridges includes deck slab, haunch, girder, shear stud, stiffener, diaphragm (cross frame), bottom lateral brace, bearing, pedestal, pier/abutment cap beam, hammer head pier, pier wall, column, footing, pile cap, drilled shaft, pile, abutment stem wall and abutment wing wall. The non-structural components of the case study bridges include railing, parapet, and sidewalk. The joint components of the case study bridges contain bolt and weld.

4.3 Structural Modeling and Structural Response

4.3.1 Structural Load

1. Types
1.1 Accidental: fire, impact, etc.
1.2 Permanent: DL
1.3 Long term: LL, settlement
1.4 Short term: wind, snow
1.5 Transient: erection, etc.

2. Applied load
2.1 Static applied load
2.1.1 Static displacement
2.1.2 Static force
2.1.3 Static pressure
2.2 Dynamic applied load
2.2.1 Dynamic acceleration
2.2.2 Dynamic velocity

3. Load
3.1 Load case
3.1.1 Documented load case
3.2 Occurrence load combination
3.3 Load element
3.3.1 Distributed element
3.3.1.1 Surface distributed element
3.3.1.1.1 Uniform surface distributed element
3.3.1.1.2 Varying surface distributed element
3.3.1.2 Curve distributed element
3.3.1.2.1 Line curve distributed element
3.3.2 Thermal element
3.3.3 Concentrated element

3.4 Node load

4. Load combination

5. Physical action
5.1 Permanent action
5.2 Variable action
5.2.1 Long term variable
5.2.2 Transient variable
5.2.3 Short term variable
5.3 Accidental action
5.4 Seismic action

4.3.1.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="GirderA" RefObj="SteelIGirder" Sta="0" TOff="0" Elev="0">
<Obj Name="DC" Type="Result">
<Obj Name="@ 1000" Type="Force">
<Param Name="Station" Value="1000"/>
<Param Name="Fx" Value="100"/>
<Param Name="Fy" Value="200"/>
<Param Name="Fz" Value="50"/>
<Param Name="Mx" Value="-100"/>
<Param Name="My" Value="0"/>
<Param Name="Mz" Value="0"/>
</Obj>
<Obj Name="@ 2000" Type="Force">
<Param Name="Station" Value="2000"/>
<Param Name="Fx" Value="0"/>
<Param Name="Fy" Value="0"/>
<Param Name="Fz" Value="0"/>
<Param Name="Mx" Value="0"/>
<Param Name="My" Value="0"/>
<Param Name="Mz" Value="0"/>
</Obj>
<Obj Name="@ 1000" Type="Deflection">
<Param Name="Station" Value="1000"/>
<Param Name="Tx" Value="0"/>
<Param Name="Ty" Value="0"/>
<Param Name="Tz" Value="0"/>
<Param Name="Rx" Value="0"/>
<Param Name="Ry" Value="0"/>
<Param Name="Rz" Value="0"/>
</Obj>
</Obj>
</Obj>
```

4.3.2 Structural Response

1. Analysis result
1.1 Node analysis result
1.2 Element node analysis result
1.3 Element analysis result
1.3.3 Point element
1.3.1 Curve element
1.3.2 Surface element
1.3.2.1 Surface stresses
1.3.2.2 Surface tractions
1.3.4 Volume element
1.3.4.1 Volume stress tensor

2. Results set
2.1 Basic results set
2.2 Combined results set
2.3 Envelop results set
2.4 Redistributed results set

3. Design result
3.1 Connection result
3.2 Joint system result
3.3 Member result
3.4 Part result
3.5 Mapped result
3.6 Resolved result

4. Reaction
4.1 Force reaction
4.2 Displacement reaction
4.3 Velocity reaction
4.4 Acceleration reaction
4.5 Dynamic reaction
4.6 Equilibrium reaction

5. Resistance
5.1 Bending resisting
5.2 Shear resisting
5.3 Axial resisting

4.4 Parts

This section covers the description and specification of generic and specific parts, which are used to model structural and non-structural components that compose a bridge. Parts are classified by the complexity of their constitution as primitive or non-primitive. Primitive parts can be further categorized by their geometry as linear or surface. While non-primitive parts can be further classified as complex or derived. Complex parts are addition of two or more parts, and derived parts are parts that derived (subtracted) from another part by using features. Features will be introduced in ‘Feature’ section.

Primitive parts are usually described by using cross section profile and a path along which the section profile is extruded. This protocol supports generic and specific cross section profiles. Their definition will be introduced in the 'Geometry and Section' chapter. The path could be either straight or curved, and either two-dimensional or three-dimensional.

A part can have one and only one associated material. Therefore, a concrete deck with stay-in- place steel form needs to be modeled as a complex part, which is an assembly of two primitive parts.

The primary methods of fabrication used in the manufacturing process include rolling, welding, cold-forming, casting, extrusion, forging, etc.

4.4.1 Linear Part

Linear part is an element with one geometric dimension that is significantly larger than the other two. The direction, in which the dimension is in, is called longitudinal direction. Examples of linear part are girder, column, rebar, etc.

4.4.2 Prismatic Linear Part

Linear parts can be categorized by the variation of its cross section as prismatic part and non-prismatic part. Prismatic part is a type of part whose cross section is constant along its longitudinal direction. It should have at least the following geometric properties.

- Cross section profile
- Straight path: start point and length
- Curved path

4.4.2.1 Straight Prismatic Linear Part

One example of straight prismatic linear part is pier column. The case study bridge, Quincy Avenue over I-25 and LRT has five cast-in-place concrete columns and five drilled shafts in each pier, as shown in Figure 4-13. Typical sections of the columns, drilled shafts and piles from the contract plans are shown in Figure 4-14.

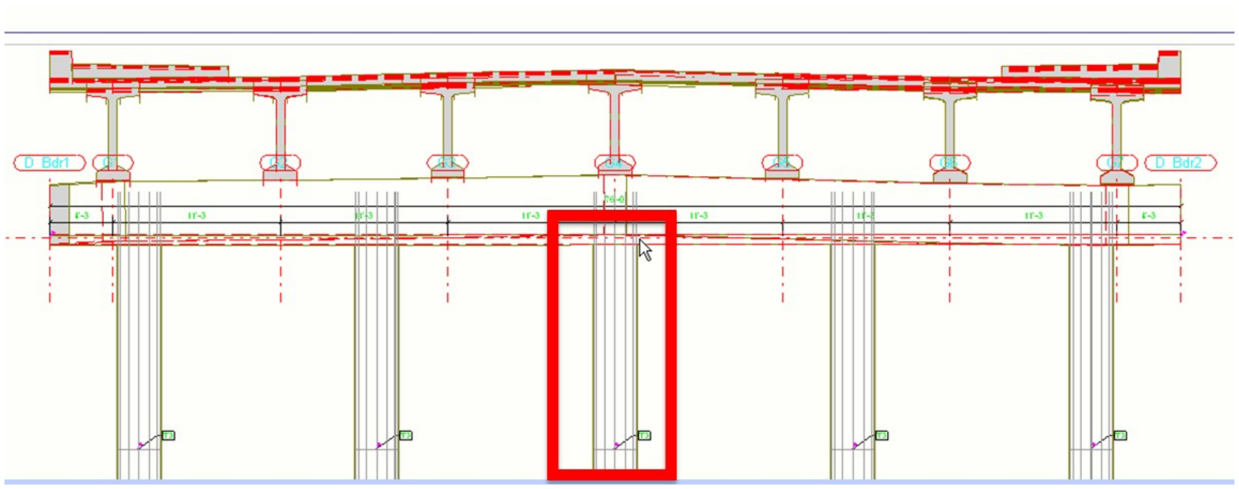


Figure 4-13. Concrete Column, Drilled Shaft and Pile

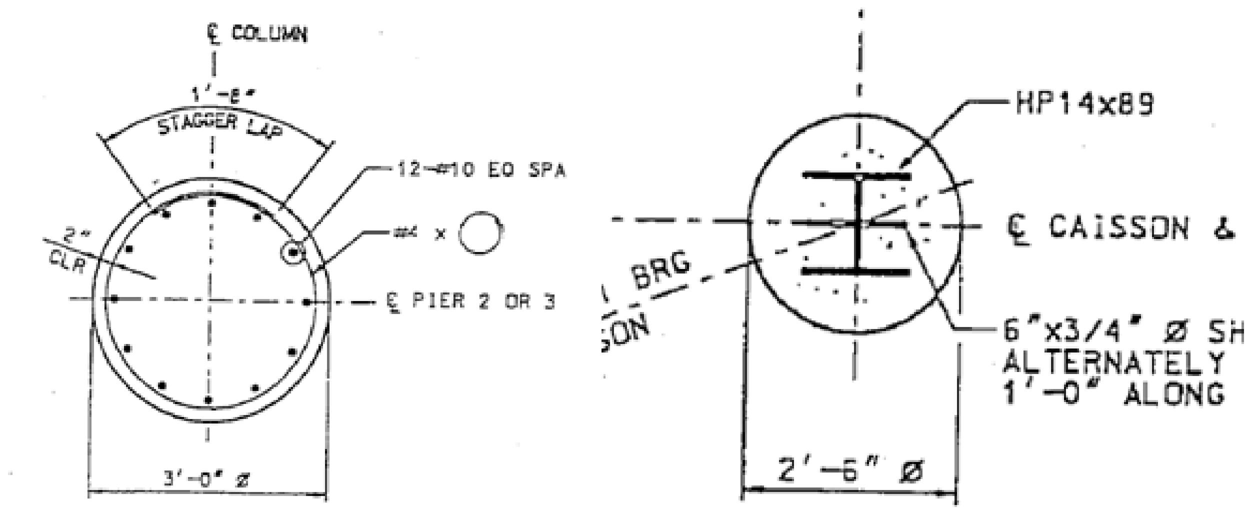


Figure 4-14. Typical Column, Drilled Shaft and Pile Sections from the Contract Plans

4.4.2.1.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="Linear Part">
<Param Name="r" Label="Radius" Value="2.5"/>
<Param Name="l" Label="Length" Value="10"/>
<Line>
<Point X="0" Y="0" Z="0"/>
<Point X="0" Y="0" Z="l"/>
<Circle X="0" Y="0" Radius="r"/>
</Line>
</Obj>
```

Figure 4-15 shows 3D view of a straight prismatic linear part using the OpenBrimViewer.

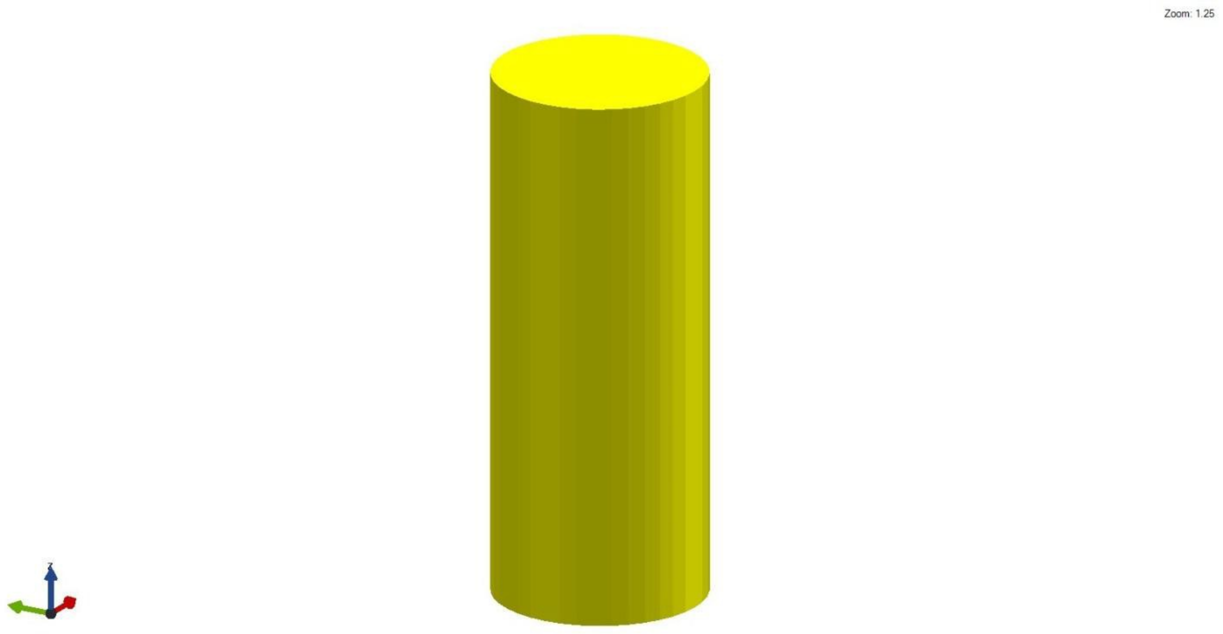


Figure 4-15. 3D View of a Straight Prismatic Part in the OpenBrIM Viewer

4.4.2.1.2 Mapping to ISM

Linear part can be modeled by using IIsmCurveMembers interface in the ISM schema. Major properties of IIsmCurveMembers are listed in Table 4-8.

Table 4-8. Major IIsmCurveMembers Properties [4]

Group	Property	Description
Physical Geometry	Section	Reference to an IIsmSection
Physical Geometry	Location	Member line
Physical Geometry	PlacementPoint	Cardinal point where Location is placed
Physical Geometry	Orientation	Orientation of cross section perpendicular to Location
Physical Geometry	MirrorShapeAboutYAxis	Section is mirrored about y-axis
Physical Geometry	Camber	Design camber
Material	SystemKind	Precast concrete, rolled steel, etc.
Material	Material	Reference to an IIsmCurveMemberMaterial
Other	Use	Beam, column, pile, etc.
Other	Member end fixity properties	Member end fixities
Other	AxialBehavior	Tension-only, compression-only or tension and compression

The main objective of the ISM repository is to store enough data so that a complete bridge model can be independently created using any modeling application. Therefore, a concentration on geometry properties would focus on Section, Location, PlacementPoint, Orientation, MirrorShapeAboutYAxis and Camber:

1) Section for Straight Prismatic Linear Part

For linear members with constant section, IIsmParametricSection and IIsmTableSection can be used to define the cross sections. The column and drilled shaft sections are modeled by using IIsmParametricSection as "solid circle." The pile section can be modeled by using IIsmTableSection. More details are described in the section for geometry and section.

2) Location for Straight Prismatic Linear Part

Member line of straight prismatic linear part such as column is located at the middle of the member, as shown in Figure 4-16. The member line is defined by specifying the location of two end points.

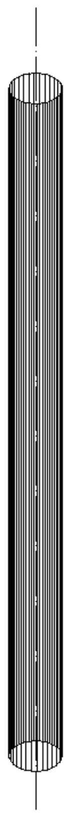


Figure 4-16. Member Line Location for Straight Prismatic Linear Part

3) Placement Point for Straight Prismatic Linear Part

Placement point of straight prismatic linear part such as column is located at the middle of the member section, as shown in Figure 4-17 and Figure 4-18.

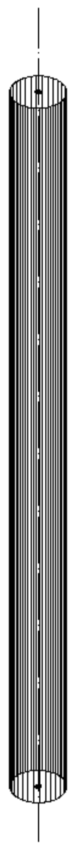


Figure 4-17. Placement Point for Straight Prismatic Linear Part (Isometric View)

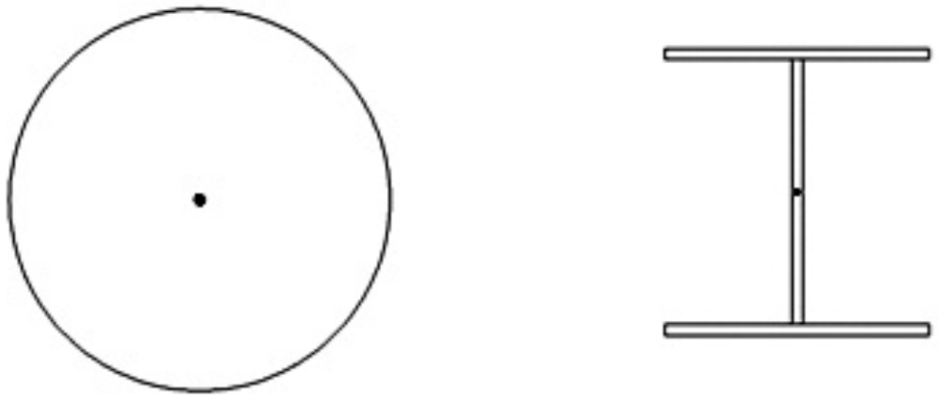


Figure 4-18. Placement Point for Straight Prismatic Linear Part (Section View)

4) Orientation for Straight Prismatic Linear Part

The orientations of cross sections of straight prismatic linear parts are perpendicular to their locations, as shown in Figure 4-19.

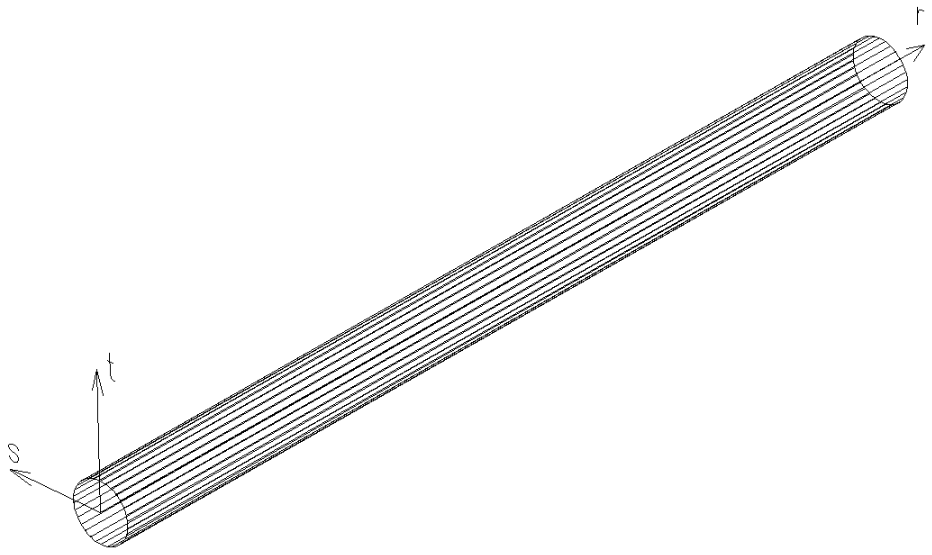


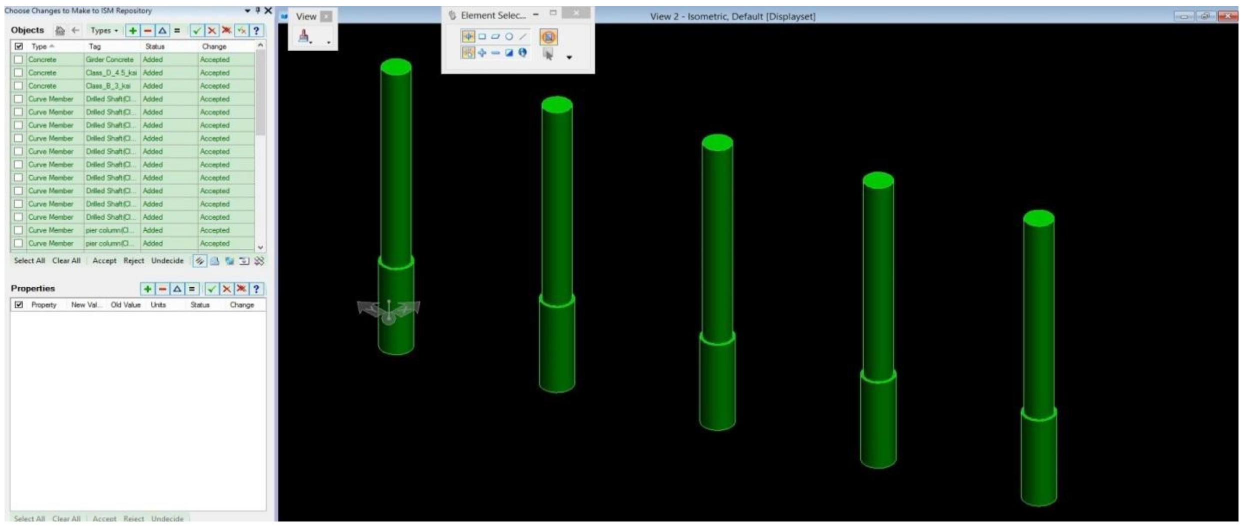
Figure 4-19. Orientation for Straight Prismatic Linear Part

5) MirrorShapeAboutYAxis for Straight Prismatic Linear Part

For linear parts such as column and pile, the Boolean value of this parameter is "True." The members sections are symmetric about Y axis.

6) Summary of Modeling Straight Prismatic Linear Part

From the analysis above, we can conclude that the for straight prismatic linear part can be modeled by using IIsmCurveMembers interface in the current ISM schema. The sections of straight prismatic linear part can be defined by using IIsmParametricSection or IIsmTableSection. Figures 4-20 and 4-21 show the ISM models of pier columns, drilled shafts and piles.


Choose Changes to Make to ISM Repository

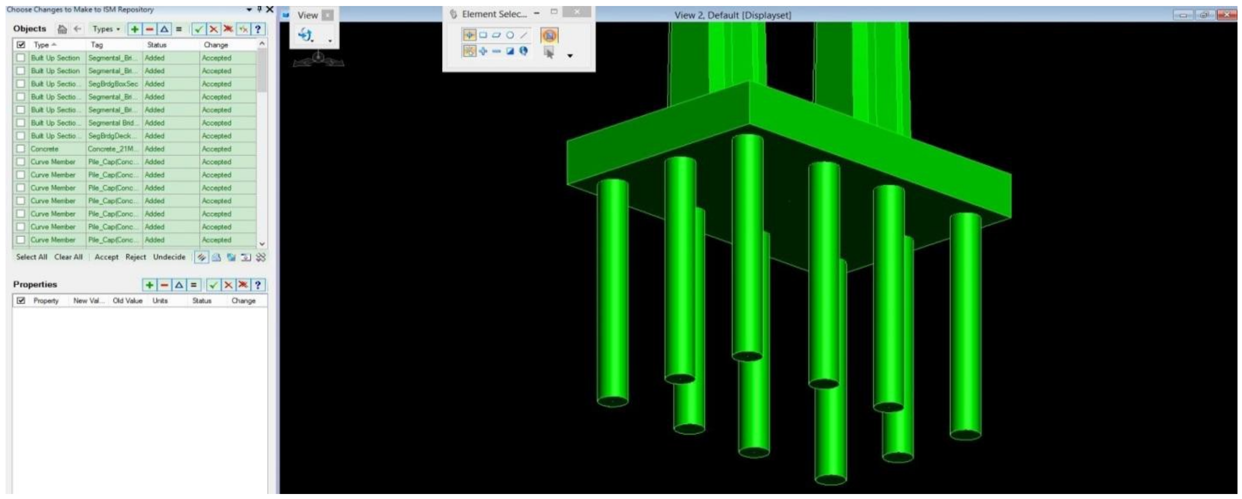
Objects

☑	Type	Tag	Status	Change
☐	Concrete	Girder Concrete	Added	Accepted
☐	Concrete	Class_D_4.5_ksi	Added	Accepted
☐	Concrete	Class_B_3_ksi	Added	Accepted
☐	Curve Member	Drilled Shaft(Cl...	Added	Accepted
☐	Curve Member	Drilled Shaft(Cl...	Added	Accepted
☐	Curve Member	Drilled Shaft(Cl...	Added	Accepted
☐	Curve Member	Drilled Shaft(Cl...	Added	Accepted
☐	Curve Member	Drilled Shaft(Cl...	Added	Accepted
☐	Curve Member	Drilled Shaft(Cl...	Added	Accepted
☐	Curve Member	Drilled Shaft(Cl...	Added	Accepted
☐	Curve Member	Drilled Shaft(Cl...	Added	Accepted
☐	Curve Member	Drilled Shaft(Cl...	Added	Accepted
☐	Curve Member	Drilled Shaft(Cl...	Added	Accepted
☐	Curve Member	pier column(Cl...	Added	Accepted
☐	Curve Member	pier column(Cl...	Added	Accepted

Properties

☑	Property	New Val...	Old Value	Units	Status	Change

Figure 4-20. ISM Models of Columns and Drilled Shafts



Choose Changes to Make to ISM Repository

Objects

☑	Type	Tag	Status	Change
☐	Built Up Section	Segmental_Bri...	Added	Accepted
☐	Built Up Section	Segmental_Bri...	Added	Accepted
☐	Built Up Sectio...	SegBrdgBoxSec	Added	Accepted
☐	Built Up Sectio...	Segmental_Bri...	Added	Accepted
☐	Built Up Sectio...	Segmental_Bri...	Added	Accepted
☐	Built Up Sectio...	Segmental Brid...	Added	Accepted
☐	Built Up Sectio...	SegBrdgDeck...	Added	Accepted
☐	Concrete	Concrete_21M...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted

Properties

☑	Property	New Val...	Old Value	Units	Status	Change

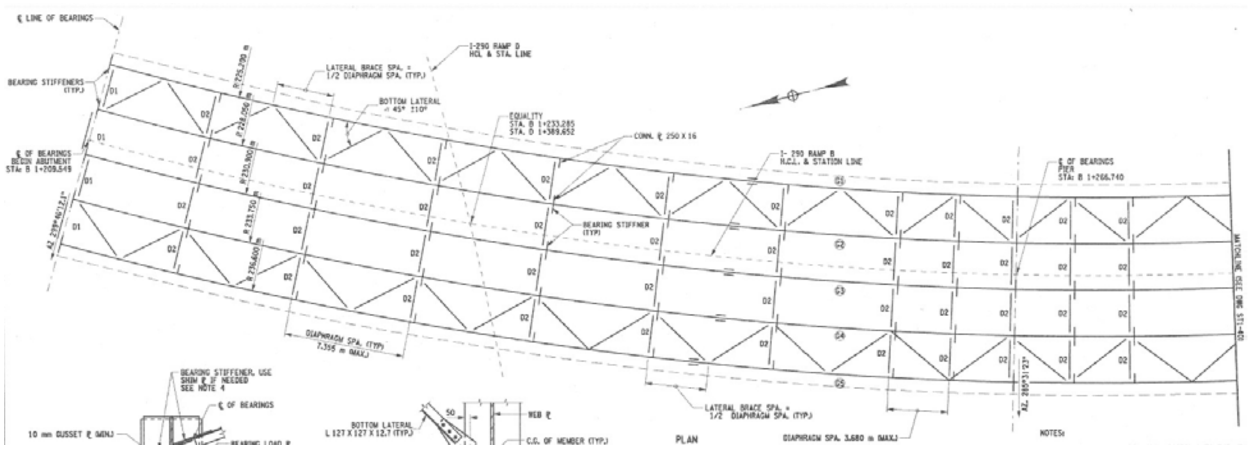
Figure 4-21. ISM Models of Piles

4.4.2.1.3 Mapping to IFC

IFC can model linear members by using IfcExtrudedAreaSolid. The IfcExtrudedAreaSolid directly defines the linear extrusion of a cross section (also referred to as profile). The extruded area solid defines the extrusion of a 2D area (given by a profile definition) by an direction and depth. The result is a solid.

4.4.2.2 Curved Prismatic Linear Part

One example of the curved prismatic linear part is a girder in a curved bridge, as shown in Figure 4-22. The only difference between modeling of curved prismatic linear part and modeling of straight prismatic linear part is the curved path (member line).



℄ LINE OF BEARINGS
BEARING STIFFENERS (TYP.)
LATERAL BRACE SPA. = 1/2 DIAPHRAGM SPA. (TYP.)
BOTTOM LATERAL
EQUALITY
CONN. ℄ 250 X 16
℄ OF BEARINGS BEGIN ABUTMENT
BEARING STIFFENER (TYP)
℄ OF BEARINGS PIER
DIAPHRAGM SPA. (TYP) 7.355 m (MAX.)
LATERAL BRACE SPA. = 1/2 DIAPHRAGM SPA. (TYP.)
DIAPHRAGM SPA. 3.680 m (MAX.)
PLAN
NOTES:

Figure 4-22. Curved Girder

4.4.2.2.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="Project 1" Alignment="Alg1">
<RoadwayGeometry Name="Alg1">
<HorizontalAlignment StartStation="0" StartAzimuth="0">
<Curve Type="Circular" Direction="Left" Length="500"
Radius="500"/>
</HorizontalAlignment>
</RoadwayGeometry>

<Obj Name="Linear Part">
<Param Name="r" Label="Radius" Value="2.5"/>
<Param Name="l" Label="Length" Value="500"/>
<Line Type="Curved">
<Point X="0" Y="0" Z="0"/>
<Point X="l" Y="0" Z="0"/>
<Circle X="0" Y="0" Radius="r" />
</Line>
</Obj>
</Obj>
```

Figure 4-23 shows 3D view of a curved prismatic linear part using the OpenBrimViewer.

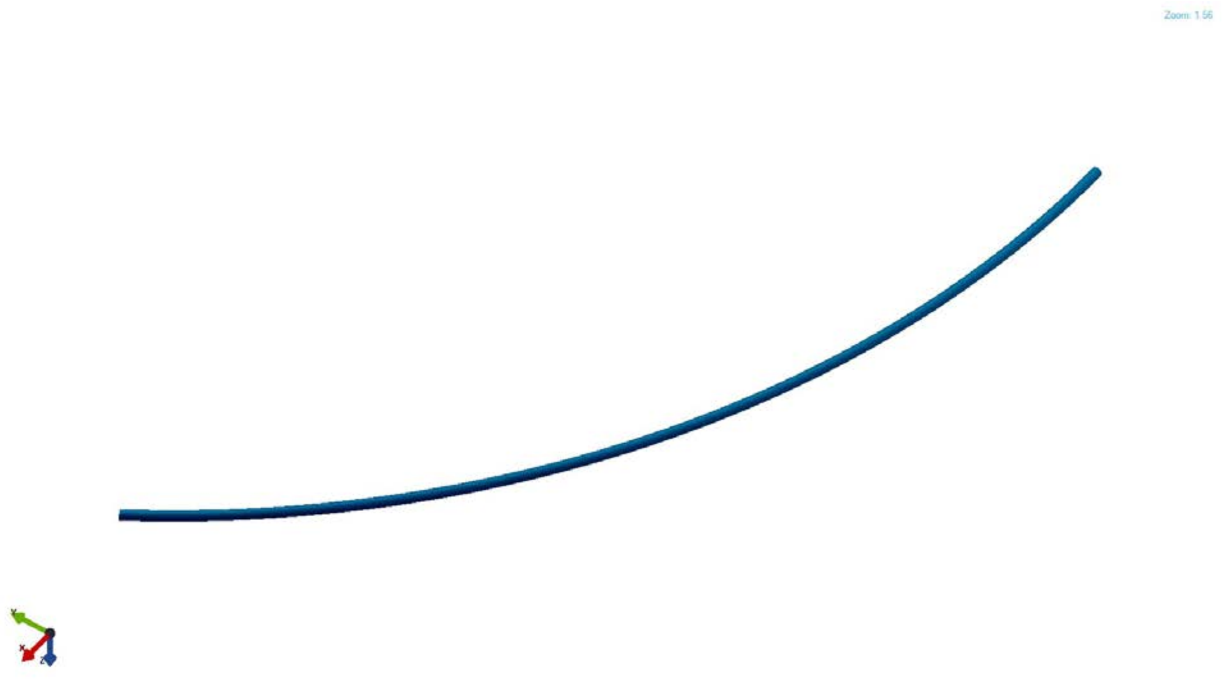


Figure 4-23. 3D View of A Curved Prismatic Part in the OpenBrIM Viewer

4.4.2.2.2 Mapping to ISM

Curved prismatic linear part can be modeled by using IIsmCurveMembers interface in the ISM schema. It is the same as modeling straight prismatic linear part except for the Location property. The Location of a curved part cannot be defined merely by the location of two end points, instead, it needs to be defined by using the implementations under the IIsmCurve3D interface, such as IIsmCircularArc3D. Figure 4-24 shows the ISM model of curved girders.

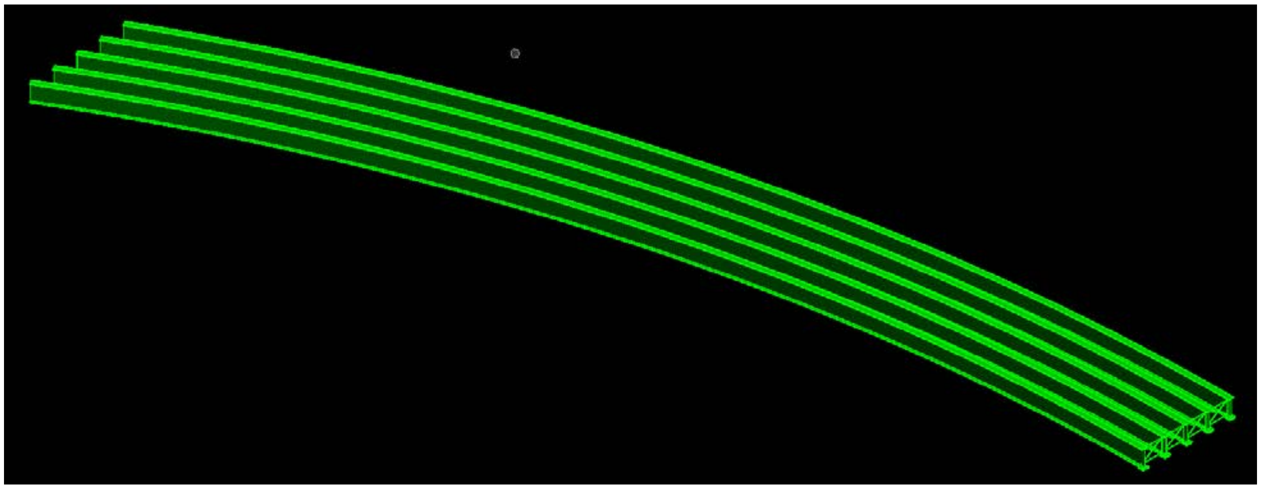


Figure 4-24. 3D View of A Curved Prismatic Part in the ISM Structural Synchronizer

4.4.2.2.3 Mapping to IFC

IFC can model curved members by using IfcExtrudedAreaSolid. The IfcExtrudedAreaSolid defines the extrusion of a cross section (also referred to as profile) along a B-spline curve. A B-spline curve is a piecewise parametric polynomial or rational curve described in terms of control points and basic functions. The B-spline curve has been selected as the most stable format to represent all types of polynomial or rational parametric curves. With appropriate attribute values, it is capable of representing single span or spline curves of explicit polynomial, rational, Bezier or B-spline type.

4.4.2.3 Non-Prismatic Linear Part

Non-prismatic linear part is a type of part whose geometry is defined as a set of cross section profiles located at a series of points on a curved path. Types of section variation include sudden, linear, parabolic, etc.

Non-prismatic linear part should have at least the following geometric attributes:

- Cross section profiles.
- Section orientations.
- Curved path.
- Straight path: start point and length.
- Locations of section.

One example of the non-prismatic linear part is the pier columns in the case study bridge, Otay River Bridge, which has multiple pier columns with a varying section. The section transition type is linear. Figure 4-25 shows the varying sections used for Otay River Bridge.

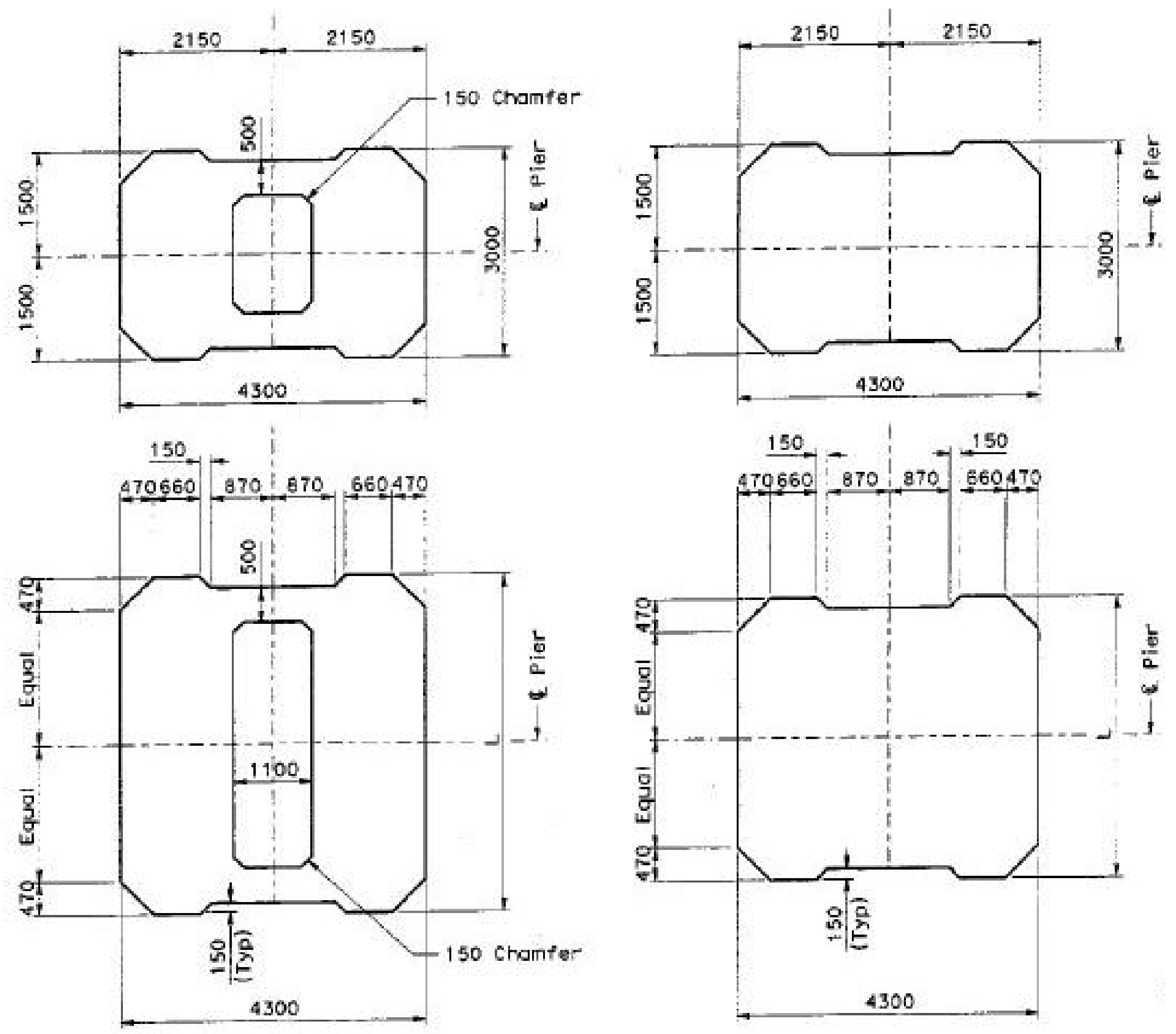


Figure 4-25. Varying Sections of Pier Column in Otay River Bridge

Another example of the non-prismatic linear part is steel plate girder. In order to save material, girder section varies along the girder length according to the moment and shear diagram calculated. This variation is reflected by flange transition. Figure 4-26 and Figure 4-27 show the section view and elevation view of steel plate girder, respectively.

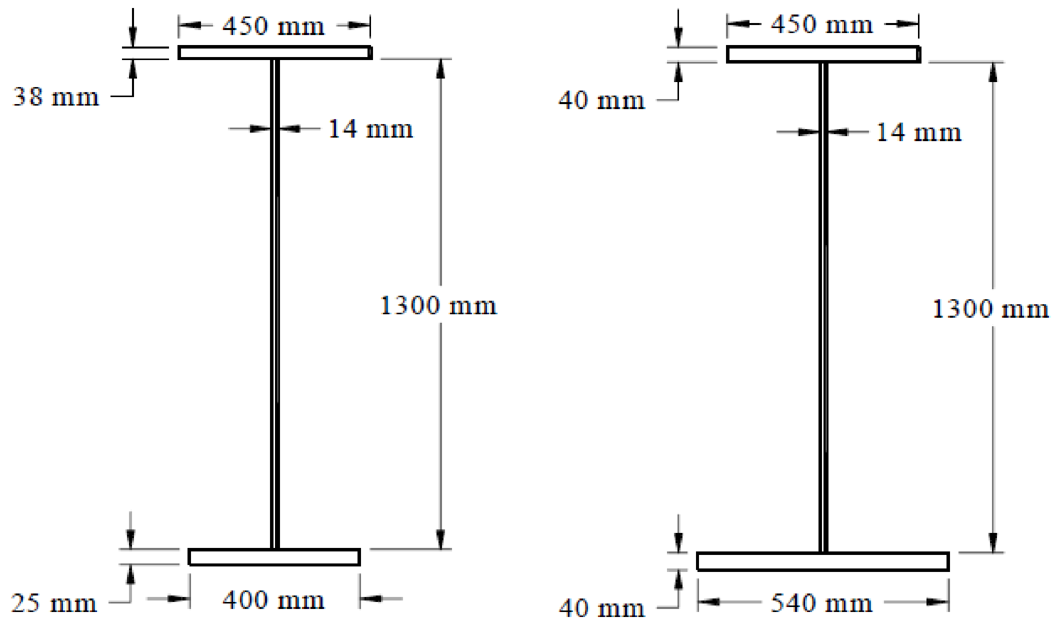


Figure 4-26. Section View of Steel Plate Girder

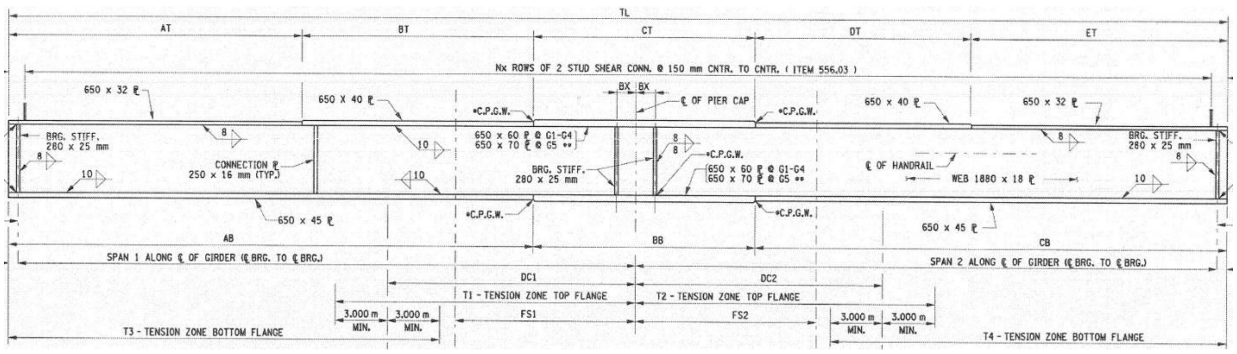

TL
AT
BT
CT
DT
ET
Nx ROWS OF 2 STUD SHEAR CONN. @ 150 mm CNTR. TO CNTR. (ITEM 556.03)
650 x 32 ℄
650 X 40 ℄
BX
BX
℄ OF PIER CAP
*C.P.G.W.
650 x 40 ℄
650 x 32 ℄
BRG. STIFF. 280 x 25 mm
CONNECTION ℄ 250 x 16 mm (TYP.)
650 x 60 ℄ @ G1-G4
650 x 70 ℄ @ G5 **
BRG. STIFF. 280 x 25 mm
℄ OF HANDRAIL
WEB 1880 x 18 ℄
650 x 45 ℄
AB
BB
CB
SPAN 1 ALONG ℄ OF GIRDER (℄ BRG. TO ℄ BRG.)
SPAN 2 ALONG ℄ OF GIRDER (℄ BRG. TO ℄ BRG.)
DC1
DC2
T1 - TENSION ZONE TOP FLANGE
T2 - TENSION ZONE TOP FLANGE
FS1
FS2
3.000 m MIN.
T3 - TENSION ZONE BOTTOM FLANGE
T4 - TENSION ZONE BOTTOM FLANGE

Figure 4-27. Elevation View of Steel Plate Girder

Sometimes, as a cost saving measure, the plate girder segments are built with different grades of steel (known as hybrid girder configurations). This is shown in Figure 4-28.

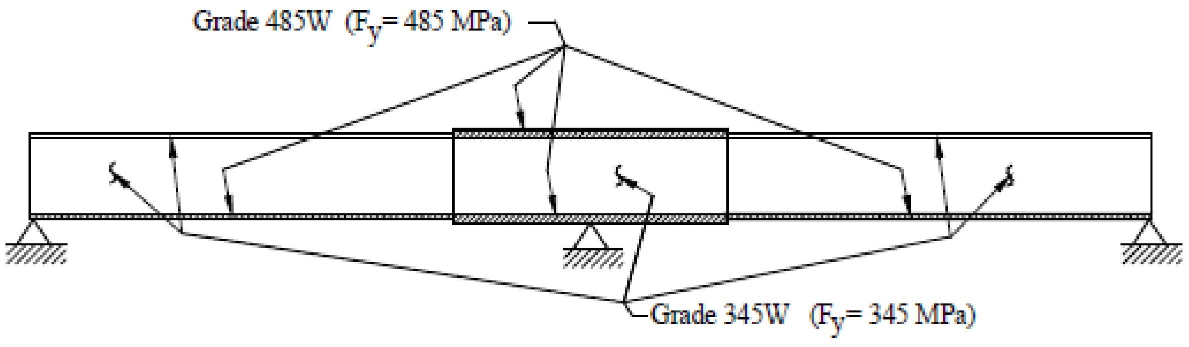


Figure 4-28. Hybrid Girder Configuration

4.4.2.3.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="Girders">
<ObjType Name="SteelPlateGirder">
<Param Name="l" Label="Length" Value="121372"/>
<Param Name="toff" Label="Transverse Offset" Value="-4800"/>
<Param Name="wd" Label="WebDepth" Value="2000"/>
<Param Name="tw" Label="WebThickness" Value="18"/>
<Param Name="tbf" Label="TopFlangeWidth" Value="650"/>
<Param Name="bbf" Label="BottomFlangeWidth" Value="700"/>
<Param Name="ttf" Label="TopFlangeThickness" Value="32"/>
<Param Name="btf" Label="BottomFlangeThickness" Value="45"/>
<Line Type="Curved" Color="Yellow">
<Point X="toff*tan(BeginAbutmentSkewAngle)" Y="0"
Z="0"/>
<Point X="l+toff*tan(EndAbutmentSkewAngle)" Y="0"
Z="0"/>
<Polygon>
<Point X="-tbf/2" Y="ttf"/>
<Point X="-tbf/2" Y="0"/>
<Point X="-tw/2" Y="0"/>
<Point X="-tw/2" Y="-wd"/>
<Point X="-bbf/2" Y="-wd"/>
<Point X="-bbf/2" Y="-wd-btf"/>
<Point X="bbf/2" Y="-wd-btf"/>
<Point X="bbf/2" Y="-wd"/>
<Point X="tw/2" Y="-wd"/>
<Point X="tw/2" Y="0"/>
<Point X="tbf/2" Y="0"/>
<Point X="tbf/2" Y="ttf"/>
</Polygon>
</Line>
</ObjType>

<Obj Name="Girder5" RefObj="SteelPlateGirder"
Sta="BeginAbutmentSta" TOff="toff" Elev="0">
<Param Name="toff" Value="-4800"/>
<Param Name="ttf"
Value="[16003~47852@40][47852~65979@60][65979~102950@40][32]"/>
<Param Name="btf" Value="[47852~65979@60][45]"/>
</Obj>
</Obj>
```

The OpenBrIM model of the plate girder is shown in Figure 4-29.

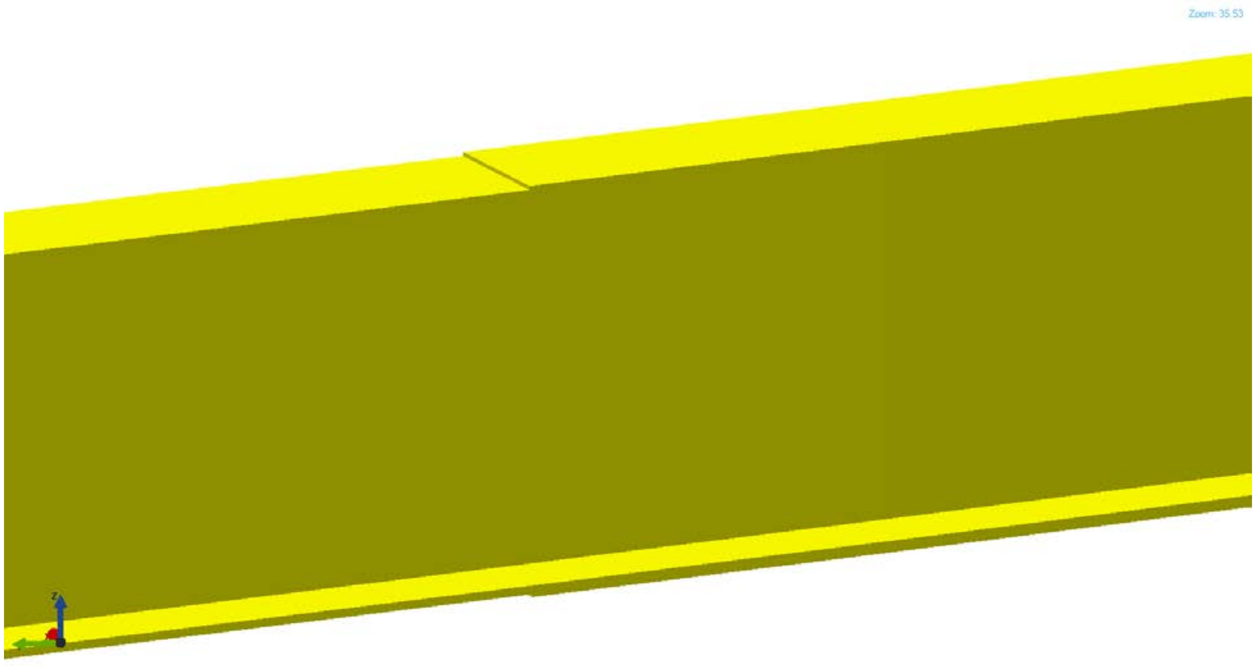
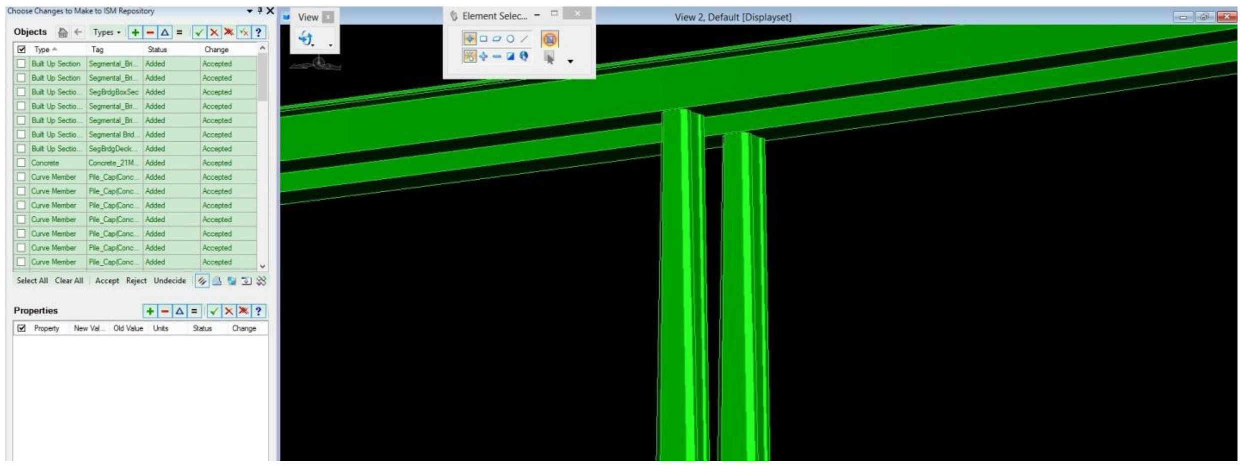


Figure 4-29. OpenBrIM Plate Girder Model

4.4.2.3.2 Mapping to ISM

A non-prismatic linear part can be modeled by using IIsmCurveMembers interface in the ISM schema. It is the same as modeling prismatic linear part except for the Section property. The Section property should be defined by using IIsmVaryingSection interface which consists of several cross sections defined by using IIsmParametricSection or IIsmTableSection interface. Figure 4-30 shows the ISM model of the pier column with a varying section.



Type	Tag	Status	Change
Built Up Section	Segmental_Bri...	Added	Accepted
Built Up Section	Segmental_Bri...	Added	Accepted
Built Up Sectio...	SegBrdgBoxSec	Added	Accepted
Built Up Sectio...	Segmental_Bri...	Added	Accepted
Built Up Sectio...	Segmental_Bri...	Added	Accepted
Built Up Sectio...	Segmental Brid...	Added	Accepted
Built Up Sectio...	SegBrdgDeck...	Added	Accepted
Concrete	Concrete_21M...	Added	Accepted
Curve Member	Pile_Cap(Conc...	Added	Accepted
Curve Member	Pile_Cap(Conc...	Added	Accepted
Curve Member	Pile_Cap(Conc...	Added	Accepted
Curve Member	Pile_Cap(Conc...	Added	Accepted
Curve Member	Pile_Cap(Conc...	Added	Accepted
Curve Member	Pile_Cap(Conc...	Added	Accepted
Curve Member	Pile_Cap(Conc...	Added	Accepted

Property	New Val...	Old Value	Units	Status	Change

Figure 4-30. 3D View of a Non-Prismatic Linear Part in the ISM Structural Synchronizer

The hybrid plate girder cannot be modeled as one curve member by using IIsmCurveMembers and IIsmVaryingSection interfaces. Because a single curve member made of multiple materials is not supported in ISM 3.0 [4]. There are two workarounds for this problem:

- One way to solve this problem is to create an interface (maybe called IIsmGroupMembers) which combines several curve members together to form a new member. Modeling hybrid plate girder can be separated into defining flanges and web as curve members and then combining them using IIsmGroupMembers interface. This method is called schedule-based modeling, which is used by AASHTOWare Opis/Virtis.

- The other method is to create an interface IIsmSectionWithMaterial that can assign different material properties to different part of the girder section.

4.4.2.4 Castellated Linear Part

Castellated linear part is a type of linear part that has a repeating feature along its length. The repeating feature may be circular, hexagonal, or octagonal holes (Figures 4-31 to 4-33). Castellated linear part should have at least the following geometric attributes:

- Castellation type.
- Horizontal end post.
- Vertical end post.
- Castellation spacing.
- Castellation dimension.

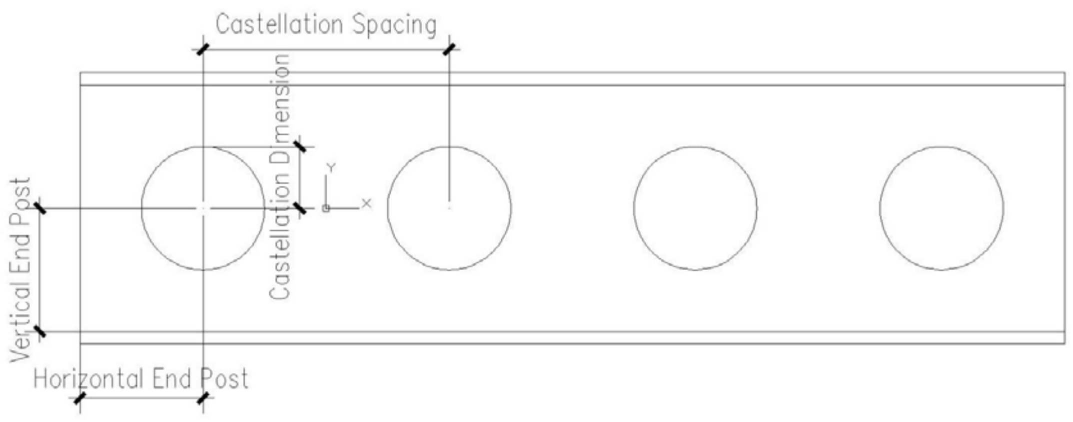


Figure 4-31. Circular Castellation Linear Part Properties

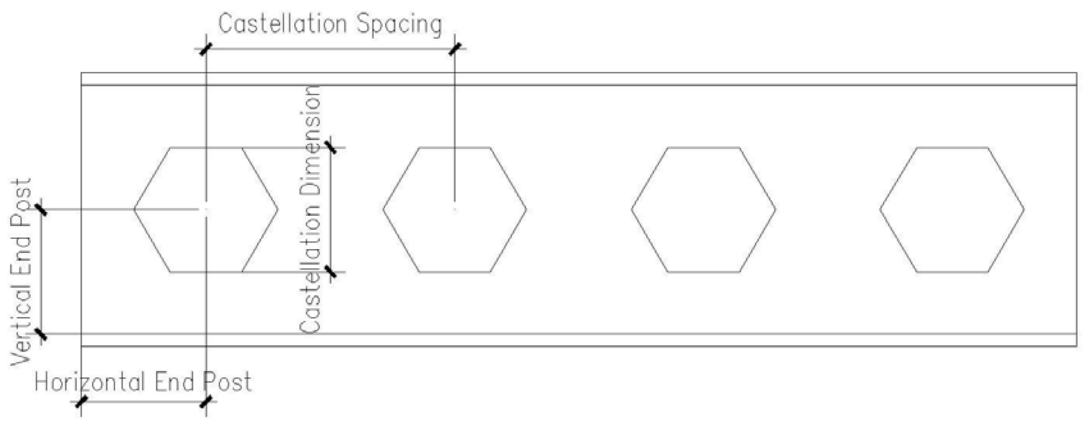


Figure 4-32. Hexagonal Castellation Linear Part Properties

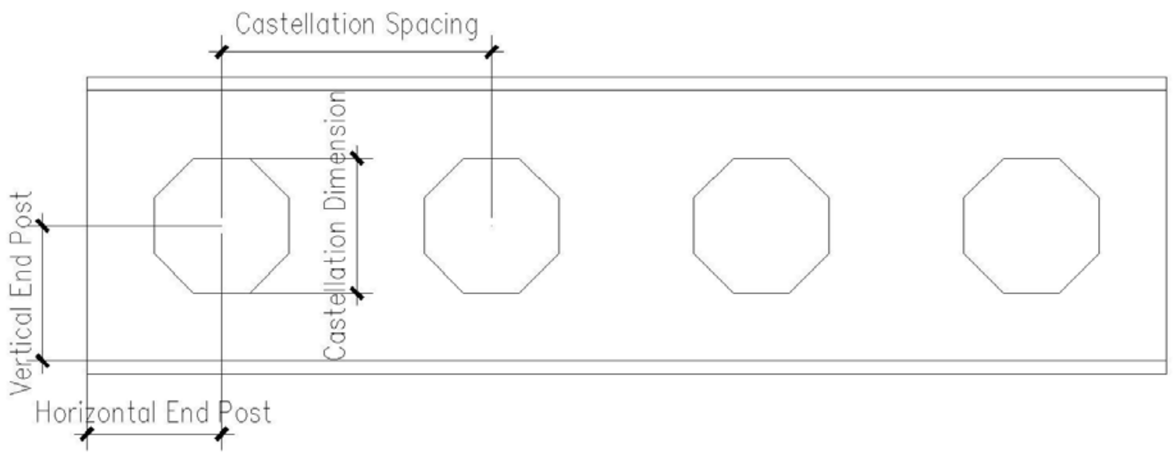


Figure 4-33. Octagonal Castellation Linear Part Properties

4.4.2.4.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="Girder1">
<Param Name="l" Label="Length" Value="50000"/>
<Param Name="wd" Label="WebDepth" Value="2000"/>
<Param Name="tw" Label="WebThickness" Value="18"/>
<Param Name="tbf" Label="TopFlangeWidth" Value="650"/>
<Param Name="bbf" Label="BottomFlangeWidth" Value="700"/>
<Param Name="ttf" Label="TopFlangeThickness" Value="32"/>
<Param Name="btf" Label="BottomFlangeThickness" Value="45"/>
<Param Name="hep" Label="HorizontalEndPost" Value="1500"/>
<Param Name="vep" Label="VerticalEndPost" Value="1000"/>
<Param Name="cs" Label="CastellationSpacing" Value="2000"/>
<Param Name="cd" Label="CastellationDimension" Value="1000"/>

<Volume Color="Yellow">
<Surface Name="tfSurface" Z="0">
<Point X="0" Y="-tbf/2" Z="0"/>
<Point X="l" Y="-tbf/2" Z="0"/>
<Point X="l" Y="tbf/2" Z="0"/>
<Point X="0" Y="tbf/2" Z="0"/>
</Surface>
<Surface RefObj="tfSurface" Z="ttf"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="webSurface" Y="-tw/2">
<Point X="0" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="0"/>
<Point X="0" Y="0" Z="0"/>
<Repeat Param="var1" StartValue="0" EndValue="5"
Increment="1">
```

```
<Polygon>
<Point X="hep+(cd/2)*Cos(pi*0.0)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*0.0)"/>
<Point X="hep+(cd/2)*Cos(pi*0.1)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*0.1)"/>
<Point X="hep+(cd/2)*Cos(pi*0.2)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*0.2)"/>
<Point X="hep+(cd/2)*Cos(pi*0.3)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*0.3)"/>
<Point X="hep+(cd/2)*Cos(pi*0.4)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*0.4)"/>
<Point X="hep+(cd/2)*Cos(pi*0.5)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*0.5)"/>
<Point X="hep+(cd/2)*Cos(pi*0.6)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*0.6)"/>
<Point X="hep+(cd/2)*Cos(pi*0.7)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*0.7)"/>
<Point X="hep+(cd/2)*Cos(pi*0.8)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*0.8)"/>
<Point X="hep+(cd/2)*Cos(pi*0.9)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*0.9)"/>
<Point X="hep+(cd/2)*Cos(pi*1.0)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*1.0)"/>
<Point X="hep+(cd/2)*Cos(pi*1.1)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*1.1)"/>
<Point X="hep+(cd/2)*Cos(pi*1.2)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*1.2)"/>
<Point X="hep+(cd/2)*Cos(pi*1.3)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*1.3)"/>
<Point X="hep+(cd/2)*Cos(pi*1.4)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*1.4)"/>
<Point X="hep+(cd/2)*Cos(pi*1.5)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*1.5)"/>
<Point X="hep+(cd/2)*Cos(pi*1.6)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*1.6)"/>
<Point X="hep+(cd/2)*Cos(pi*1.7)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*1.7)"/>
<Point X="hep+(cd/2)*Cos(pi*1.8)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*1.8)"/>
<Point X="hep+(cd/2)*Cos(pi*1.9)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*1.9)"/>
<Point X="hep+(cd/2)*Cos(pi*2.0)+cs*var1"
Y="0" Z="-vep+(cd/2)*Sin(pi*2.0)"/>
</Polygon>
</Repeat>
</Surface>
<Surface RefObj="webSurface" Y="tw"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="bfSurface" Z="-wd-btf">
<Point X="0" Y="-bbf/2" Z="0"/>
```

```
<Point X="l" Y="-bbf/2" Z="0"/>
<Point X="l" Y="bbf/2" Z="0"/>
<Point X="0" Y="-bbf/2" Z="0"/>
<Point X="0" Y="-bbf/2" Z="0"/>
</Surface>
</Surface RefObj="bfSruface" Z="wd"/>
</Volume>
</Obj>
```

Figure 4-34 shows 3D view of the model of a circular castellation linear part using the OpenBrimViewer.

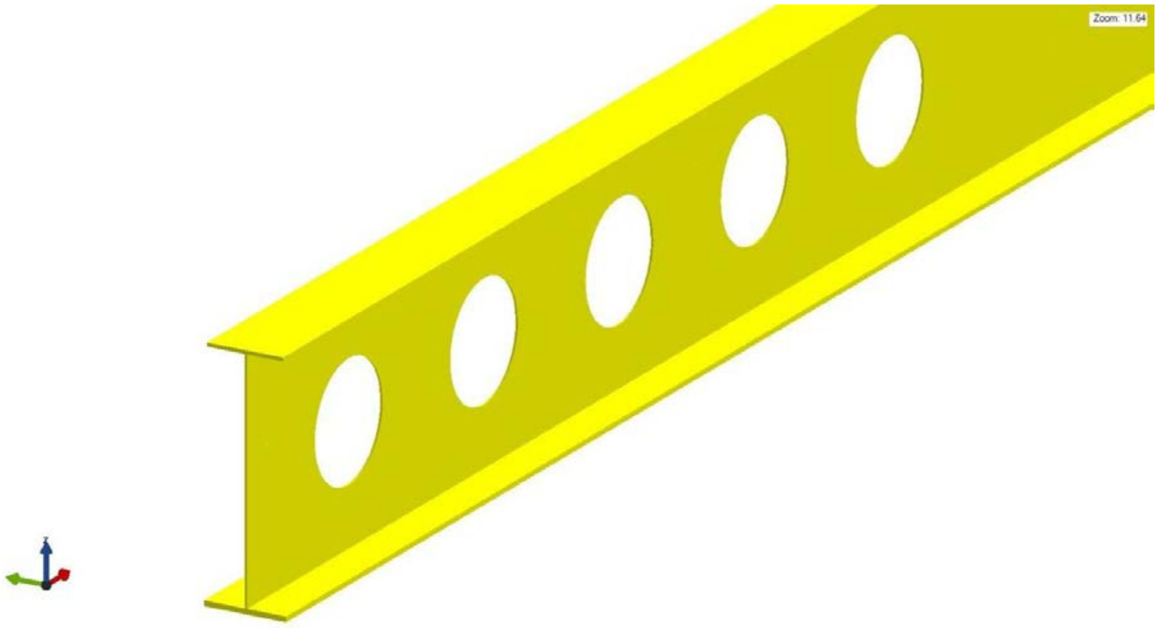


Figure 4-34. 3D View of a Circular Castellation Linear Part in the OpenBrIM Viewer

```
<Obj Name="Girder1">
<Param Name="l" Label="Length" Value="50000"/>
<Param Name="wd" Label="WebDepth" Value="2000"/>
<Param Name="tw" Label="WebThickness" Value="18"/>
<Param Name="tbf" Label="TopFlangeWidth" Value="650"/>
<Param Name="bbf" Label="BottomFlangeWidth" Value="700"/>
<Param Name="ttf" Label="TopFlangeThickness" Value="32"/>
<Param Name="btf" Label="BottomFlangeThickness" Value="45"/>
<Param Name="hep" Label="HorizontalEndPost" Value="1500"/>
<Param Name="vep" Label="VerticalEndPost" Value="1000"/>
<Param Name="cs" Label="CastellationSpacing" Value="2000"/>
<Param Name="cd" Label="CastellationDimension" Value="1000"/>
<Volume Color="Yellow">
<Surface Name="tfSurface" Z="0">
<Point X="0" Y="-tbf/2" Z="0"/>
<Point X="l" Y="-tbf/2" Z="0"/>
<Point X="l" Y="tbf/2" Z="0"/>
<Point X="0" Y="tbf/2" Z="0"/>
```

```
</Surface>
<Surface RefObj="tfSurface" Z="ttf"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="webSurface" Y="-tw/2">
<Point X="0" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="0"/>
<Point X="0" Y="0" Z="0"/>
<Repeat Param="var1" StartValue="0" EndValue="5"
Increment="1">
<Polygon>
<Point
X="hep+cs*0.0+(cd/2)*tan(pi/6)+cs*var1" Y="0" Z="-vep+(cd/2)"/>
<Point
X="hep+cs*0.0+(cd/2)/cos(pi/6)+cs*var1" Y="0" Z="-vep"/>
<Point
X="hep+cs*0.0+(cd/2)*tan(pi/6)+cs*var1" Y="0" Z="-vep-(cd/2)"/>
<Point X="hep+cs*0.0-
(cd/2)*tan(pi/6)+cs*var1" Y="0" Z="-vep-(cd/2)"/>
<Point X="hep+cs*0.0-
(cd/2)/cos(pi/6)+cs*var1" Y="0" Z="-vep"/>
<Point X="hep+cs*0.0-
(cd/2)*tan(pi/6)+cs*var1" Y="0" Z="-vep+(cd/2)"/>
</Polygon>
</Repeat>
</Surface>
<Surface RefObj="webSurface" Y="tw"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="bfSurface" Z="-wd-btf">
<Point X="0" Y="-bbf/2" Z="0"/>
<Point X="l" Y="-bbf/2" Z="0"/>
<Point X="l" Y="bbf/2" Z="0"/>
<Point X="0" Y="bbf/2" Z="0"/>
</Surface>
<Surface RefObj="bfSurface" Z="-wd"/>
</Volume>
</Obj>
```

Figure 4-35 shows 3D view of the model of a hexagonal castellation linear part using the OpenBrimViewer.

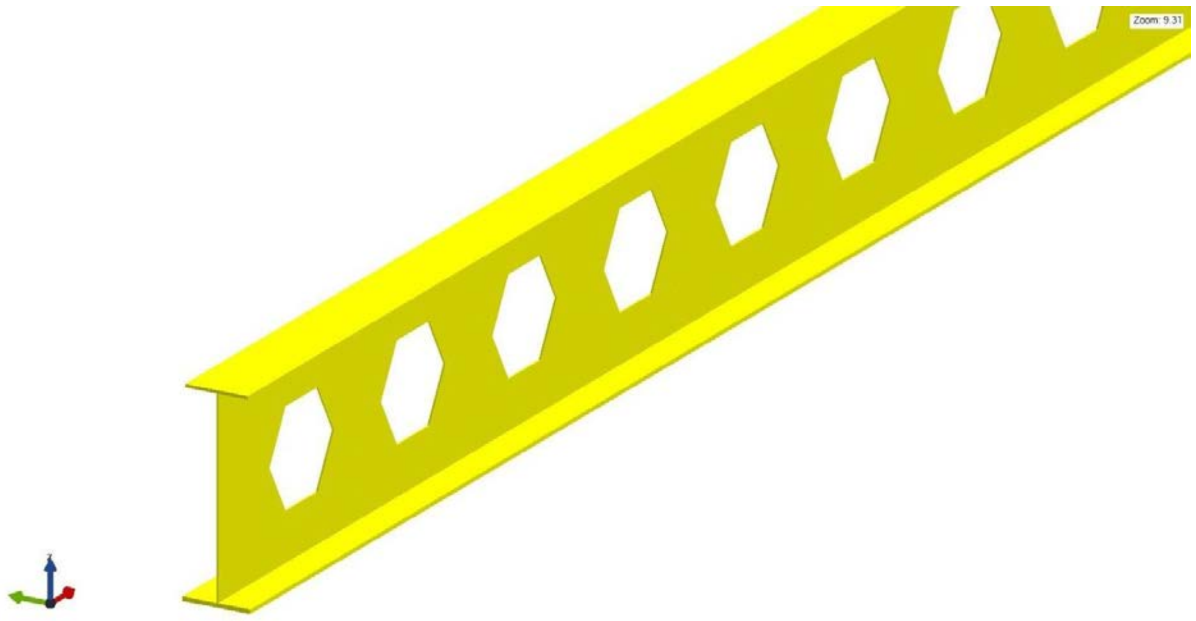


Figure 4-35. 3D View of a Hexagonal Castellation Linear Part in the OpenBrIM Viewer

```
<Obj Name="Girder1">
<Param Name="l" Label="Length" Value="50000"/>
<Param Name="wd" Label="WebDepth" Value="2000"/>
<Param Name="tw" Label="WebThickness" Value="18"/>
<Param Name="tbf" Label="TopFlangeWidth" Value="650"/>
<Param Name="bbf" Label="BottomFlangeWidth" Value="700"/>
<Param Name="ttf" Label="TopFlangeThickness" Value="32"/>
<Param Name="btf" Label="BottomFlangeThickness" Value="45"/>
<Param Name="hep" Label="HorizontalEndPost" Value="1500"/>
<Param Name="vep" Label="VerticalEndPost" Value="1000"/>
<Param Name="cs" Label="CastellationSpacing" Value="2000"/>
<Param Name="cd" Label="CastellationDimension" Value="1000"/>
<Volume Color="Yellow">
<Surface Name="tfSurface" Z="0">
<Point X="0" Y="-tbf/2" Z="0"/>
<Point X="l" Y="-tbf/2" Z="0"/>
<Point X="l" Y="tbf/2" Z="0"/>
<Point X="0" Y="tbf/2" Z="0"/>
</Surface>
<Surface RefObj="tfSurface" Z="ttf"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="webSurface" Y="-tw/2">
<Point X="0" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="0"/>
<Point X="0" Y="0" Z="0"/>
<Repeat Param="var1" StartValue="0" EndValue="5"
```

```
Increment="1">
<Polygon>
<Point
X="hep+cs*0.0+(cd/2)*tan(pi/8)+cs*var1" Y="-tw/2" Z="-vep+(cd/2)"/>
<Point
X="hep+cs*0.0+(cd/2)+cs*var1" Y="- tw/2" Z="-vep+(cd/2)*tan(pi/8)"/>
<Point X="hep+cs*0.0+(cd/2)+cs*var1" Y="-
tw/2" Z="-vep-(cd/2)*tan(pi/8)"/>
<Point X="hep+cs*0.0+(cd/2)*tan(pi/8)+cs*var1"
Y="-tw/2" Z="-vep-(cd/2)"/>
<Point X="hep+cs*0.0(cd/2)*tan(pi/8)+cs*var1"
Y="-tw/2" Z="-vep-(cd/2)"/>
<Point X="hep+cs*0.0-
cd/2)+cs*var1" Y="- tw/2" Z="-vep-(cd/2)*tan(pi/8)"/>
<Point X="hep+cs*0.0-(cd/2)+cs*var1" Y="-
tw/2" Z="-vep+(cd/2)*tan(pi/8)"/>
<Point X="hep+cs*0.0-(cd/2)*tan(pi/8)+cs*var1"
Y="-tw/2" Z="-vep+(cd/2)"/>
</Polygon>
</Repeat>
</Surface>
<Surface RefObj="webSurface" Y="tw"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="bfSurface" Z="-wd-btf">
<Point X="0" Y="-bbf/2" Z="0"/>
<Point X="l" Y="-bbf/2" Z="0"/>
<Point X="l" Y="bbf/2" Z="0"/>
<Point X="0" Y="bbf/2" Z="0"/>
</Surface>
<Surface RefObj="bfSurface" Z="-wd"/>
</Volume>
</Obj>
```

Figure 4-36 shows 3D view of the model of an octagonal castellation linear part using the OpenBrimViewer.

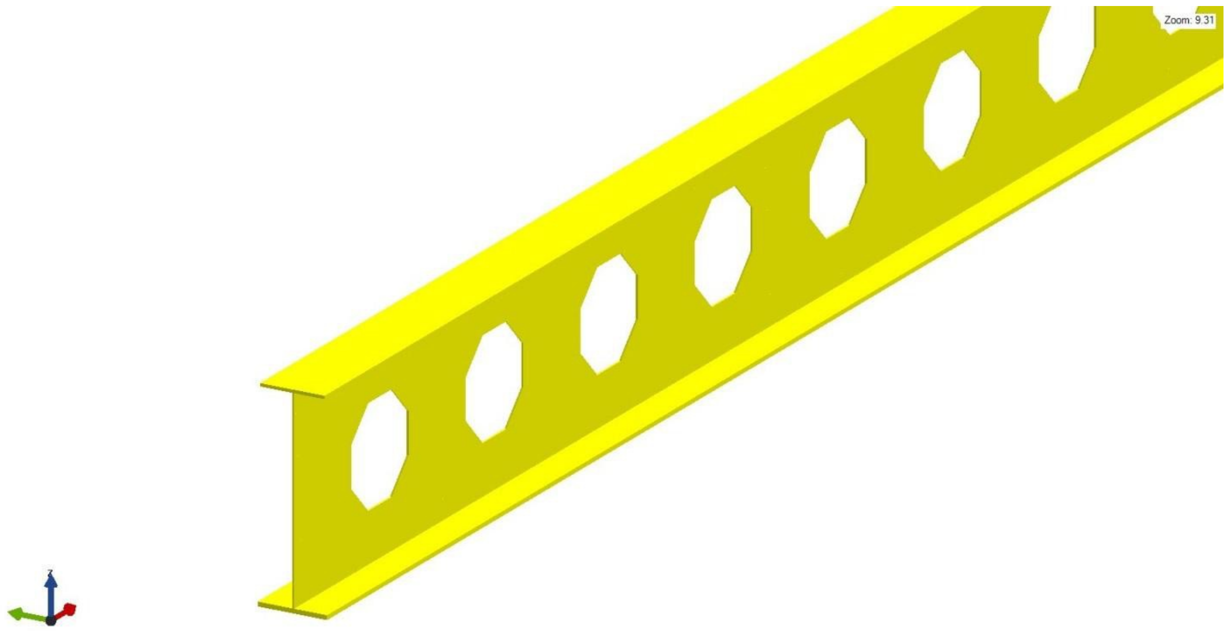


Figure 4-36. 3D View of an Octagonal Castellation Linear Part in the OpenBrIM Viewer

4.4.2.4.2 Mapping to ISM

A Castellation linear part can be modeled by using IIsmCurveMembers interface. The voids can be modeled by using IIsmFeatureSubtraction interface.

4.4.2.4.3 Mapping to IFC

A Solid member in IFC can be modeled using IfcExtrudedAreaSolid. If the planar area has inner boundaries, i.e., holes defined, then those holes shall be swept into holes of the solid.

4.4.3 Surface Part

Otay River Bridge has multiple pile caps, as shown in Figure 4-37. Typical section of the pile caps from the contract plans of Otay River Bridge is shown in Figures 4-38 and 4-39.

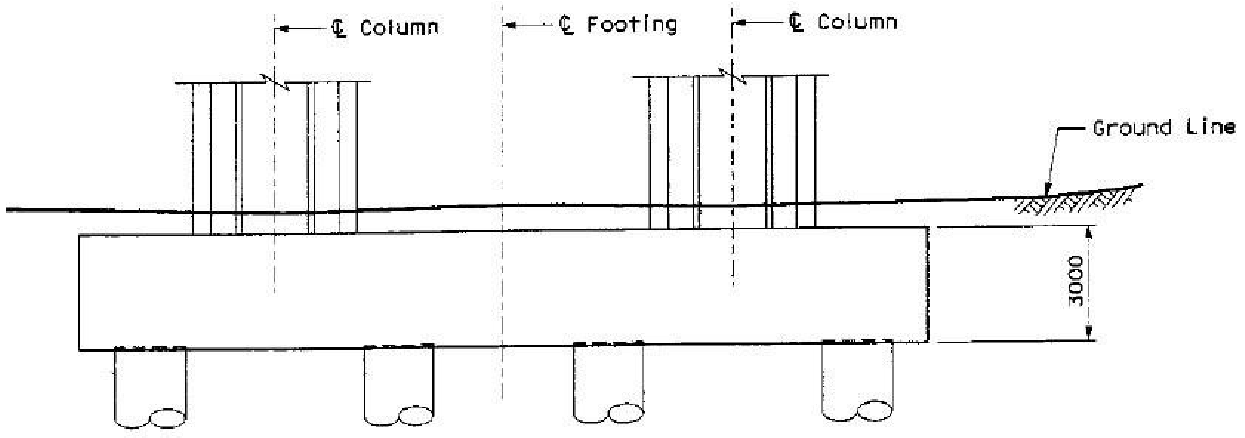


Figure 4-37. Concrete Pile Cap

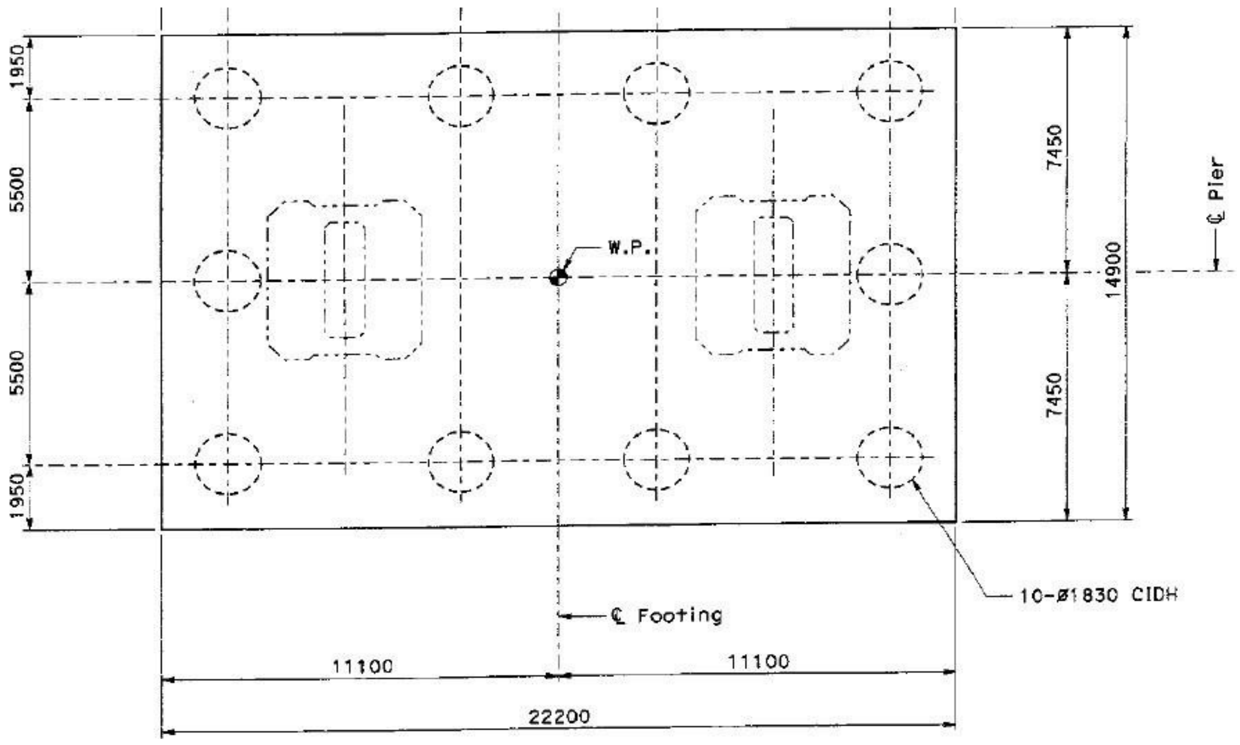


Figure 4-38. Typical Pile Cap Section for Otay River Bridge

1. Bounded surface part
1.1 Complex bounded surface part
1.2 Simple bounded surface part

2. Profiled surface part
2.1 Sheet profile

4.4.3.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="Deck" Sta="BridgeBeginsSta" TOff="girder.toff"
Elev="girder.Z+girder.Haunch.ht">
<Param Name="girder" Value="Girder3" Type="Reference"/>
<Param Name="dl" Label="Deck Length" Value="BridgeEndsSta-
BridgeBeginsSta"/>
<Param Name="dw" Label="Deck Slab Width" Value="12914"/>
<Param Name="dt" Label="Deck Slab Thickness" Value="240"/>
<Volume Type="Curved" Color="#8fc8d8">
<Surface Name="DeckBottomSurface" Type="Curved">
<Point X="-dw/2*tan(BeginAbutmentSkewAngle)" Y="-
dw/2"/>
<Point X="dl-dw/2*tan(EndAbutmentSkewAngle)" Y="-
dw/2"/>
<Point X="dl+dw/2*tan(EndAbutmentSkewAngle)"
Y="dw/2"/>
<Point X="dw/2*tan(BeginAbutmentSkewAngle)"
Y="dw/2"/>
</Surface>
<Surface RefObj="DeckBottomSurface"
Elev="girder.Z+girder.Haunch.ht+dt" Type="Curved"/>
</Volume>
</Obj>
```

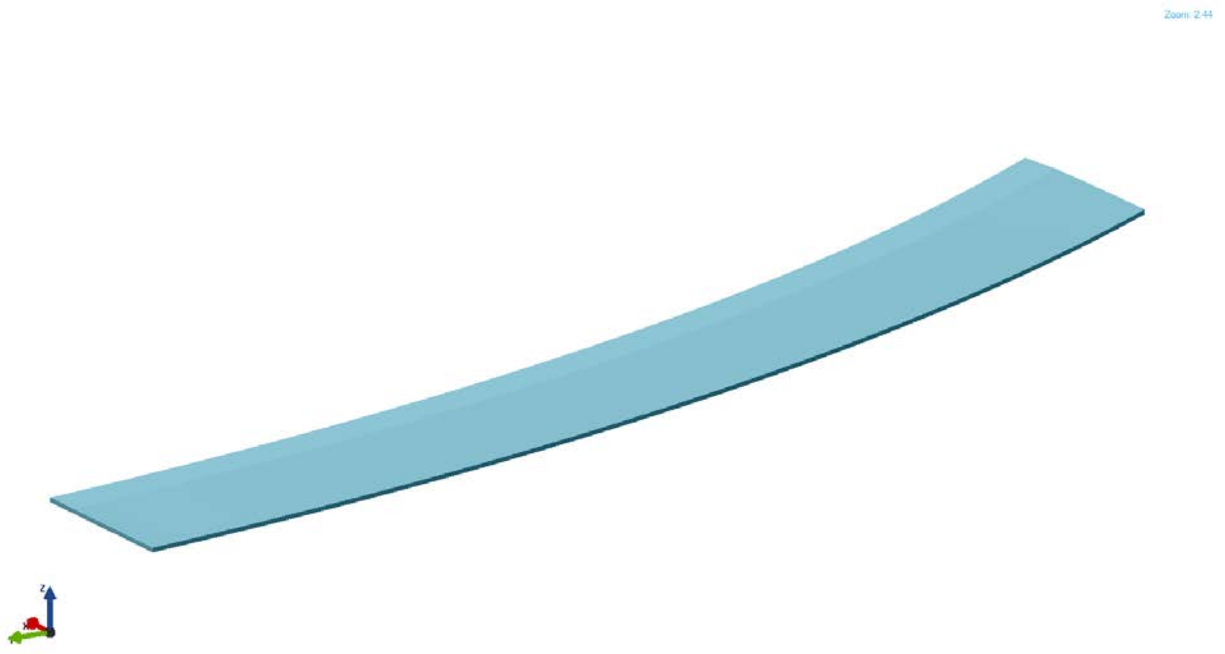


Figure 4-39. Typical Pile Cap Section for Otay River Bridge

4.4.3.2 Mapping to ISM

Surface part can be modeled as surface members by using IIsmSurfaceMembers interface.

1) Physical Geometry Properties of IIsmSurfaceMembers

Physical geometry properties of IIsmSurfaceMembers, including Boundary, SetUniformThickness, PlacementSurface, SetUniformOffset will be demonstrated in this chapter.

2) Boundary

The boundary of surface part is defined as a polyline that consists of several critical points, or is defined as a polyline that consists of curves and straight lines. In addition, the X, Y and Z coordinates of these points are defined by using three arrays.

3) SetUniformThickness

This property demonstrates the uniform thickness of surface part, as shown in Figure 4-40.

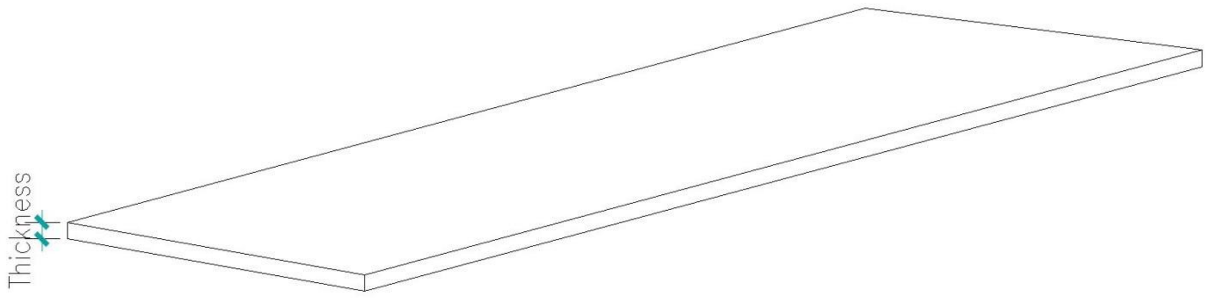


Figure 4-40. Uniform Thickness of Surface Part

4) PlacementSurface

This property defines the relative location of the boundary of the surface member. It has four options: top, center, bottom, and unset, as shown in Figure 4-41.

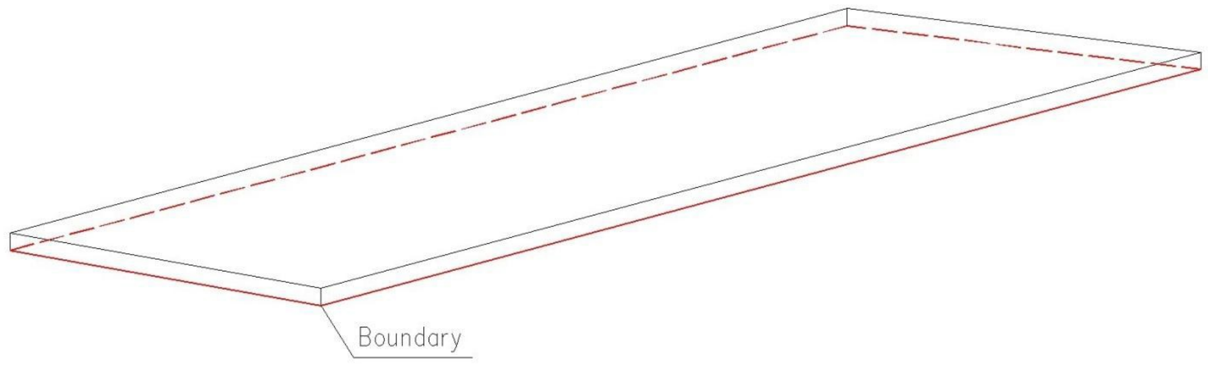


Figure 4-41. PlacementSurface of Surface Part

5) SetUniformOffset

This property is also to define the relative location of the boundary and the surface member, as shown in Figure 4-42. In this case, the value of offset is 0.

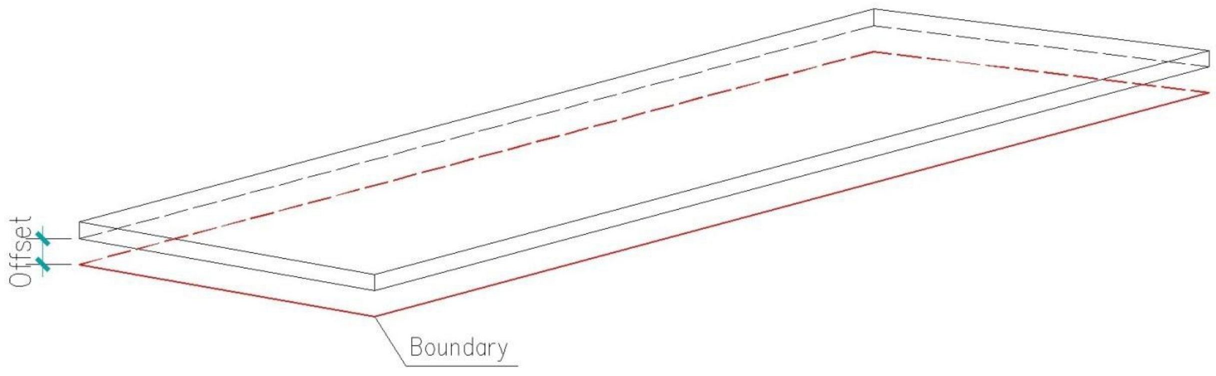
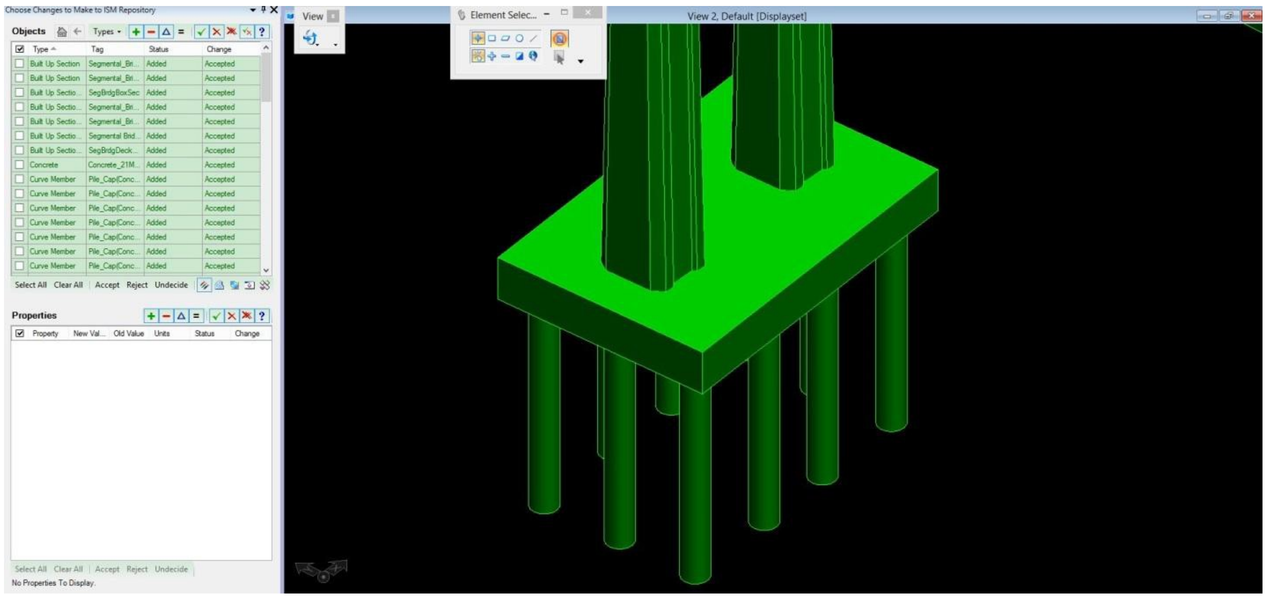


Figure 4-42. Uniform Offset of the Boundary to the Surface Member

6) Summary of Modeling Surface Part

From the analysis above, we can conclude that surface part can be modeled by using IIsmSurfaceMembers interface in the current ISM schema. Figure 4-43a and 4-43b shows the ISM models of the pile cap in Otay River Bridge.


Choose Changes to Make to ISM Repository

Objects

☑	Type	Tag	Status	Change
☐	Built Up Section	Segmental_Bri...	Added	Accepted
☐	Built Up Section	Segmental_Bri...	Added	Accepted
☐	Built Up Sectio...	SegBrdgBoxSec	Added	Accepted
☐	Built Up Sectio...	Segmental_Bri...	Added	Accepted
☐	Built Up Sectio...	Segmental_Bri...	Added	Accepted
☐	Built Up Sectio...	Segmental Brid...	Added	Accepted
☐	Built Up Sectio...	SegBrdgDeck...	Added	Accepted
☐	Concrete	Concrete_21M...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted
☐	Curve Member	Pile_Cap(Conc...	Added	Accepted

Properties

☑	Property	New Val...	Old Value	Units	Status	Change

No Properties To Display.

Figure 4-43a. ISM Models of Pile Cap

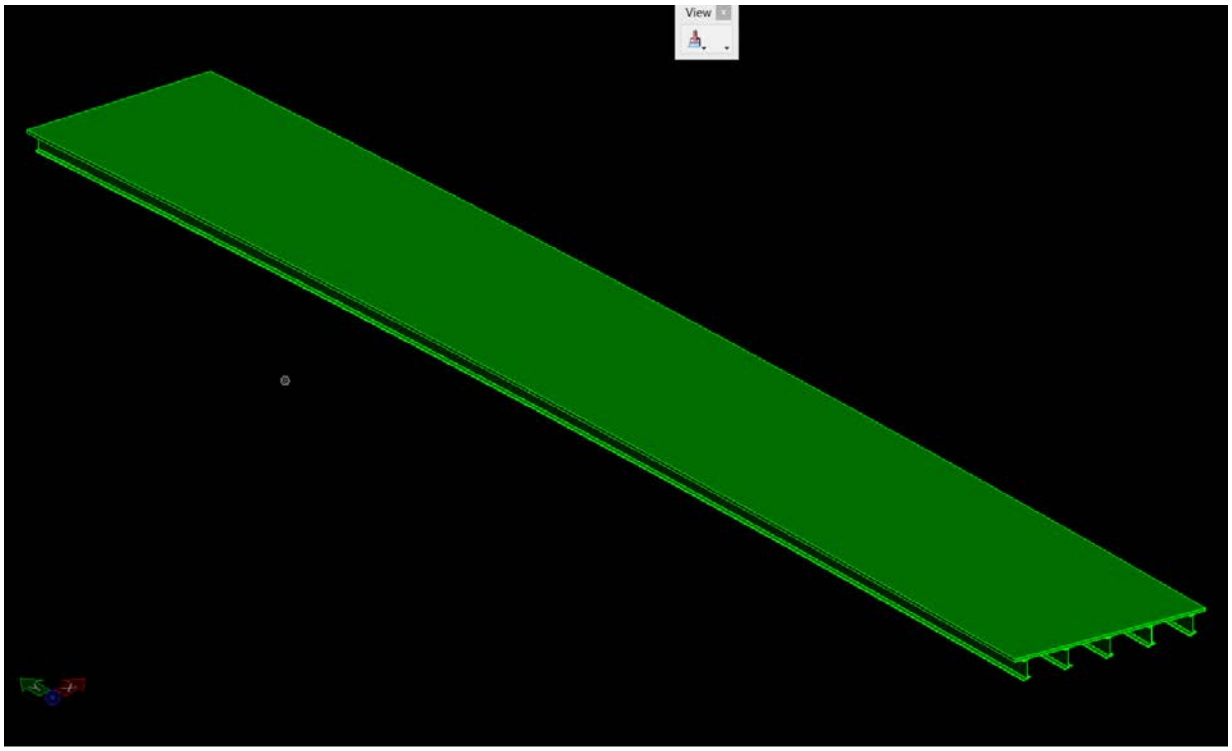


Figure 4-43b. ISM Models of Deck

4.4.3.3 Mapping to IFC

Surfaces in IFC are defined using IfcShapeRepresentation. IfcShapeRepresentation represents the concept of a particular geometric representation of a product or a product component within a specific geometric representation context. The inherited attribute RepresentationType is used to define the geometric model used for the shape representation. The inherited attributeRepresentationIdentifier is used to denote the part of the representation captured by the IfcShapeRepresentation (e.g., Axis, Body, etc.). Face based and shell based surface models are included as predefined types in IFC.

4.4.4 Volume Part

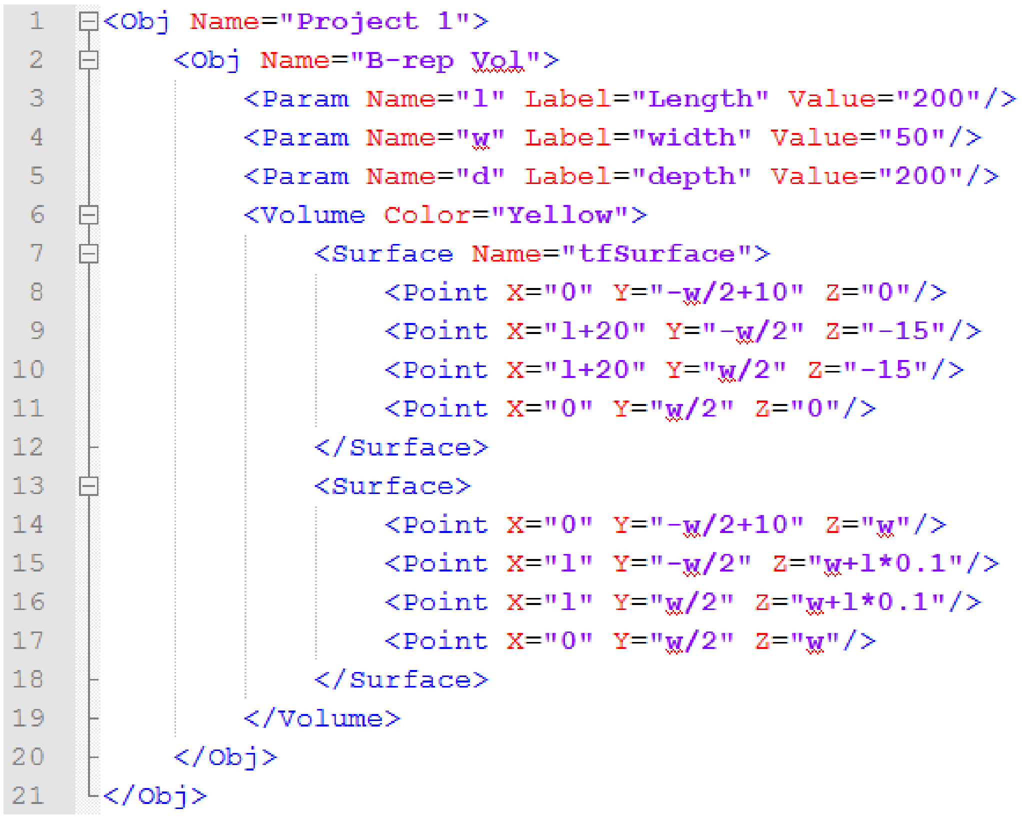
Volume part is the bridge members that can be modeled using boundary representation (B-rep) or revolve function.

4.4.4.1 Boundary Representation

Boundary representation defines a volume using its two surfaces. It can be used to model an abutment wall or protection wall.

4.4.4.1.1 Mapping to OpenBrIM Schema

The XML code created for B-rep volume part based on the OpenBrIM XSD is shown in Code 4.1.



```
 1  <Obj Name="Project 1">
 2      <Obj Name="B-rep Vol">
 3          <Param Name="l" Label="Length" Value="200"/>
 4          <Param Name="w" Label="width" Value="50"/>
 5          <Param Name="d" Label="depth" Value="200"/>
 6          <Volume Color="Yellow">
 7              <Surface Name="tfSurface">
 8                  <Point X="0" Y="-w/2+10" Z="0"/>
 9                  <Point X="l+20" Y="-w/2" Z="-15"/>
10                  <Point X="l+20" Y="w/2" Z="-15"/>
11                  <Point X="0" Y="w/2" Z="0"/>
12              </Surface>
13              <Surface>
14                  <Point X="0" Y="-w/2+10" Z="w"/>
15                  <Point X="l" Y="-w/2" Z="w+l*0.1"/>
16                  <Point X="l" Y="w/2" Z="w+l*0.1"/>
17                  <Point X="0" Y="w/2" Z="w"/>
18              </Surface>
19          </Volume>
20      </Obj>
21  </Obj>
```

Code 4.1 OpenBrIM XML Code for B-rep Volume Part

In the above XML code, the volume part is defined by two "surface" nodes in line 7 to line 18. These two surfaces are not parallel. Figure 4-44 shows the viewer-generated 3D view of the B- rep volume part.

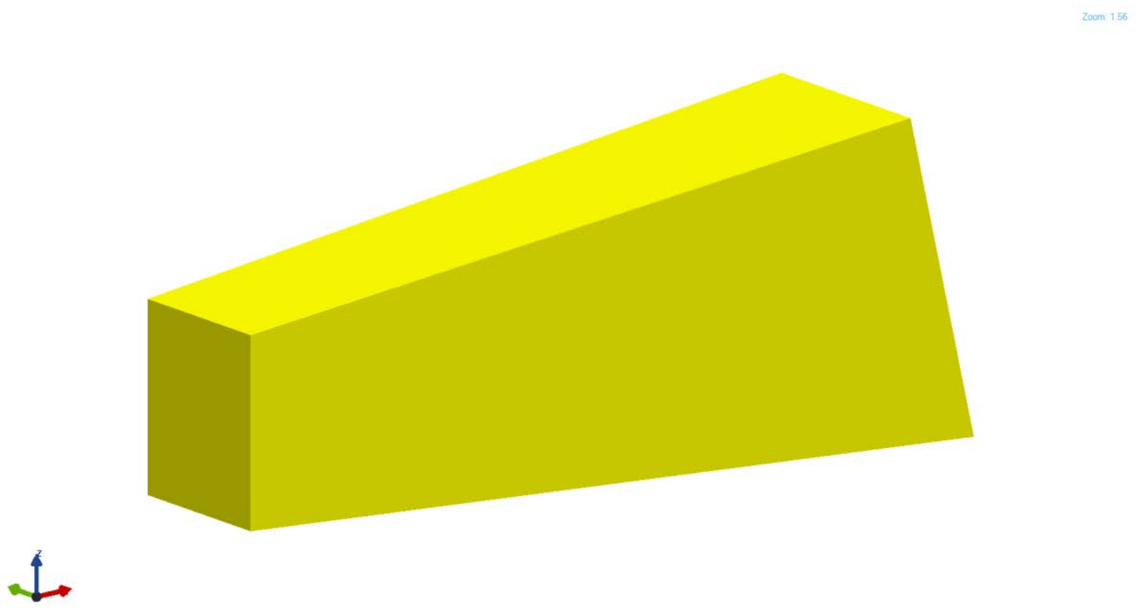


Figure 4-44. 3D View of B-rep Volume Part

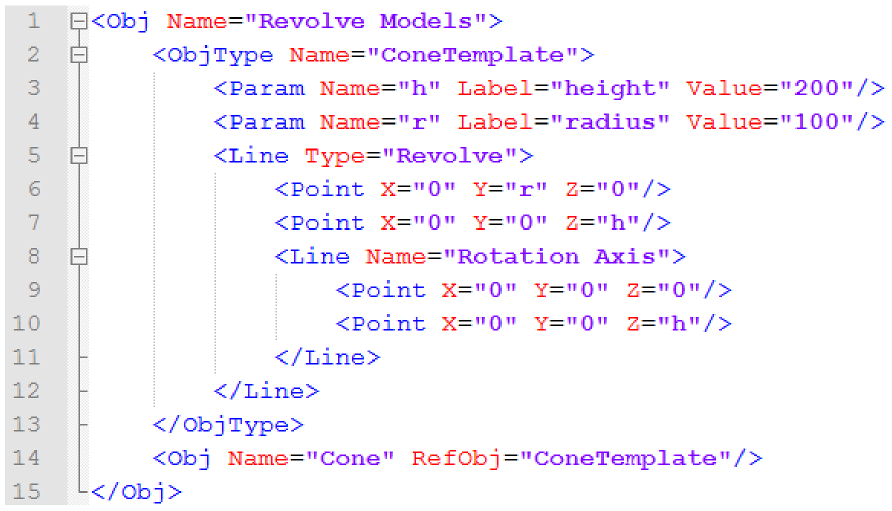
4.4.4.2 Revolve

The friction pendulum friction bearing is one bridge component that needs to be modeled using the revolve function. The revolve function should have at least two geometric properties.

- Rotation axis
- Boundary curve which rotates around the rotation axis

4.4.4.2.1 Mapping to OpenBrIM Schema

The XML code created for revolve volume part based on the OpenBrIM XSD is shown in Code 4.2.



```xml
<Obj Name="Revolve Models">
    <ObjType Name="ConeTemplate">
        <Param Name="h" Label="height" Value="200"/>
        <Param Name="r" Label="radius" Value="100"/>
        <Line Type="Revolve">
            <Point X="0" Y="r" Z="0"/>
            <Point X="0" Y="0" Z="h"/>
            <Line Name="Rotation Axis">
                <Point X="0" Y="0" Z="0"/>
                <Point X="0" Y="0" Z="h"/>
            </Line>
        </Line>
    </ObjType>
    <Obj Name="Cone" RefObj="ConeTemplate"/>
</Obj>
```

Code 4.2 OpenBrIM XML Code for Revolve Volume Part

In the above XML code, the rotation axis is defined by "Line" node in line 8 to line 11. The boundary line is defined by "Line" node in line 5 to line 7. Figure 4-45 illustrates the XML code.

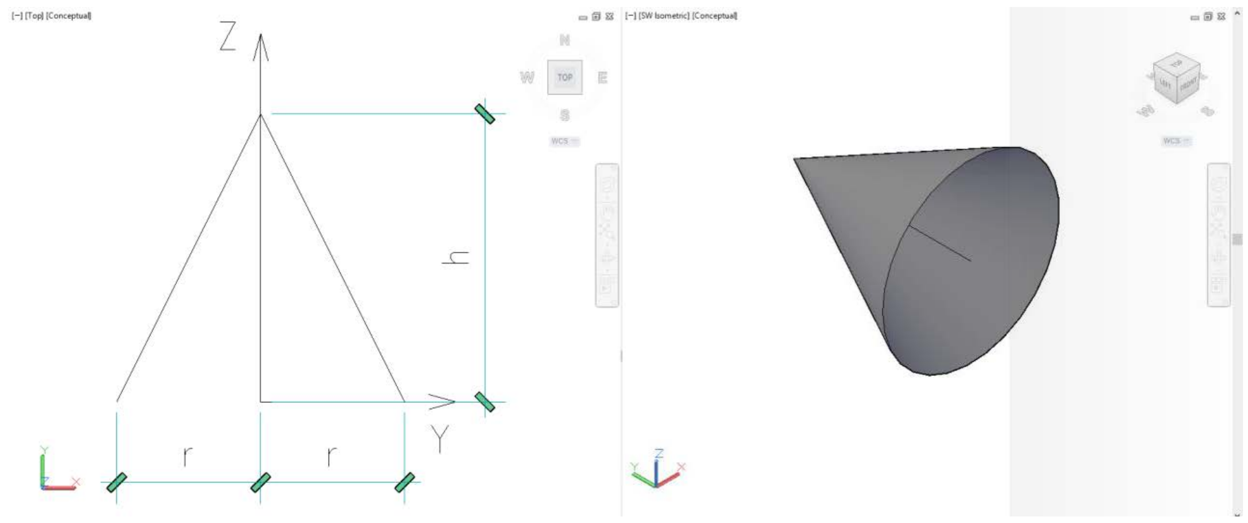


Figure 4-45. Illustration of Revolve Volume Part

4.4.5 Complex Part

Sometimes to further save cost of material, a steel plate girder is built of several pieces of steel with different grades (also known as hybrid girder configuration), as shown in Figure 4-46. Since a part can have one and only one associated material, this hybrid steel plate girder needs to be modeled as a complex part which is addition of several small parts, i.e. webs and flanges. This method is called schedule-based modeling which is used by AASHTO are BrD/BrR [1].

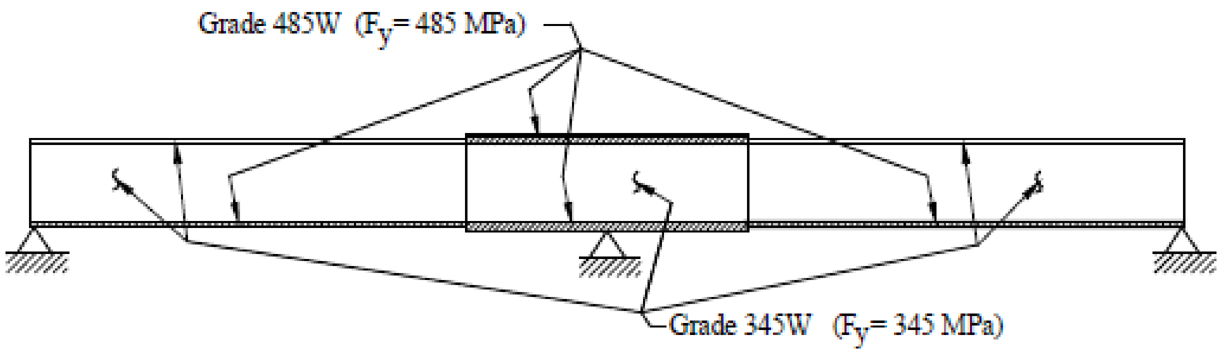
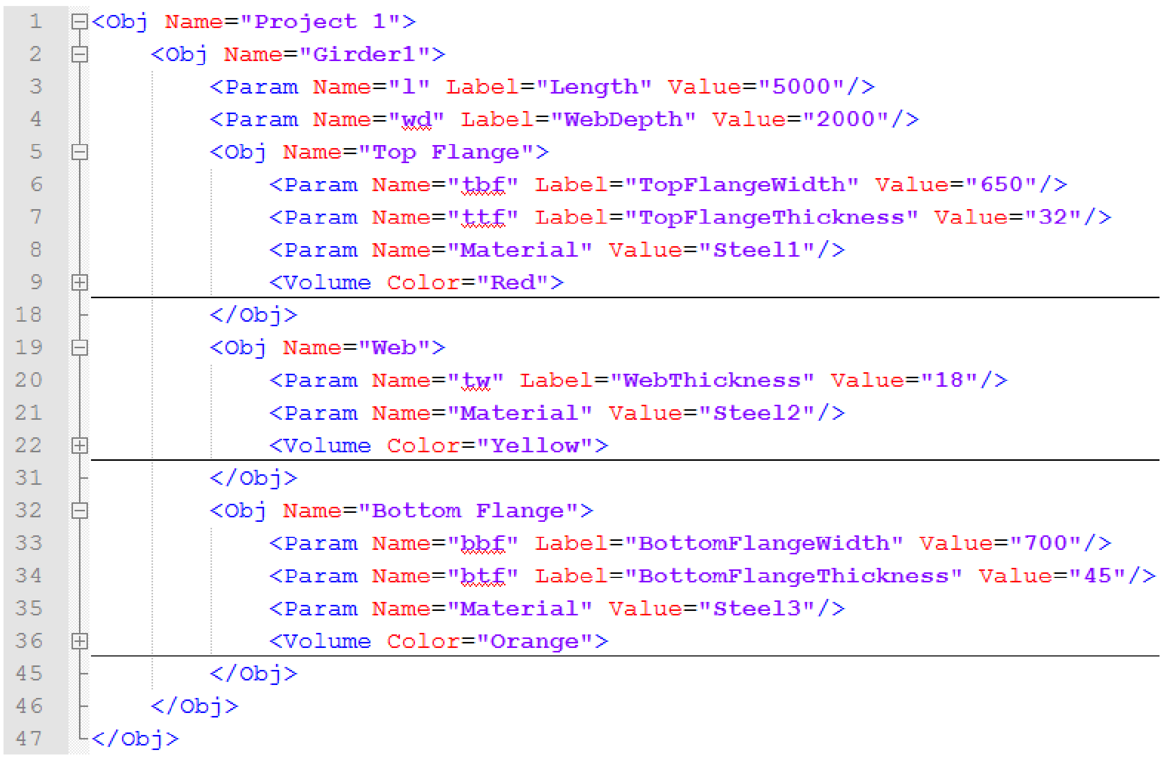


Figure 4-46. Hybrid Girder Configuration

4.4.5.1 Mapping to OpenBrIM Schema

The XML code created for complex part (i.e., hybrid steel plate girder) based on the OpenBrIM XSD is shown in Code 4.3.



```
1  <Obj Name="Project 1">
2      <Obj Name="Girder1">
3          <Param Name="l" Label="Length" Value="5000"/>
4          <Param Name="wd" Label="WebDepth" Value="2000"/>
5          <Obj Name="Top Flange">
6              <Param Name="tbf" Label="TopFlangeWidth" Value="650"/>
7              <Param Name="ttf" Label="TopFlangeThickness" Value="32"/>
8              <Param Name="Material" Value="Steel1"/>
9              <Volume Color="Red">
18         </Obj>
19         <Obj Name="Web">
20             <Param Name="tw" Label="WebThickness" Value="18"/>
21             <Param Name="Material" Value="Steel2"/>
22             <Volume Color="Yellow">
31         </Obj>
32         <Obj Name="Bottom Flange">
33             <Param Name="bbf" Label="BottomFlangeWidth" Value="700"/>
34             <Param Name="btf" Label="BottomFlangeThickness" Value="45"/>
35             <Param Name="Material" Value="Steel3"/>
36             <Volume Color="Orange">
45         </Obj>
46     </Obj>
47 </Obj>
```

Code 4.3 OpenBrIM XML Code for Complex Part

In the XML code above, the steel plate girder is composed of "Obj" node named "Top Flange" (line 5 – line 18), "Obj" node named "Web" (line 19 – line 31) and "Obj" node named "Bottom Flange" (line 32 – line 45). Each has its own material properties. Figure 4-47 shows the viewer- generated 3D view of the girder.

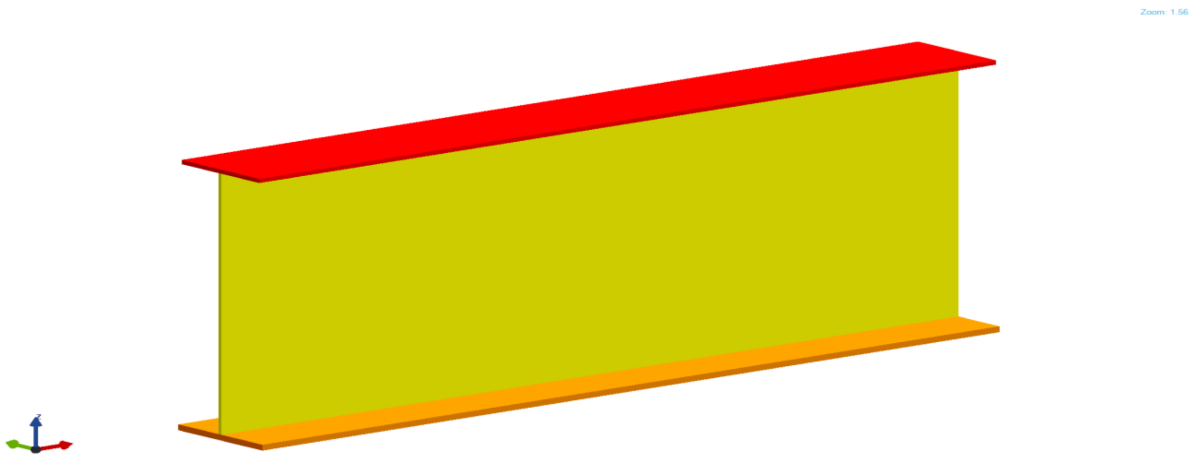


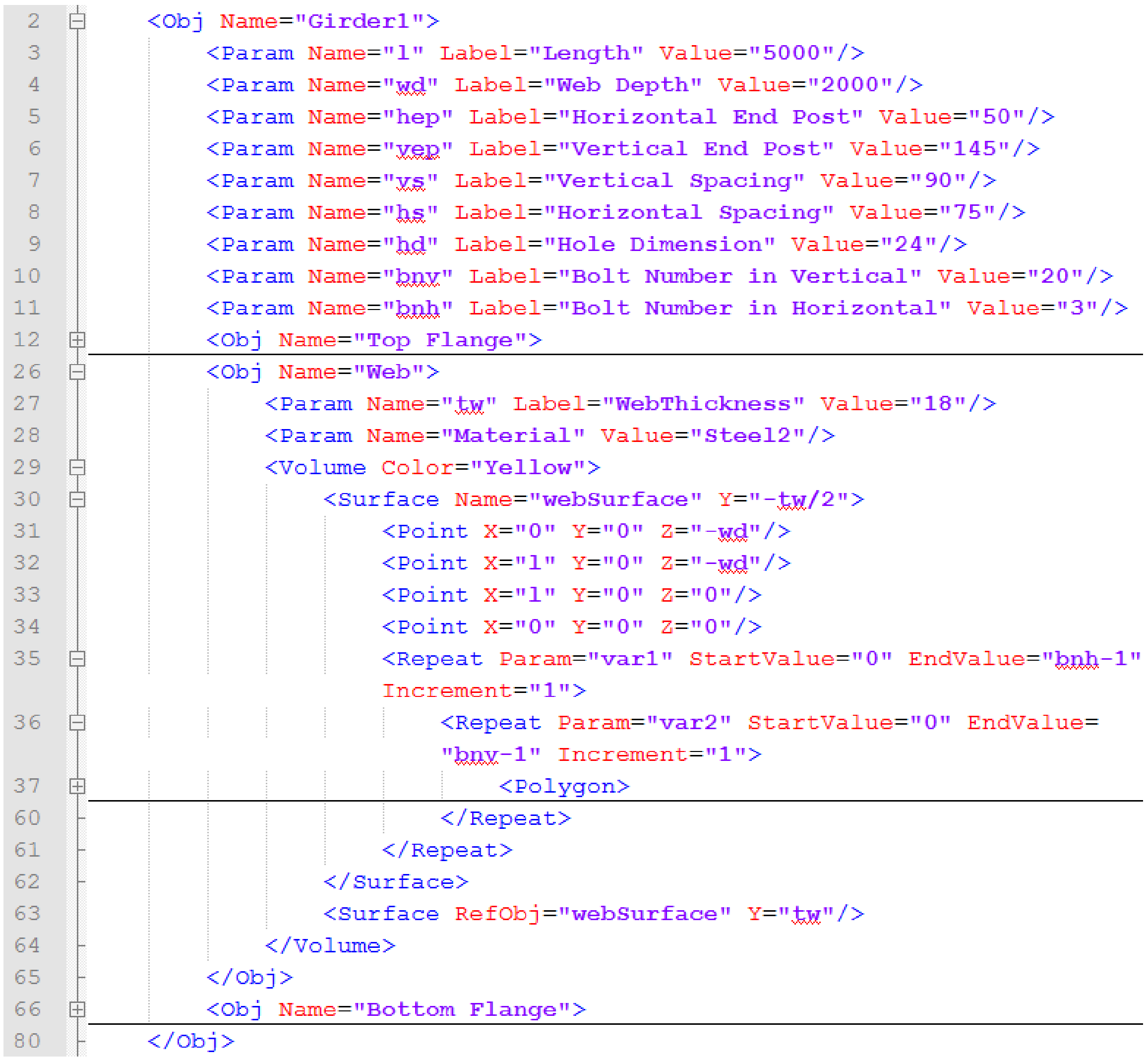
Figure 4-47. View of Complex Part

4.4.6 Derived Part

Derived parts are parts that derived (subtracted) from another part by using features. One example of derived part is steel plate girder with bolted connection. It can be derived from steel plate girder and hole features.

4.4.6.1 Mapping to OpenBrIM Schema

The XML code created for derived part (i.e., steel plate girder with bolted connection) based on the OpenBrIM XSD is shown in Code 4.4.



```
2       <Obj Name="Girder1">
3           <Param Name="l" Label="Length" Value="5000"/>
4           <Param Name="wd" Label="Web Depth" Value="2000"/>
5           <Param Name="hep" Label="Horizontal End Post" Value="50"/>
6           <Param Name="vep" Label="Vertical End Post" Value="145"/>
7           <Param Name="vs" Label="Vertical Spacing" Value="90"/>
8           <Param Name="hs" Label="Horizontal Spacing" Value="75"/>
9           <Param Name="hd" Label="Hole Dimension" Value="24"/>
10          <Param Name="bnv" Label="Bolt Number in Vertical" Value="20"/>
11          <Param Name="bnh" Label="Bolt Number in Horizontal" Value="3"/>
12          <Obj Name="Top Flange">
26          <Obj Name="Web">
27              <Param Name="tw" Label="WebThickness" Value="18"/>
28              <Param Name="Material" Value="Steel2"/>
29              <Volume Color="Yellow">
30                  <Surface Name="webSurface" Y="-tw/2">
31                      <Point X="0" Y="0" Z="-wd"/>
32                      <Point X="l" Y="0" Z="-wd"/>
33                      <Point X="l" Y="0" Z="0"/>
34                      <Point X="0" Y="0" Z="0"/>
35                      <Repeat Param="var1" StartValue="0" EndValue="bnh-1"
                        Increment="1">
36                          <Repeat Param="var2" StartValue="0" EndValue=
                            "bnv-1" Increment="1">
37                              <Polygon>
60                          </Repeat>
61                      </Repeat>
62                  </Surface>
63                  <Surface RefObj="webSurface" Y="tw"/>
64              </Volume>
65          </Obj>
66          <Obj Name="Bottom Flange">
80      </Obj>
```

Code 4.4 OpenBrIM XML Code for Derived Part

The definition of plate girder is the same as Code 4.3. The bolt holes are defined by "Polygon" node in line 37. Two level "Repeat" nodes are used to define a 2D array of holes (in longitudinal and vertical directions). Figure 4-48 shows viewer-generated 3D view of the plate girder.

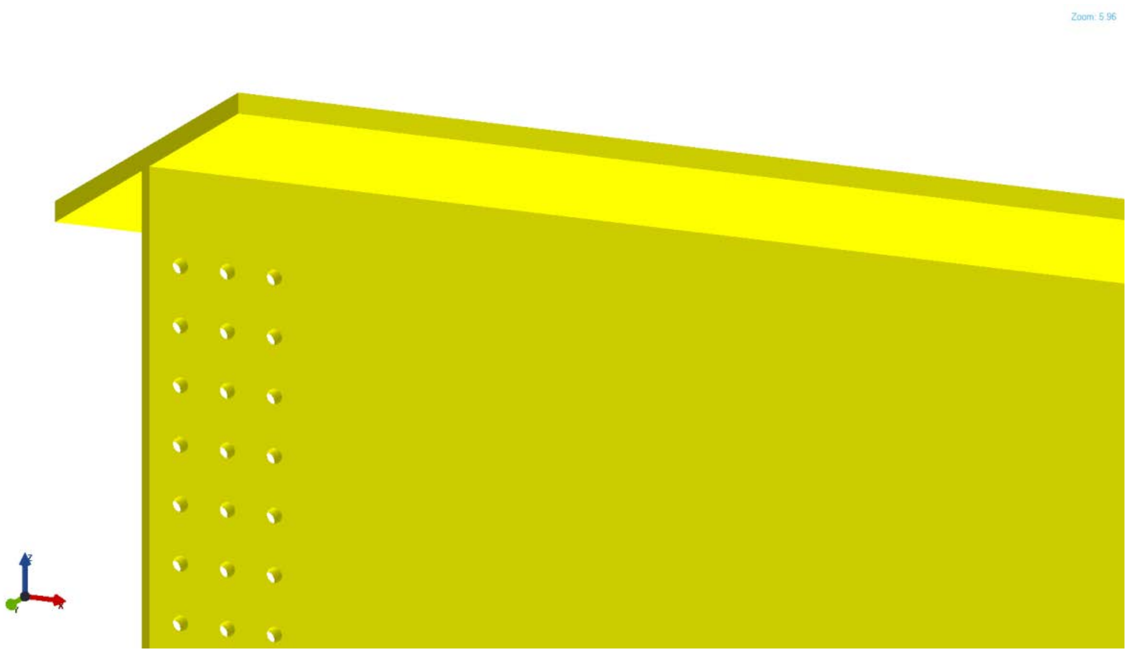


Figure 4-48. 3D View of Derived Part

4.5 Features

Features are variations to the geometry of primitive objects (i.e., parts) and non-primitive objects (e.g. assemblies, joint systems). This section covers the descriptions and specifications of generic and specific features.

Features can be categorized into edge features, surface features (e.g., surface treatment) and volumetric features (e.g., holes, thread, etc.). In the CIMsteel Integration Standards Release 2.1 (CIS/2.1) data model, all the implemented volumetric features are subtractive, which is they remove material from the original part. However, in the Integrated Structural Modeling (ISM) data model, features can be both subtractive and additive. This interoperable data protocols follow the concept of the CIS/2.1 data model.

Features have their own coordinate system, local coordinate system, other than global coordinate system (i.e., longitudinal, transverse and vertical). In some cases, the local coordinate system and the transformation between local and global coordinate systems are implicit.

4.5.1 Hole

Hole features is a type of volumetric features, which is defined by a 2D closed path and a thickness. Usually the thickness of hole is the same as the thickness of parts to which the feature applies. However, this does not imply that an unconstrained hole applied to the top flange of a plate girder will also pass through the bottom flange. In this case, the hole only passes through the top flange because plate girders are modeled as assemblies, and the top flange is one part in the associated assembly.

One implementation of the hole feature is to model the ducts embedded in girders, as shown in Figure 4-49.

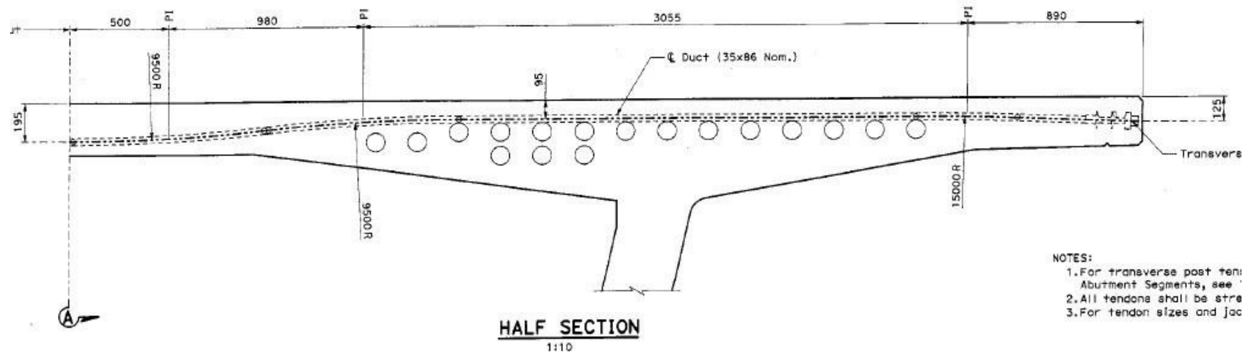


Figure 4-49. Ducts in Concrete Girders

Holes can be categorized into three types according to their shapes: circular holes, rectangular holes (with fillet), and slotted holes.

4.5.1.1 Circular Hole

Circular holes are defined by a circular cutting path and a thickness. The center of the circular path is the place at where the feature's cutting axis lies and the origin of the feature's coordinate system. The attribute of the circular hole is radius of hole (Figures 4-50 and 4-51). The attributes of the circular hole feature are listed below.

- Horizontal end post
- Vertical end post
- Hole radius

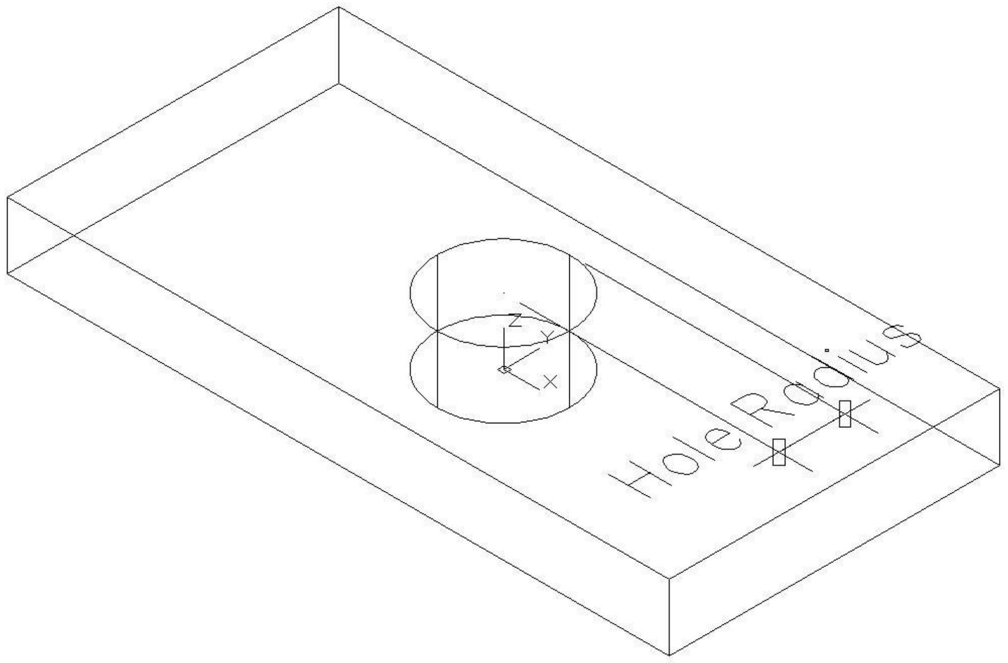


Figure 4-50. 3D View of Circular Hole Feature

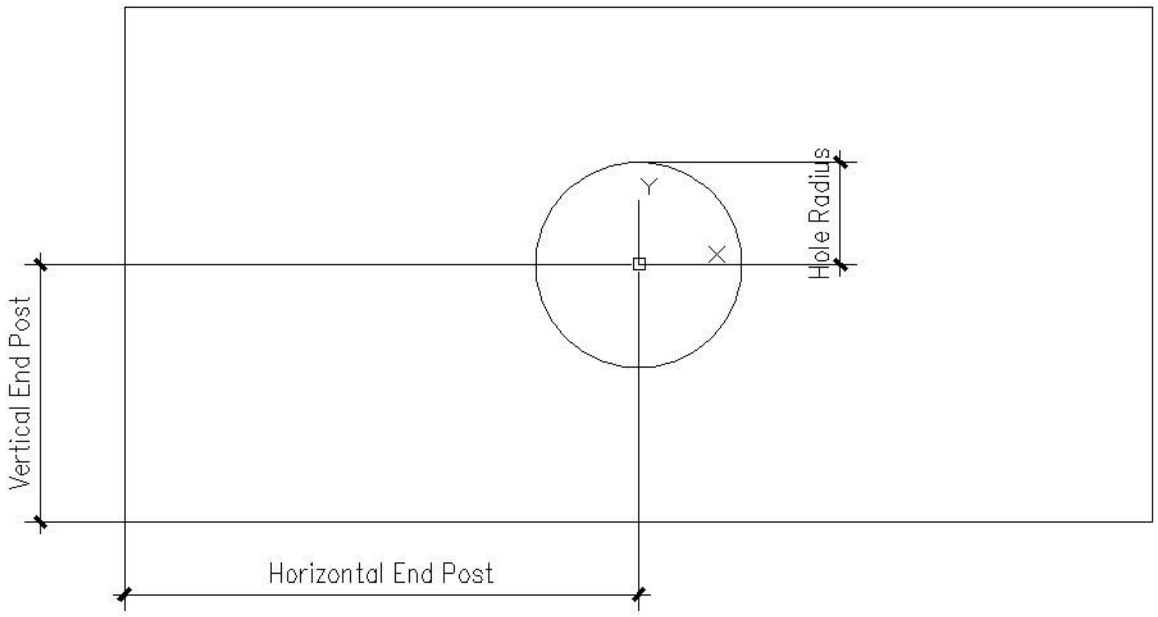


Figure 4-51. Circular Hole Feature Properties

4.5.1.1.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="Girder1">
<Param Name="l" Label="Length" Value="50000"/>
<Param Name="wd" Label="WebDepth" Value="2000"/>
<Param Name="tw" Label="WebThickness" Value="18"/>
<Param Name="tbf" Label="TopFlangeWidth" Value="650"/>
<Param Name="bbf" Label="BottomFlangeWidth" Value="700"/>
<Param Name="ttf" Label="TopFlangeThickness" Value="32"/>
<Param Name="btf" Label="BottomFlangeThickness" Value="45"/>
<Param Name="hep" Label="HorizontalEndPost" Value="1500"/>
<Param Name="vep" Label="VerticalEndPost" Value="1000"/>
<Param Name="hr" Label="Radius of the hole" Value="50"/>
<Param Name="hs" Label="Horizontal Spacing" Value="1000"/>
<Volume Color="Yellow">
<Surface Name="tfSurface" Z="0">
<Point X="0" Y="-tbf/2" Z="0"/>
<Point X="l" Y="-tbf/2" Z="0"/>
<Point X="l" Y="tbf/2" Z="0"/>
<Point X="0" Y="tbf/2" Z="0"/>
</Surface>
<Surface RefObj="tfSurface" Z="ttf"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="webSurface" Y="-tw/2">
```

```
<Point X="0" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="0"/>
<Point X="0" Y="0" Z="0"/>
<Repeat Param="var1" StartValue="0" EndValue="5"
Increment="1">
<Polygon>
<Point X="hep+hs*var1+hr*Cos(pi*0.0)" Y="0"
Z="-vep+hr*Sin(pi*0.0)"/>
<Point X="hep+hs*var1+hr*Cos(pi*0.1)" Y="0"
Z="-vep+hr*Sin(pi*0.1)"/>
<Point X="hep+hs*var1+hr*Cos(pi*0.2)" Y="0"
Z="-vep+hr*Sin(pi*0.2)"/>
<Point X="hep+hs*var1+hr*Cos(pi*0.3)" Y="0"
Z="-vep+hr*Sin(pi*0.3)"/>
<Point X="hep+hs*var1+hr*Cos(pi*0.4)" Y="0"
Z="-vep+hr*Sin(pi*0.4)"/>
<Point X="hep+hs*var1+hr*Cos(pi*0.5)" Y="0"
Z="-vep+hr*Sin(pi*0.5)"/>
<Point X="hep+hs*var1+hr*Cos(pi*0.6)" Y="0"
Z="-vep+hr*Sin(pi*0.6)"/>
<Point X="hep+hs*var1+hr*Cos(pi*0.7)" Y="0"
Z="-vep+hr*Sin(pi*0.7)"/>
<Point X="hep+hs*var1+hr*Cos(pi*0.8)" Y="0"
Z="-vep+hr*Sin(pi*0.8)"/>
<Point X="hep+hs*var1+hr*Cos(pi*0.9)" Y="0"
Z="-vep+hr*Sin(pi*0.9)"/>
<Point X="hep+hs*var1+hr*Cos(pi*1.0)" Y="0"
Z="-vep+hr*Sin(pi*1.0)"/>
<Point X="hep+hs*var1+hr*Cos(pi*1.1)" Y="0"
Z="-vep+hr*Sin(pi*1.1)"/>
<Point X="hep+hs*var1+hr*Cos(pi*1.2)" Y="0"
Z="-vep+hr*Sin(pi*1.2)"/>
<Point X="hep+hs*var1+hr*Cos(pi*1.3)" Y="0"
Z="-vep+hr*Sin(pi*1.3)"/>
<Point X="hep+hs*var1+hr*Cos(pi*1.4)" Y="0"
Z="-vep+hr*Sin(pi*1.4)"/>
<Point X="hep+hs*var1+hr*Cos(pi*1.5)" Y="0"
Z="-vep+hr*Sin(pi*1.5)"/>
<Point X="hep+hs*var1+hr*Cos(pi*1.6)" Y="0"
Z="-vep+hr*Sin(pi*1.6)"/>
<Point X="hep+hs*var1+hr*Cos(pi*1.7)" Y="0"
Z="-vep+hr*Sin(pi*1.7)"/>
<Point X="hep+hs*var1+hr*Cos(pi*1.8)" Y="0"
Z="-vep+hr*Sin(pi*1.8)"/>
<Point X="hep+hs*var1+hr*Cos(pi*1.9)" Y="0"
Z="-vep+hr*Sin(pi*1.9)"/>
<Point X="hep+hs*var1+hr*Cos(pi*2.0)" Y="0"
Z="-vep+hr*Sin(pi*2.0)"/>
</Polygon>
</Repeat>
```

```
</Surface>
<Surface RefObj="webSurface" Y="tw"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="bfSurface" Z="-wd-btf">
<Point X="0" Y="-bbf/2" Z="0"/>
<Point X="l" Y="-bbf/2" Z="0"/>
<Point X="l" Y="bbf/2" Z="0"/>
<Point X="0" Y="bbf/2" Z="0"/>
</Surface> <Surface RefObj="bfSurface" Z="-wd"/>
</Volume>
</Obj>
```

Figure 4-52 shows 3D view of the model of a part with a circular hole using the OpenBrimViewer.

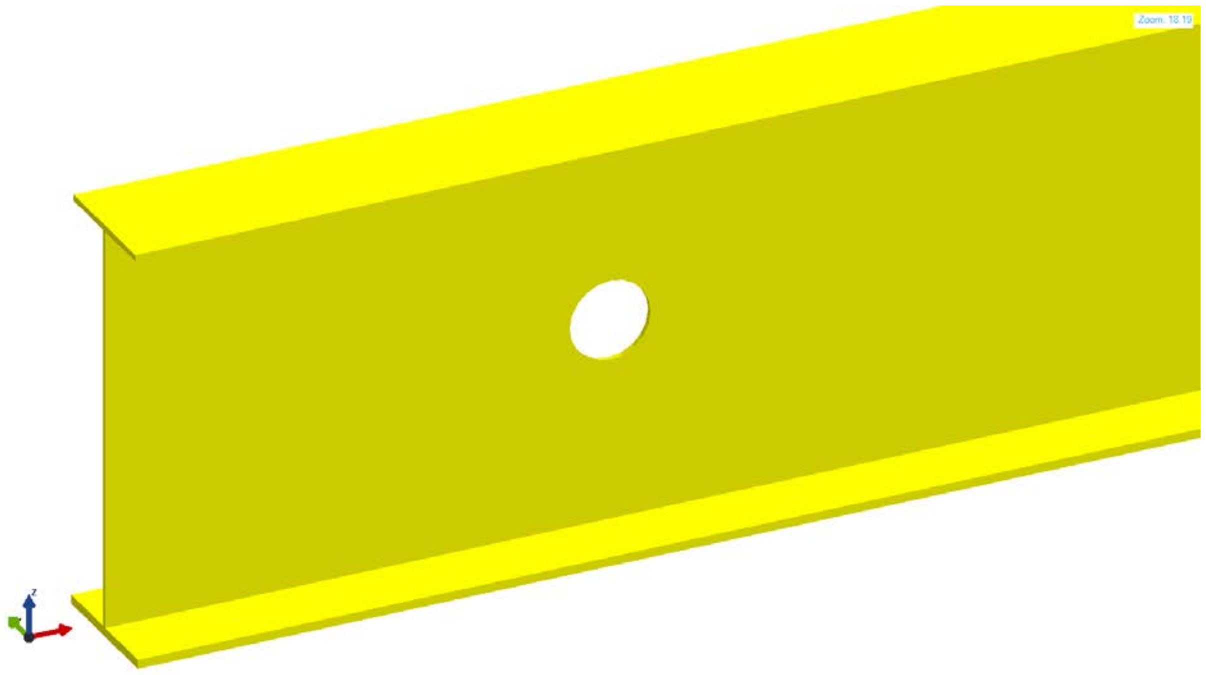


Figure 4-52. 3D View of Circular Hole Feature in the OpenBrIM Viewer

4.5.1.1.2 Mapping to ISM

The circular hole feature can be modeled by using IIsmFeatureSubtraction interface.

4.5.1.1.3 Mapping to IFC

Properties	**IFC Entities/Attributes**
Size of hole - diameter	IfcCircleProfileDef \ Radius IfcProperty \ NominalValue
Location of holes relative to critical surface	IfcProduct \ ObjectPlacement
Location of holes from end point	IfcProduct \ ObjectPlacement IfcProperty \ NominalValue
Location of holes from specified datum point of beam	IfcProduct \ ObjectPlacement
Identification of through holes	IfcProperty \ NominalValue
Reference to bolt details	Refer to IfcMechanicalFastener for mapping
Partial depth holes	Set IfcObject \ ObjectType to 'Partial Depth Hole' Geometry representation defines the depth
Beveled holes (counter sunk)	Set IfcObject \ ObjectType to 'Beveled Hole' Geometry representation defines shape
Tapped/threaded holes	Set IfcObject \ ObjectType to 'Threaded Hole' Geometry representation defines shape
Holes that are not round	Set IfcObject \ ObjectType to 'Arbitrary Hole' Geometry representation defines shape
Burned or Drilled	IfcTask \ WorkMethod

4.5.1.2 Rectangular Hole

Rectangular holes are defined by a rectangular cutting path and thickness. The center of the rectangular cutting path is the place at where the feature's cutting axis lies and the origin of the feature's coordinate system. The attributes of this feature include length of hole, height of hole and fillet radius. If the hole corners are square, the fillet radius is zero. For square cornered holes, the fillet radius shall be set to zero. For square holes, the length of hole shall be set equal to the height of hole. For square holes with square corners, the length of hole shall be set equal to the height of hole, and the fillet radius shall be set to zero (Figure 4-53).

The attributes of the rectangular hole feature are listed below.

- Horizontal end post
- Vertical end post
- Hole height
- Hole length
- Fillet radius

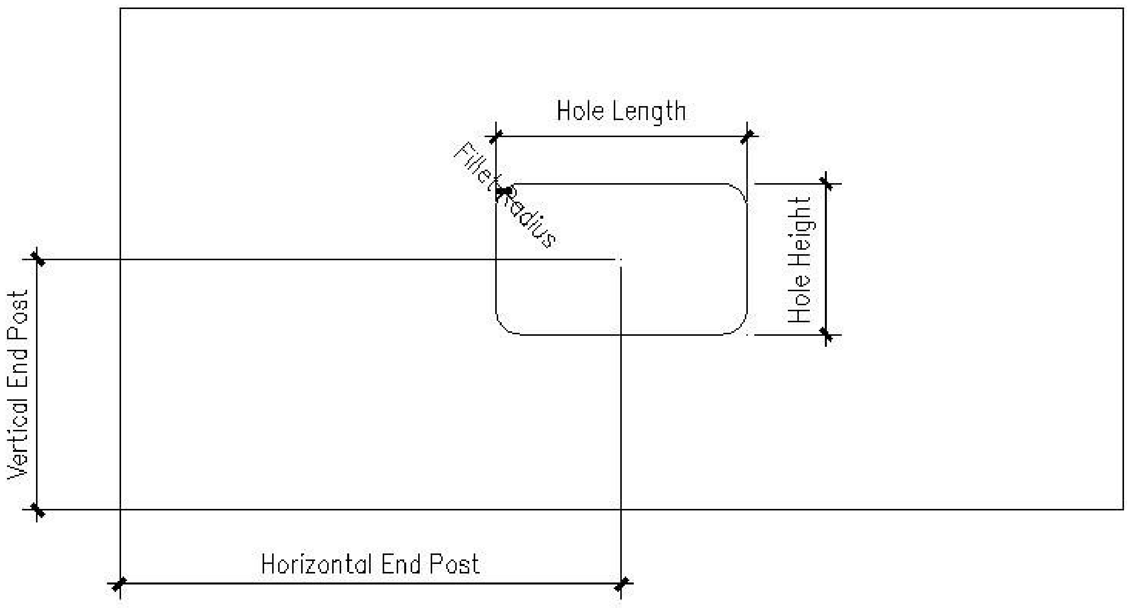


Figure 4-53. Rectangular Hole Feature Properties

4.5.1.2.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="Girder1">
<Param Name="l" Label="Length" Value="50000"/>
<Param Name="wd" Label="WebDepth" Value="2000"/>
<Param Name="tw" Label="WebThickness" Value="18"/>
<Param Name="tbf" Label="TopFlangeWidth" Value="650"/>
<Param Name="bbf" Label="BottomFlangeWidth" Value="700"/>
<Param Name="ttf" Label="TopFlangeThickness" Value="32"/>
<Param Name="btf" Label="BottomFlangeThickness" Value="45"/>
<Param Name="hep" Label="HorizontalEndPost" Value="1500"/>
<Param Name="vep" Label="VerticalEndPost" Value="1000"/>
<Param Name="hr" Label="Radius of the hole" Value="50"/>
<Param Name="hs" Label="Horizontal Spacing" Value="1000"/>
<Volume Color="Yellow">
<Surface Name="tfSurface" Z="0">
<Point X="0" Y="-tbf/2" Z="0"/>
<Point X="l" Y="-tbf/2" Z="0"/>
<Point X="l" Y="tbf/2" Z="0"/>
<Point X="0" Y="tbf/2" Z="0"/>
</Surface>
<Surface RefObj="tfSurface" Z="ttf"/>
</Volume> <Volume Color="Yellow">
<Surface Name="webSurface" Y="-tw/2">
<Point X="0" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="-wd"/>
```

```
<Point X="l" Y="0" Z="0"/>
<Point X="0" Y="0" Z="0"/>
<Repeat Param="var1" StartValue="0" EndValue="5"
Increment="1">
<Polygon>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.0))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.0))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.1))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-Sin(3.14/2*0.1))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.2))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.2))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.3))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.3))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.4))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.4))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.5))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.5))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.6))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.6))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.7))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.7))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.8))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.8))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.9))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.9))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*1.0))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*1.0))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*1.0))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*1.0))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.9))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.9))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.8))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.8))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.7))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.7))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.6))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.6))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.5))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.5))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.4))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.4))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.3))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.3))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.2))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.2))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.1))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.1))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.0))+hs*var1" Y="0" Z="-vep+hh/2-fr*(1-sin(3.14/2*0.0))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.0))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.0))"/>
```

```
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.1))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.1))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.2))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.2))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.3))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.3))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.4))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.4))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.5))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.5))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.6))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.6))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.7))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.7))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.8))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.8))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*0.9))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.9))"/>
<Point X="hep+hl/2-fr*(1-
cos(3.14/2*1.0))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*1.0))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*1.0))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*1.0))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.9))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.9))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.8))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.8))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.7))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.7))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.6))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.6))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.5))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.5))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.4))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.4))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.3))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.3))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.2))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.2))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.1))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.1))"/>
<Point X="hep-hl/2+fr*(1-
cos(3.14/2*0.0))+hs*var1" Y="0" Z="-vep-hh/2+fr*(1-sin(3.14/2*0.0))"/>
</Polygon>
</Repeat>
</Surface>
<Surface RefObj="webSurface" Y="tw"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="bfSurface" Z="-wd-btf">
<Point X="0" Y="-bbf/2" Z="0"/>
<Point X="l" Y="-bbf/2" Z="0"/>
```

```
<Point X="l" Y="bbf/2" Z="0"/>
<Point X="0" Y="bbf/2" Z="0"/>
</Surface>
<Surface RefObj="bfSurface" Z="-wd"/>
</Volume>
</Obj>
```

Figures 4-54 to 4-56 show 3D view of the model of a part with a rectangular hole using the OpenBrim Viewer.

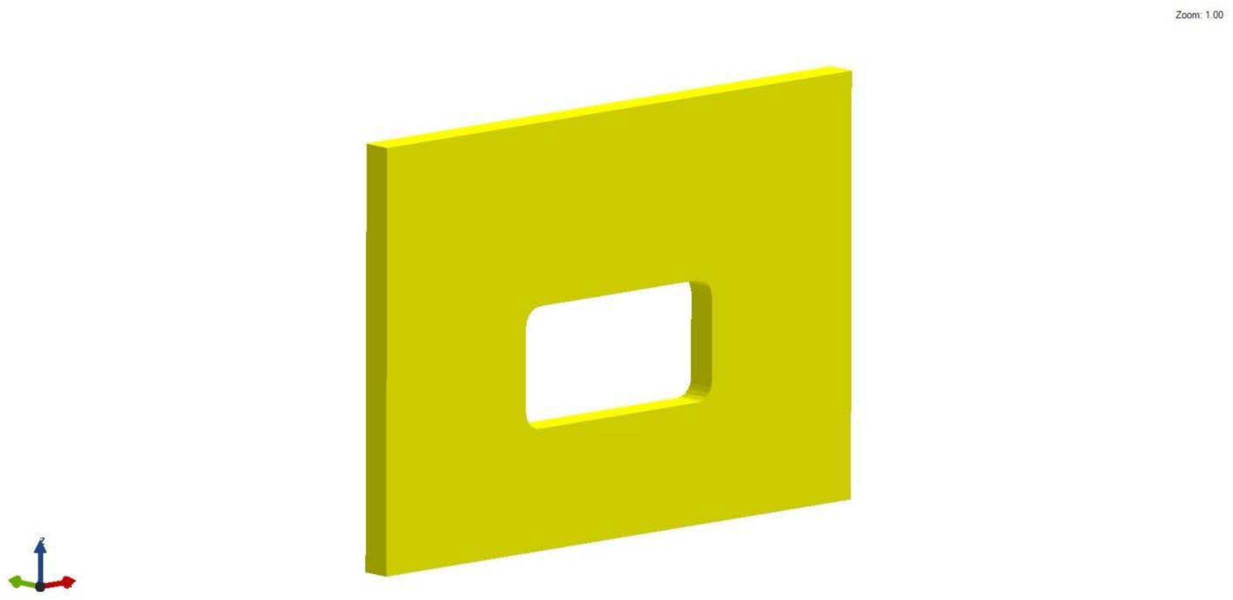


Figure 4-54. 3D View of Rectangular Hole with Fillet in the OpenBrIM Viewer

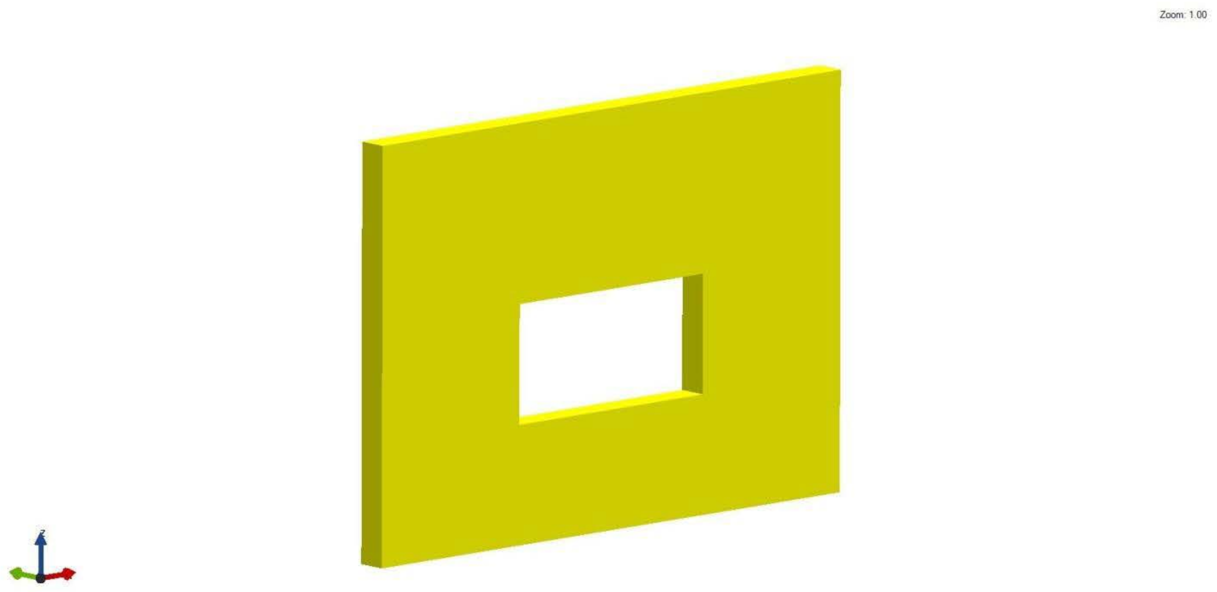


Figure 4-55. 3D View of Rectangular Hole without Fillet in the OpenBrIM Viewer

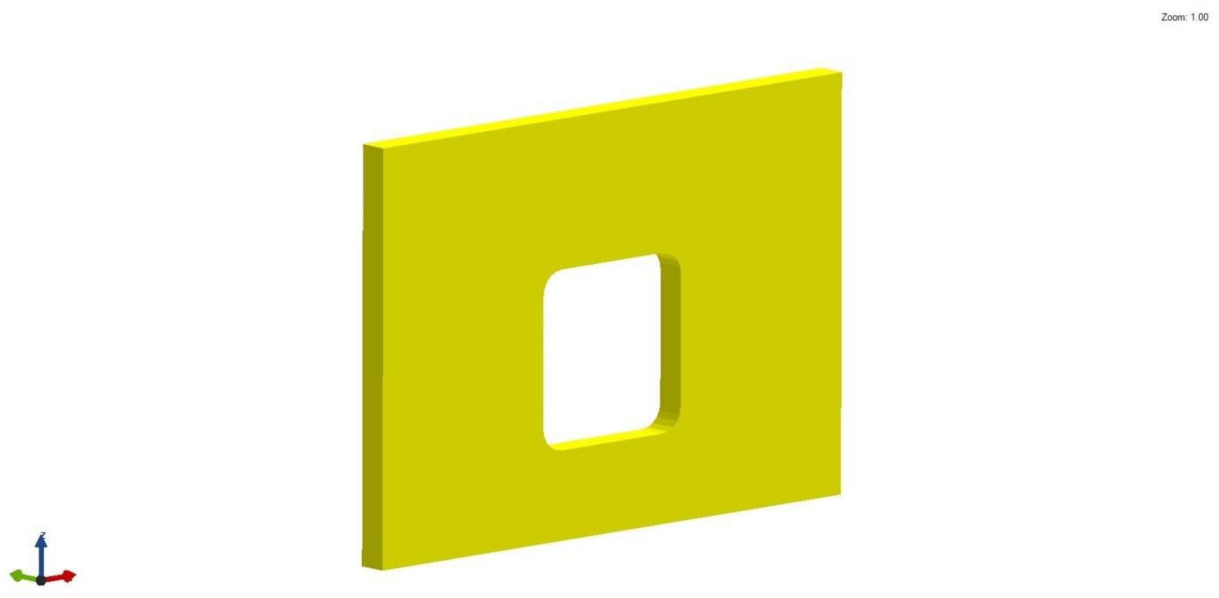


Figure 4-56. 3D View of Square Hole with Fillet in the OpenBrIM Viewer

4.5.1.2.2 Mapping to ISM

The rectangular hole feature can be modeled by using IIsmFeatureSubtraction interface.

4.5.1.3 Slotted Hole

Slotted holes are defined by a thickness, and a cutting path that is implicitly defined as a slot; i.e., an elongated circle. The center of the slotted cutting path is the place at where the feature’s cutting axis lies and the origin of the feature’s coordinate system. The attributes of this feature include length of slot and height of slot. The attributes of the slotted hole feature are listed below (Figure 4-57):

- Horizontal end post
- Vertical end post
- Slot length
- Slot height

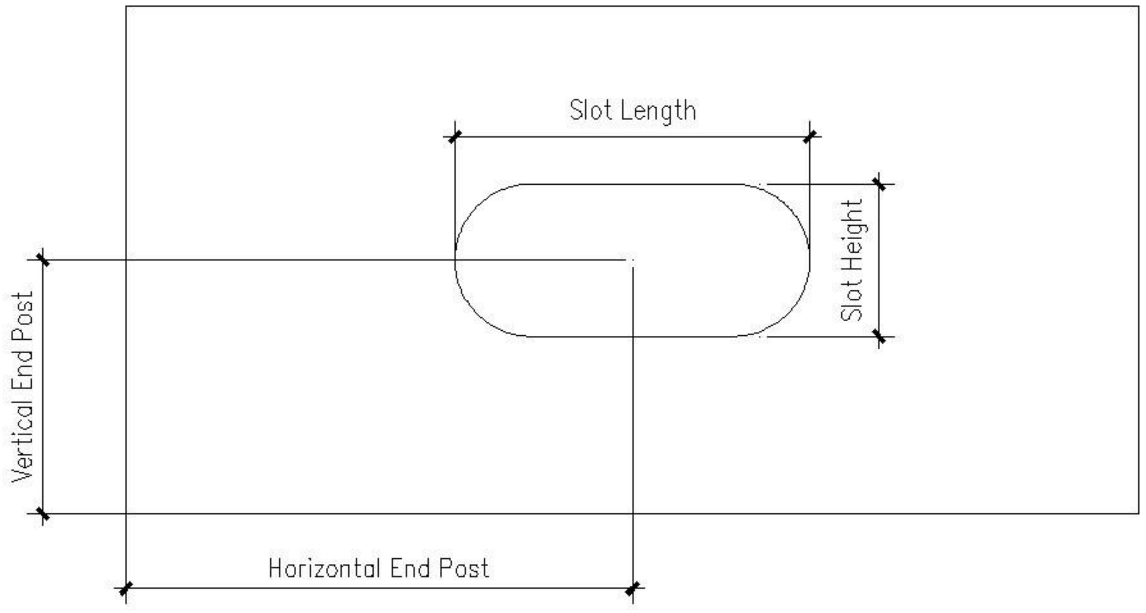


Figure 4-57. Slotted Hole Feature Properties

4.5.1.3.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="Girder1">
<Param Name="l" Label="Length" Value="50000"/>
<Param Name="wd" Label="WebDepth" Value="2000"/>
<Param Name="tw" Label="WebThickness" Value="18"/>
<Param Name="tbf" Label="TopFlangeWidth" Value="650"/>
<Param Name="bbf" Label="BottomFlangeWidth" Value="700"/>
<Param Name="ttf" Label="TopFlangeThickness" Value="32"/>
<Param Name="btf" Label="BottomFlangeThickness" Value="45"/>
<Param Name="hep" Label="HorizontalEndPost" Value="1500"/>
<Param Name="vep" Label="VerticalEndPost" Value="1000"/>
<Param Name="hr" Label="Radius of the hole" Value="50"/>
<Param Name="hs" Label="Horizontal Spacing" Value="1000"/>
<Volume Color="Yellow">
<Surface Name="tfSurface" Z="0">
<Point X="0" Y="-tbf/2" Z="0"/>
<Point X="l" Y="-tbf/2" Z="0"/>
<Point X="l" Y="tbf/2" Z="0"/>
<Point X="0" Y="tbf/2" Z="0"/>
</Surface>
<Surface RefObj="tfSurface" Z="ttf"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="webSurface" Y="-tw/2">
<Point X="0" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="-wd"/>
```

```
<Point X="l" Y="0" Z="0"/>
<Point X="0" Y="0" Z="0"/>
<Repeat Param="var1" StartValue="0" EndValue="5"
Increment="1">
<Polygon>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*0.0))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*0.0))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*0.1))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*0.1))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*0.2))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*0.2))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*0.3))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*0.3))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*0.4))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*0.4))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*0.5))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*0.5))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*0.6))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*0.6))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*0.7))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*0.7))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*0.8))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*0.8))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*0.9))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*0.9))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*1.0))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*1.0))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*1.1))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*1.1))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*1.2))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*1.2))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*1.3))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*1.3))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*1.4))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*1.4))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*1.5))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*1.5))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*1.6))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*1.6))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*1.7))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*1.7))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*1.8))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*1.8))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*1.9))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*1.9))"/>
<Point X="hep-(sl/2)+sh/2*(1-
sin(pi/2*2.0))+hs*var1" Y="0" Z="-vep-sh/2*(cos(pi/2*2.0))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*0.0))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*0.0))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*0.1))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*0.1))"/>
```

```
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*0.2))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*0.2))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*0.3))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*0.3))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*0.4))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*0.4))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*0.5))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*0.5))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*0.6))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*0.6))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*0.7))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*0.7))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*0.8))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*0.8))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*0.9))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*0.9))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*1.0))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*1.0))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*1.1))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*1.1))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*1.2))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*1.2))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*1.3))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*1.3))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*1.4))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*1.4))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*1.5))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*1.5))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*1.6))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*1.6))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*1.7))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*1.7))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*1.8))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*1.8))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*1.9))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*1.9))"/>
<Point X="hep+(sl/2)-sh/2*(1-
sin(pi/2*2.0))+hs*var1" Y="0" Z="-vep+sh/2*(cos(pi/2*2.0))"/>
</Polygon>
</Repeat>
</Surface>
<Surface RefObj="webSurface" Y="tw"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="bfSurface" Z="-wd-btf">
<Point X="0" Y="-bbf/2" Z="0"/>
<Point X="l" Y="-bbf/2" Z="0"/>
<Point X="l" Y="bbf/2" Z="0"/>
<Point X="0" Y="bbf/2" Z="0"/>
</Surface>
<Surface RefObj="bfSurface" Z="-wd"/>
```

```
</Volume>
</Obj>
```

Figure 4-58 show 3D view of the model of a part with a slotted hole using the OpenBrim Viewer.

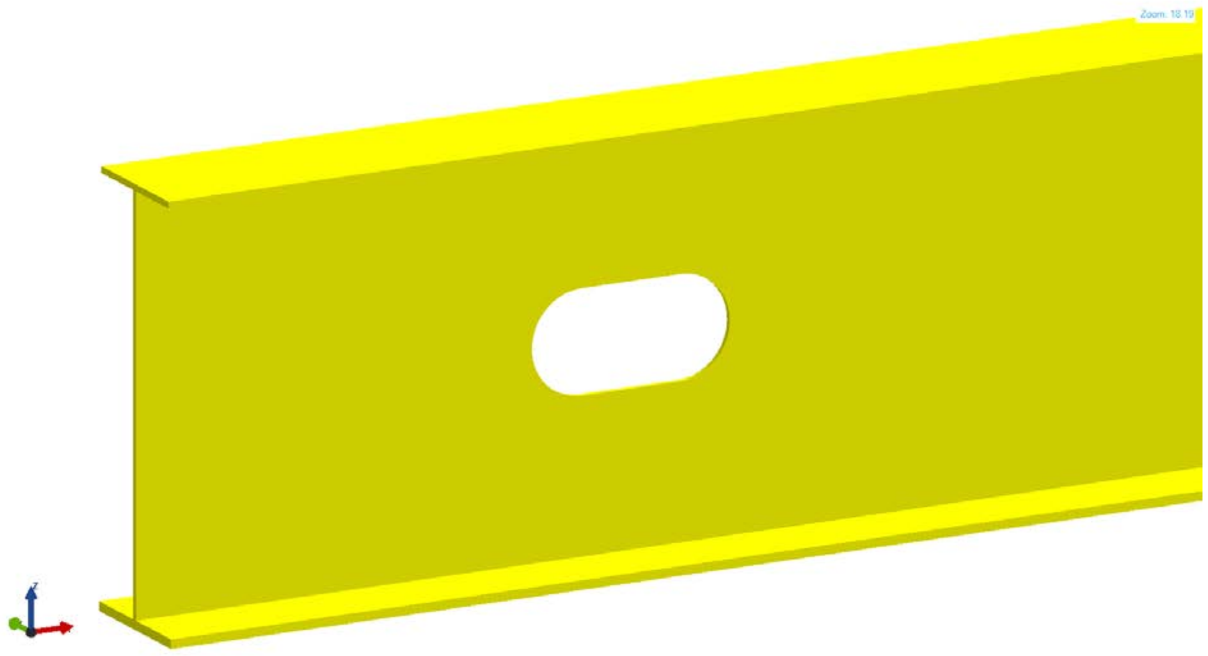


Figure 4-58. 3D View of Slotted Hole in the OpenBrIM Viewer

4.5.1.3.2 Mapping to ISM

The slotted hole feature can be modeled by using IIsmFeatureSubtraction interface.

4.5.1.3.3 Mapping to IFC

Properties	**IFC Entities/Attributes**
Dimension	IfcArbitraryClosedProfileDef IfcProperty \ NominalValue
Other Information	Follows the mapping of Holes for Fasteners

4.5.1.4 Curved Slotted Hole

Curved slotted hole is a type of slotted holes which is defined by rolling a circle on a circular arc. The location point shown in Figure 4-59 is the place at where the feature’s cutting axis lies and the origin of the feature’s coordinate system.

The attributes of the curved slotted hole feature are listed below:

- Horizontal end post.
- Vertical end post.

- Curve radius.
- Start angle.
- Sector angle.
- Slot height.

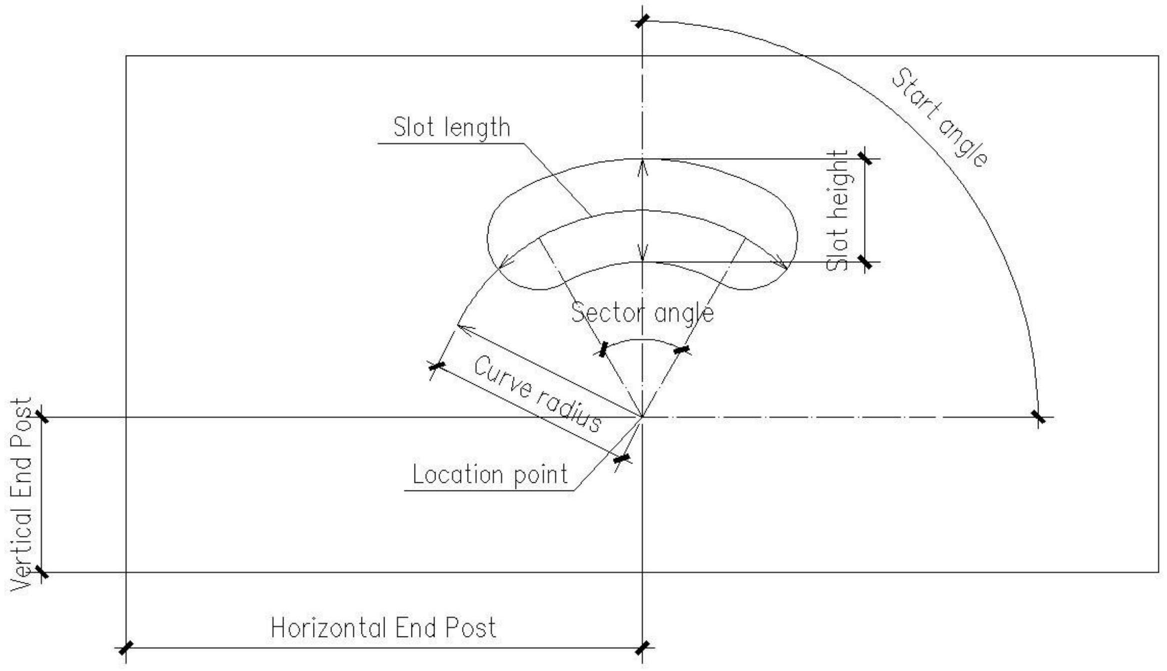


Figure 4-59. Curved Slotted Hole Feature Properties

4.5.1.4.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIM schema is shown as follows:

```
<Obj Name="Girder1">
<Param Name="l" Label="Length" Value="50000"/>
<Param Name="wd" Label="WebDepth" Value="2000"/>
<Param Name="tw" Label="WebThickness" Value="18"/>
<Param Name="tbf" Label="TopFlangeWidth" Value="650"/>
<Param Name="bbf" Label="BottomFlangeWidth" Value="700"/>
<Param Name="ttf" Label="TopFlangeThickness" Value="32"/>
<Param Name="btf" Label="BottomFlangeThickness" Value="45"/>
<Param Name="hep" Label="HorizontalEndPost" Value="1500"/>
<Param Name="vep" Label="VerticalEndPost" Value="1000"/>
<Param Name="hr" Label="Radius of the hole" Value="50"/>
<Param Name="hs" Label="Horizontal Spacing" Value="1000"/>
<Volume Color="Yellow">
<Surface Name="tfSurface" Z="0">
<Point X="0" Y="-tbf/2" Z="0"/>
<Point X="l" Y="-tbf/2" Z="0"/>
<Point X="l" Y="tbf/2" Z="0"/>
<Point X="0" Y="tbf/2" Z="0"/>
```

```
</Surface>
<Surface RefObj="tfSurface" Z="ttf"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="webSurface" Y="-tw/2">
<Point X="0" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="-wd"/>
<Point X="l" Y="0" Z="0"/>
<Point X="0" Y="0" Z="0"/>
<Repeat Param="var1" StartValue="0" EndValue="5"
Increment="1">
<Polygon>
<Point X="hep+(cr-sh/2)*cos(stang-
(secang/2)*1.0)+hs*var1" Y="0" Z="-vep+(cr-sh/2)*sin(stang- (secang/2)*1.0)"/>
<Point X="hep+(cr-
sh/2)*cos(stang-(secang/2)*0.9)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang-(secang/2)*0.9)"/>
<Point X="hep+(cr-
sh/2)*cos(stang-(secang/2)*0.8)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang-(secang/2)*0.8)"/>
<Point X="hep+(cr-
sh/2)*cos(stang-(secang/2)*0.7)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang-(secang/2)*0.7)"/>
<Point X="hep+(cr-
sh/2)*cos(stang-(secang/2)*0.6)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang-(secang/2)*0.6)"/>
<Point X="hep+(cr-
sh/2)*cos(stang-(secang/2)*0.5)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang-(secang/2)*0.5)"/>
<Point X="hep+(cr-
sh/2)*cos(stang-(secang/2)*0.4)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang-(secang/2)*0.4)"/>
<Point X="hep+(cr-
sh/2)*cos(stang-(secang/2)*0.3)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang-(secang/2)*0.3)"/>
<Point X="hep+(cr-
sh/2)*cos(stang-(secang/2)*0.2)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang-(secang/2)*0.2)"/>
<Point X="hep+(cr-
sh/2)*cos(stang-(secang/2)*0.1)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang-(secang/2)*0.1)"/>
<Point X="hep+(cr-
sh/2)*cos(stang-(secang/2)*0.0)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang-(secang/2)*0.0)"/>
<Point X="hep+(cr-
sh/2)*cos(stang+(secang/2)*0.1)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang+(secang/2)*0.1)"/>
<Point X="hep+(cr-
sh/2)*cos(stang+(secang/2)*0.2)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang+(secang/2)*0.2)"/>
<Point X="hep+(cr-
sh/2)*cos(stang+(secang/2)*0.3)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang+(secang/2)*0.3)"/>
<Point X="hep+(cr-
sh/2)*cos(stang+(secang/2)*0.4)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang+(secang/2)*0.4)"/>
<Point X="hep+(cr-
sh/2)*cos(stang+(secang/2)*0.5)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang+(secang/2)*0.5)"/>
<Point X="hep+(cr-
sh/2)*cos(stang+(secang/2)*0.6)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang+(secang/2)*0.6)"/>
<Point X="hep+(cr-
sh/2)*cos(stang+(secang/2)*0.7)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang+(secang/2)*0.7)"/>
<Point X="hep+(cr-
sh/2)*cos(stang+(secang/2)*0.8)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang+(secang/2)*0.8)"/>
<Point X="hep+(cr-
sh/2)*cos(stang+(secang/2)*0.9)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang+(secang/2)*0.9)"/>
<Point X="hep+(cr-
sh/2)*cos(stang+(secang/2)*1.0)+hs*var1" Y="0" Z="-vep+(cr-
sh/2)*sin(stang+(secang/2)*1.0)"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+0.1*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+0.1*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+0.2*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+0.2*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+0.3*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+0.3*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+0.4*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+0.4*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+0.5*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+0.5*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+0.6*pi/2))+hs*var1" Y="0" Z="-
```

```
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+0.6*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+0.7*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+0.7*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+0.8*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+0.8*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+0.9*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+0.9*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+1.0*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+1.0*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+1.1*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+1.1*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+1.2*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+1.2*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+1.3*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+1.3*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+1.4*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+1.4*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+1.5*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+1.5*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+1.6*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+1.6*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+1.7*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+1.7*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+1.8*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+1.8*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+1.9*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+1.9*pi/2))"/>
<Point
X="hep+cr*cos(stang+(secang/2)*1.0)+(sh/2)*cos(((pi-
(stang+(secang/2)))+2.0*pi/2))+hs*var1" Y="0" Z="-
vep+cr*sin(stang+(secang/2)*1.0)-(sh/2)*sin(((pi-
(stang+(secang/2)))+2.0*pi/2))"/>
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*1.0)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*1.0)"/>
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*0.9)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*0.9)"/>
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*0.8)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*0.8)"/>
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*0.7)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*0.7)"/>
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*0.6)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*0.6)"/>
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*0.5)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*0.5)"/>
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*0.4)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*0.4)"/>
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*0.3)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*0.3)"/>
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*0.2)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*0.2)"/>
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*0.1)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*0.1)"/>
```

```
<Point
X="hep+(cr+sh/2)*cos(stang+(secang/2)*0.0)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang+(secang/2)*0.0)"/>
<Point
X="hep+(cr+sh/2)*cos(stang-(secang/2)*0.1)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang-(secang/2)*0.1)"/>
<Point
X="hep+(cr+sh/2)*cos(stang-(secang/2)*0.2)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang-(secang/2)*0.2)"/>
<Point
X="hep+(cr+sh/2)*cos(stang-(secang/2)*0.3)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang-(secang/2)*0.3)"/>
<Point
X="hep+(cr+sh/2)*cos(stang-(secang/2)*0.4)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang-(secang/2)*0.4)"/>
<Point
X="hep+(cr+sh/2)*cos(stang-(secang/2)*0.5)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang-(secang/2)*0.5)"/>
<Point
X="hep+(cr+sh/2)*cos(stang-(secang/2)*0.6)+hs*var1" Y="0" Z="- vep+(cr+sh/2)*sin(stang-
(secang/2)*0.6)"/>
<Point
X="hep+(cr+sh/2)*cos(stang-(secang/2)*0.7)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang-(secang/2)*0.7)"/>
<Point
X="hep+(cr+sh/2)*cos(stang-(secang/2)*0.8)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang-(secang/2)*0.8)"/>
<Point
X="hep+(cr+sh/2)*cos(stang-(secang/2)*0.9)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang-(secang/2)*0.9)"/>
<Point
X="hep+(cr+sh/2)*cos(stang-(secang/2)*1.0)+hs*var1" Y="0" Z="-
vep+(cr+sh/2)*sin(stang-(secang/2)*1.0)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-0.1*pi/2)+hs*var1" Y="0"
Z="-vep+cr*sin(stang-(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))-
0.1*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-0.2*pi/2)+hs*var1" Y="0"
Z="-vep+cr*sin(stang-(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))-
0.2*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-0.3*pi/2)+hs*var1" Y="0"
Z="-vep+cr*sin(stang-(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))-
0.3*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-0.4*pi/2)+hs*var1" Y="0"
Z="-vep+cr*sin(stang-(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))-
0.4*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-0.5*pi/2)+hs*var1" Y="0"
```

```
Z="-vep+cr*sin(stang-(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))-
0.5*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-0.6*pi/2)+hs*var1" Y="0"
Z="-vep+cr*sin(stang-(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))-
0.6*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-0.7*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 0.7*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-0.8*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 0.8*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-0.9*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 0.9*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-1.0*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 1.0*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-1.1*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 1.1*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-1.2*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 1.2*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-1.3*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 1.3*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-1.4*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 1.4*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-1.5*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 1.5*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-1.6*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 1.6*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-1.7*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 1.7*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-1.8*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 1.8*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-1.9*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 1.9*pi/2)"/>
<Point X="hep+cr*cos(stang-
(secang/2)*1.0)+(sh/2)*cos((stang-(secang/2))-2.0*pi/2)+hs*var1" Y="0" Z="-vep+cr*sin(stang-
(secang/2)*1.0)+(sh/2)*sin((stang-(secang/2))- 2.0*pi/2)"/>
</Polygon>
</Repeat>
</Surface>
```

```
<Surface RefObj="webSurface" Y="tw"/>
</Volume>
<Volume Color="Yellow">
<Surface Name="bfSurface" Z="-wd-btf">
<Point X="0" Y="-bbf/2" Z="0"/>
<Point X="l" Y="-bbf/2" Z="0"/>
<Point X="l" Y="bbf/2" Z="0"/>
<Point X="0" Y="bbf/2" Z="0"/>
</Surface>
<Surface RefObj="bfSurface" Z="-wd"/>
</Volume>
</Obj>
```

Figure 4-60 shows 3D view of the model of a part with a curved slotted hole using the OpenBrim Viewer.

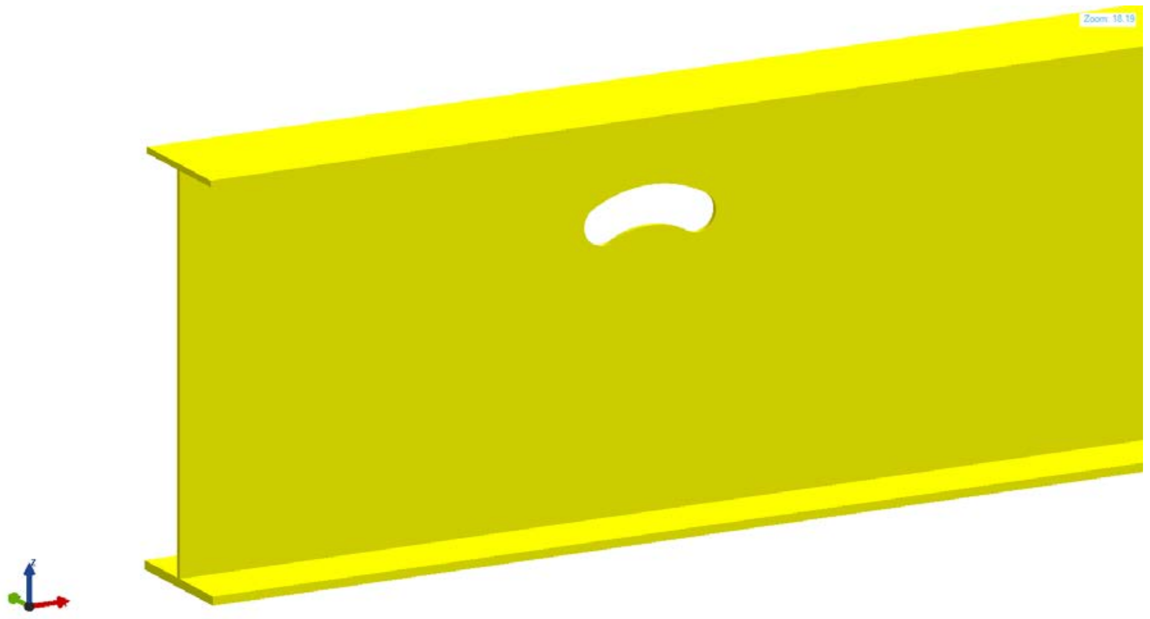


Figure 4-60. 3D View of Curved Slotted Hole in the OpenBrIM Viewer

4.5.1.4.2 Mapping to ISM

The curved slotted hole feature can be modeled by using IIsmFeatureSubtraction interface.

4.5.1.4.3 Mapping to IFC

Properties	**IFC Entities/Attributes**
Dimension	IfcArbitraryClosedProfileDef IfcProperty \ NominalValue
Other Information	Follows the mapping of Holes for Fasteners

4.6 Joint Systems

Joint systems are used to connect bridge components. They can be categorized into the following types: mechanical (e.g., bolt assembly), chemical (e.g., sealant), welded, amorphous (e.g., grout) and compound (e.g., welded shear studs). This section covers the description and specification of generic and specific joint systems.

A joint system is the combination of ways that two bridge components are connected and the geometric description of that connection.

4.6.1 Fastener

A fastener is a hardware device that mechanically connects or affixes two or more components. Fasteners include bolt, nut, washer, stud, nail, pin, screw, shear stud and countersunk fastener. Fastener should have at least the following attributes:

- Fastener grade
- Fastener sequence
- Nominal diameter
- Nominal length

4.6.2 Bolted Joint System

Bolt assembly is classified as mechanical joint system in this protocol, because a bolt assembly acts as a mechanical fastener in a joint system. A bolt assembly includes a bolt, a nut and two washers (i.e., bolt side washer and nut side washer), as shown in Figure 4-61.

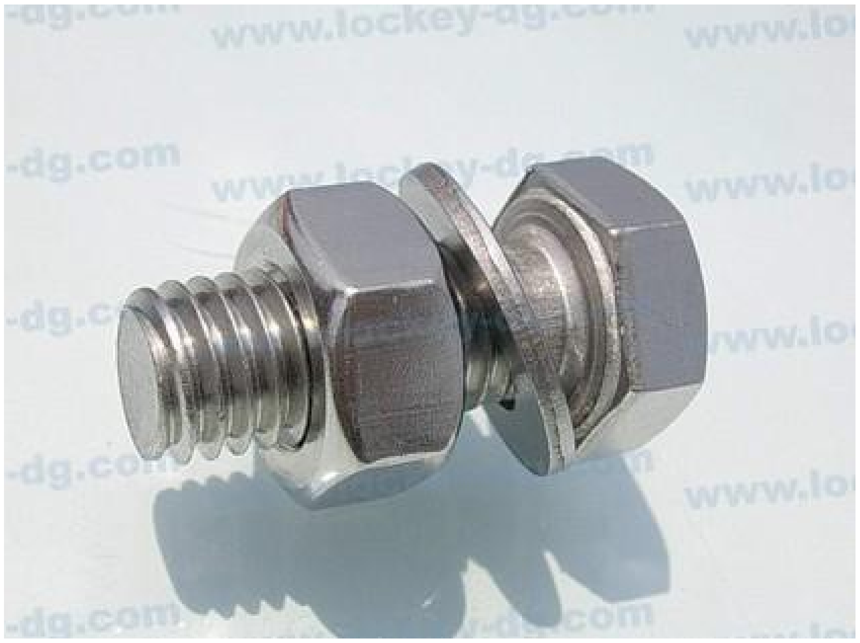


Figure 4-61. Bolt Assembly

Usually, steel bridge components are coupled by bolt connections. For example, long-span steel plate girders are assembled by using a bolted field splice, as shown in Figure 4-62.

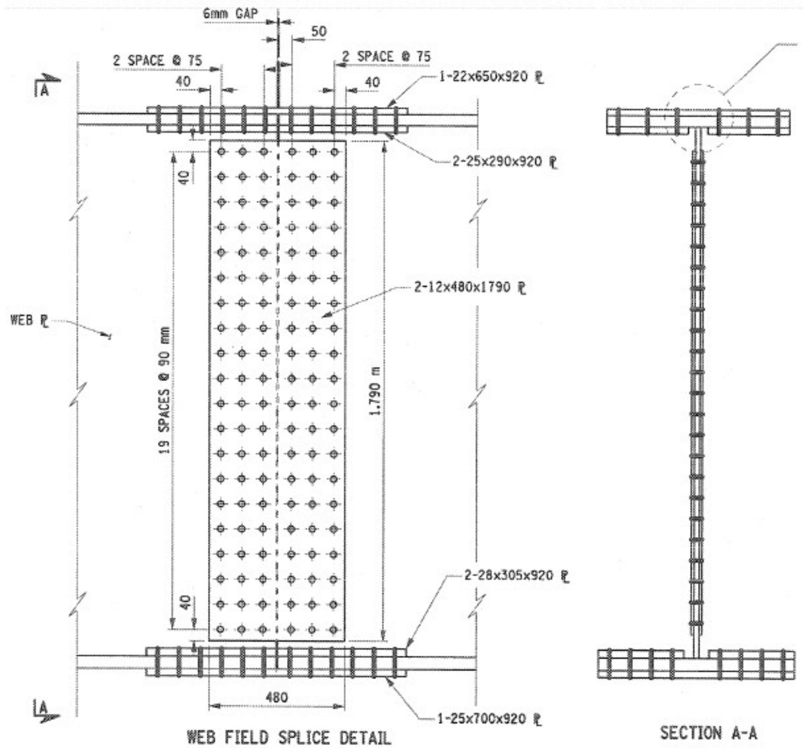


Figure 4-62. Bolted Field Splice

4.6.2.1 Hexagonal Head Bolt

Hexagonal head bolt is a type of bolt whose head shape is hexagonal. It should have at least the following attributes for geometric description:

- Bolt head distance across flats.
- Bolt head thickness.
- Bolt shank diameter.
- Bolt shank length.

4.6.2.1.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<ObjType Name="Hex Head Bolt">
<Param Name="bhs" Label="Bolt Head Size" Value="12.7"/>
<Param Name="bht" Label="Bolt Head Thickness" Value="7"/>
<Param Name="bss" Label="Bolt Shank Size" Value="6"/>
<Param Name="bsl" Label="Bolt Shank Length" Value="50"/>
<Line Color="Yellow">
<Point X="0" Y="0" Z="0"/>
<Point X="bht" Y="0" Z="0"/>
<Surface>
<Point X="bhs*tan(pi/6)" Y="bhs"/>
<Point X="bhs/cos(pi/6)" Y="0"/>
<Point X="bhs*tan(pi/6)" Y="-bhs"/>
```

```
<Point X="-bhs*tan(pi/6)" Y="-bhs"/>
<Point X="-bhs/cos(pi/6)" Y="0"/>
<Point X="-bhs*tan(pi/6)" Y="bhs"/>
</Surface>
</Line>
<Line Color="Yellow">
<Point X="bht" Y="0" Z="0"/>
<Point X="bht+bsl" Y="0" Z="0"/>
<Circle X="0" Y="0" Radius="bss"/>
</Line>
</ObjType>
<Obj Name="Bolt 1" RefObj="Hex Head Bolt"/>
```

Figure 4-63 shows 3D view of the model of a hexagonal head bolt in the OpenBrimViewer.

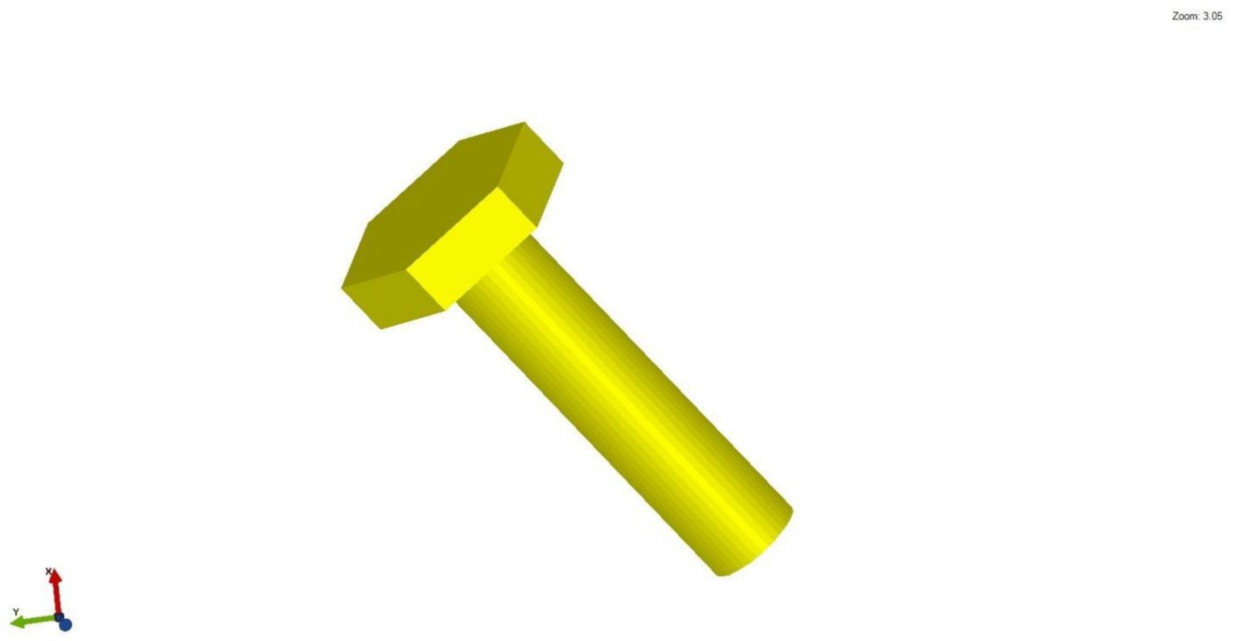


Figure 4-63. 3D View of Hexagonal Head Bolt

4.6.2.1.2 Mapping to IFC

Bolts in IFC are defined as IfcMechanicalFastener. This group contains fasteners connecting building elements mechanically. The exact type information of the IfcMechanicalFastener is given in the ObjectType attribute inherited from IfcObject. A bolt in IFC is modeled using threaded cylindrical rod that engages with a similarly threaded hole in a nut or any other part to form a fastener.

4.6.2.1.3 Mapping to ISM

1) Physical Geometry Properties of IIsmCurveMembers for Bolt

Physical geometry properties of bolt including Section, Location, PlacementPoint, Orientation, and MirrorShapeAboutYAxis will be demonstrated in this section.

2) Section of Bolt

To simplify the modeling, screw and countersink are ignored. The components of a bolt assembly can be modeled as linear member with constant section. The hexagonal head bolt has two sections. The hexagonal head section can be modeled using IIsmCustomSection interface, while the shank section can be modeled using IIsmParametricSection interface. They will be grouped by using IIsmVaryingSection interface. The sections of nut and washer have a circular void in the center. They can be modeled by using IIsmBuiltUpSection.

3) Location of Bolt

Member line of a bolt assembly which is defined by using locations of two end points is located in the center, as shown in Figure 4-64.

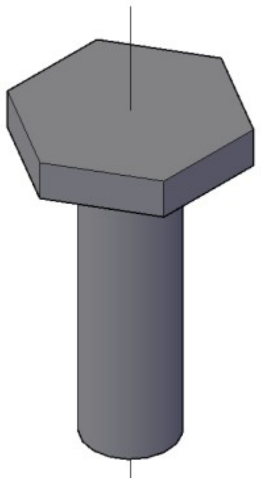


Figure 4-64. Member Line Location of Bolt

4) Placement Point of Bolt

Placement point of a bolt assembly is located at the center, as shown in Figure 4-65.

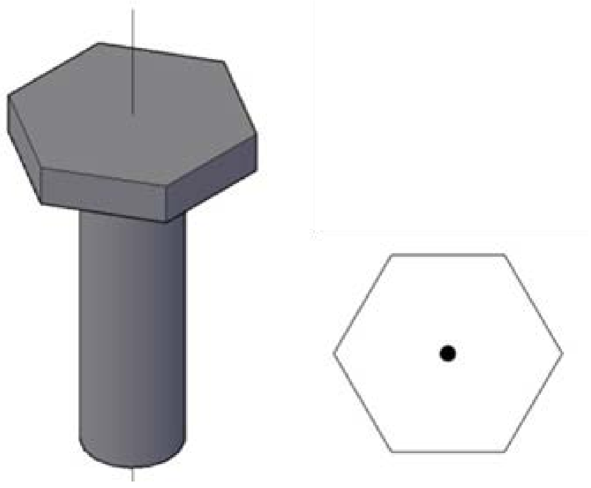


Figure 4-65. Placement Point of Bolt Assembly (Isometric View and Section View)

5) Orientation of Bolt

Orientation of cross sections of bolt assembly is perpendicular to the location, as shown in Figure 4-66.

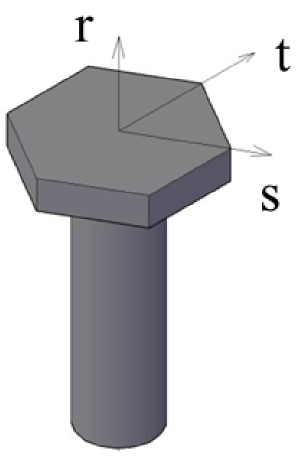


Figure 4-66. Orientation of Bolt

6) MirrorShapeAboutYAxis of Bolt

The Boolean value of this parameter is “True” for bolt assembly.

7) Summary of Modeling Bolt

From the analysis above, we can conclude that the bolt assembly can be modeled by using IIsmCurveMembers interface in the current ISM schema. Their section can be defined by using IIsmParametricSection, IIsmCustomSection and IIsmBuiltUpSection interfaces.

4.6.2.2 Circular Head Bolt

Circular head bolt is a type of bolt whose head shape is circular. It should have at least the following attributes for geometric description:

- Bolt head diameter.
- Bolt head thickness.
- Bolt shank diameter.
- Bolt shank length.

4.6.2.2.1 Mapping to OpenBrIM schema

The XML code created based on the OpenBrIm schema is shown as follows:

```
<ObjType Name="Cir Head Bolt">
<Param Name="bhs" Label="Bolt Head Size" Value="12.7"/>
<Param Name="bht" Label="Bolt Head Thickness" Value="7"/>
<Param Name="bss" Label="Bolt Shank Size" Value="6"/>
<Param Name="bsl" Label="Bolt Shank Length" Value="50"/>
<Line Color="Yellow">
```

```
<Point X="0" Y="0" Z="0"/>
<Point X="bht" Y="0" Z="0"/>
<Circle X="0" Y="0" Radius="bhs"/>
</Line>
<Line Color="Yellow">
<Point X="bht" Y="0" Z="0"/>
<Point X="bht+bsl" Y="0" Z="0"/>
<Circle X="0" Y="0" Radius="bss"/>
</Line>
</ObjType>
<Obj Name="Bolt 2" RefObj="Cir Head Bolt" Y="100"/>
```

Figure 4-67 shows 3D view of the model of a circular head bolt in the OpenBrimViewer.

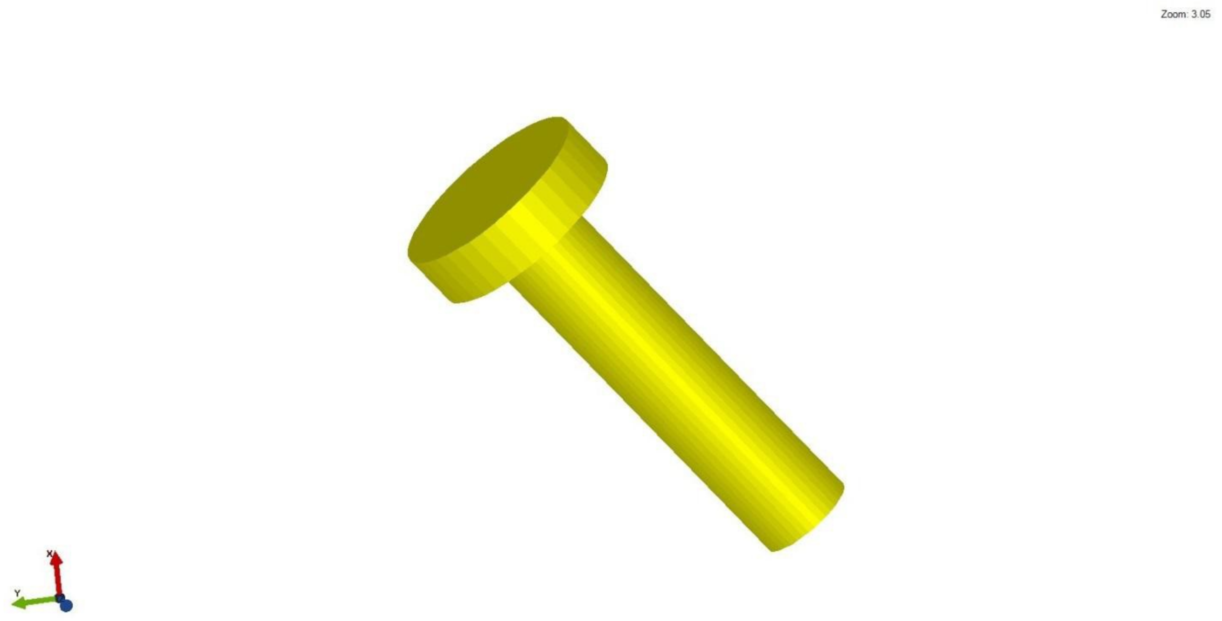


Figure 4-67. 3D View of Circular Head Bolt

4.6.2.2.2 Mapping to IFC

Bolts in IFC are defined as IfcMechanicalFastener. This group contains fasteners connecting building elements mechanically. The exact type information of the IfcMechanicalFastener is given in the ObjectType attribute inherited from IfcObject. A bolt in IFC is modeled using threaded cylindrical rod that engages with a similarly threaded hole in a nut or any other part to form a fastener.

4.6.2.2.3 Mapping to ISM

Modeling of circular head bolt using ISM schema is the same as modeling of hexagonal head bolt.

4.6.2.3 Square Head Bolt

A square head bolt is a type of bolt whose head shape is square. It should have at least the following attributes for geometric description:

- Bolt head distance across flats.
- Bolt head thickness.
- Bolt shank diameter.
- Bolt shank length.

4.6.2.3.1 Mapping to OpenBrIM schema

The XML code created based on the OpenBrIm schema is shown as follows:

```
<ObjType Name="Rect Head Bolt">
<Param Name="bhs" Label="Bolt Head Size" Value="25.4"/>
<Param Name="bht" Label="Bolt Head Thickness" Value="7"/>
<Param Name="bss" Label="Bolt Shank Size" Value="6"/>
<Param Name="bsl" Label="Bolt Shank Length" Value="50"/>
<Line Color="Yellow">
<Point X="0" Y="0" Z="0"/>
<Point X="bht" Y="0" Z="0"/>
<Surface>
<Point X="bhs/2" Y="bhs/2"/>
<Point X="-bhs/2" Y="bhs/2"/>
<Point X="-bhs/2" Y="-bhs/2"/>
<Point X="bhs/2" Y="-bhs/2"/>
</Surface>
</Line>
<Line Color="Yellow">
<Point X="bht" Y="0" Z="0"/>
<Point X="bht+bsl" Y="0" Z="0"/>
<Circle X="0" Y="0" Radius="bss"/>
</Line>
</ObjType>
<Obj Name="Bolt 3" RefObj="Rect Head Bolt" Y="200"/>
```

Figure 4-68 shows 3D view of the model of a square head bolt in the OpenBrimViewer.

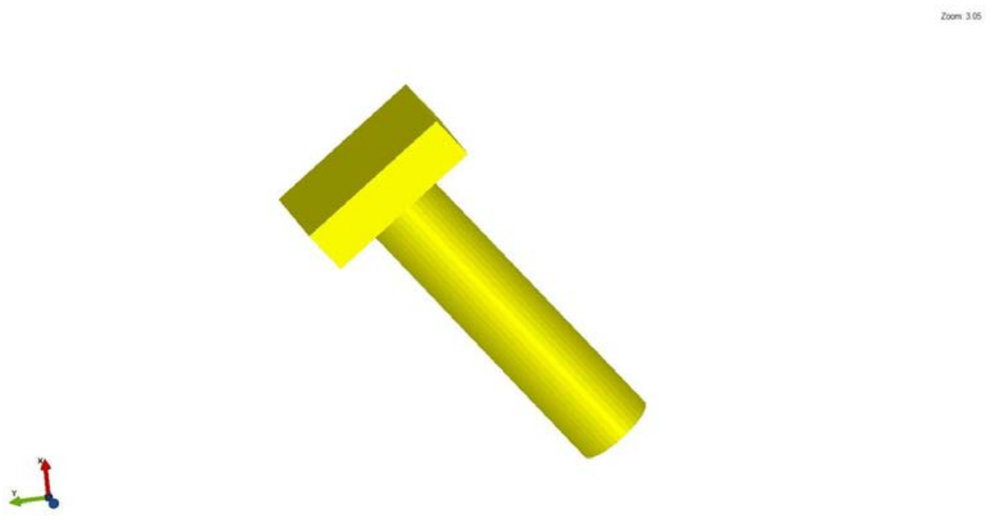


Figure 4-68. 3D View of Square Head Bolt

4.6.2.3.2 Mapping to IFC

Bolts in IFC are defined as IfcMechanicalFastener. This group contains fasteners connecting building elements mechanically. The exact type information of the IfcMechanicalFastener is given in the ObjectType attribute inherited from IfcObject. A bolt in IFC is modeled using threaded cylindrical rod that engages with a similarly threaded hole in a nut or any other part to form a fastener.

4.6.2.3.3 Mapping to ISM

Modeling of a square head bolt using ISM schema is the same asmodeling of a hexagonal head bolt.

4.6.2.4 Hexagonal Nut

Hexagonal nut is a type of nut whose shape is hexagonal. It should have at least the following attributes for geometric description.

- Nut distance across flats
- Nut inside diameter
- Nut thickness

4.6.2.4.1 Mapping to OpenBrIM Schema

The XML code created based on the OpenBrIm schema is shown as follows:

```
<ObjType Name="Hex Head Nut">
<Param Name="nos" Label="Nut Outer Size" Value="12.7"/>
<Param Name="nis" Label="Nut Inner Size" Value="6"/>
<Param Name="nt" Label="Nut thickness" Value="7"/>
<Param Name="bolt" Value="" Type="Reference"/>
<Line Color="Yellow">
<Point X="0" Y="0" Z="0"/>
<Point X="nt" Y="0" Z="0"/>
<Surface>
<Point X="nos*tan(pi/6)" Y="nos"/>
<Point X="nos/cos(pi/6)" Y="0"/>
<Point X="nos*tan(pi/6)" Y="-nos"/>
<Point X="-nos*tan(pi/6)" Y="-nos"/>
<Point X="-nos/cos(pi/6)" Y="0"/>
<Point X="-nos*tan(pi/6)" Y="nos"/>
<Surface>
<Point X="nis*Cos(pi*0.0)" Y="nis*Sin(pi*0.0)"/>
<Point X="nis*Cos(pi*0.1)" Y="nis*Sin(pi*0.1)"/>
<Point X="nis*Cos(pi*0.2)" Y="nis*Sin(pi*0.2)"/>
<Point X="nis*Cos(pi*0.3)" Y="nis*Sin(pi*0.3)"/>
<Point X="nis*Cos(pi*0.4)" Y="nis*Sin(pi*0.4)"/>
<Point X="nis*Cos(pi*0.5)" Y="nis*Sin(pi*0.5)"/>
<Point X="nis*Cos(pi*0.6)" Y="nis*Sin(pi*0.6)"/>
<Point X="nis*Cos(pi*0.7)" Y="nis*Sin(pi*0.7)"/>
<Point X="nis*Cos(pi*0.8)" Y="nis*Sin(pi*0.8)"/>
<Point X="nis*Cos(pi*0.9)" Y="nis*Sin(pi*0.9)"/>
<Point X="nis*Cos(pi*1.0)" Y="nis*Sin(pi*1.0)"/>
```

```
<Point X="nis*Cos(pi*1.1)" Y="nis*Sin(pi*1.1)"/>
<Point X="nis*Cos(pi*1.2)" Y="nis*Sin(pi*1.2)"/>
<Point X="nis*Cos(pi*1.3)" Y="nis*Sin(pi*1.3)"/>
<Point X="nis*Cos(pi*1.4)" Y="nis*Sin(pi*1.4)"/>
<Point X="nis*Cos(pi*1.5)" Y="nis*Sin(pi*1.5)"/>
<Point X="nis*Cos(pi*1.6)" Y="nis*Sin(pi*1.6)"/>
<Point X="nis*Cos(pi*1.7)" Y="nis*Sin(pi*1.7)"/>
<Point X="nis*Cos(pi*1.8)" Y="nis*Sin(pi*1.8)"/>
<Point X="nis*Cos(pi*1.9)" Y="nis*Sin(pi*1.9)"/>
<Point X="nis*Cos(pi*2.0)" Y="nis*Sin(pi*2.0)"/>
</Surface>
</Surface>
</Line>
</ObjType>
<Obj Name="Nut 1" RefObj="Hex Head Nut" X="bolt.bht+bolt.bsl">
<Param Name="bolt" Value="Bolt 1" Type="Reference"/>
</Obj>
```

Figure 4-69 shows 3D view of the model of a hexagonal nut in the OpenBrimViewer.

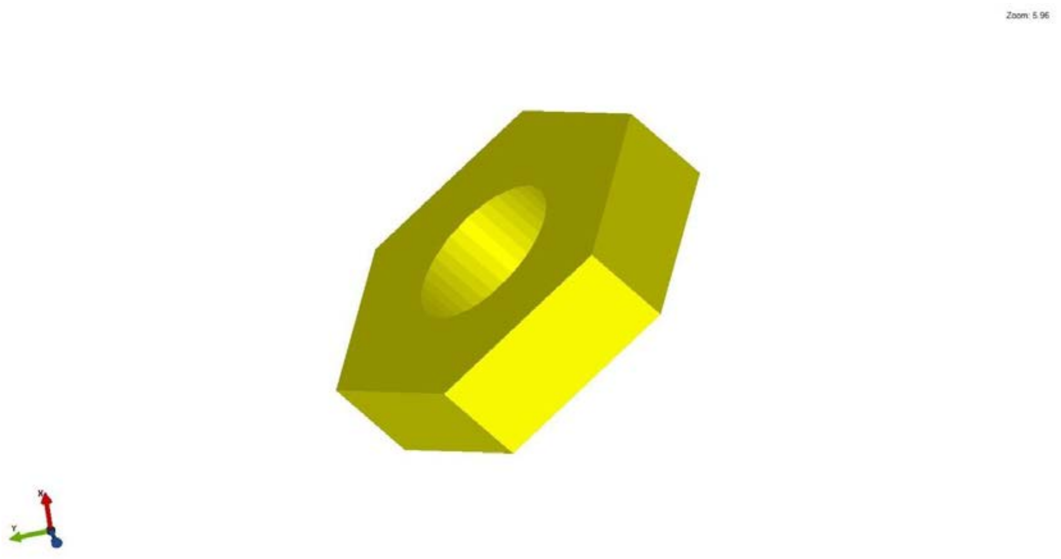


Figure 4-69. 3D View of Hexagonal Nut

4.6.2.4.2 Mapping to IFC

Nuts in IFC are defined as IfcMechanicalFastener. This group contains fasteners connecting building elements mechanically. The exact type information of the IfcMechanicalFastener is given in the ObjectType attribute inherited from IfcObject. A nut in IFC is modeled using a small square or hexagonal metal block with internal screw thread to be fitted onto a bolt.

4.6.2.4.3 Mapping to ISM

Modeling of hexagonal nut using ISM schema is the same as modeling of hexagonal head bolt.

4.6.2.5 Circular Nut

A circular nut is a type of nut whose shape is circular. It should have at least the following attributes for geometric description.

- Nut outer diameter
- Nut inside diameter
- Nut thickness

4.6.2.5.1 Mapping to OpenBrIM Schema

The XML code created based on the OpenBrIm schema is shown as follows:

```
<ObjType Name="Cir Head Nut">
<Param Name="nos" Label="Nut Outer Size" Value="12.7"/>
<Param Name="nis" Label="Nut Inner Size" Value="6"/>
<Param Name="nt" Label="Nut thickness" Value="7"/>
<Line Color="Yellow">
<Point X="0" Y="0" Z="0"/>
<Point X="nt" Y="0" Z="0"/>
<Surface>
<Point  X="nos*Cos(pi*0.0)"  Y="nos*Sin(pi*0.0)"/>
<Point  X="nos*Cos(pi*0.1)"  Y="nos*Sin(pi*0.1)"/>
<Point  X="nos*Cos(pi*0.2)"  Y="nos*Sin(pi*0.2)"/>
<Point  X="nos*Cos(pi*0.3)"  Y="nos*Sin(pi*0.3)"/>
<Point  X="nos*Cos(pi*0.4)"  Y="nos*Sin(pi*0.4)"/>
<Point  X="nos*Cos(pi*0.5)"  Y="nos*Sin(pi*0.5)"/>
<Point  X="nos*Cos(pi*0.6)"  Y="nos*Sin(pi*0.6)"/>
<Point  X="nos*Cos(pi*0.7)"  Y="nos*Sin(pi*0.7)"/>
<Point  X="nos*Cos(pi*0.8)"  Y="nos*Sin(pi*0.8)"/>
<Point  X="nos*Cos(pi*0.9)"  Y="nos*Sin(pi*0.9)"/>
<Point  X="nos*Cos(pi*1.0)"  Y="nos*Sin(pi*1.0)"/>
<Point  X="nos*Cos(pi*1.1)"  Y="nos*Sin(pi*1.1)"/>
<Point  X="nos*Cos(pi*1.2)"  Y="nos*Sin(pi*1.2)"/>
<Point  X="nos*Cos(pi*1.3)"  Y="nos*Sin(pi*1.3)"/>
<Point  X="nos*Cos(pi*1.4)"  Y="nos*Sin(pi*1.4)"/>
<Point  X="nos*Cos(pi*1.5)"  Y="nos*Sin(pi*1.5)"/>
<Point  X="nos*Cos(pi*1.6)"  Y="nos*Sin(pi*1.6)"/>
<Point  X="nos*Cos(pi*1.7)"  Y="nos*Sin(pi*1.7)"/>
<Point  X="nos*Cos(pi*1.8)"  Y="nos*Sin(pi*1.8)"/>
<Point  X="nos*Cos(pi*1.9)"  Y="nos*Sin(pi*1.9)"/>
<Point  X="nos*Cos(pi*2.0)"  Y="nos*Sin(pi*2.0)"/>
<Surface>
<Point X="nis*Cos(pi*0.0)" Y="nis*Sin(pi*0.0)"/>
<Point X="nis*Cos(pi*0.1)" Y="nis*Sin(pi*0.1)"/>
<Point X="nis*Cos(pi*0.2)" Y="nis*Sin(pi*0.2)"/>
<Point X="nis*Cos(pi*0.3)" Y="nis*Sin(pi*0.3)"/>
<Point X="nis*Cos(pi*0.4)" Y="nis*Sin(pi*0.4)"/>
<Point X="nis*Cos(pi*0.5)" Y="nis*Sin(pi*0.5)"/>
<Point X="nis*Cos(pi*0.6)" Y="nis*Sin(pi*0.6)"/>
<Point X="nis*Cos(pi*0.7)" Y="nis*Sin(pi*0.7)"/>
```

```
<Point X="nis*Cos(pi*0.8)" Y="nis*Sin(pi*0.8)"/>
<Point X="nis*Cos(pi*0.9)" Y="nis*Sin(pi*0.9)"/>
<Point X="nis*Cos(pi*1.0)" Y="nis*Sin(pi*1.0)"/>
<Point X="nis*Cos(pi*1.1)" Y="nis*Sin(pi*1.1)"/>
<Point X="nis*Cos(pi*1.2)" Y="nis*Sin(pi*1.2)"/>
<Point X="nis*Cos(pi*1.3)" Y="nis*Sin(pi*1.3)"/>
<Point X="nis*Cos(pi*1.4)" Y="nis*Sin(pi*1.4)"/>
<Point X="nis*Cos(pi*1.5)" Y="nis*Sin(pi*1.5)"/>
<Point X="nis*Cos(pi*1.6)" Y="nis*Sin(pi*1.6)"/>
<Point X="nis*Cos(pi*1.7)" Y="nis*Sin(pi*1.7)"/>
<Point X="nis*Cos(pi*1.8)" Y="nis*Sin(pi*1.8)"/>
<Point X="nis*Cos(pi*1.9)" Y="nis*Sin(pi*1.9)"/>
<Point X="nis*Cos(pi*2.0)" Y="nis*Sin(pi*2.0)"/>
</Surface>
</Surface>
</Line>
</ObjType>
<Obj Name="Nut 2" RefObj="Cir Head Nut" X="bolt.bht+bolt.bsl" Y="bolt.Y">
<Param Name="bolt" Value="Bolt 2" Type="Reference"/>
</Obj>
```

Figure 4-70 shows 3D view of the model of a circular nut in the OpenBrimViewer.

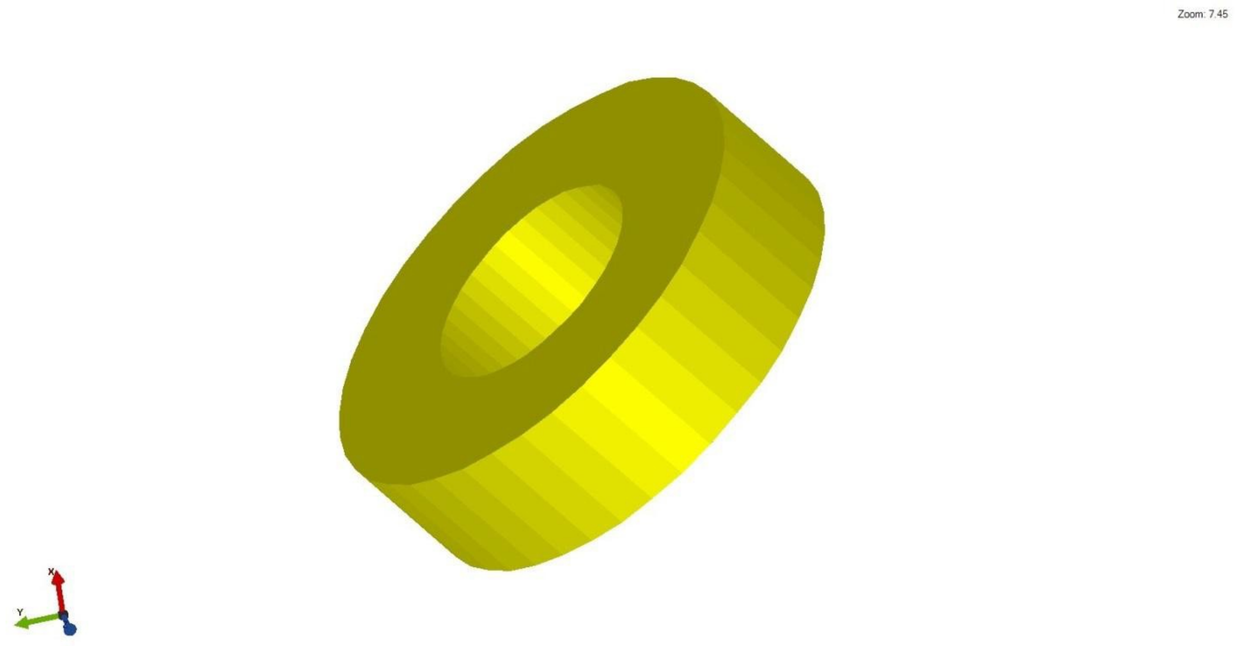


Figure 4-70. 3D View of Circular Nut

4.6.2.5.2 Mapping to IFC

Nuts in IFC are defined as IfcMechanicalFastener. This group contains fasteners connecting building elements mechanically. The exact type information of the IfcMechanicalFastener is given in the

ObjectType attribute inherited from IfcObject. A nut in IFC is modeled using a small square or hexagonal metal block with internal screw thread to be fitted onto a bolt.

4.6.2.5.3 Mapping to ISM

Modeling of a circular nut using ISM schema is the same as modeling of a hexagonal head bolt.

4.6.2.6 Square Nut

Square nut is a type of nut whose shape is square. It should have at least the following attributes for geometric description.

- Nut distance across flats
- Nut inside diameter
- Nut thickness

4.6.2.6.1 Mapping to OpenBrIM schema

```
<ObjType Name="Rect Head Nut">
<Param Name="nos" Label="Nut Outer Size" Value="25.4"/>
<Param Name="nis" Label="Nut Inner Size" Value="6"/>
<Param Name="nt" Label="Nut thickness" Value="7"/>
<Line Color="Yellow">
<Point X="0" Y="0" Z="0"/>
<Point X="nt" Y="0" Z="0"/>
<Surface>
<Point X="nos/2" Y="nos/2"/>
<Point X="-nos/2" Y="nos/2"/>
<Point X="-nos/2" Y="-nos/2"/>
<Point X="nos/2" Y="-nos/2"/>
<Surface>
<Point X="nis*Cos(pi*0.0)" Y="nis*Sin(pi*0.0)"/>
<Point X="nis*Cos(pi*0.1)" Y="nis*Sin(pi*0.1)"/>
<Point X="nis*Cos(pi*0.2)" Y="nis*Sin(pi*0.2)"/>
<Point X="nis*Cos(pi*0.3)" Y="nis*Sin(pi*0.3)"/>
<Point X="nis*Cos(pi*0.4)" Y="nis*Sin(pi*0.4)"/>
<Point X="nis*Cos(pi*0.5)" Y="nis*Sin(pi*0.5)"/>
<Point X="nis*Cos(pi*0.6)" Y="nis*Sin(pi*0.6)"/>
<Point X="nis*Cos(pi*0.7)" Y="nis*Sin(pi*0.7)"/>
<Point X="nis*Cos(pi*0.8)" Y="nis*Sin(pi*0.8)"/>
<Point X="nis*Cos(pi*0.9)" Y="nis*Sin(pi*0.9)"/>
<Point X="nis*Cos(pi*1.0)" Y="nis*Sin(pi*1.0)"/>
<Point X="nis*Cos(pi*1.1)" Y="nis*Sin(pi*1.1)"/>
<Point X="nis*Cos(pi*1.2)" Y="nis*Sin(pi*1.2)"/>
<Point X="nis*Cos(pi*1.3)" Y="nis*Sin(pi*1.3)"/>
<Point X="nis*Cos(pi*1.4)" Y="nis*Sin(pi*1.4)"/>
<Point X="nis*Cos(pi*1.5)" Y="nis*Sin(pi*1.5)"/>
<Point X="nis*Cos(pi*1.6)" Y="nis*Sin(pi*1.6)"/>
<Point X="nis*Cos(pi*1.7)" Y="nis*Sin(pi*1.7)"/>
<Point X="nis*Cos(pi*1.8)" Y="nis*Sin(pi*1.8)"/>
<Point X="nis*Cos(pi*1.9)" Y="nis*Sin(pi*1.9)"/>
```

```
<Point X="nis*Cos(pi*2.0)" Y="nis*Sin(pi*2.0)"/>
</Surface>
</Surface>
</Line>
</ObjType>
<Obj    Name="Nut    3"      RefObj="Rect  Head   Nut"    X="bolt.bht+bolt.bsl" Y="bolt.Y">
<Param Name="bolt" Value="Bolt 3" Type="Reference"/>
</Obj>
```

Figure 4-71 shows 3D view of the model of a square nut in the OpenBrimViewer.

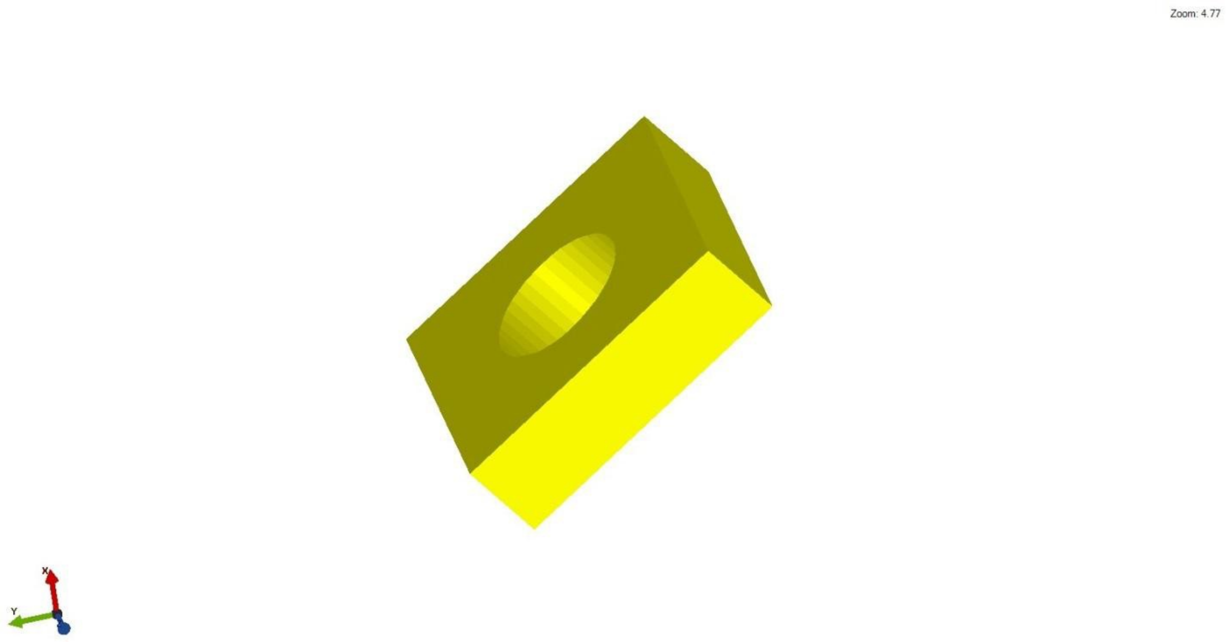


Figure 4-71. 3D View of Square Nut

4.6.2.6.2 Mapping to IFC

Nuts in IFC are defined as IfcMechanicalFastener. This group contains fasteners connecting building elements mechanically. The exact type information of the IfcMechanicalFastener is given in the ObjectType attribute inherited from IfcObject. A nut in IFC is modeled using a small square or hexagonal metal block with an internal screw thread to be fitted onto a bolt.

4.6.2.6.3 Mapping to ISM

Modeling of a square nut using ISM schema is the same as modeling of a hexagonal head bolt.

4.6.2.7 Washer

A washer (Figure 4-72) should have at least the following attributes:

- Washer outer diameter
- Washer inside diameter

- Washer thickness
- Associated component

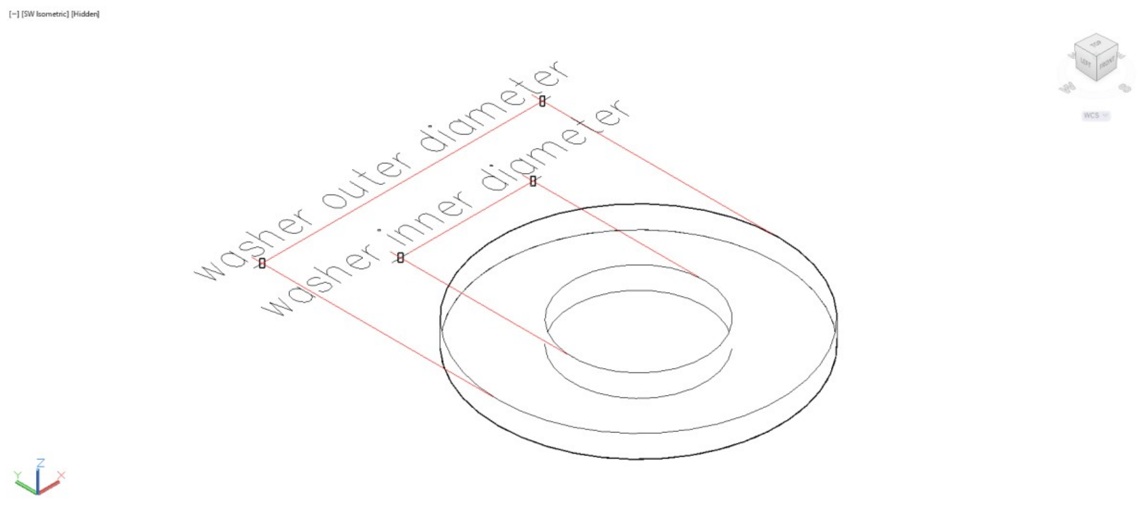


Figure 4-72. Washer Properties

4.6.2.7.1 Mapping to OpenBrIM Schema

The XML code created based on the OpenBrIm schema is shown as follows:

```
<ObjType Name="Washer">
<Param Name="wos" Label="Washer Outer Size" Value="15"/>
<Param Name="wis" Label="Washer Inner Size" Value="6"/>
<Param Name="wt" Label="Washer Thickness" Value="2"/>
<Param Name="bolt" Value="" Type="Reference"/>
<Param Name="nut" Value="" Type="Reference"/>
<Line Color="Cyan">
<Point X="0" Y="0" Z="0"/>
<Point X="wt" Y="0" Z="0"/>
<Surface>
<Point X="wos*Cos(pi*0.0)" Y="wos*Sin(pi*0.0)"/>
<Point X="wos*Cos(pi*0.1)" Y="wos*Sin(pi*0.1)"/>
<Point X="wos*Cos(pi*0.2)" Y="wos*Sin(pi*0.2)"/>
<Point X="wos*Cos(pi*0.3)" Y="wos*Sin(pi*0.3)"/>
<Point X="wos*Cos(pi*0.4)" Y="wos*Sin(pi*0.4)"/>
<Point X="wos*Cos(pi*0.5)" Y="wos*Sin(pi*0.5)"/>
<Point X="wos*Cos(pi*0.6)" Y="wos*Sin(pi*0.6)"/>
<Point X="wos*Cos(pi*0.7)" Y="wos*Sin(pi*0.7)"/>
<Point X="wos*Cos(pi*0.8)" Y="wos*Sin(pi*0.8)"/>
<Point X="wos*Cos(pi*0.9)" Y="wos*Sin(pi*0.9)"/>
<Point X="wos*Cos(pi*1.0)" Y="wos*Sin(pi*1.0)"/>
<Point X="wos*Cos(pi*1.1)" Y="wos*Sin(pi*1.1)"/>
<Point X="wos*Cos(pi*1.2)" Y="wos*Sin(pi*1.2)"/>
<Point X="wos*Cos(pi*1.3)" Y="wos*Sin(pi*1.3)"/>
<Point X="wos*Cos(pi*1.4)" Y="wos*Sin(pi*1.4)"/>
```

```
<Point X="wos*Cos(pi*1.5)" Y="wos*Sin(pi*1.5)"/>
<Point X="wos*Cos(pi*1.6)" Y="wos*Sin(pi*1.6)"/>
<Point X="wos*Cos(pi*1.7)" Y="wos*Sin(pi*1.7)"/>
<Point X="wos*Cos(pi*1.8)" Y="wos*Sin(pi*1.8)"/>
<Point X="wos*Cos(pi*1.9)" Y="wos*Sin(pi*1.9)"/>
<Point X="wos*Cos(pi*2.0)" Y="wos*Sin(pi*2.0)"/>
<Surface>
<Point X="wis*Cos(pi*0.0)" Y="wis*Sin(pi*0.0)"/>
<Point X="wis*Cos(pi*0.1)" Y="wis*Sin(pi*0.1)"/>
<Point X="wis*Cos(pi*0.2)" Y="wis*Sin(pi*0.2)"/>
<Point X="wis*Cos(pi*0.3)" Y="wis*Sin(pi*0.3)"/>
<Point X="wis*Cos(pi*0.4)" Y="wis*Sin(pi*0.4)"/>
<Point X="wis*Cos(pi*0.5)" Y="wis*Sin(pi*0.5)"/>
<Point X="wis*Cos(pi*0.6)" Y="wis*Sin(pi*0.6)"/>
<Point X="wis*Cos(pi*0.7)" Y="wis*Sin(pi*0.7)"/>
<Point X="wis*Cos(pi*0.8)" Y="wis*Sin(pi*0.8)"/>
<Point X="wis*Cos(pi*0.9)" Y="wis*Sin(pi*0.9)"/>
<Point X="wis*Cos(pi*1.0)" Y="wis*Sin(pi*1.0)"/>
<Point X="wis*Cos(pi*1.1)" Y="wis*Sin(pi*1.1)"/>
<Point X="wis*Cos(pi*1.2)" Y="wis*Sin(pi*1.2)"/>
<Point X="wis*Cos(pi*1.3)" Y="wis*Sin(pi*1.3)"/>
<Point X="wis*Cos(pi*1.4)" Y="wis*Sin(pi*1.4)"/>
<Point X="wis*Cos(pi*1.5)" Y="wis*Sin(pi*1.5)"/>
<Point X="wis*Cos(pi*1.6)" Y="wis*Sin(pi*1.6)"/>
<Point X="wis*Cos(pi*1.7)" Y="wis*Sin(pi*1.7)"/>
<Point X="wis*Cos(pi*1.8)" Y="wis*Sin(pi*1.8)"/>
<Point X="wis*Cos(pi*1.9)" Y="wis*Sin(pi*1.9)"/>
<Point X="wis*Cos(pi*2.0)" Y="wis*Sin(pi*2.0)"/>
</Surface>
</Surface>
</Line>
</ObjType>
<Obj Name="Washer 5" RefObj="Washer" X="bolt.bht" Y="bolt.Y">
<Param Name="bolt" Value="Bolt 3" Type="Reference"/>
</Obj>
```

Figure 4-73 shows 3D view of the model of a washer in the OpenBrimViewer.

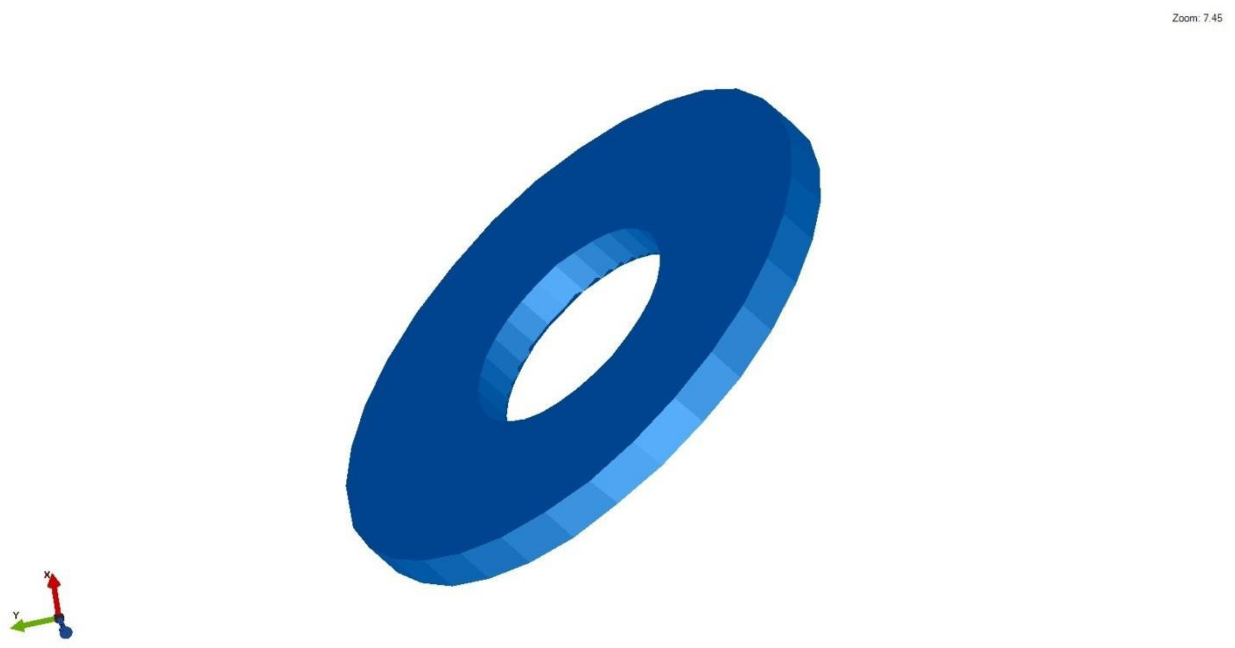


Figure 4-73. 3D View of Washer

4.6.2.7.2 Mapping to IFC

Washers in IFC are defined as IfcMechanicalFastener. This group contains fasteners connecting building elements mechanically. The exact type information of the IfcMechanicalFastener is given in the ObjectType attribute inherited from IfcObject. A washer in IFC is modeled as a disk, and of metal, plastic, rubber or other material. It is placed beneath a nut or at an axle bearing or a joint to relieve friction, prevent leakage, or distribute pressure.

4.6.2.7.3 Mapping to ISM

Modeling of washer using ISM schema is the same as modeling of hexagonal head bolt.

4.6.2.8 Tapered Washer

A tapered washer is a washer that does not have a constant thickness. It should have at least the following attributes (Figure 4-74):

- Washer outer diameter
- Washer inside diameter
- Washer thickness
- Associated component
- Taper (ratio of difference between the maximum thickness and the minimum thickness to the outer diameter)

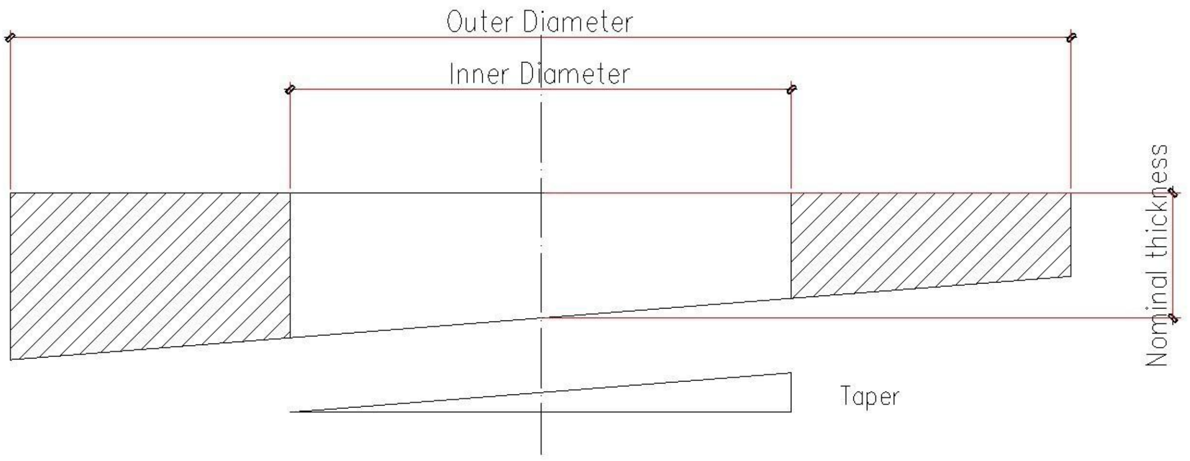


Figure 4-74. Tapered Washer Properties

4.6.2.8.1 Mapping to OpenBrIM schema

```
<ObjType Name=”TaperedWasherTemplate”>
<Param Name=”wod” Label="Washer Outer Diameter" Value="14.6247"/>
<Param Name=”wid” Label=”Washer Inner Diameter" Value="6"/>
<Param Name=”wt” Label="WasherThickness" Value="2"/>
<Param Name=”component” Label="AssociatedComponent" Value="Nut
3"/>

<Param Name="Taper" Value="0.08"/>
</ObjType>
```

4.6.2.8.2 Mapping to ISM

Modeling of a tapered washer using ISM schema is similar to the modeling of a washer. The taper feature can be defined by using the IIsmFeatureSubtraction interface.

4.6.2.9 Bolt Assembly

The geometric attributes of bolt are listed below:

- Bolt head diameter (for circular head).
- Bolt head distance across flats (for hexagonal head or square head) Bolt head thickness.
- Bolt shank diameter Bolt shank length.

Washer can be categorized into bolt side washer and nut side washer. Its geometric attributes are listed below:

- Washer outer diameter.
- Washer inner diameter.
- Washer thickness.

The geometric attributes of nut are listed below:

- Nut distance across flats.
- Nut inner diameter.
- Nut thickness.

The following attributes are necessary to locate a bolt assembly on the steel assembly:

- Parts that bolt assembly fastens together.
- End distance to a critical point in longitudinal direction.
- End distance to a critical point in transverse direction.
- Spacing of bolt assemblies in longitudinal direction.
- Spacing of bolt assemblies in transverse direction.
- Bolt preload.

Additional attributes for necessary information of bolt assembly include:

- Bolt standard.
- Washer standard.
- Nut standard.

4.6.2.9.1 Mapping to OpenBrIM Schema

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Object Name="TfBolt1" Template="Top Flange Bolt Object Template">
<Location Sta="" TOff="" Elev=""/>
<Parameters>
<Parameter Name="nbd" Value="12.7"/>
<Parameter Name="bht" Value="7"/>
<Parameter Name="bsd" Value="6"/>
<Parameter Name="bsl" Value="50"/>

<Parameter Name="wod" Value="14.6247"/>
<Parameter Name="wid" Value="6"/>
<Parameter Name="wt" Value="2"/>

<Parameter Name="ndaf" Value="12.7"/>
<Parameter Name="nid" Value="6"/>
<Parameter Name="nt" Value="7"/>

<Parameter Name="girder" Value="Girder 1"/>
<Parameter Name="part1" Value="Flange Connecting Plate 1a"/>
<Parameter Name="part2" Value="Flange Connecting Plate 1e"/>

<Parameter Name="led" Value="40"/>
<Parameter Name="ved" Value="40"/>
<Parameter Name="ls" Value="75"/>
<Parameter Name="ts" Value="105"/>
</Parameters>
</Object>
```

Figures 4-75 to 4-77 show 3D view of the model of a single bolt assembly in the OpenBrimViewer.

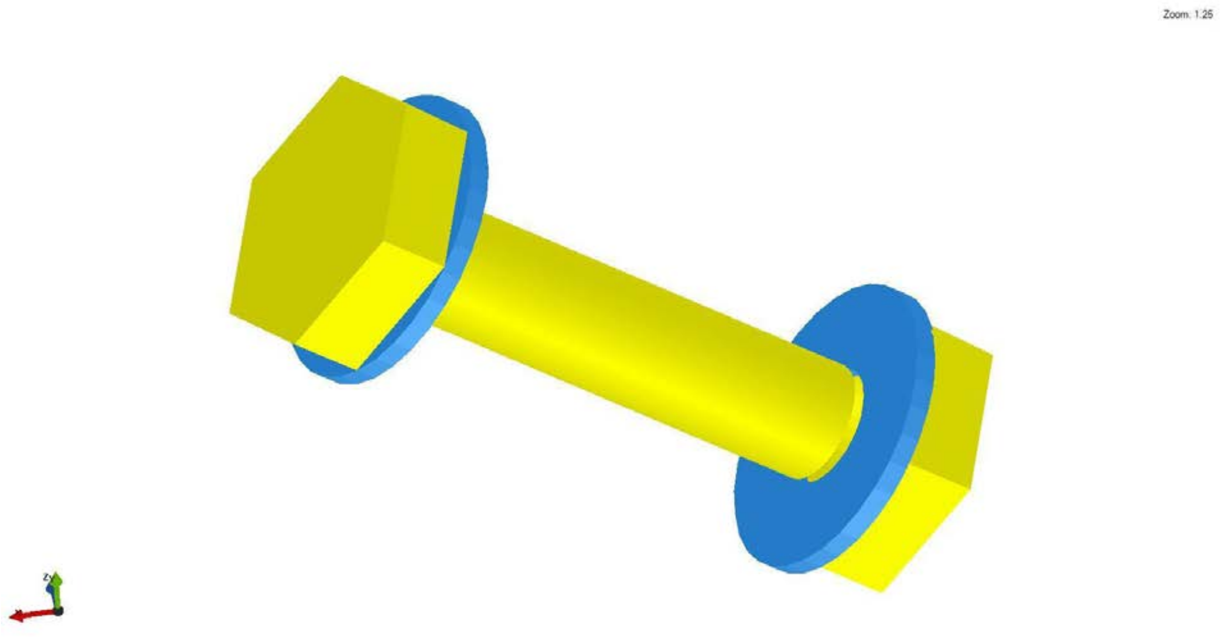


Figure 4-75. 3D View of Bolt Assembly with Hexagonal Head

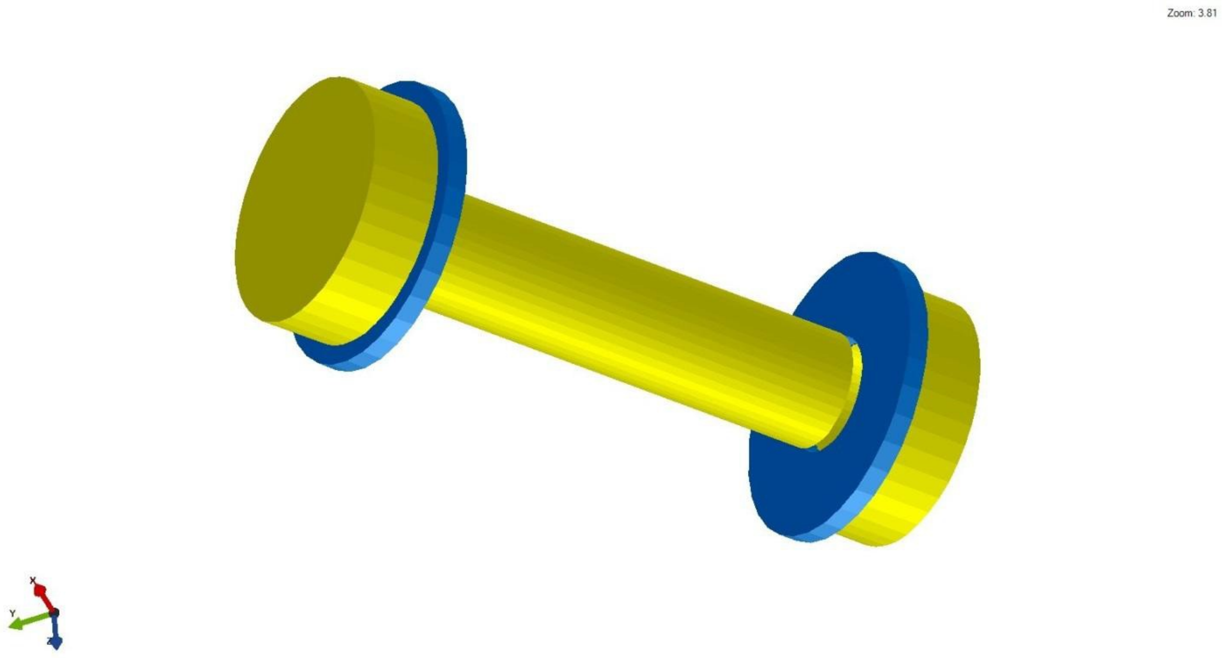


Figure 4-76. 3D View of Bolt Assembly with Circular Head

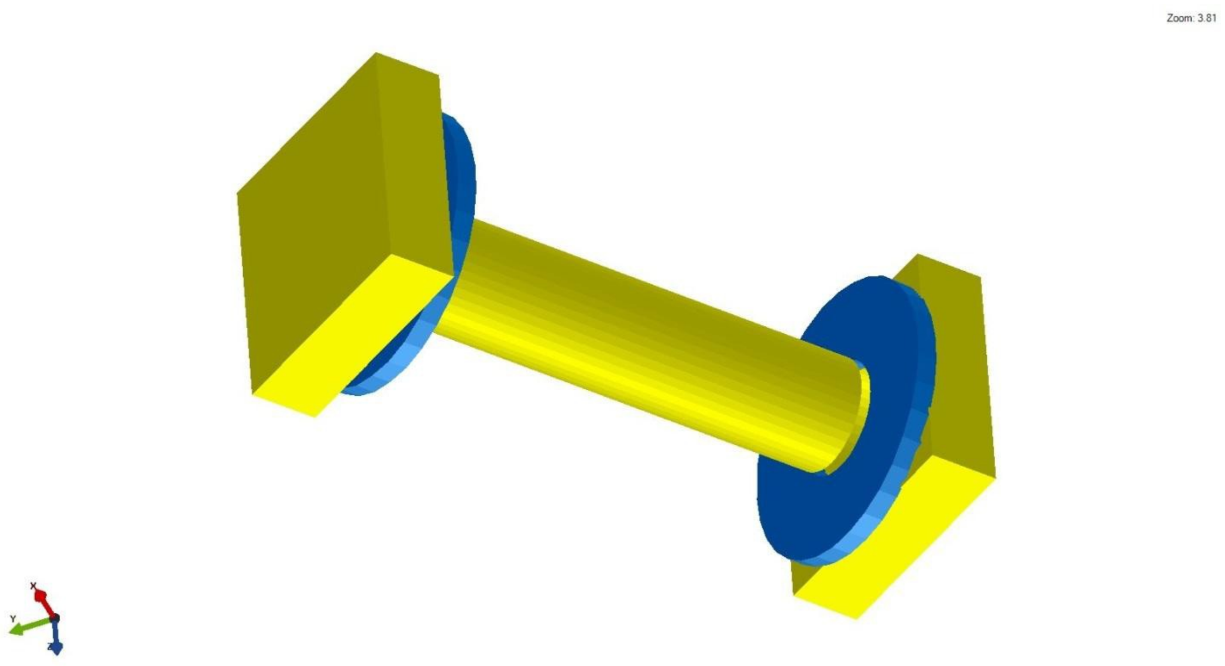


Figure 4-77. 3D View of Bolt Assembly with Rectangular Head

4.6.2.9.2 Mapping to IFC

Properties	**IFC Entities/Attributes**
Bolt Assembly	IfcMechanicalFastener
Bolt side washer	IfcRelAggregates \ RelatedObjects IfcProperty \ NominalValue
Nut side washer	IfcRelAggregates \ RelatedObjects IfcProperty \ NominalValue
Nut	IfcRelAggregates \ RelatedObjects
Location on steel element	IfcProduct \ ObjectPlacement
Part(s) it fastens to and together	IfcRelConnectsElements \ RelatedElement
Fastening direction	N/A
Fastening location	IfcProperty \ NominalValue
Bolt standard	IfcMaterial, IfcClassificationReference
Nominal bolt diameter	IfcProperty \ NominalValue
Bolt length	IfcProperty \ NominalValue
Location on bolt assembly	IfcProduct \ ObjectPlacement
Washer Standard	IfcMaterial, IfcClassificationReference
Nominal Washer Size	IfcProperty \ NominalValue
Washer Thickness	IfcProperty \ NominalValue
Location on bolt assembly	IfcProduct \ ObjectPlacement

Properties	**IFC Entities/Attributes**
Nut standard	IfcMaterial, IfcClassificationReference
Nominal Nut Size	IfcProperty \ NominalValue
Nut Type	IfcObject \ ObjectType
Location on bolt assembly	IfcProduct \ ObjectPlacement

4.6.3 Weld

Weld is categorized as welded joint system in this protocol. It can be categorized by the extent of its penetration as deep, full or partial penetration (or undefined). Weld can also be classified by its type as butt, fillet, spot or plug (or undefined).

In the AISC EM.11 Final Steel Detailing Model Documentation, four types of representation of a weld are introduced:

- A textual note or comment showing two components are joined by a welded connection.
- A dimensional illustration showing the length and extent of a weld.
- A physical representation of a weld using its theoretical size and shape.
- Additional information; e.g., start and end points, to indicate weld geometry.

It is recommended by AISC EM.11 that the third type of modeling is the best practice. And the CIS/2.1 data model also defines the geometry of a weld. This data protocol uses the combination of above two modeling methods. Therefore, weld is classified by its geometry as point weld, curve weld or surface weld.

4.6.3.1 Weld Mechanism

Weld mechanism have the following attributes:

- Weld type.
- Weld penetration Weld name.
- Weld design strength.

4.6.3.2 Curve Weld

A curve weld should have at least the following attributes for geometric description:

- Weld size.
- Weld section.
- Weld path.
- Location from critical point.
- Associated bridge component 1.
- Associated bridge component 2.

4.6.3.2.1 Mapping to OpenBrIM Schema

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="Weld">
<Param Name="ws" Label="Weld Size" Value="12.5"/>
<Param Name="stwted" Label="Stiffener Weld Transverse End
Distance" Value="6"/>
<Param Name="stwved" Label="Stiffener Weld Vertical End Distance"
Value="6"/>
<Repeat Param="var1" StartValue="0" EndValue="ns" Increment="1">
<Line Color="#CC3333">
<Point X="ss*var1" Y="girder.tw/2+w2+stwted" Z="-
(h1+h2+h3)"/>
<Point X="ss*var1" Y="girder.tw/2+w2+w1-stwted" Z="-
(h1+h2+h3)"/>
<Polygon>
<Point X="ws*cos(pi/2*0.0)" Y="ws*sin(pi/2*0.0)"/>
<Point X="ws*cos(pi/2*0.1)" Y="ws*sin(pi/2*0.1)"/>
<Point X="ws*cos(pi/2*0.2)" Y="ws*sin(pi/2*0.2)"/>
<Point X="ws*cos(pi/2*0.3)" Y="ws*sin(pi/2*0.3)"/>
<Point X="ws*cos(pi/2*0.4)" Y="ws*sin(pi/2*0.4)"/>
<Point X="ws*cos(pi/2*0.5)" Y="ws*sin(pi/2*0.5)"/>
<Point X="ws*cos(pi/2*0.6)" Y="ws*sin(pi/2*0.6)"/>
<Point X="ws*cos(pi/2*0.7)" Y="ws*sin(pi/2*0.7)"/>
<Point X="ws*cos(pi/2*0.8)" Y="ws*sin(pi/2*0.8)"/>
<Point X="ws*cos(pi/2*0.9)" Y="ws*sin(pi/2*0.9)"/>
<Point X="ws*cos(pi/2*1.0)" Y="ws*sin(pi/2*1.0)"/>
<Point X="0" Y="0"/>
</Polygon>
</Line>
</Repeat>
</Obj>
```

Figure 4-78 shows 3D view of the model of a stiffener with weld connection in the OpenBrIM Viewer.

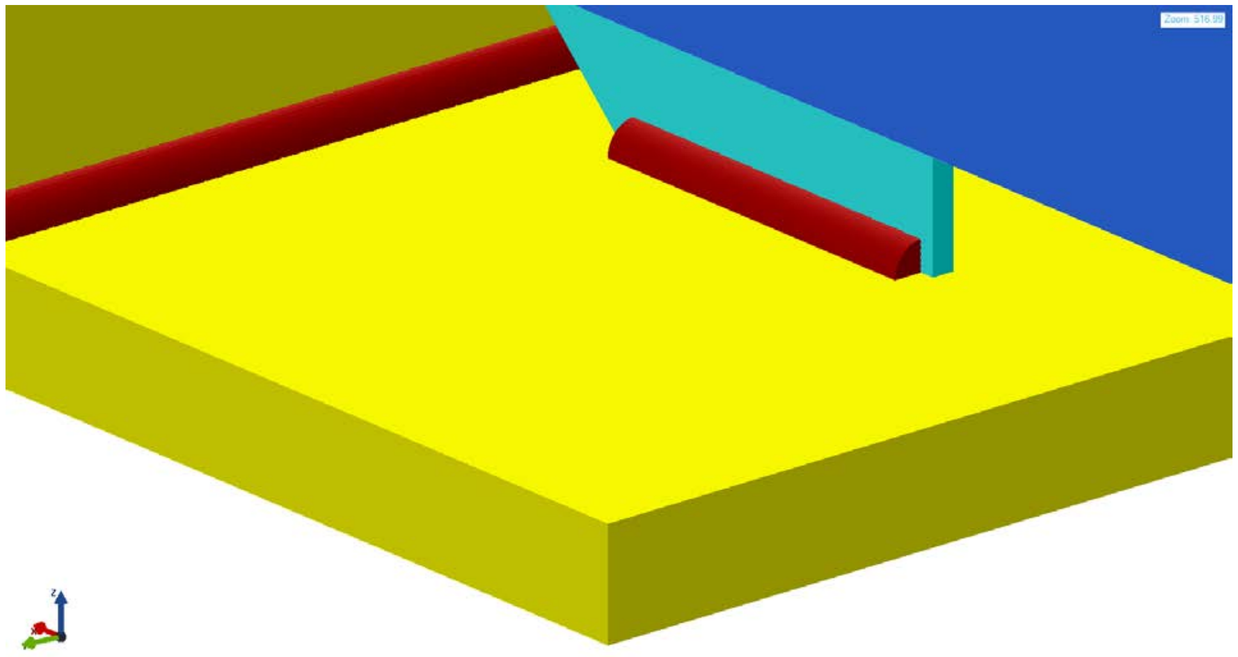


Figure 4-78. 3D View of Curve Weld

4.6.3.2.2 Mapping to IFC

Weld in IFC is categorized under IfcFastener. IfcFasteners are defined as Representations of fixing parts, which are used as fasteners to connect or join elements with other elements. The exact type information of the IfcFastener is given in the ObjectType attribute inherited from IfcObject. Standard type designations are provided for guideline below. Note that mechanical fasteners are represented by instances of the subtype IfcMechanicalFastener.

4.6.3.2.3 Mapping to ISM

1) Physical Geometry Properties of IIsmCurveMembers for Weld

Physical geometry properties of weld including Section, Location, PlacementPoint, Orientation, and MirrorShapeAboutYAxis will be demonstrated in this section.

2) Section of Weld

The cross section of a weld is treated as a quarter of a circle. Therefore, it can be defined by using IIsmCustomSection.

3) Location of Weld

Member line of a weld which is defined by using locations of two end points is located at the corner, as shown in Figure 4-79.

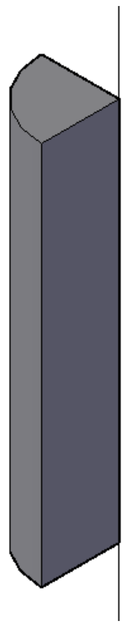


Figure 4-79. Member Line Location of Weld

4) Placement Point of Weld

Placement point of a weld is located at the corner, as shown in Figure 4-80.

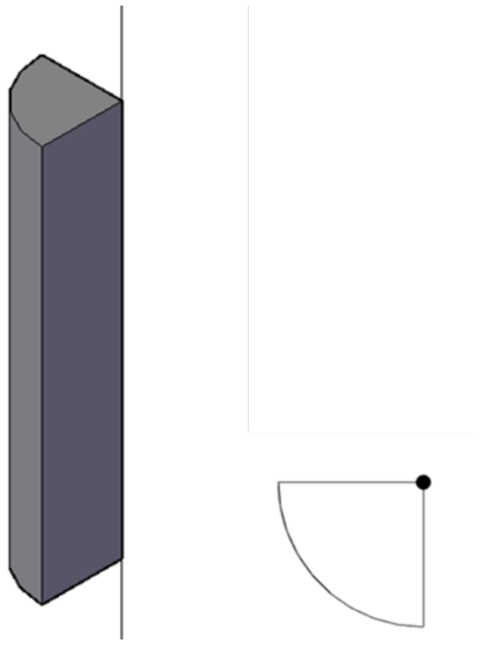


Figure 4-80. Placement Point of Weld (Isometric View and Section View)

5) Orientation of Weld

Orientation of cross sections of weld is perpendicular to the location.

6) MirrorShapeAboutYAxis of Weld

The Boolean value of this parameter is “False” for weld.

7) Summary of Modeling Weld

From the analysis above, it can be concluded that the weld can be modeled by using IIsmCurveMembers interface in the current ISM schema. Their section can be defined by using IIsmCustomSection interfaces.

4.6.4 Shear Stud

Shear stud is used to transfer shear force between a metal (usually steel) component and a concrete component in a composite structure. Shear stud is welded to the metal component, but its mechanical characteristic is similar to bolt assembly. Therefore, it is categorized in this protocol as compound joint system. Figure 4-81 shows the elevation view and details of shear stud.

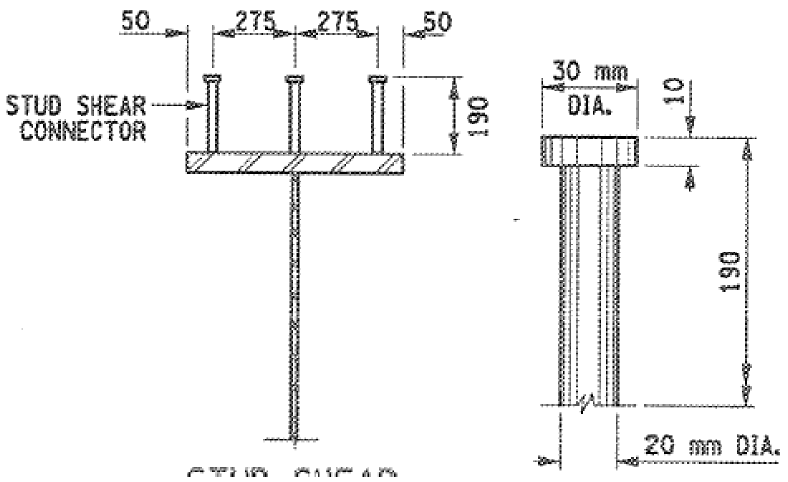


Figure 4-81. Elevation View and Details of Shear Stud

Shear stud should have at least the following attributes:

- Shear stud stem dimension/section.
- Shear stud cap dimension/section.
- Shear stud stem length.
- Shear stud cap thickness.
- Distance from critical point(s).
- Spacing between shear studs.
- Associate bridge component.

4.6.4.1 Mapping to OpenBrIM schema

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="ShearStud" Z="ttf">
<Param Name="sssr" Label="Shear Stud Stem Radius" Value="10"/>
```

```
<Param Name="sssl" Label="Shear Stud Stem Length" Value="180"/>
<Param Name="sscr" Label="Shear Stud Cap Radius" Value="15"/>
<Param Name="sscl" Label="Shear Stud Cap Length" Value="10"/>
<Param Name="led" Label="Longitudinal End Distance" Value="200"/>
<Param Name="ted" Label="Transverse End Distance" Value="50"/>
<Param Name="ls" Label="Longitudinal Spacing" Value="250"/>
<Param Name="ts" Label="TransverseSpacing" Value="275"/>
<Repeat Param="var1" StartValue="0" EndValue="15" Increment="1">
<Repeat Param="var2" StartValue="-1" EndValue="1" Increment="1">
<Line Color="#66CCFF">
<Point X="led+ls*var1" Y="ts*var2" Z="0"/>
<Point X="led+ls*var1" Y="ts*var2" Z="sssl"/>
<Circle X="0" Y="0" Radius="sssr" />
</Line>
<Line Color="#66CCFF">
<Point X="led+ls*var1" Y="ts*var2" Z="sssl"/>
<Point X="led+ls*var1" Y="ts*var2" Z="sssl+sscl"/>
<Circle X="0" Y="0" Radius="sscr" />
</Line>
</Repeat>
</Repeat>
</Obj>
```

Figure 4-82 and 4-83 show 3D view of shear stud in the OpenBrIM Viewer.

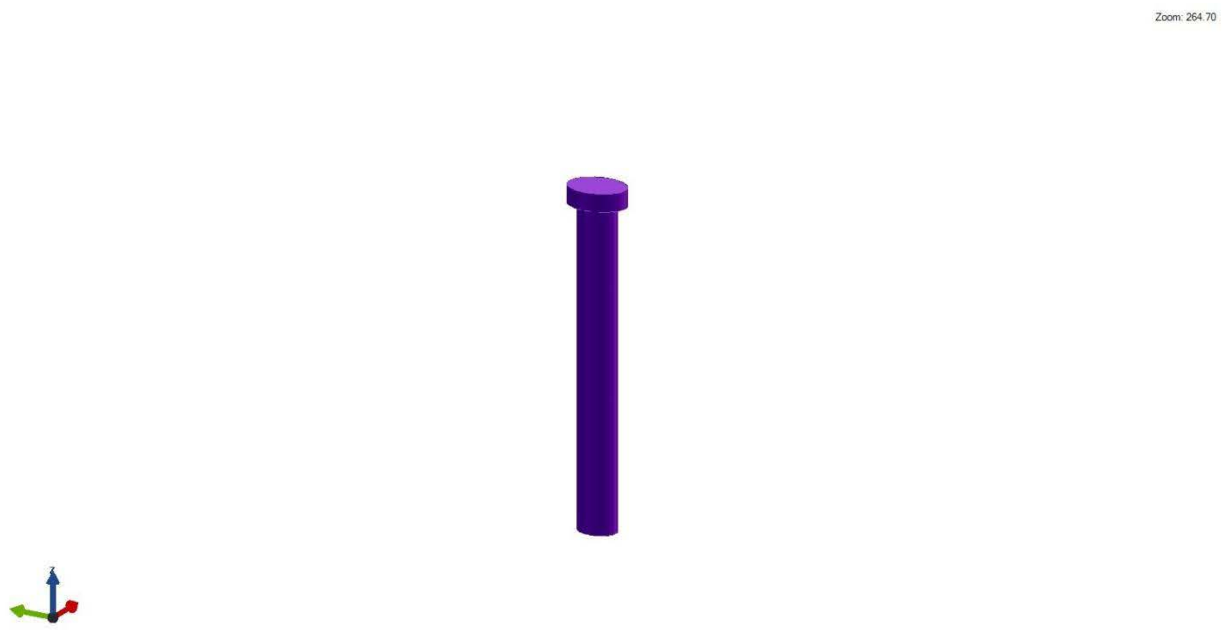


Figure 4-82. 3D View of a Shear Stud

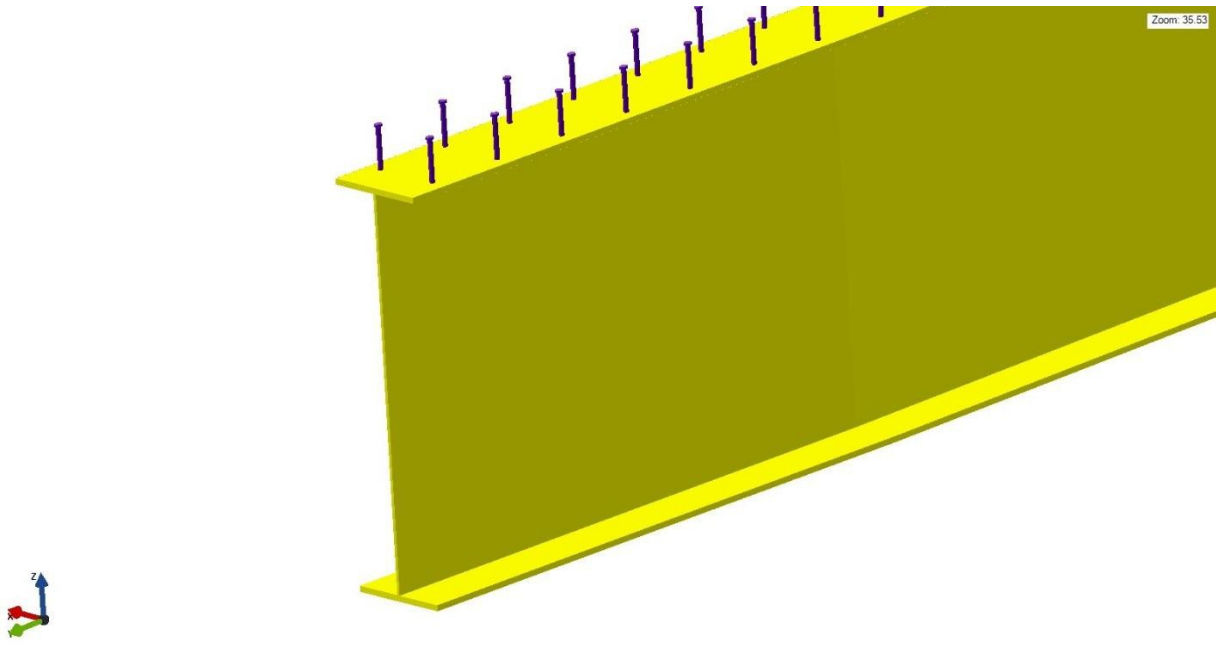


Figure 4-83. 3D View of Shear Studs on a Girder

4.6.4.2 Mapping to IFC

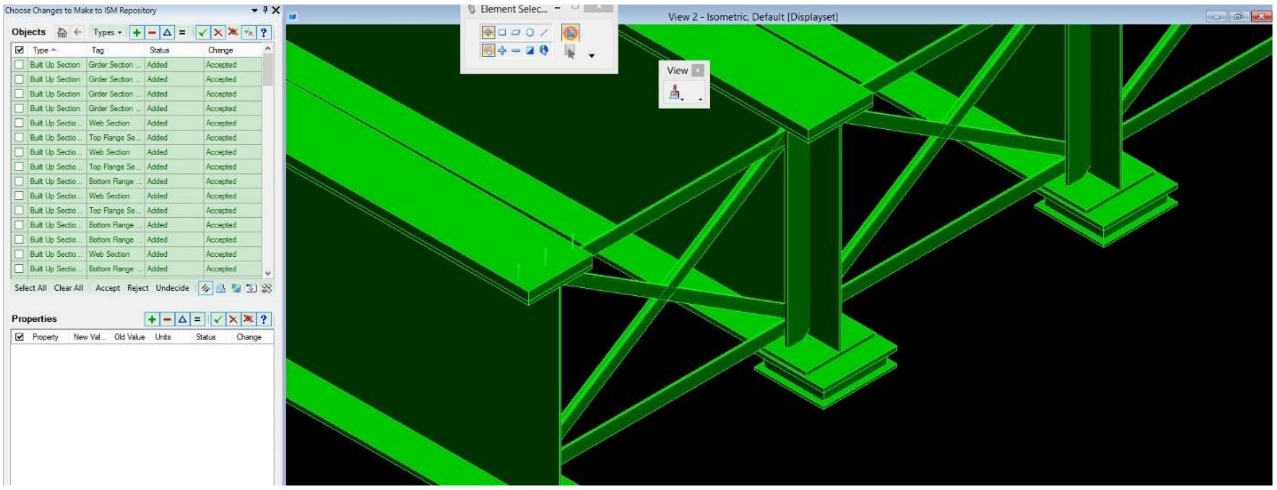
The Stud in IFC is categorized as an enumeration of IfcReinforcingBar and can be found under IfcReinforcingBarRoleEnum.

4.6.4.3 Mapping to ISM

Shear stud can be defined as accessories for a steel girder (IIsmCurveMember) by using the IIsmShearStudZone interface provided in ISM version 3. As the name indicated, shear stud is defined in a "zone" of the steel girder. IIsmShearStudZone defines the following geometric properties [4]

- StudCount: the number of studs in the zone.
- StudDiameter: the diameter of the stud shank.
- StudLength: the overall height of the shear stud.
- ZoneLength: the length over which the studs are distributed.

These properties are not enough for defining the shear studs shown in Figure 4-81. In order to model shear stud itself, diameter and height of the head of shear stud need to be specified. In order to define the location of shear studs, end distance and spacing in both longitudinal and transverse directions need to be specified. A workaround is to model shear studs as curve members. Figure 4-84 shows the ISM model of shear studs.


Choose Changes to Make to ISM Repository

Objects

☑	Type	Tag	Status	Change
☐	Built Up Section	Girder Section ...	Added	Accepted
☐	Built Up Section	Girder Section ...	Added	Accepted
☐	Built Up Section	Girder Section ...	Added	Accepted
☐	Built Up Section	Girder Section ...	Added	Accepted
☐	Built Up Sectio...	Web Section	Added	Accepted
☐	Built Up Sectio...	Top Flange Se...	Added	Accepted
☐	Built Up Sectio...	Web Section	Added	Accepted
☐	Built Up Sectio...	Top Flange Se...	Added	Accepted
☐	Built Up Sectio...	Bottom Flange ...	Added	Accepted
☐	Built Up Sectio...	Web Section	Added	Accepted
☐	Built Up Sectio...	Top Flange Se...	Added	Accepted
☐	Built Up Sectio...	Bottom Flange ...	Added	Accepted
☐	Built Up Sectio...	Bottom Flange ...	Added	Accepted
☐	Built Up Sectio...	Web Section	Added	Accepted
☐	Built Up Sectio...	Bottom Flange ...	Added	Accepted

Properties

☑	Property	New Val...	Old Value	Units	Status	Change

Figure 4-84. ISM Model of Shear Stud

4.7 Reinforcement / Prestressing Strands

Rebar bend objects can be defined as per BrIM standards parametrically based on CRSI standard bar bending chart. These standard rebar bends can be defined using object and section templates as extrusion solids driven by parameters defined in CRSI standard bar bending chart.

For example, Type-1 and Type-T1 rebar bends have been defined parametrically in template files that can be accessed by object instances in project file to create a model view of the rebar bend.

4.7.1 CRSI Type-1 Rebar Bend

As shown in Figure 4-85, CRSI Rebar Bend Type-1, following parameters are required to define type-1 rebar bend:

- A, Hook length.
- B, detailing dimension G, Hook length.
- J, Hook width.
- Dia, Rebar diameter.
- Material, Rebar material.

Figure 8-86 shows the block diagram depicting the interaction between object/section instances and their reference templates in the project template file. “Test_Rebar_Bend_1” (object instance) uses dimension parameters (A, B, G & J) and rebar diameter parameter (dia) to call its reference template, “crsi.Bar.1” and generate a model view of the CRSI type-1 rebar bend. Object template, “crsi.Bar.1” creates a solid extrusion model of this rebar bend based on the values of parameters in “Test_Rebar_Bend_1” (object instance). Rendering of this rebar bend in OpenBrIM viewer is shown in Figure 4-87. As shown in Schema Implementation-2, Sta, TOff and Elev key words can be used to place the rebar bend object. Placement reference point of the rebar bend defined in the template is shown in Figure 4-85.

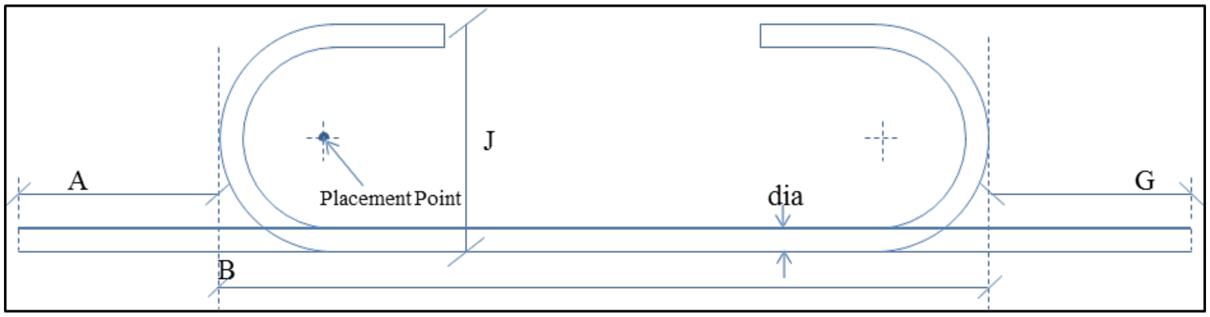


Figure 4-85. CRSI Rebar Bend Type–1

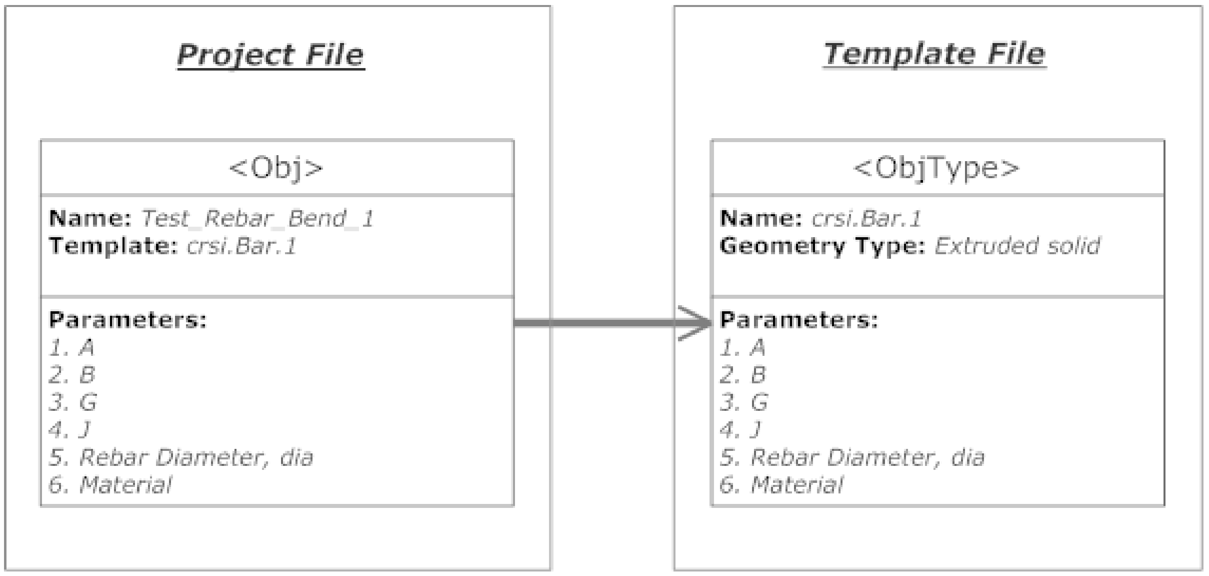


Figure 4-86. Block Diagram

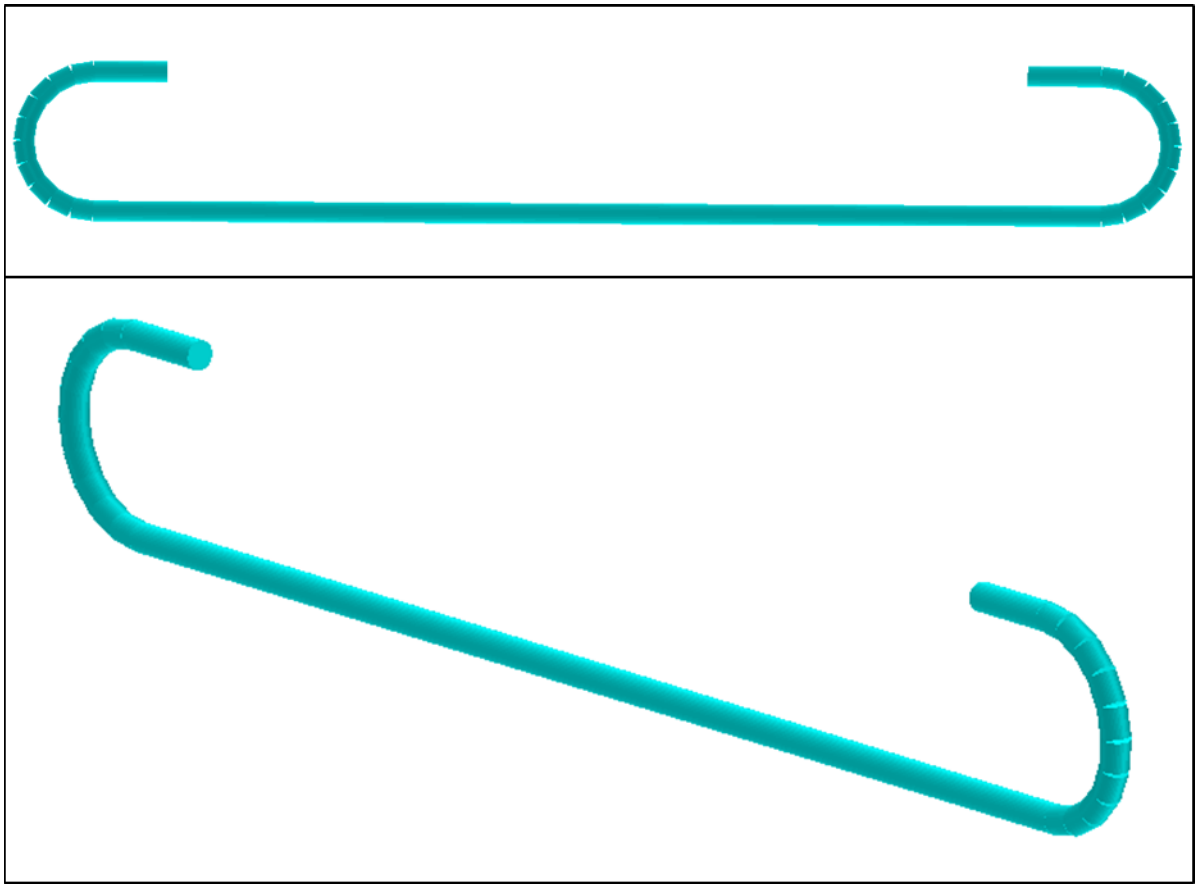


Figure 4-87. CRSI Type -1 Rebar Bend Rendering in OpenBrIM Viewer

Schema Implementation 1: Object Template for CRSI Type – 1 Rebar Bend

```
<ObjType Name="crsi.Bar.1" X="0" Y="-w/2+sideCover" Z="d/2-topCover">
//Define parameters using <Param>
<Param Name="A" Label="A" Value="75" Type="Length" />
<Param Name="B" Label="B" Value="434" Type="Length" />
<Param Name="G" Label="G" Value="75" Type="Length" />
<Param Name="J" Label="J" Value="(16.13+6.35*2)*2" Type="Length" />
<Param Name="dia" Label="dia" Value="6.35*2" Type="Length" />
//Define straight part using <Line>
<Line Color="cyan">
//Define orientation of the section using <Orientation>
<Orientation X="1" Y="0" Z="0"/>
//Define extrusion path using <Point>
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*1.5)-(G+J*0.5-3.142*0.5*(J-dia))" Z="(J
dia)*0.5 * sin(3.142*1.5)" />
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*1.5)" Z="(J-dia)*0.5 * sin(3.142*1.5)" />
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*1.4)" Z="(J-dia)*0.5 * sin(3.142*1.4)" />
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*1.3)" Z="(J-dia)*0.5 * sin(3.142*1.3)" />
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*1.2)" Z="(J-dia)*0.5 * sin(3.142*1.2)" />
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*1.1)" Z="(J-dia)*0.5 * sin(3.142*1.1)" />
```

```
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*1)" Z="(J-dia)*0.5 * sin(3.142*1)" />
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*0.9)" Z="(J-dia)*0.5 * sin(3.142*0.9)" />
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*0.8)" Z="(J-dia)*0.5 * sin(3.142*0.8)" />
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*0.7)" Z="(J-dia)*0.5 * sin(3.142*0.7)" />
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*0.6)" Z="(J-dia)*0.5 * sin(3.142*0.6)" />
<Point X="0" Y="(B-J)-(J-dia)*0.5 * cos(3.142*0.5)" Z="(J-dia)*0.5 * sin(3.142*0.5)" />
<Point X="0" Y="(B-J)" Z="(J-dia)*0.5" />
<Point X="0" Y="0" Z="(J-dia)*0.5" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*0.5)" Z="(J-dia)*0.5 * sin(3.142*0.5)" />
 <Point X="0" Y="(J-dia)*0.5 * cos(3.142*0.6)" Z="(J-dia)*0.5 * sin(3.142*0.6)" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*0.7)" Z="(J-dia)*0.5 * sin(3.142*0.7)" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*0.8)" Z="(J-dia)*0.5 * sin(3.142*0.8)" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*0.9)" Z="(J-dia)*0.5 * sin(3.142*0.9)" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*1)" Z="(J-dia)*0.5 * sin(3.142*1)" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*1.1)" Z="(J-dia)*0.5 * sin(3.142*1.1)" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*1.2)" Z="(J-dia)*0.5 * sin(3.142*1.2)" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*1.3)" Z="(J-dia)*0.5 * sin(3.142*1.3)" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*1.4)" Z="(J-dia)*0.5 * sin(3.142*1.4)" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*1.5)" Z="(J-dia)*0.5 * sin(3.142*1.5)" />
<Point X="0" Y="(J-dia)*0.5 * cos(3.142*1.5)+A+J/2-3.142*0.5*(J-dia)" Z="(J-dia)*0.5 *
sin(3.142*1.5)" />
<Circle X="0" Y="0" Radius="dia*0.5" Refinement="Low" />
</Line>
</ObjType>
```

Schema Implementation 2: Object Instance for CRSI Type – 1 Rebar Bend in the Project File

```
//Assign placement location and orientation using "Sta", "TOff", "Elev", "RX", "RY", "RZ"
<Obj Name=" BarTest1" RefObj=" crsi.Bar.1" Sta="0" TOff="0" Elev="0" RX="0" RY="0"
RZ="0">
<Parameter Name="rebarsec" Value="Rebar Section #13"/>
<Parameter Name="A" Value="70"/>
<Parameter Name="B" Value="434"/>
<Parameter Name="G" Value="70"/>
<Parameter Name="J" Value="(16.13+6.35*2)*2"/>
<Parameter Name="material" Value="Steel Grade 60"/>
</Obj>
```

4.7.2 CRSI Type – T1 Rebar Bend

As shown Figure 4-88, following parameters are required to define Type-T1 rebar bend.

- A, Hook length
- B, detailing dimension
- C, detailing dimension
- D, detailing dimension
- E, detailing dimension
- G, Hook length
- bendDia, bend diameter
- dia, Rebar diameter

Figure4-89 shows the block diagram depicting the interaction between object instances and their reference templates in the project template file. “Test_Rebar_Bend_T1” (object instance) uses dimension parameters (A, B, C, D, E, G & bendDia) and rebar diameter (dia) to call its reference template, “crsi.Bar.T1” and generate a model view of the CRSI type-1 rebar bend. Object template, “crsi.Bar.T1” creates a solid extrusion model of this rebar bend based on the values of parameters in “Test_Rebar_Bend_T1” (object instance). Rendering of this rebar bend in OpenBrIM viewer is shown in Figure 4-90**.** As shown in Schema Implementation 4, Sta, TOff and Elev key words can be used to place the rebar bend object. The placement reference point of the rebar bend defined in the template is shown in Figure 4-88**.**

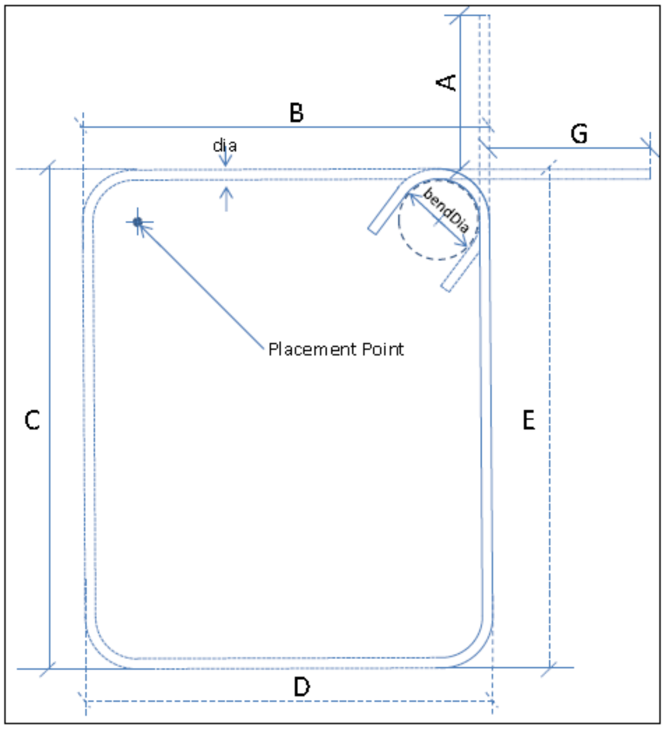
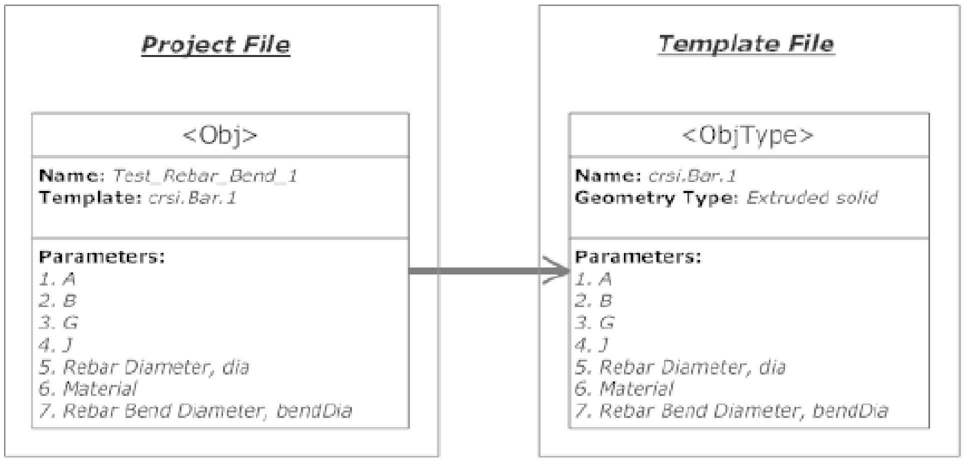


Figure 4-88. CRSI Rebar Bend Type = T1



<Obj>	<ObjType>
Name: *Test_Rebar_Bend_1*	**Name:** *crsi.Bar.1*
Template: *crsi.Bar.1*	**Geometry Type:** *Extruded solid*
Parameters:	**Parameters:**
1. A	*1. A*
2. B	*2. B*
3. G	*3. G*
4. J	*4. J*
5. Rebar Diameter, dia	*5. Rebar Diameter, dia*
6. Material	*6. Material*
7. Rebar Bend Diameter, bendDia	*7. Rebar Bend Diameter, bendDia*

Figure 4-89. Block Diagram

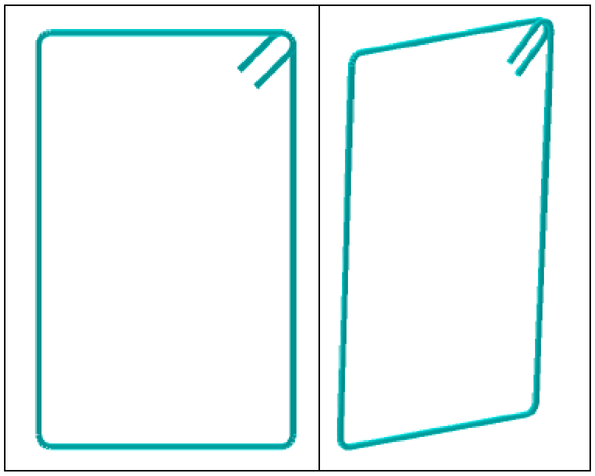


Figure 4-90. CRSI Type - T1 Rebar Bend Rendering in OpenBrIM Viewer

Schema Implementation 3: Object Template for CRSI Type – 1 Rebar Bend

```
<ObjType Name="crsi.Bar.T1">
 //Define parameters using <Parameter>
   <Param Name="bendDia" Label="Bend Diameter" Value="16.13+6.35" Type="Length" />
   <Param Name="A" Label="Hook-1 Length" Value="100" Type="Length" />
   <Param Name="B" Label="Width" Value="434" Type="Length" />
   <Param Name="C" Label="Depth" Value="500" Type="Length" />
<Param Name="D" Label="Width" Value="500" Type="Length" />
<Param Name="E" Label="Depth" Value="500" Type="Length" />
<Param Name="G" Label="Hook-2 length" Value="100" Type="Length" />
<Param Name="dia" Label="dia" Value="6.35*2" Type="Length" />
<Line Color="cyan">
         //Define extrusion path using <Point>
<Repeat Param="r" StartValue="0.5" EndValue="1.2" Increment="0.1">
<Point X="0" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.5)"
Z="(bendDia/2 +dia/2) * sin(3.142*0.5)" />
</Repeat>
<Point X="0" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*1.25)-(G+bendDia/2+dia-
0.75*3.142*bendDia/2)*cos(3.142*0.25)" Z="(bendDia/2 +dia/2) * sin(3.142*1.25)-(G+bendDia/2+dia-
0.75*3.142*bendDia/2)*cos(3.142*0.25)" />
<Point X="0" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*1.25)" Z="(bendDia/2 +dia/2) *
sin(3.142*1.25)" />
<Point X="0" Y="B-bendDia-dia*2" Z="bendDia/2+dia/2" />
<Point X="0" Y="0" Z="bendDia/2+dia/2" />
<Point X="0" Y="(bendDia/2 +dia/2)* cos(3.142*0.5)" Z="(bendDia/2 +dia/2)*sin(3.142*0.5)" />
<Point X="0" Y="(bendDia/2 +dia/2)* cos(3.142*0.6)" Z="(bendDia/2 +dia/2)*sin(3.142*0.6)" />
<Point X="0" Y="(bendDia/2 +dia/2)* cos(3.142*0.7)" Z="(bendDia/2 +dia/2)*sin(3.142*0.7)" />
<Point X="0" Y="(bendDia/2 +dia/2)* cos(3.142*0.8)" Z="(bendDia/2 +dia/2)*sin(3.142*0.8)" />
<Point X="0" Y="(bendDia/2 +dia/2)* cos(3.142*0.9)" Z="(bendDia/2 +dia/2)*sin(3.142*0.9)" />
<Point X="0" Y="(bendDia/2 +dia/2)* cos(3.142*1.0)" Z="(bendDia/2 +dia/2)*sin(3.142*1.0)" />
<Point X="0" Y="-bendDia/2-dia/2" Z="0" />
<Point X="0" Y="-bendDia/2-dia/2" Z="-(C-bendDia-dia*2)" />
```

```
<Point X="0" Y="(bendDia/2 +dia/2) * cos(3.142*1)" Z="-(C-bendDia-dia*2)-(bendDia/2
+dia/2) * sin(3.142*1)" />
<Point X="0" Y="(bendDia/2 +dia/2) * cos(3.142*0.9)" Z="-(C-bendDia-dia*2)-(bendDia/2
+dia/2) * sin(3.142*0.9)" />
<Point X="0" Y="(bendDia/2 +dia/2) * cos(3.142*0.8)" Z="-(C-bendDia-dia*2)-(bendDia/2
+dia/2) * sin(3.142*0.8)" />
<Point X="0" Y="(bendDia/2 +dia/2) * cos(3.142*0.7)" Z="-(C-bendDia-dia*2)-(bendDia/2
+dia/2) * sin(3.142*0.7)" />
<Point X="0" Y="(bendDia/2 +dia/2) * cos(3.142*0.6)" Z="-(C-bendDia-dia*2)-(bendDia/2
+dia/2) * sin(3.142*0.6)" />
<Point X="0" Y="(bendDia/2 +dia/2) * cos(3.142*0.5)" Z="-(C-bendDia-dia*2)-(bendDia/2
+dia/2) * sin(3.142*0.5)" />
<Point X="0" Y="0" Z="-(C-bendDia/2-dia*1.5)" />
                <Point X="0" Y="D-(bendDia +dia*2)" Z="-(C-bendDia/2-dia*1.5)" />
<Point X="0" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.5)" Z="-(E-bendDia-dia*2)-
(bendDia/2 +dia/2) * sin(3.142*0.5)" />
<Point X="0" Y="D-(bendDia +dia*2)-(bendDia/2 +dia2) * cos(3.142*0.6)" Z="-(E-bendDia-dia*2)-
(bendDia/2 +dia/2) * sin(3.142*0.6)" />
<Point X="0" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.7)" Z="-(E-bendDia-dia*2)-
(bendDia/2 +dia/2) * sin(3.142*0.7)" />
<Point X="0" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.8)" Z="-(E-bendDia-dia*2)-
(bendDia/2 +dia/2) * sin(3.142*0.8)" />
<Point X="0" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.9)" Z="-(E-bendDia-dia*2)-
(bendDia/2 +dia/2) * sin(3.142*0.9)" />
<Point X="0" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*1)" Z="-(E-bendDiadia*2)-
(bendDia/2 +dia/2) * sin(3.142*1)" />
<Point X="0" Y="D-(bendDia/2 +dia*1.5)" Z="-(E-bendDia-dia*2)" />
<Point X="dia" Y="D-(bendDia/2 +dia*1.5)" Z="0" />
<Point X="dia" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*1)" Z="(bendDia/2 +dia/2) *
sin(3.142*1)" />
<Point X="dia" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.9)" Z="(bendDia/2 +dia/2) *
sin(3.142*0.9)" />
<Point X="dia" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.8)" Z="(bendDia/2 +dia/2) *
sin(3.142*0.8)" />
<Point X="dia" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.7)" Z="(bendDia/2 +dia/2) *
sin(3.142*0.7)" />
<Point X="dia" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.6)" Z="(bendDia/2 +dia/2) *
sin(3.142*0.6)" />
<Point X="dia" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.5)" Z="(bendDia/2 +dia/2) *
sin(3.142*0.5)" />
<Point X="dia" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.4)" Z="(bendDia/2 +dia/2) *
sin(3.142*0.4)" />
<Point X="dia" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.3)" Z="(bendDia/2 +dia/2) *
sin(3.142*0.3)" />
<Point X="dia" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.25)" Z="(bendDia/2 +dia/2) *
sin(3.142*0.25)" />
<Point X="dia" Y="D-(bendDia +dia*2)-(bendDia/2 +dia/2) * cos(3.142*0.25)-(A+bendDia/2+dia-
0.75*3.142*bendDia/2)*cos(3.142*0.25)" Z="(bendDia/2 +dia/2) * sin(3.142*0.25)-(A+bendDia/2+dia-
0.75*3.142*bendDia/2)*cos(3.142*0.25)" />
<Circle X="0" Y="0" Radius="dia*0.5" Refinement="Low" />
```

```
</Line>
</ObjType>
```

Schema Implementation 4: Object Instance for CRSI Type – 1 Rebar Bend in the Project File

```
//Assign placement location and orientation using "Sta", "TOff", "Elev", "RX", "RY", "RZ"
<Obj Name=" BarTest1" RefObj=" crsi.Bar.1" Sta="0" TOff="0" Elev="0" RX="0" RY="0"
RZ="0">
<Param Name="bendDia" Value="16.13+6.35" />
<Param Name="A" Value="100" />
<Param Name="B" Value="434" />
<Param Name="C" Value="1000" />
<Param Name="D" Value="500" />
<Param Name="E" Value="1000" />
<Param Name="G" Value="100" />
<Param Name="dia" Value="6.35*2" />
<Parameter Name="material" Value="Steel Grade 60"/> </Obj>
```

4.7.3 Introduction to Reinforcement

In this section, the geometry properties of the reinforcement and prestressing steel in the case study bridges will be defined. Reinforcement used for bridge components can be categorized into four sets: longitudinal, transverse, surface and other.

4.7.4 Longitudinal Reinforcement

Longitudinal reinforcement exists in bridge structural components such as girder, concrete diaphragm, pier cap beam, pier column, drilled shaft, etc. Figure 4-91 shows the longitudinal reinforcement in the cap beam and the pier column in Quincy Avenue over I-25 and LRT, respectively.

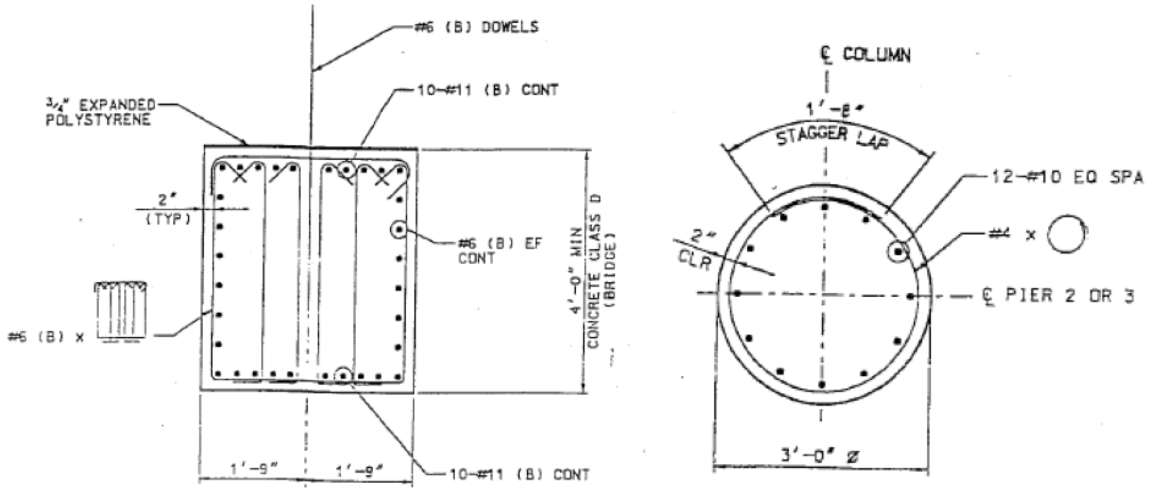


Figure 4-91. Longitudinal Reinforcement in Cap Beam and Column

The longitudinal rebars in the cap beam with a rectangular section can be modeled by using IIsmRectanguleParallelRebar interface. #6 dowels can be modeled by using a sub-interface of

IIsmParallelRebar interface. The longitudinal rebars in the pier columns with a circular section can be modeled by using IIsmCircleParallelRebar interface.

Figure 4-92 shows the longitudinal reinforcement in the diaphragm and the railing in Quincy Avenue over I-25 and LRT, respectively. These longitudinal rebars can be modeled by using IIsmGridParallelRebar or IIsmLayerParallelRebar interfaces.

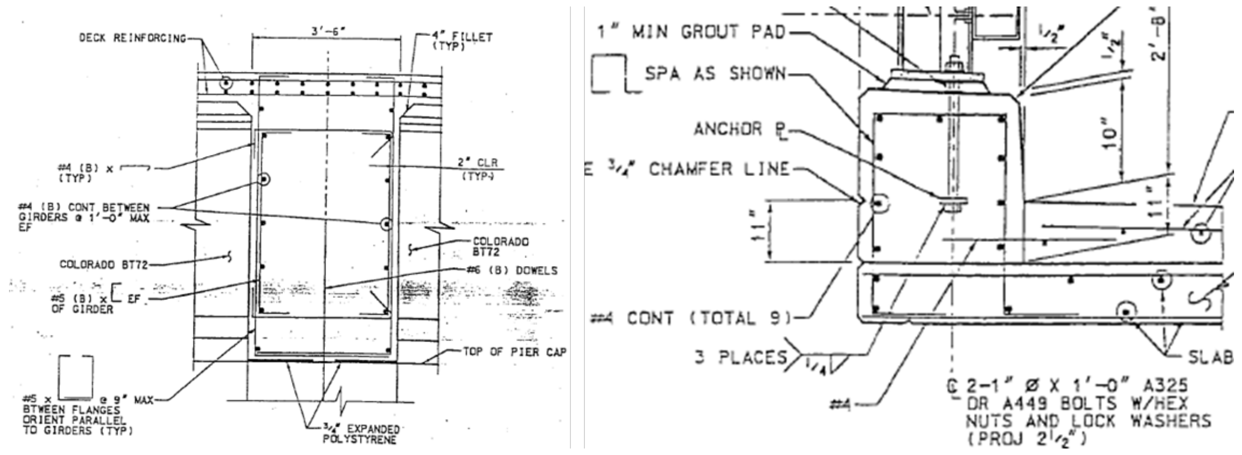


Figure 4-92. Longitudinal Reinforcement in Diaphragm and Railing

Figure 4-93 shows the longitudinal reinforcement in the parapet and sidewalk in Quincy Avenue over I-25 and LRT. The longitudinal rebars can be modeled by using IIsmCustomParallelRebar interface.

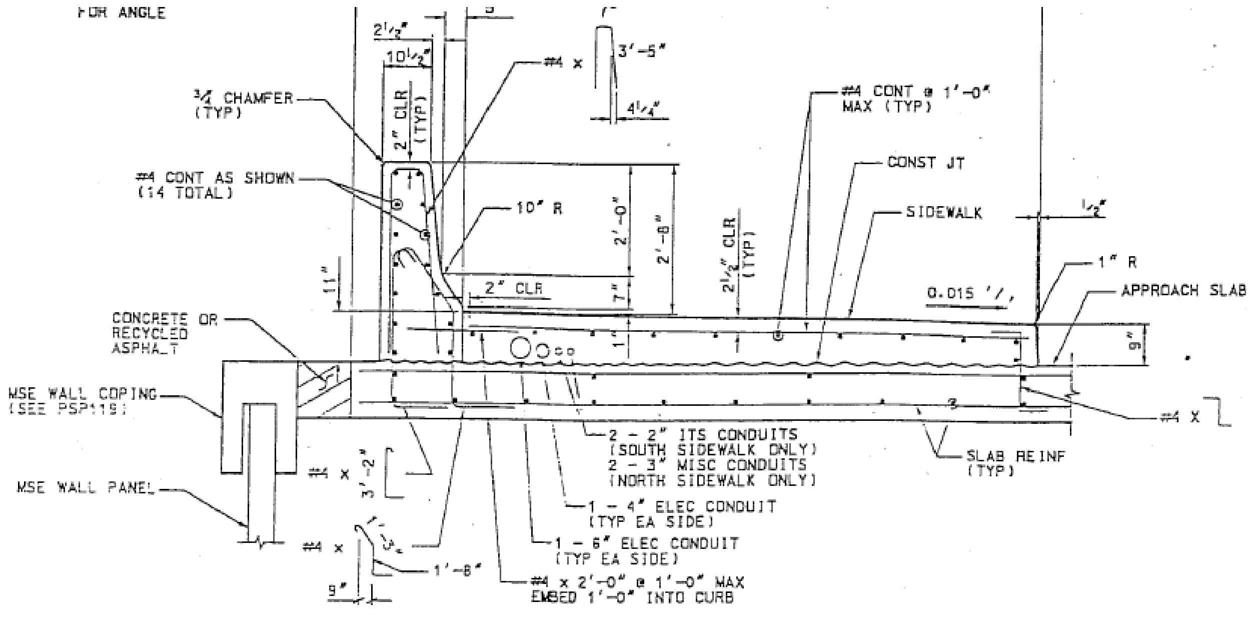


Figure 4-93. Longitudinal Reinforcement in Parapet and Sidewalk

Sometimes, the longitudinal reinforcement in vertical members such as column can be rebar bundle, as shown in Figure 4-94 This type of longitudinal reinforcement cannot be handled by using IIsmCircleParallelRebar interface.

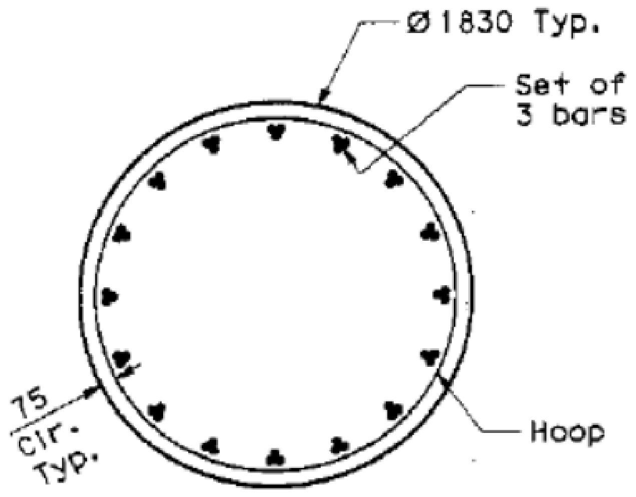


Figure 4-94. Longitudinal Reinforcement as Rebar Bundle

4.7.5 Transverse Reinforcement

Transverse reinforcement, which is used for confinement exists in nearly all the concrete bridge components. The transverse ties in the circular pier column can be modeled by using IIsmCircleTieRebar interface. The transverse reinforcement in the rectangular cap beam can be modeled by using IIsmRectangleTieRebar and IIsmStraightPerpendicularRebar interfaces, respectively. The shapes of these rebars are shown in Figure 4-95.

However, some transverse reinforcements do not have a regular shape (i.e., circular, rectangular, etc.), as shown in Figure 4-93 and Figure 4-95. These rebars can be modeled by using IIsmCustomPerpendicularRebar interface.

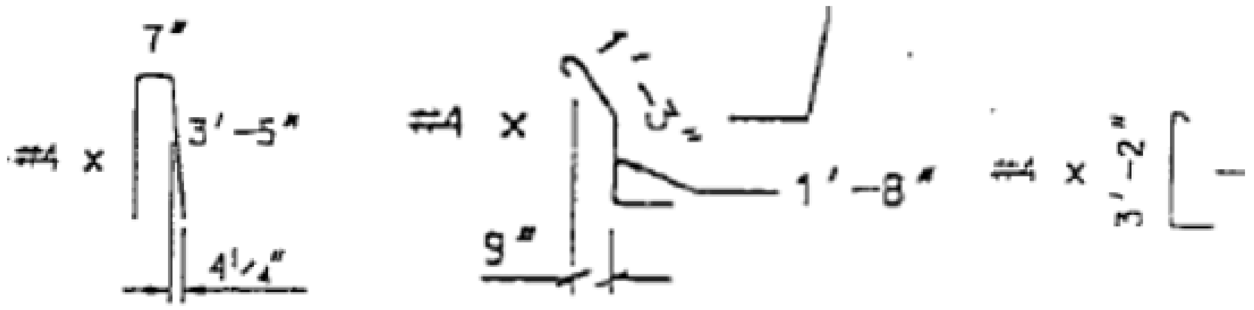


Figure 4-95. Transverse Reinforcement with Non-Regular Shapes

4.7.6 Surface Reinforcement

Figure 4-96 shows the reinforcement in deck slab. The reinforcement embedded in surface members such as deck slab, stem, backwall, wingwall, pier wall can be modeled by using IIsmAreaSurfaceRebar and/or IIsmConcentratedSurfaceRebar interfaces, because this interface defines a planar area filled with bars with a specified orientation and spacing [4].

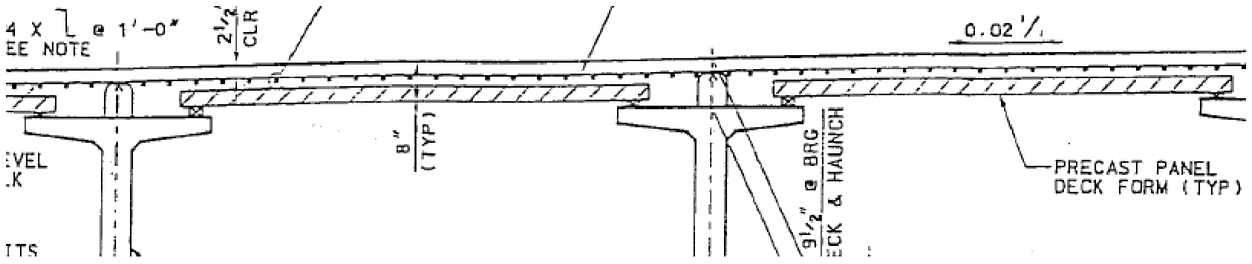


Figure 4-96. Reinforcement in Deck Slab

4.7.7 Other Reinforcement

Figure 4-97 shows a typical bar list from the contract plans for the case study bridge. Some of the rebars shown in this list cannot be modeled using IIsmCircleParallelRebar or IIsmRectangleParallelRebar interfaces. IIsmGridParallelRebar or IIsmCustomParallelRebar interface can be used to model rebar with special shapes.

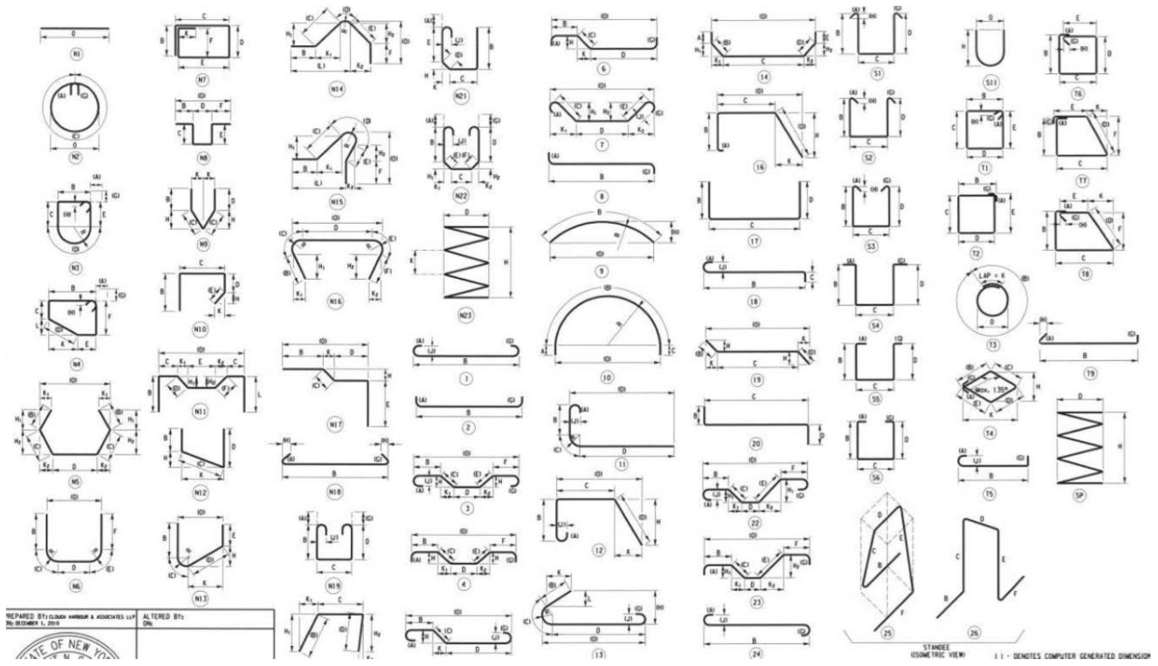


Figure 4-97. Bar List

4.7.8 Prestressing Steel

Figure 4-98 shows the prestressing steels embedded in Bulb Tee girder. Prestressing steels are commonly used in precast prestressed concrete members. However, the current release of ISM (version 3) does not include interfaces for modeling prestressing steel (either pre-tensioned or post-tensioned) [4].

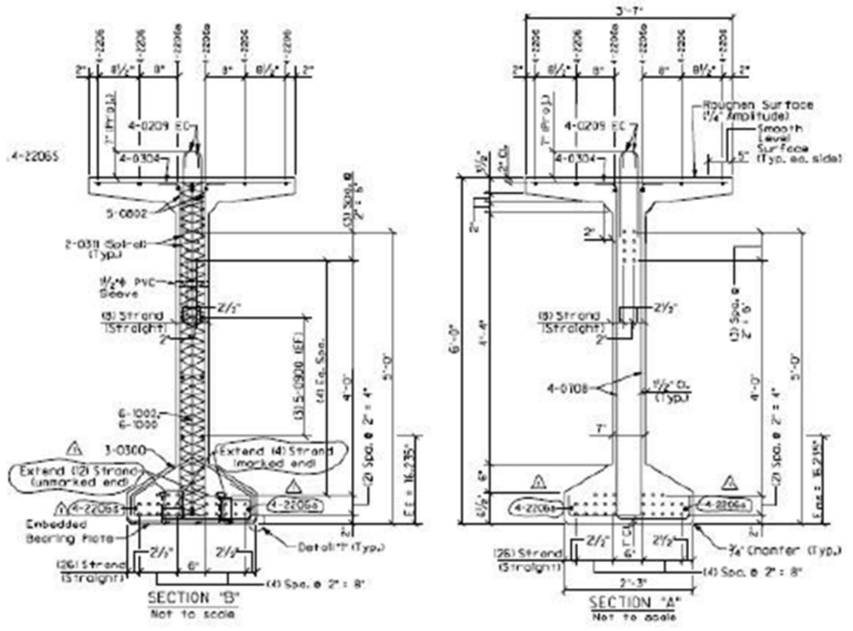


Figure 4-98. Prestressing Steel in Bulb Tee Girder

4.8 Materials

Material objects are used to specify material identities, properties, types, and applied conditions for bridge structural and non-structural members. This section covers the description and specification of common used materials. The exchange data protocol is flexible to be extended to include other materials and attributes.

4.8.1 Material Identification

The following attributes are used to define the identification of a material:

- Material designation
- Material name
- Material description

4.8.2 Material Properties

Standard properties for common used materials; e.g., steel, concrete, timber, etc., are shown as follows. The following attributes represent one of the mechanical characteristics for the material:

- Poisson’s ratio. This property provides the numerical value of the Poisson’s ratio of the associated material, which is the ratio of the lateral strain to the axial strain.

- Elastic modulus (Young’s modulus). This property provides the numerical value of the elastic modulus of the associated material, which is the slope of the stress–strain curve in the elastic deformation region of a material.

- Shear modulus. This property provides the numerical value of the shear modulus of the associated material, which is the ratio of shear stress to the shear strain. This property can be calculated by using the above two properties.

- Secant modulus. This property provides the numerical value of the secant modulus of the associated material, which reflects the slope of the line that connects the origin and a point of interest on the stress-strain curve, or the average ratio of stress to strain. This attribute is used for materials with variable elastic modulus, or for linear materials operating in the nonlinear region.

- Yield strength. This property provides the numerical value of the yield strength of the associated material, which is the stress that accompanies the beginning of plastic region.

- Ultimate tensile strength. This property provides the numerical value of the ultimate tensile strength of the associated material, which is the maximum stress the material can support before failure.

- Compressive strength. This property provides the numerical value of the compressive strength of the associated material, which is the maximum stress a material can sustain in compressive axial loading.

- Unit mass. This property provides the numerical value of the mass of the associated material per unit volume or area.

- Thermal expansion coefficient. This property provides the numerical value of the thermal expansion coefficient. It can be used in linear, area, and volumetric thermal expansion.

- Hardness. This property provides the numerical value of the hardness of the associated material. It shows the capacity of the material surface to resist deformation.

- Toughness. This property provides the numerical value of the toughness of the associated material. It is a measure of the material's ability to absorb energy, which is applied locally and rapidly, and deform plastically without fracturing.

4.8.3 Material Types

The materials described in this exchange data protocol can be categorized into three types. They are isotropic materials, anisotropic materials and orthotropic materials:

- Isotropic materials. Isotropic materials are the materials that have the same properties in all directions (e.g., steel), which are defined using one set of material properties in section 4.8.5.

- Anisotropic materials. Anisotropic materials are the materials that have different properties in different directions (e.g., timber), which are defined using two or more sets of material properties, and two or more associated orientations to represent directions.

- Orthotropic materials. Orthotropic materials are the materials that have different properties in different orthogonal directions, which are defined using two sets of material properties in two directions; i.e., in plane and out of plane.

4.8.4 Property Ranges (where the associated material properties are valid)

One set of material properties are valid for analysis and design in a certain property range. For example, in a bilinear material model, elastic modulus is only available within a particular strain range. Therefore, it is necessary to define property ranges:

- Dimensional range: lower value for dimension, upper value for dimension
- Strain range: lower value for strain, upper value for strain
- Stress range: lower value for stress, upper value for stress
- Temperature range: lower value for temperature, upper value for temperature

4.8.5 Mapping to Data Models

4.8.5.1 Mapping to OpenBrIM schema

- Isotropic materials

```
<Obj Name="Material 1" Label="Isotropic">
        <Param Name="MaterialDesignation" Value=""/>
        <Param Name="MaterialName" Value=""/>
        <Param Name="MaterialDescription" Value=""/>
        <Param Name="PoissonsRatio" Value="0.3"/>
        <Param Name="ElasticModulus" Value="200000"/>
        <Param Name="ShearModulus" Value="76923.1"/>
        <Param Name="secantModulus" Value="300000"/>
        <Param Name="YieldStrength" Value="250"/>
        <Param Name="UltimateTensileStrength" Value="400"/>
        <Param Name="CompressiveStrength" Value="10"/>
        <Param Name="UnitMass" Value="7860"/>
        <Param Name="ThermalExpansionCoefficient" Value="0.0000065"/>
        <Param Name="Hardness" Value="50"/>
        <Param Name="Toughness" Value="600"/>
        <Param Name="LowerValueOfDimension" Value="100"/>
        <Param Name="UpperValueOfDimension" Value="200"/>
        <Param Name="LowerValueOfStrain" Value="0.002"/>
        <Param Name="UpperValueOfStrain" Value="0.005"/>
        <Param Name="LowerValueOfStress" Value="100"/>
        <Param Name="UpperValueOfStress" Value="200"/>
        <Param Name="LowerValueOfTemperature" Value="10"/>
        <Param Name="UpperValueOfTemperature" Value="40"/>
</Obj>
```

- Orthotropic materials

```
<Obj Name="Material 2" Label="Orthotropic">
<Obj Name="InPlaneProperties">
        <Param Name="MaterialDesignation" Value=""/>
        <Param Name="MaterialName" Value=""/>
                <Param Name="MaterialDescription" Value=""/>
                <Param Name="PoissonsRatio" Value="0.3"/>
                <Param Name="ElasticModulus" Value="200000"/>
                <Param Name="ShearModulus" Value="76923.1"/>
                <Param Name="secantModulus" Value="300000"/>
                <Param Name="YieldStrength" Value="250"/>
                <Param Name="UltimateTensileStrength" Value="400"/>
                <Param Name="CompressiveStrength" Value="10"/>
                <Param Name="UnitMass" Value="7860"/>
                <Param Name="ThermalExpansionCoefficient"
Value="0.0000065"/>
                <Param Name="Hardness" Value="50"/>
                <Param Name="Toughness" Value="600"/>
        </Obj>
        <Obj Name="OutOfPlaneProperties">
                <Param Name="PoissonsRatio" Value="0.3"/>

        </Obj>
</Obj>
```

4.8.5.2 Mapping to ISM

Data Protocol	Mapping to ISM
Material designation	Not applicable
Material name	Not applicable
Material description	Not applicable
Poisson’s ratio	IIsmSteel::PoissonsRatio
Elastic modulus	IIsmSteel::ElasticModulus IIsmConcrete::ElasticModulusCalc IIsmConcrete::SpecifiedElastic Modulus
Shear modulus	Not applicable
Secant modulus	Not applicable
Yield strength	IIsmSteel::YieldStress
Ultimate tensile strength	IIsmSteel::TensileStrength IIsmConcrete::TensileStrengthCalc IIsmConcrete::SpecifiedTensileStrength
Compressive strength	IIsmConcrete::CompressiveStrength IIsmConcrete::CompressiveStrengthTest IIsmConcrete::FirstLoadCompressiveStrength
Unit mass	IIsmSteel::UnitMass IIsmSteel::UnitMassForLoads
Thermal expansion coefficient	IIsmSteel::ThermalExpansionCoefficient
Hardness	Not applicable
Toughness	Not applicable
Lower value for dimension	Not applicable
Upper value for dimension	Not applicable
Lower value for strain	Not applicable

Data Protocol	Mapping to ISM
Upper value for strain	Not applicable
Lower value for stress	Not applicable
Upper value for stress	Not applicable
Lower value for temperature	Not applicable
Upper value for temperature	Not applicable

4.8.5.3 Mapping to IFC

Both homogeneous, or inhomogeneous substance that can be used to form elements (physical products or their components) in IFC are defined under IfcMaterial.

4.9 Accessories

4.9.1 Bearing

4.9.1.1 Introduction to Bearing

This section defines the geometry properties of bearing. Except for the case study bridge with integral pier, other case study bridges have several bearings under girders at pier. There are more than one type of bearings in the case study bridges, such as elastomeric bearing and concave friction pendulum bearing. Figure 4-99 shows the elevation view of the bearings under the steel girders at pier.

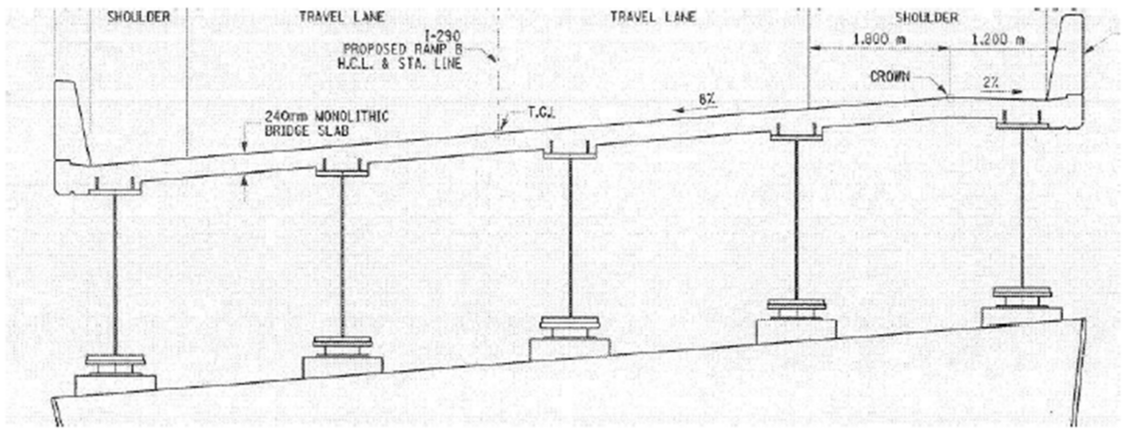


Figure 4-99. Bearings at Pier

Figure 4-100 shows the elevation view of an elastomeric bearing, while Figure 4-101 shows the plan view of the bearing.

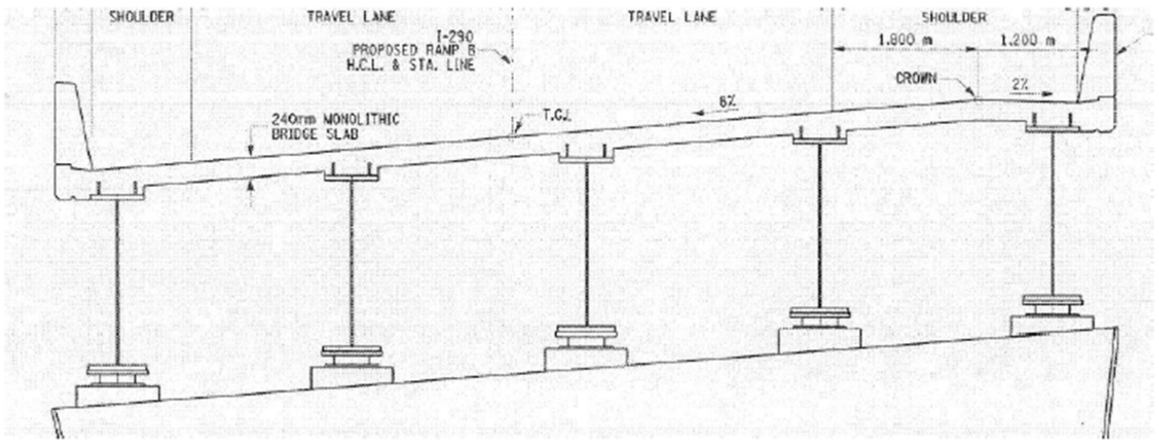


Figure 4-100. Elevation View of Elastomeric Bearing

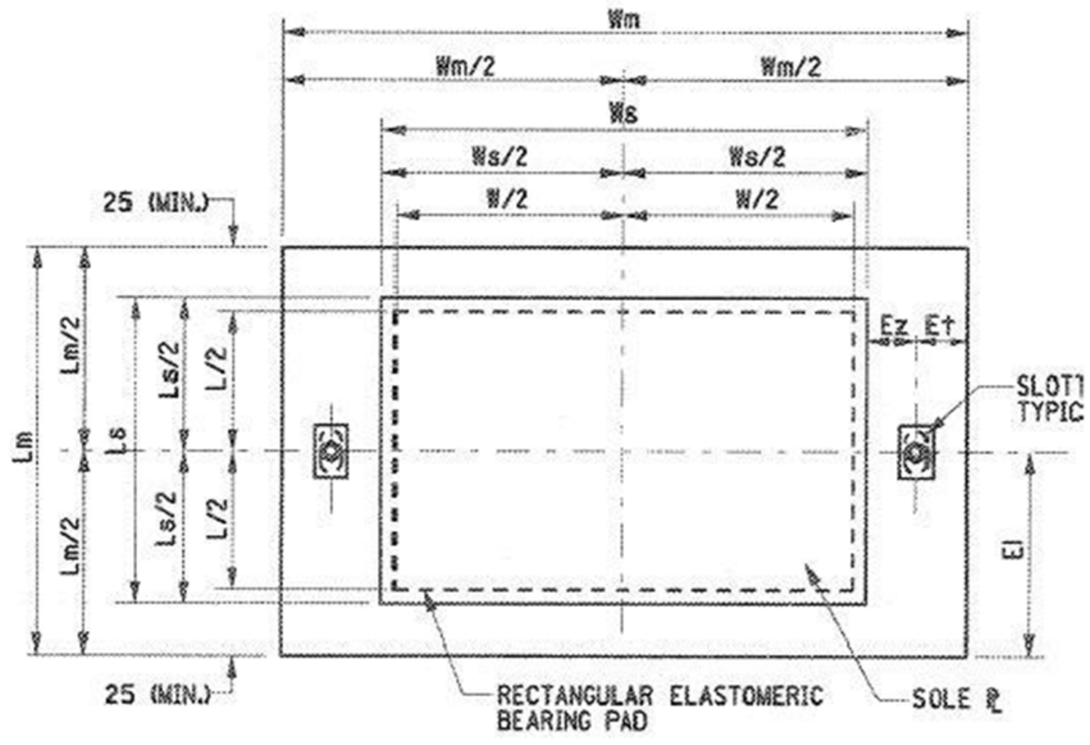


Figure 4-101. Plan View of Elastomeric Bearing

The elastomeric bearing can be defined as linear part. The concave friction pendulum bearing, needs to be defined by using revolve modeling method. Figure 4-102 shows a 3D view of an elastomeric bearing.

4.9.1.2 Mapping Bearing to Schemas

4.9.1.2.1 Mapping to OpenBrIM

```
<Obj Name="Bearing" X="Span1Length" Z="-wd-btf">
        <Param Name="tpd" Label="Top Plate Depth" Value="700"/>
        <Param Name="tpw" Label="Top Plate Width" Value="750"/>
        <Param Name="tpt" Label="Top Plate Thickness" Value="40"/>
        <Param Name="rbd" Label="Rubber Bearing Depth" Value="600"/>
        <Param Name="rbw" Label="Rubber Bearing Width" Value="650"/>
        <Param Name="rbt" Label="Rubber Bearing Thickness" Value="120"/>
        <Param Name="bpd" Label="Bottom Plate Depth" Value="700"/>
        <Param Name="bpw" Label="Bottom Plate Width" Value="750"/>
        <Param Name="bpt" Label="Bottom Plate Thickness" Value="25"/>
        <Param Name="ssd" Label="Steel Shim Depth" Value="595"/>
        <Param Name="ssw" Label="Steel Shim Width" Value="645"/>
        <Param Name="sst" Label="Steel Shim Thickness" Value="3"/>
        <Param Name="nss" Label="Number of Steel Shims" Value="8"/>
        <Line Color="Cyan">
                <Point X="0" Y="0" Z="0"/> <Point X="0" Y="0" Z="-tpt"/>
                <Polygon>
                        <Point X="-tpw/2" Y="tpd/2"/>
                        <Point X="-tpw/2" Y="-tpd/2"/>
                        <Point X="tpw/2" Y="-tpd/2"/>
                        <Point X="tpw/2" Y="tpd/2"/>
                </Polygon>
        </Line>
        <Repeat Param="var1" StartValue="1" EndValue="nss" Increment="1">
                <Line Color="Black">
                        <Point X="0" Y="0" Z="-tpt-(rbt-nss*sst)/(1+nss)*var1sst*(var1-1)"/>
                        <Point X="0" Y="0" Z="-tpt-(rbt-nss*sst)/(1+nss)*var1sst*var1"/>
                        <Polygon>
                                <Point X="-ssw/2" Y="ssd/2"/>
                                <Point X="-ssw/2" Y="-ssd/2"/>
                                <Point X="ssw/2" Y="-ssd/2"/>
                                <Point X="ssw/2" Y="ssd/2"/>
                        </Polygon>
                </Line>
        </Repeat>
        <Line Color="#ffa52c">
                <Point X="0" Y="0" Z="-tpt"/>
                <Point X="0" Y="0" Z="-tpt-rbt"/>
                <Polygon>
                        <Point X="-rbw/2" Y="rbd/2"/>
                        <Point X="-rbw/2" Y="-rbd/2"/>
                        <Point X="rbw/2" Y="-rbd/2"/>
                        <Point X="rbw/2" Y="rbd/2"/>
                </Polygon>
        </Line>
        <Line Color="Cyan">
                <Point X="0" Y="0" Z="-tpt-rbt"/>
                <Point X="0" Y="0" Z="-tpt-rbt-bpt"/>
                <Polygon>
                        <Point X="-bpw/2" Y="bpd/2"/>
                        <Point X="-bpw/2" Y="-bpd/2"/>
                        <Point X="bpw/2" Y="-bpd/2"/>
                        <Point X="bpw/2" Y="bpd/2"/>
                </Polygon>
        </Line>
</Obj>
```

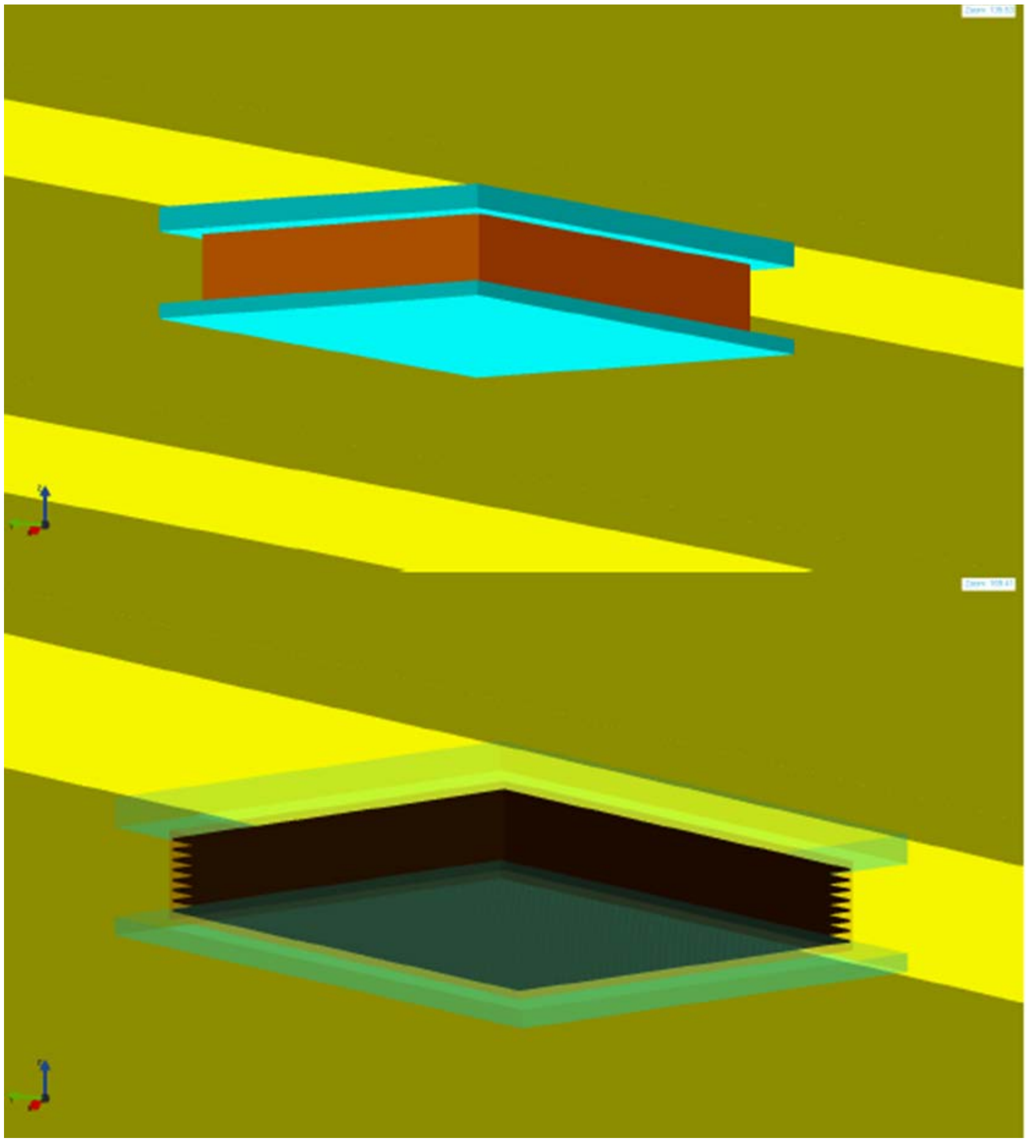


Figure 4-102. 3D View of a Elastomeric Bearing

4.9.1.2.2 Mapping to ISM

1) Physical Geometry Properties of IIsmCurveMembers for Bearing

Physical geometry properties of elastomeric bearing including Section, Location, PlacementPoint, Orientation, and MirrorShapeAboutYAxis will be demonstrated in this section.

2) Section of Bearing

Elastomeric bearings consist of steel segments and rubber segments. Because one linear member can only have one material type, the bearings should be modeled as multiple parts. Each part has a constant rectangular cross section. Therefore, IIsmParametricSection interface can be used.

3) Location of Bearing

The member line of an elastomeric bearing defined by the location of two endpoints is located in the center, as shown in Figure 4-103.

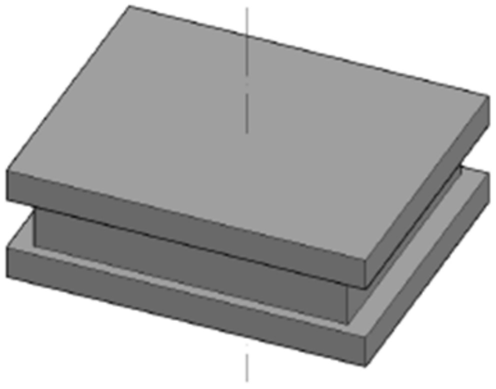


Figure 4-103. Member Line Location of Bearing

4) Placement Point of Bearing

Placement point of elastomeric bearing is located at the center, as shown in Figure 4-104.

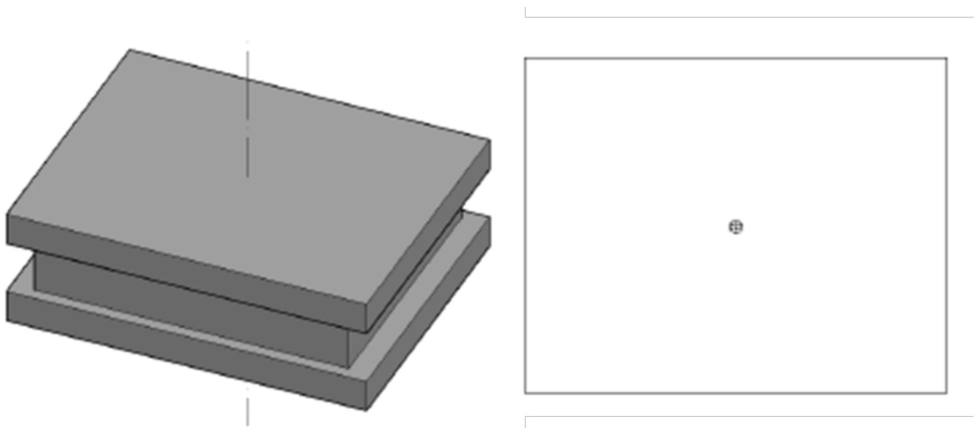


Figure 4-104. Placement Point of Bearing (Isometric View and Section View)

5) Orientation of Bearing

Orientation of cross sections of bearing is perpendicular to the location, as shown in Figure 4-105.

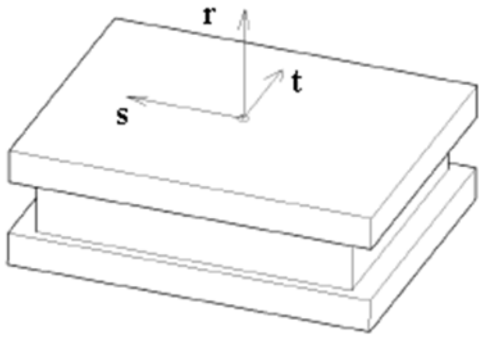


Figure 4-105. Orientation of Bearing

6) MirrorShapeAboutYAxis for Bearing

The Boolean value of this parameter is “True.” The section of elastomeric bearing and is symmetric about Y axis.

7) Summary of Modeling Bearing

From the analysis above, it can be concluded that the elastomeric bearing may be modeled by using IIsmCurveMembers interface in the current ISM schema. Their section may be defined by using IIsmParametricSection interface. Figure 4-106 shows the ISM model of the elastomeric bearing.

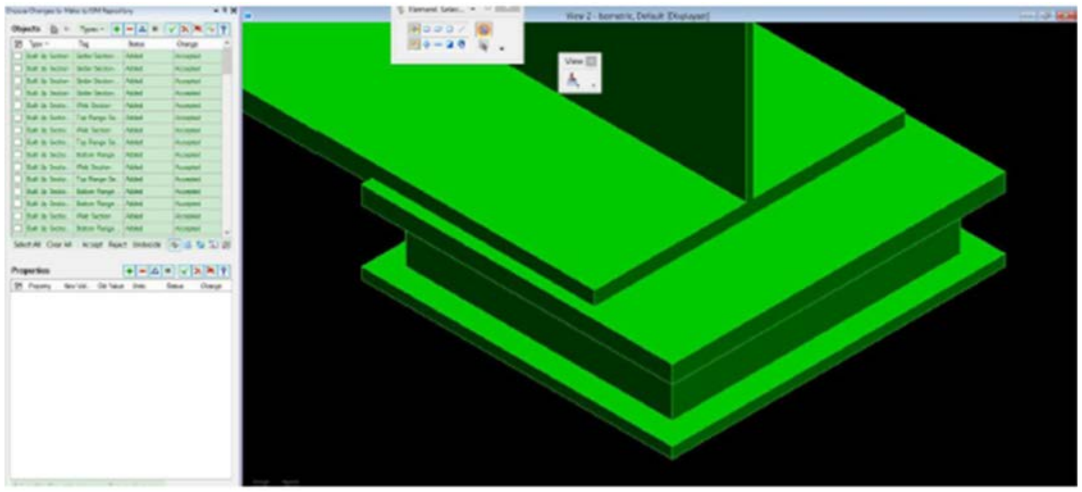


Figure 4-106. ISM Model of Elastomeric Bearing

Although the current ISM is able to model elastomeric bearing, it is not capable of defining a concave friction pendulum bearing, as shown in Figure 4-107. The current ISM can only model two sets of members; i.e., linear member and surface member [4]. To model concave friction pendulum bearing, revolve modeling function is required.

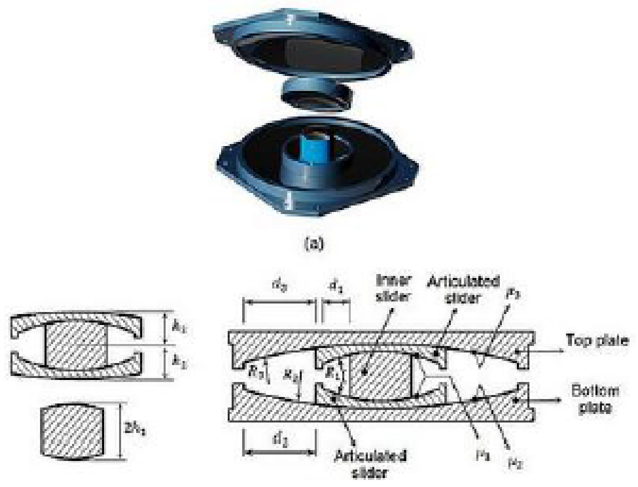


Figure 4-107. Concave Friction Pendulum Bearing

4.9.2 Parapet and Sidewalk

4.9.2.1 Introduction to Parapet and Sidewalk

The case study bridge, Quincy Avenue over I-25 and LRT has two concrete parapets on two sides, as shown in Figure 4-108. Section view of the parapet from the contract plans is shown in Figure 4-109. The parapets can be defined as linear members by using IIsmCurveMembers interface.

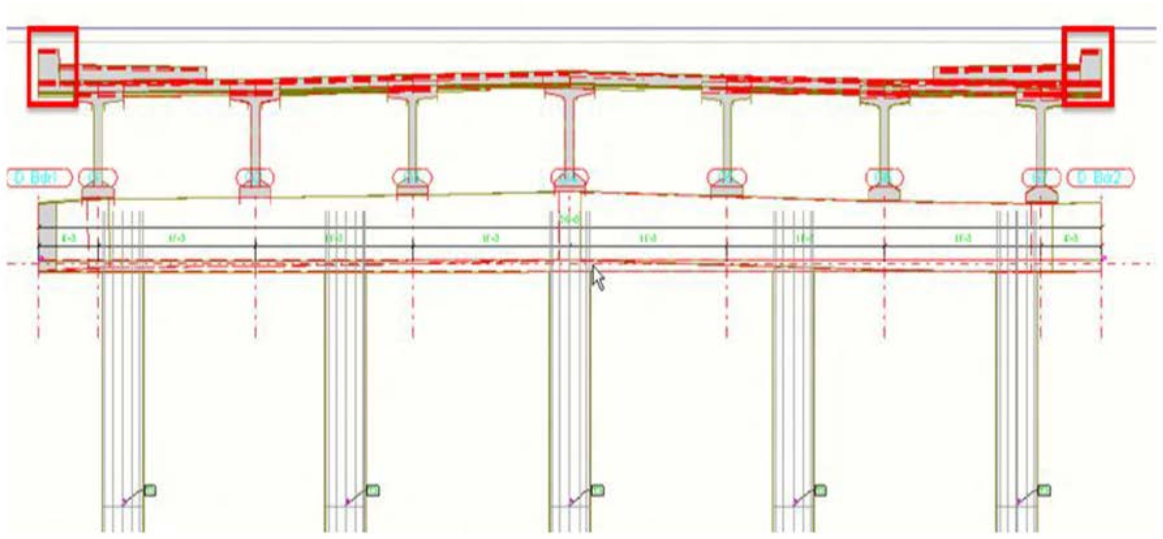


Figure 4-108. Concrete Parapet

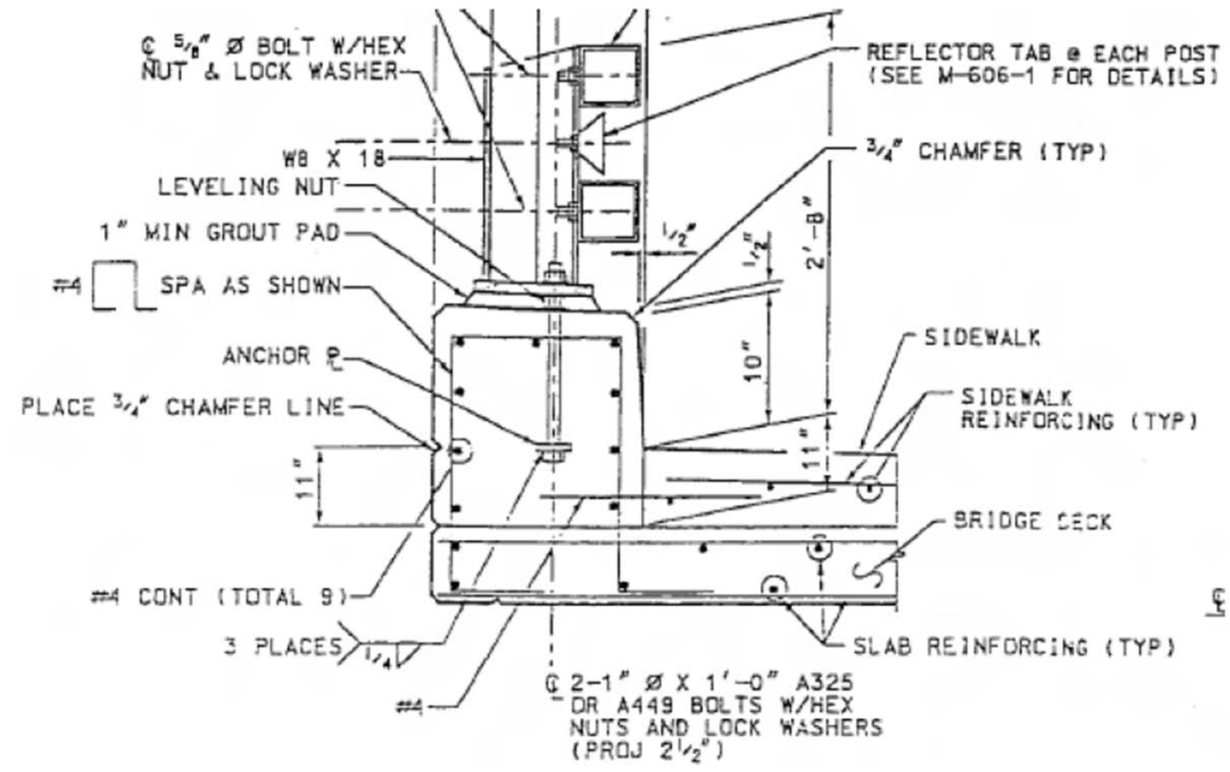


Figure 4-109. Typical Parapet Section from the Contract Plans

4.9.2.2 Mapping Parapet and Sidewalk to Schemas

Mapping to OpenBrIM Modeling parapet and sidewalk is the same as modeling linear part. Mapping to ISM Physical geometry properties of the concrete parapet and sidewalk including Section, Location,

PlacementPoint, Orientation, and MirrorShapeAboutYAxis will be demonstrated in this section.

1) Section of Parapet and Sidewalk

Because parapet and sidewalk is linear member with constant section, and their sections are not defined in any standard, IIsmParametricSection or IIsmCustomSection can be used for the sections. To simplify the bridge model, the chamfer is ignored.

The parameters of parapet proposed for ISM schema are listed below. The parameters have their meanings defined in Figure 4-110.

- Vertical: H1, H2, H3
- Horizontal: ParapetWidth, W1
- Other: Use, SystemKind, SectionDescription

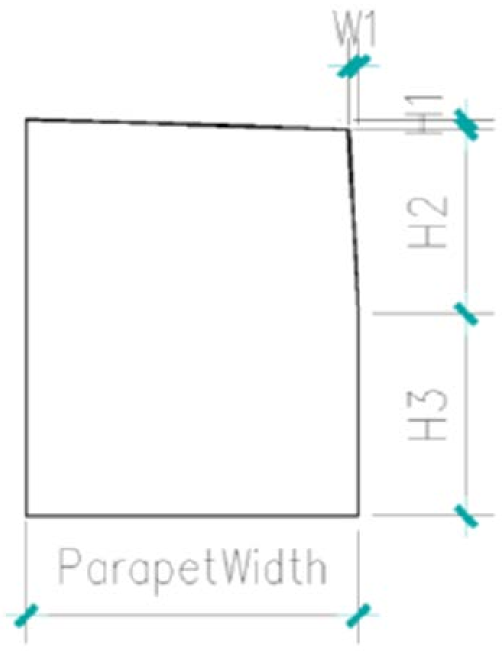


Figure 4-110. Definition of Parapet for ISM

The parameters proposed for ISM schema are listed below. The parameters have their meanings defined in Figure 4-111.

- Vertical: H1, H2
- Horizontal: SidewalkWidth
- Other: Use, SystemKind, SectionDescription

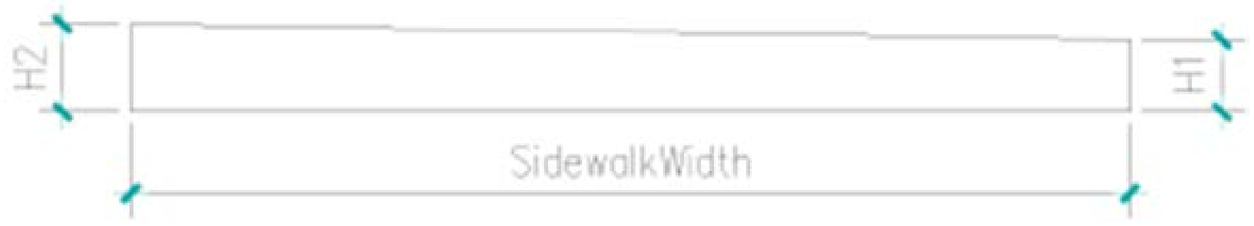


Figure 4-111. Definition of Sidewalk for ISM

2) Location of Parapet and Sidewalk

Member line of parapet and sidewalk which is defined by using locations of two end points is located at the bottom left (or the bottom right) of parapet and sidewalk, as shown in Figure 4-112 and Figure 4-113, respectively.

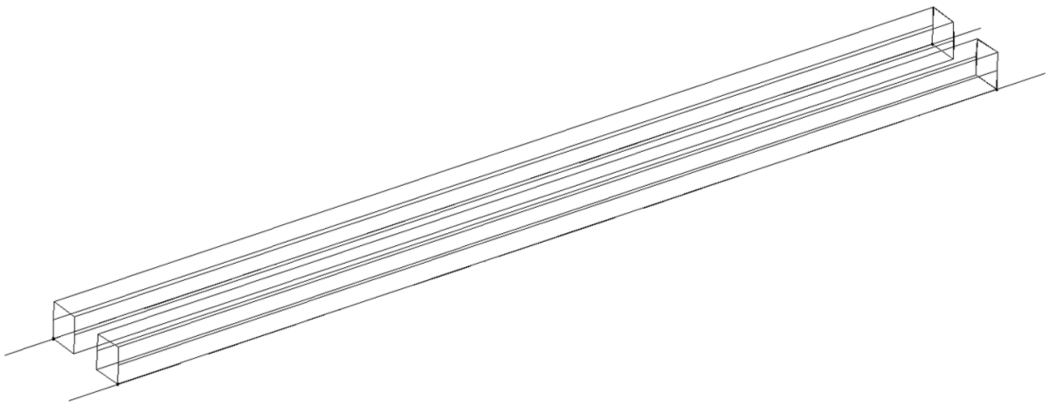


Figure 4-112. Parapet Member Line Location

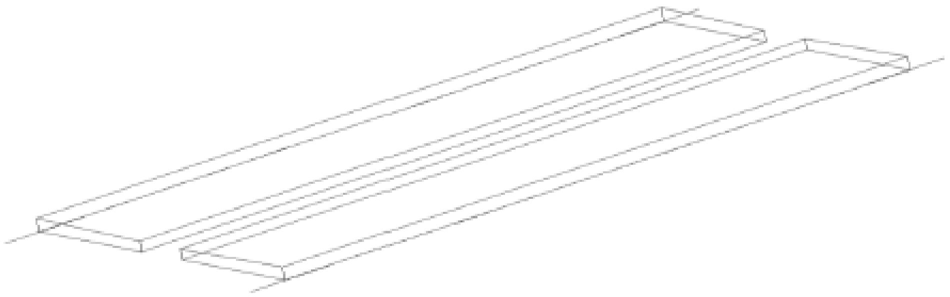


Figure 4-113. Sidewalk Member Line Location

3) Placement Point of Parapet and Sidewalk

Placement point of the parapet is located at the bottom left (or the bottom right) of the parapets, as shown in Figure 4-114 and Figure 4-115.

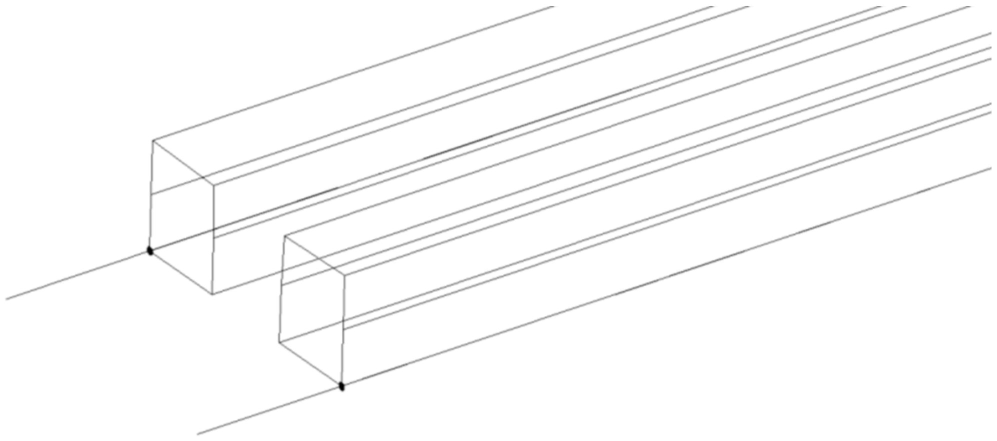


Figure 4-114. Parapet Placement Point (Isometric View)

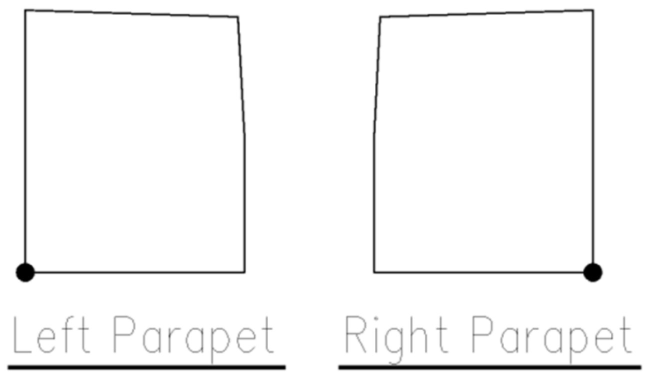


Figure 4-115. Parapet Placement Point (Section View)

Placement point of the sidewalk is located at the bottom left (or the bottom right) of the sidewalks, as shown in Figure 4-116 and Figure 4-117.

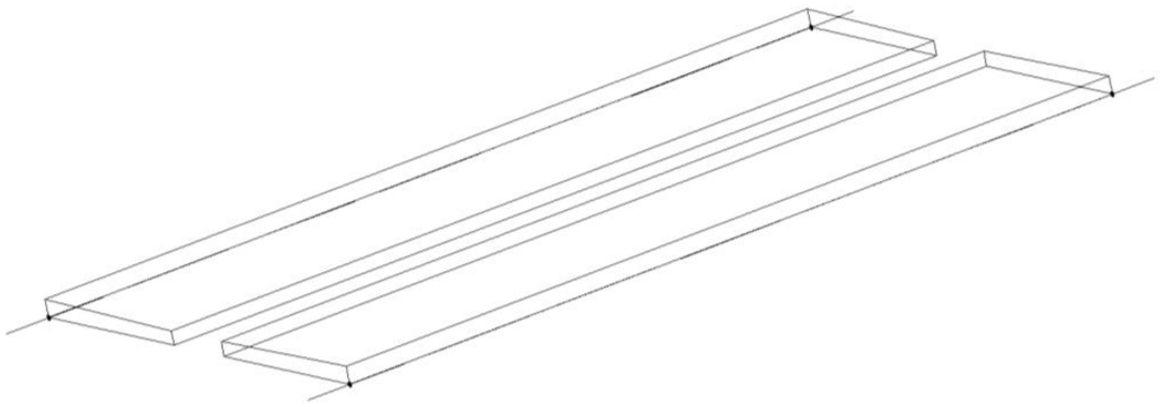


Figure 4-116. Sidewalk Placement Point (Isometric View)

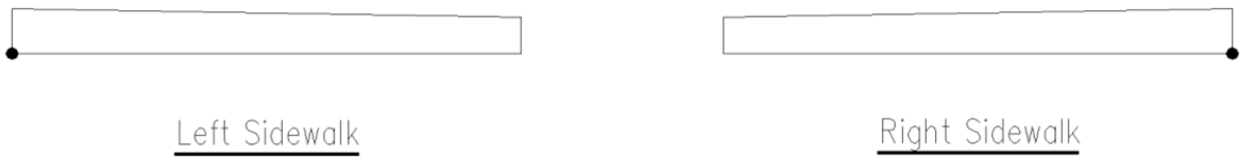


Figure 4-117. Sidewalk Placement Point (Section View)

4) Orientation of Parapet and Sidewalk

Orientation of the parapet cross section is perpendicular to the parapet location, as shown in Figure 4-118.

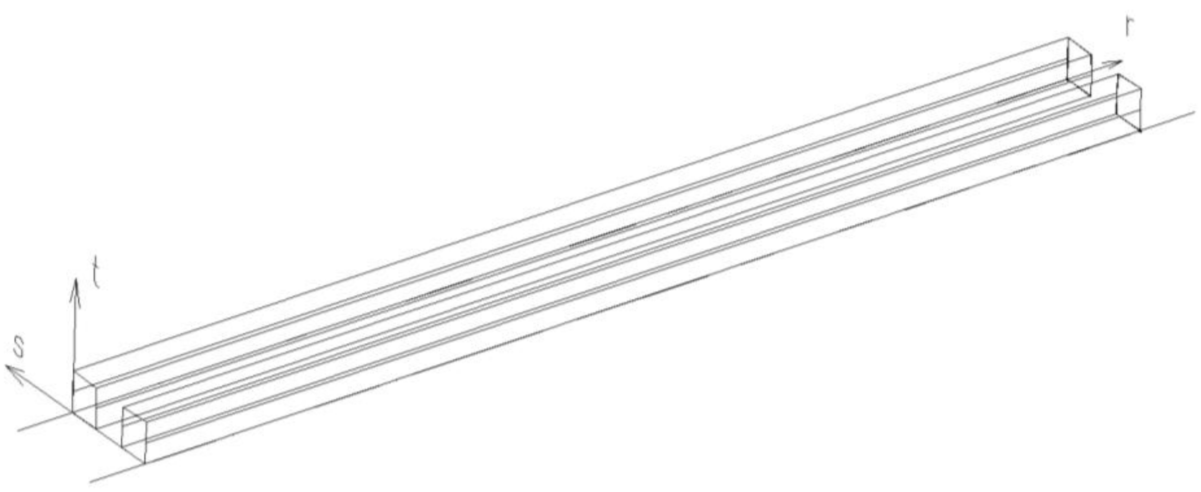


Figure 4-118. Parapet Orientation

Orientation of the sidewalk cross section is perpendicular to the sidewalk location, as shown in Figure 4-119.

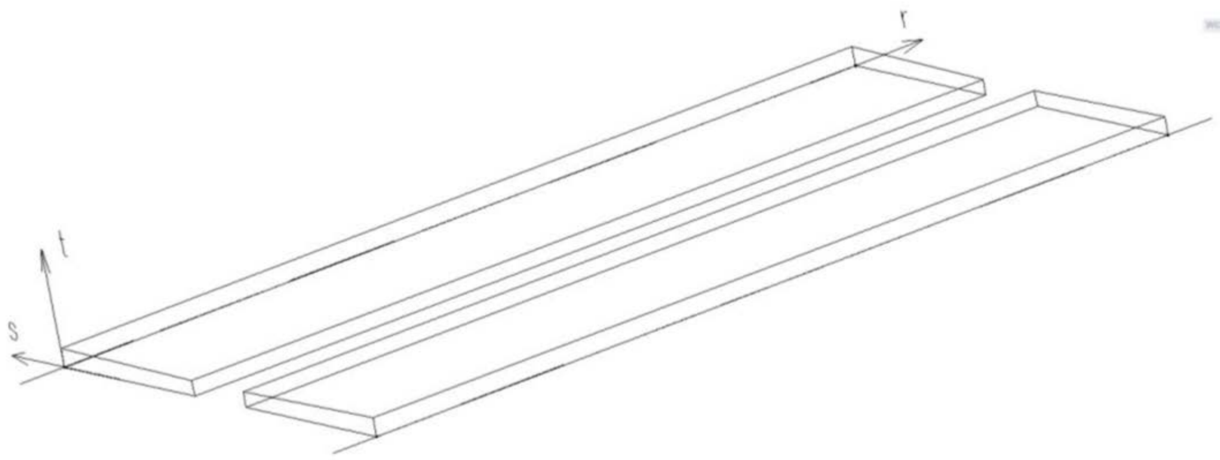


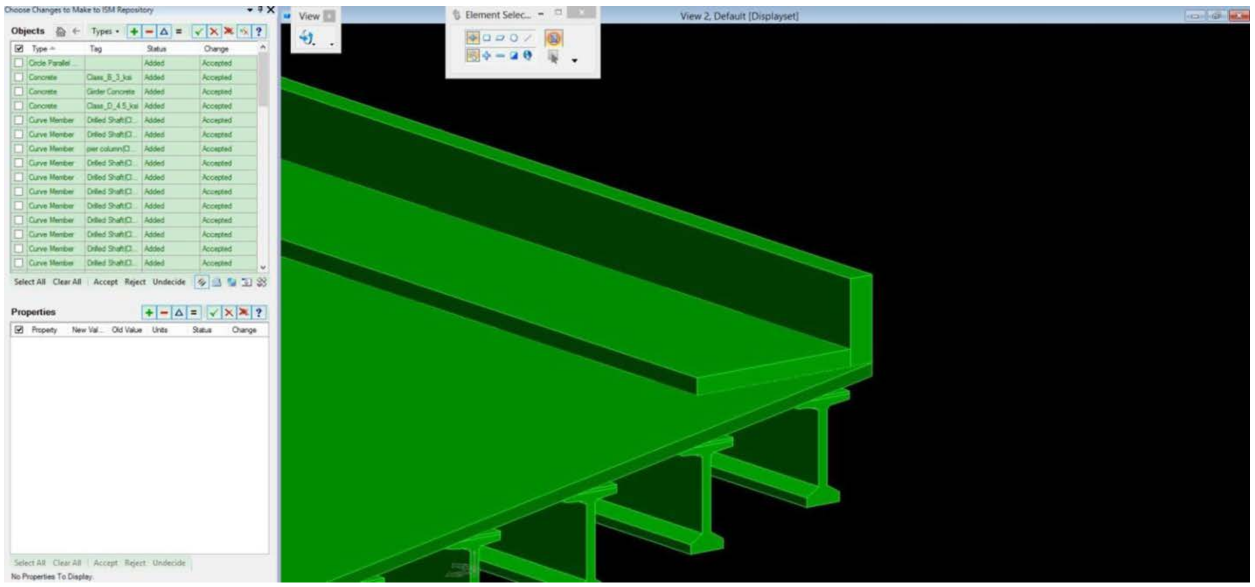
Figure 4-119. Sidewalk Orientation

5) MirrorShapeAboutYAxis of Parapet and Sidewalk

The Boolean value of this parameter is “False.” The section of parapet and sidewalk is asymmetric about Y axis.

6) Summary of Modeling Parapet and Sidewalk

From the analysis above, we can conclude that concrete parapet and sidewalk can be modeled by using IIsmCurveMembers interface in the current ISM schema. Their sections can be defined either by using IIsmCustomSection interface, or by using IIsmParametricSection interface with the parameters defined in this section. Figure 4-120 shows the ISM model of parapet and sidewalk.



Type	Tag	Status	Change
Circle Parallel ...		Added	Accepted
Concrete	Class_B_3_ksi	Added	Accepted
Concrete	Girder Concrete	Added	Accepted
Concrete	Class_D_4.5_ksi	Added	Accepted
Curve Member	Drilled Shaft(C...	Added	Accepted
Curve Member	Drilled Shaft(C...	Added	Accepted
Curve Member	pier column(C...	Added	Accepted
Curve Member	Drilled Shaft(C...	Added	Accepted
Curve Member	Drilled Shaft(C...	Added	Accepted
Curve Member	Drilled Shaft(C...	Added	Accepted
Curve Member	Drilled Shaft(C...	Added	Accepted
Curve Member	Drilled Shaft(C...	Added	Accepted
Curve Member	Drilled Shaft(C...	Added	Accepted
Curve Member	Drilled Shaft(C...	Added	Accepted
Curve Member	Drilled Shaft(C...	Added	Accepted

Property	New Val...	Old Value	Units	Status	Change

Figure 4-120. ISM Model of Sidewalk and Parapet

4.9.2.2.1 Mapping to IFC

Parapet in IFC is defined under IfcWallTypeEnum as a wall-like barrier to protect human occupants from falling, or to prevent the spread of fires. Frequently, it is designed on the edge of balconies, terraces or roofs.

4.10 Geometry and Sections

This section covers the description and specification of generic and specific geometries and sections.

4.10.1 Camber

A cambered component is a type of linear part whose longitudinal axis is distorted from the nominal axis during fabrication in one or two directions which are perpendicular to the longitudinal axis. The nominal axis of the part is defined by the locus of the cardinal point of the section profile that defines the shape of the part between its two ends.

For concrete girders, camber is usually defined in contract drawings using the deflection value at the mid-span under certain load cases, including prestressed force, beam dead load, slab and haunch dead load, other non-composite dead load and superimposed dead load. Figure 4-121 shows a typical camber table used by the NYSDOT.

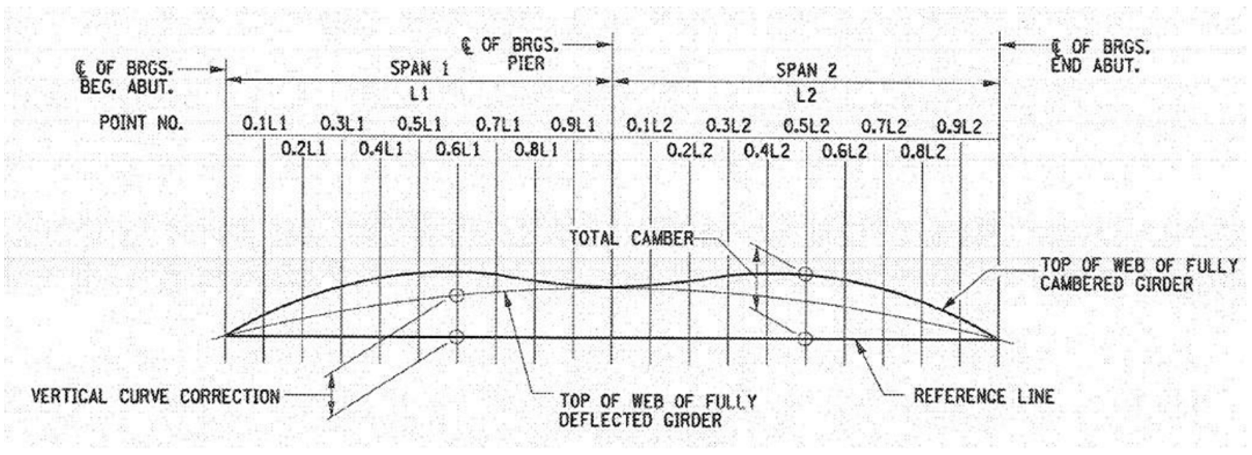
CAMBER TABLE		
CAMBER DUE TO PRESTRESSED FORCE (WITHOUT GROWTH) @ TRANSFER	↑	60
DEFLECTION DUE TO BEAM DEAD LOAD (WITHOUT GROWTH) @ TRANSFER	↓	-21
DEFLECTION DUE TO SLAB DEAD LOAD	↓	-11
DEFLECTION DUE TO SUPERIMPOSED DEAD LOAD	↓	-3
NET TOTAL DEFLECTION AT BEAM DUE TO CAMBER AND DEAD LOAD DEFLECTIONS	↑	25

Figure 4-121. Camber Table for Concrete Girder

Camber may be defined using the deflection values at the 10th point along the longitudinal direction of a steel girder. However, sometimes camber is defined using the deflection values at arbitrary points along the longitudinal direction. Common camber cases for steel girder include steel dead load, concrete dead load, superimposed dead load, and vertical curve. Figure 4-122 and Figure 4-123 show a typical camber table and a camber diagram for steel plate girder.

CAMBER TABLE		CL BRG. BEGIN ABUT,	0.10L	0.20L	0.30L	0.40L	0.50L	0.60L	0.70L	0.80L	0.90L	CL BRG. PIER
GIRDER 1	I STEEL D.L. (m)	0.000	0.011	0.020	0.027	0.030	0.029	0.025	0.019	0.011	0.005	0.000
	II CONCRETE D.L. (m)	0.000	0.027	0.051	0.066	0.074	0.073	0.063	0.047	0.029	0.012	0.000
	III SUPERIMPOSED D.L. (m)	0.000	0.007	0.013	0.016	0.018	0.018	0.016	0.012	0.007	0.003	0.000
	IV VERTICAL CURVE (m)	0.000	0.018	0.035	0.049	0.061	0.071	0.079	0.084	0.088	0.089	0.088
	TOTAL = I+II+III+IV	0.000	0.063	0.119	0.158	0.183	0.191	0.183	0.162	0.135	0.109	0.088

Figure 4-122. Camber Table for Steel Girder



℄ OF BRGS. BEG. ABUT.
℄ OF BRGS. PIER
℄ OF BRGS. END ABUT.
SPAN 1
L1
SPAN 2
L2
POINT NO.
0.1L1 0.3L1 0.5L1 0.7L1 0.9L1
0.2L1 0.4L1 0.6L1 0.8L1
0.1L2 0.3L2 0.5L2 0.7L2 0.9L2
0.2L2 0.4L2 0.6L2 0.8L2
TOTAL CAMBER
TOP OF WEB OF FULLY CAMBERED GIRDER
VERTICAL CURVE CORRECTION
TOP OF WEB OF FULLY DEFLECTED GIRDER
REFERENCE LINE

Figure 4-123. Camber Diagram of Steel Plate Girder

Other current data models consider but are unable to fully support exchange of bridge camber data. In the CIMsteel Integration Standards (CIS/2), camber is described as offsets in two directions at one certain point between bearings [4]. In the Industry Foundation Classes (IFC4), camber is defined as an offset only at the mid-span of the girder [7]. These data models cannot capture all the camber ordinates at spaces along span length. However, the XML schema proposed herein can well address this problem.

Based on the above camber tables in contract drawings, camber should have at least the following attribute:

- Location of points of interest.
- Camber value in the vertical direction.
- Camber value in the transverse direction.

4.10.1.1 Mapping to OpenBrIM

The XML code created based on the OpenBrIm schema is shown as follows:

```
<Obj Name="Cambers">
        <Obj Name=”CamberUnderSteelDeadLoad”>
                <Point Sta="BridgeBeginsSta*Span1Length*0.0" Trans="0" Vert="0.000"/>
                <Point Sta="BridgeBeginsSta*Span1Length*0.1" Trans="0" Vert="0.008"/>
                <Point Sta="BridgeBeginsSta*Span1Length*0.2" Trans="0" Vert="0.014"/>
                <Point Sta="BridgeBeginsSta*Span1Length*0.3" Trans="0" Vert="0.018"/>
                <Point Sta="BridgeBeginsSta*Span1Length*0.4" Trans="0" Vert="0.019"/>
                <Point Sta="BridgeBeginsSta*Span1Length*0.5" Trans="0" Vert="0.018"/>
                <Point Sta="BridgeBeginsSta*Span1Length*0.6" Trans="0" Vert="0.014"/>
                <Point Sta="BridgeBeginsSta*Span1Length*0.7" Trans="0" Vert="0.009"/>
                <Point Sta="BridgeBeginsSta*Span1Length*0.8" Trans="0"
        Vert="0.004"/>
                <Point Sta="BridgeBeginsSta*Span1Length*0.9" Trans="0"
        Vert="0.000"/>
                <Point Sta="BridgeBeginsSta*Span1Length*1.0" Trans="0"
        Vert="0.000"/>
        </Obj>
</Obj>
```

In the XML code, a set of camber ordinates is given a name (Name=”Camber1”) so that it can be referenced by multiple girders. CamberCase names the (load) effect that induces deflections opposed by camber. The obj nodes inside the CamberType node include a vertical camber value and a transverse deflection value (placeholder) for web panel distortion. In order to capture camber values at any number of spaces along the span length, there is no limitation on the number of the obj nodes containing camber values. The XML code for camber can be read by the OpenBrIM Viewer and rendered in amplified manner for a cambered steel girder as shown in Figures 4-124, 4-125, 4-126, 4-127, and 4-128.

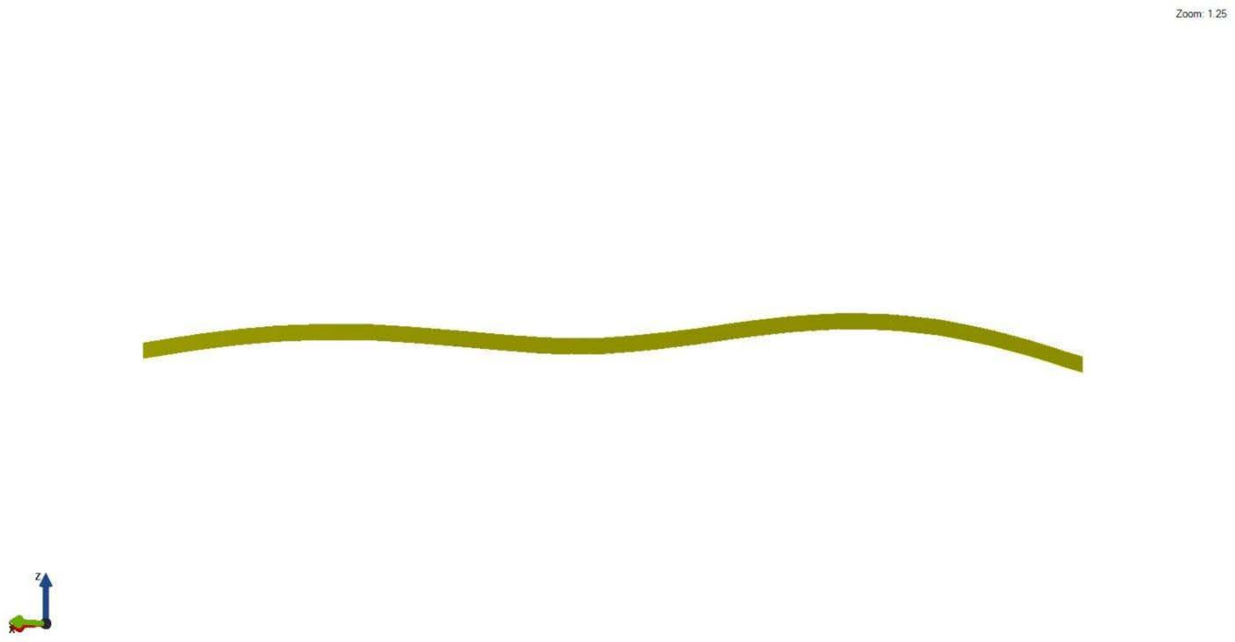


Figure 4-124. 3D View of a Cambered Girder (Steel Dead Load, Scale Factor: 100000)

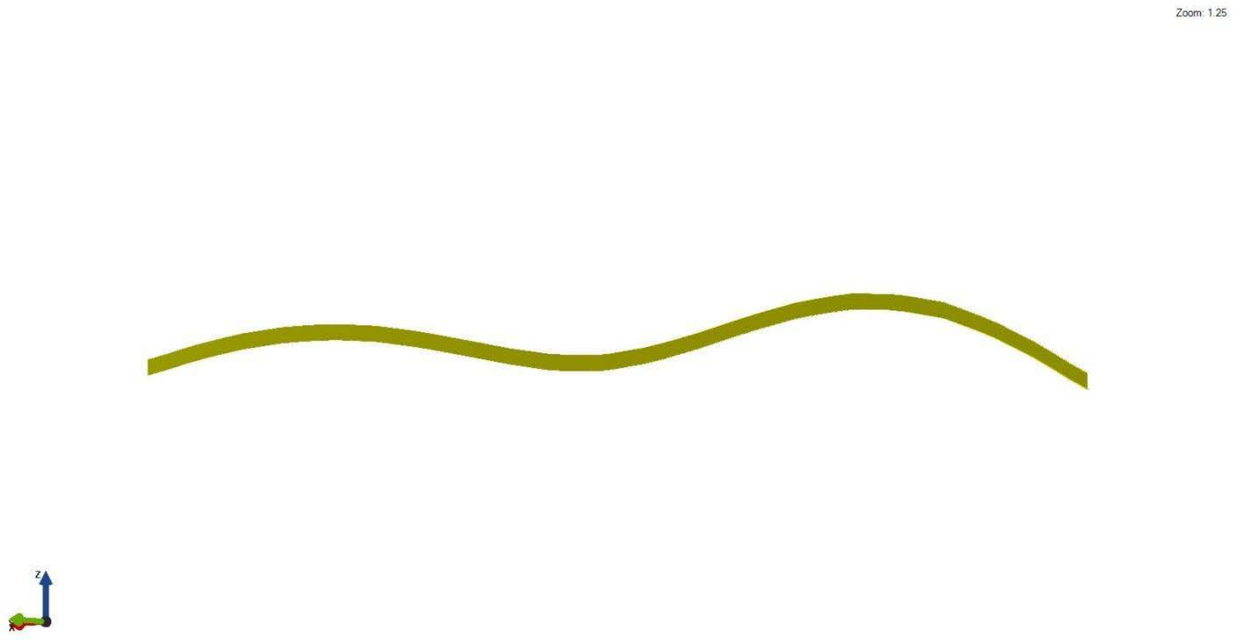


Figure 4-125. 3D View of a Cambered Girder (Concrete Dead Load, Scale Factor: 100000)

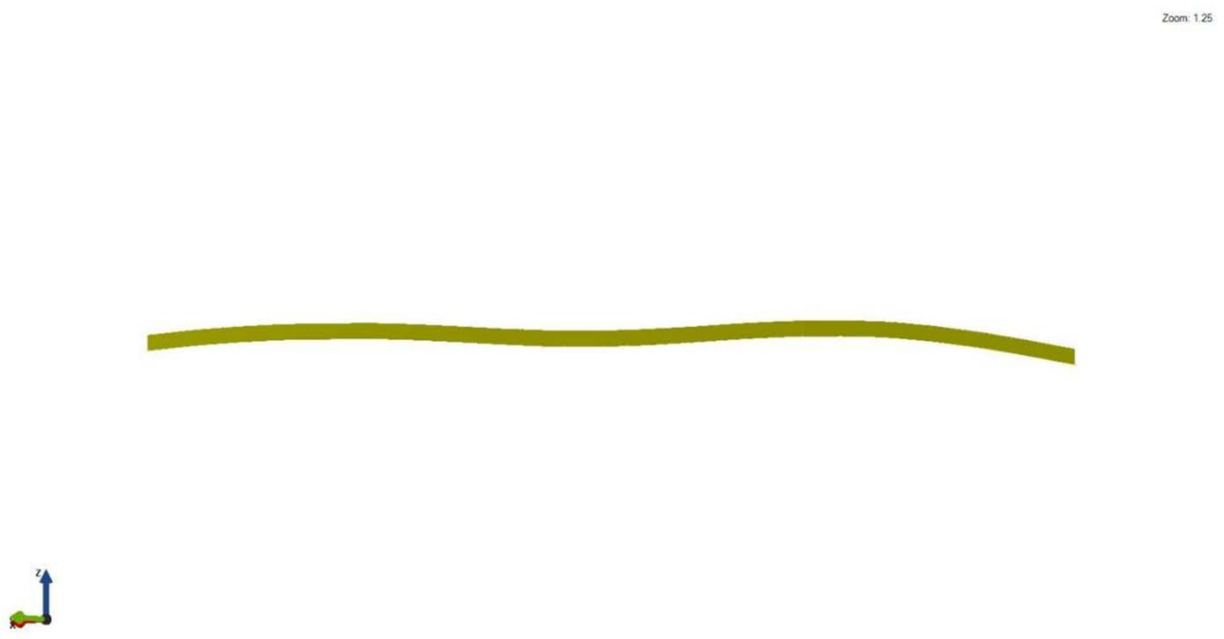


Figure 4-126. 3D View of a Cambered Girder (Superimposed Dead Load, Scale Factor: 100000)

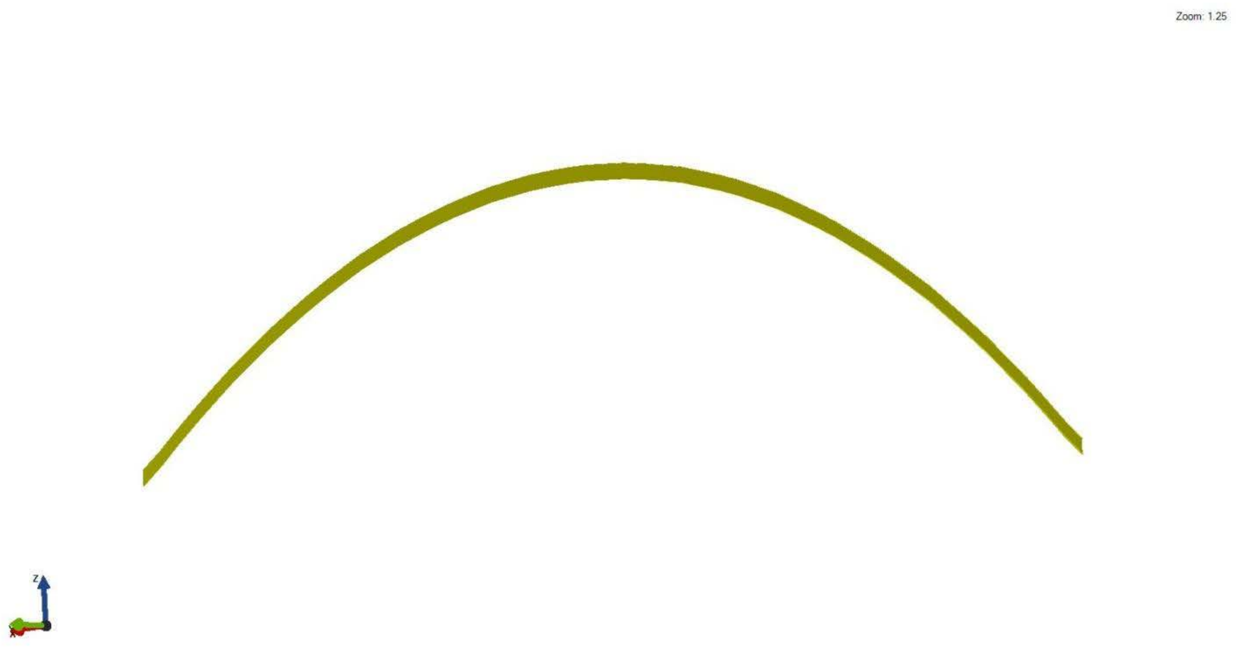


Figure 4-127. 3D View of a Cambered Girder (Vertical Curve, Scale Factor: 50000)

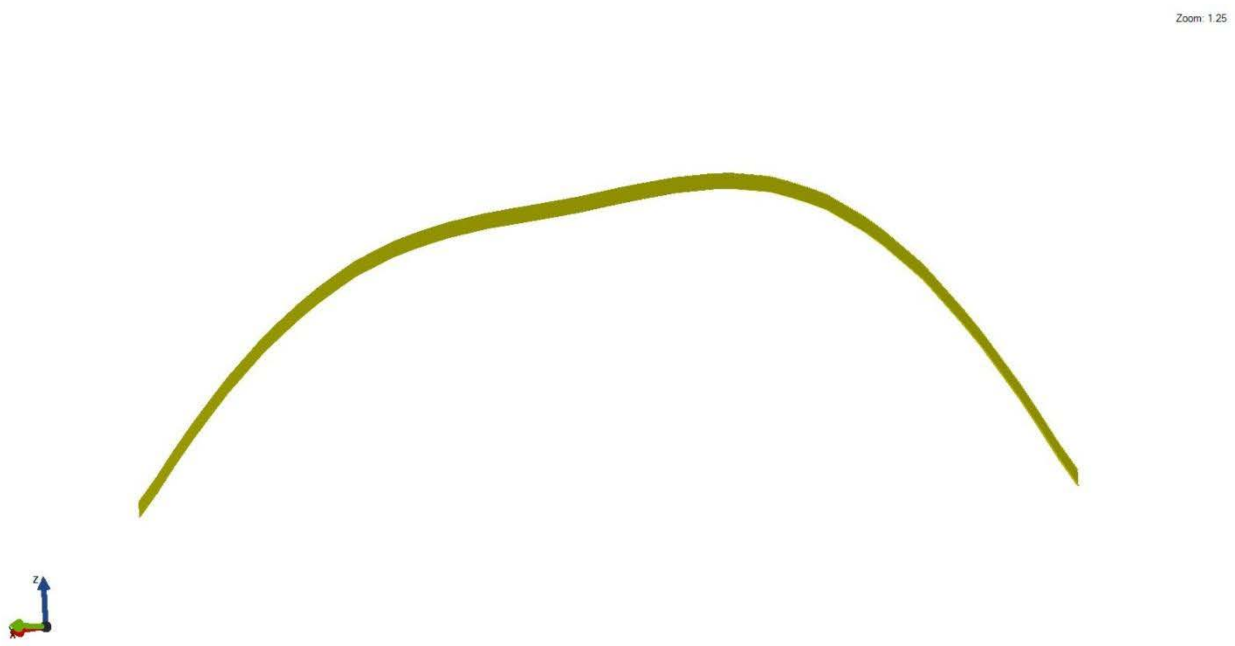


Figure 4-128. 3D View of a Cambered Girder (Total, Scale Factor: 50000)

4.10.1.2 Mapping to ISM

IIsmCurveMember interface has a property called “Camber” which takes a single value as the camber ordinate. However, this property cannot suffice definition of camber because it has three drawbacks. Firstly, it does not specify the location of that camber ordinate. Secondly, for a multi-span continuous plate girder modeled as one IsmCurveMember, each span of the girder should have one associated camber ordinate, i.e. the IsmCurveMember should have multiple camber value. Therefore, the current camber property cannot suffice this. Thirdly, this property does not specify the effect case associated with camber ordinate. For example, the girder could have different camber values under different load cases.

4.10.2 Section

4.10.2.1 Parametric Section

The case study bridge, Quincy Avenue over I-25 and LRT has seven precast prestressed concrete Bulb Tee girders, as shown in Figure 4-129. Section view of the girder is shown in the Figure 4-130.

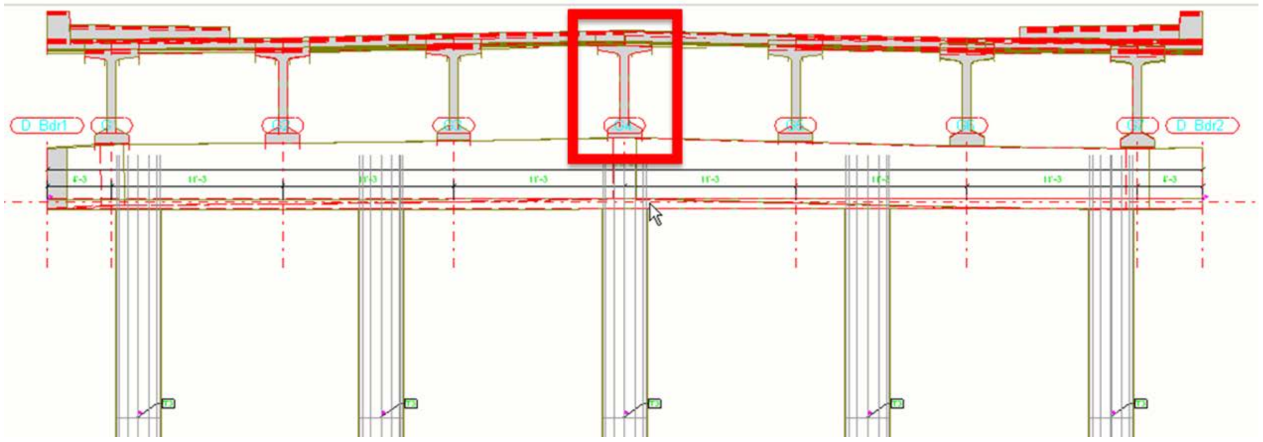


Figure 4-129. Precast Prestressed Concrete Bulb Tee Girder

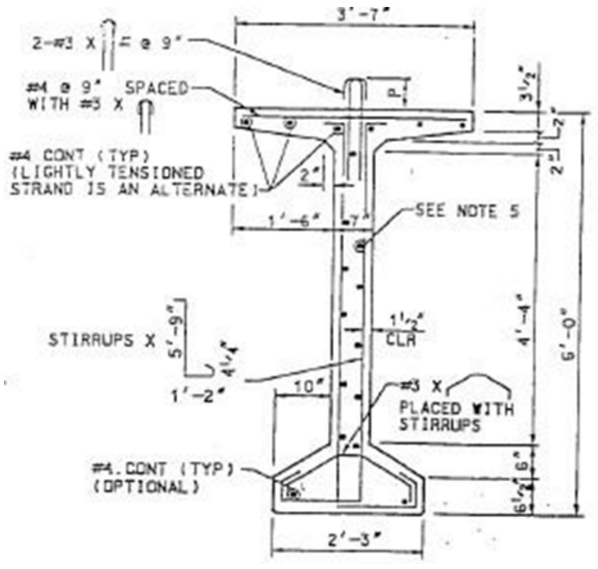


Figure 4-130. Section View of the Bulb Tee Girder from the Contract Plans

The girder type is Colorado BT-72. Its dimensions are shown in Figure 4-122, which is from the contract plans. The Colorado BT-72 girder is slightly different from the conventional BT-72 girder defined in the PCI standard in terms of dimensions. For example, the top flange width of Colorado BT-72 girder is 43 inches and that of PCI BT-72 girder is 42 inches.

The case study bridge, Staten Island Expressway over Slosson Avenue (westbound) has nine precast prestressed concrete adjacent box beams. A section view of the beam is shown in Figure 4-131.

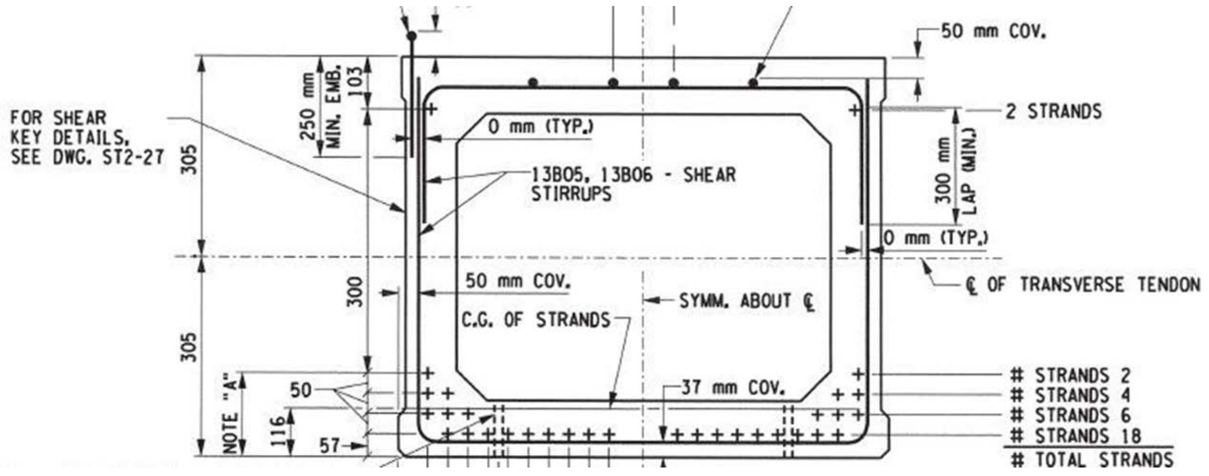


Figure 4-131. Section View of the Adjacent Box Beam from the Contract Plans

The case study bridge, Staten Island Expressway over Slosson Avenue (westbound) has ten precast prestressed NEXT beams. A section view of the beam is shown in Figure 4-132.

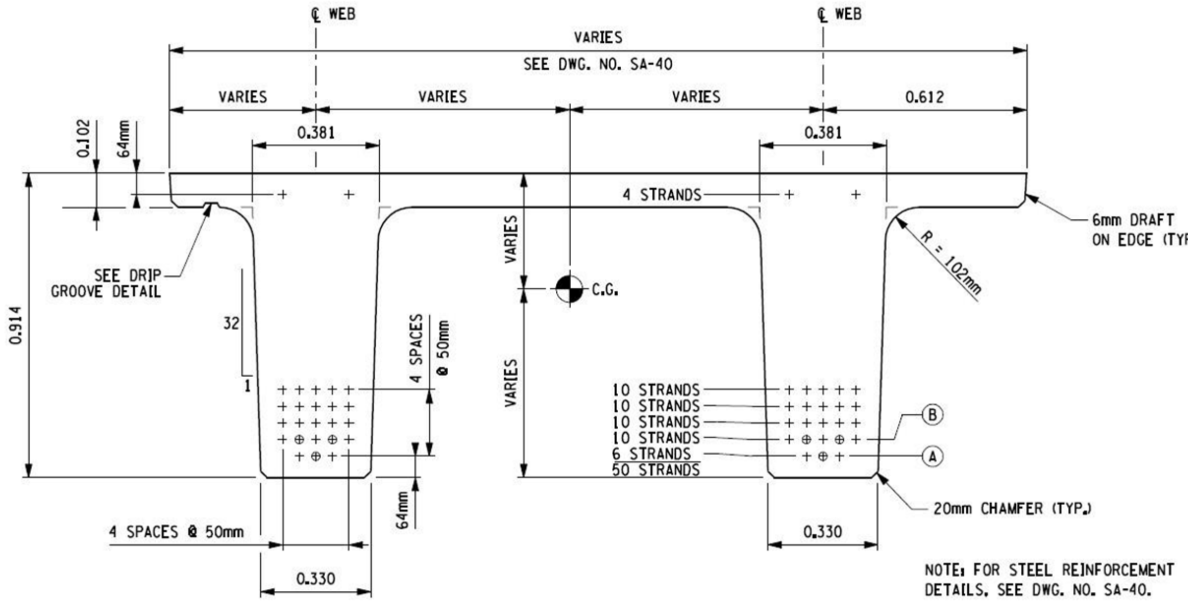


Figure 4-132. Section View of the NEXT Beam from the Contract Plans

The case study bridge, Otay River Bridge has two precast prestressed California box beams. A section view of the beam is shown in Figure 4-133.

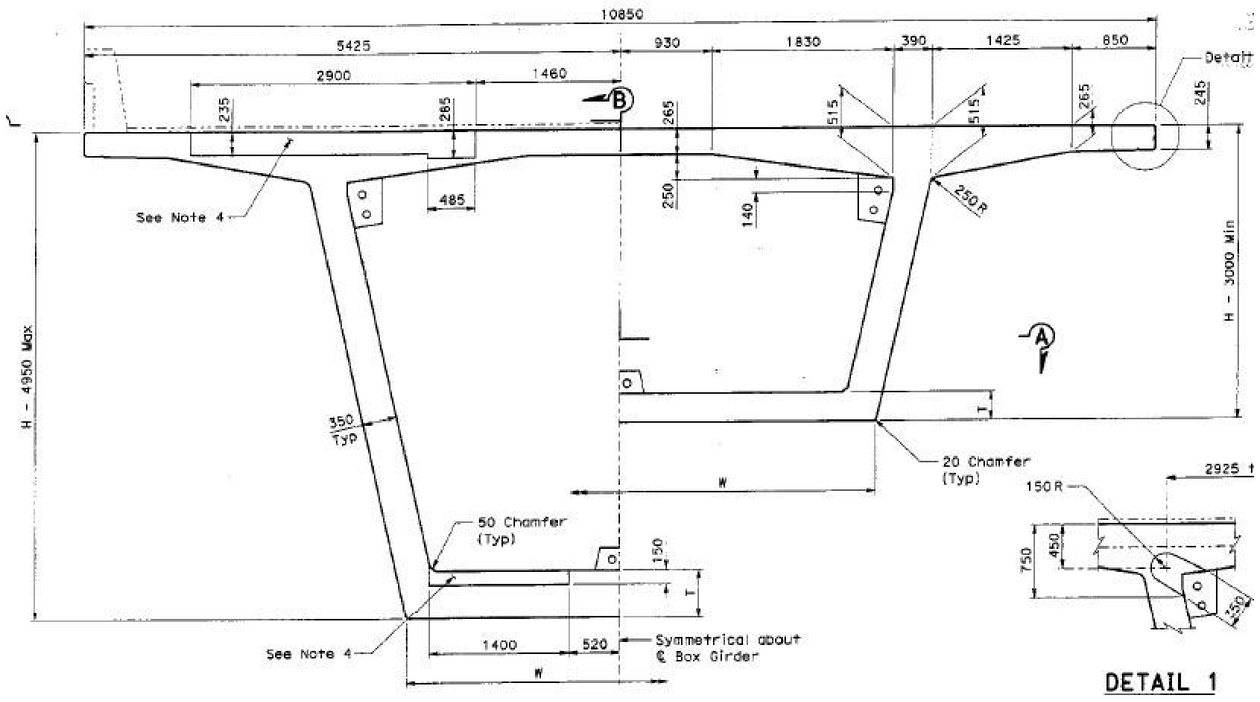


Figure 4-133. Section View of the California Box Beam from the Contract Plans

4.10.2.2 Mapping to ISM

The interface hierarchy for linear bridge components in a tree format is shown below [4].

IIsmObject (abstract)
- ➔IIsmSection (abstract)
  - ➔IIsmConstantSection (abstract)
    - ➔IIsmFixedParametricSection (abstract)
      - ➔IIsmParametricSection (concrete)
      - ➔IIsmTableSection (concrete)
    - ➔IIsmBuiltUpSection (concrete)
    - ➔IIsmCustomSection (concrete)
  - ➔IIsmVaryingSection (concrete)
- ➔IIsmSubObject (abstract)
  - ➔IIsmBuiltUpSectionComponent (concrete)
  - ➔IIsmVaryingSectionSegment (concrete)

Because bridge girders/beams are linear members with constant section, IIsmTableSection, IIsmParametricSection, IIsmCustomSection and IIsmBuiltUpSection can be used to define the girder/beam cross sections.

1) IIsmTableSection

IIsmTableSection defines a cross section shape that is defined in a table in a standard or a specification [4]. Because most of the precast prestressed concrete girder sections are predefined in standards such as PCI, this interface is suitable for Bulb Tee girder, adjacent box beam and California box beam.

Table 4-9 lists the properties of the interface.

Table 4-9. Girder IIsmTableSection Properties

Property	Value	Value	Value
StandOrganization	AASHTO/PCI (Precast/Prestressed Concrete Institute)	AASHTO/PCI (Precast/Prestressed Concrete Institute)	AASHTO – PCI – ASBI (American Segmental Bridge Institute)
Revision	Third Edition, First Release, Nov. 2011	Third Edition, First Release, Nov. 2011	1997
Manufacturer	N/A	N/A	N/A
SectionName	AASHTO I-Beams	AASHTO Box Beams	Segmental Box Girder

ISM uses the SPC (Structural Property Catalog) library of table section. However, the currently supported SPC/ISM StandardsOrganizations do not include PCI or other bridge organizations. Therefore, if it is too time-consuming to add the bridge standards to the library, IIsmParametricSection, IIsmCustomSection or IIsmBuiltUpSection can be used to define the girder/beam cross sections.

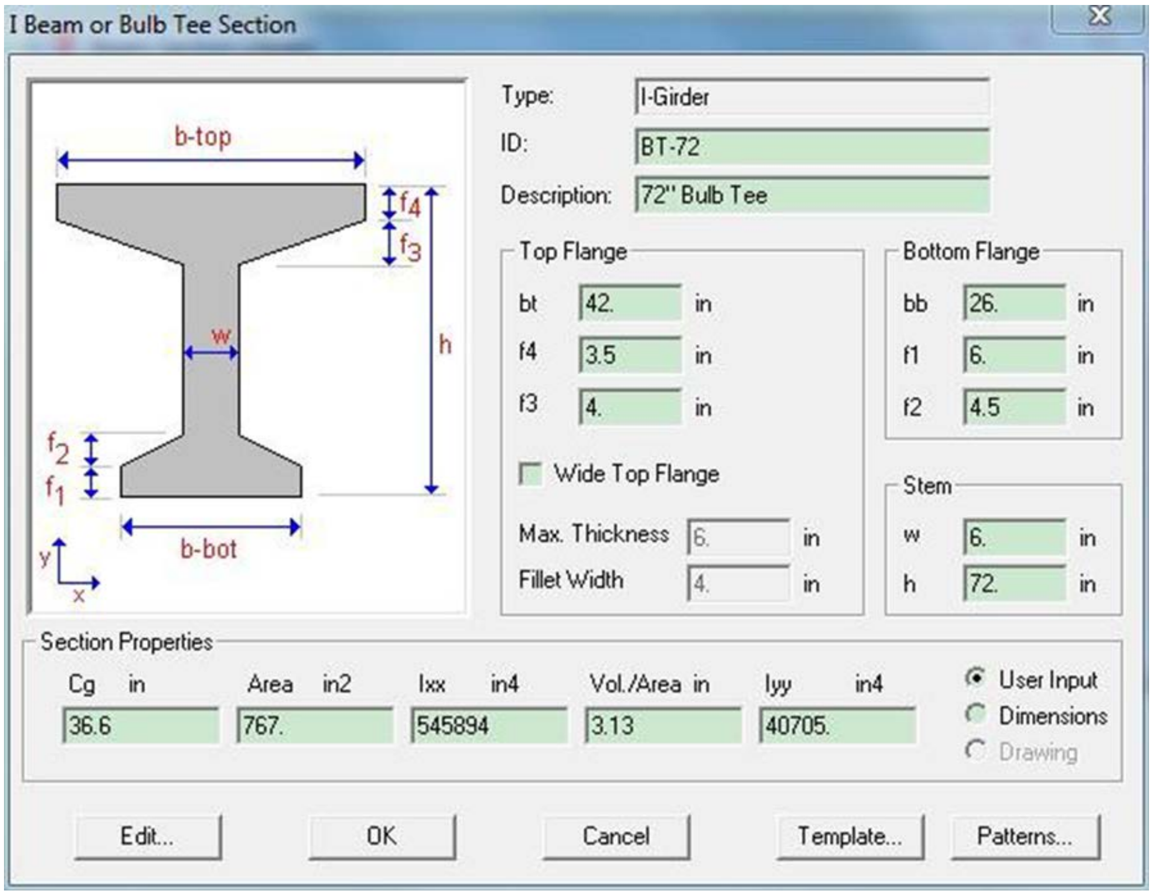
2) IIsmParametricSection

IIsmParametricSection defines a parametric section whose parameters can be set as well as read [4]. In the parametric section types supported, no such a type can be used to model the four kinds of girders/beams described in the section before. Therefore, these girder sections need to be added to those parametric section types. The necessary parameters to define it can be retrieved from other software applications. LEAP Conspan and SAP2000 Bridge Module (CSiBridge) are used to illustrate possible parametric definitions next.

3) Bulb Tee Girder Section Parameters in LEAP Conspan

The parameters of Bulb Tee girder section defined in the LEAP Conspan girder section library [5] are shown below. The parameters have their meanings defined in Figure 4-134.

- Vertical: H, f1, f2, f3, f4.
- Horizontal: b-top, w, b-bot.
- Section Properties: Cg, Area, Ixx, Vol./Area, Iyy.
- Other: Type, ID, Description.



Type: I-Girder
ID: BT-72
Description: 72" Bulb Tee

Top Flange

bt	42.	in
f4	3.5	in
f3	4.	in

☐ Wide Top Flange

Max. Thickness	6.	in
Fillet Width	4.	in

Bottom Flange

bb	26.	in
f1	6.	in
f2	4.5	in

Stem

w	6.	in
h	72.	in

Section Properties

Cg in	Area in2	Ixx in4	Vol./Area in	Iyy in4
36.6	767.	545894	3.13	40705.

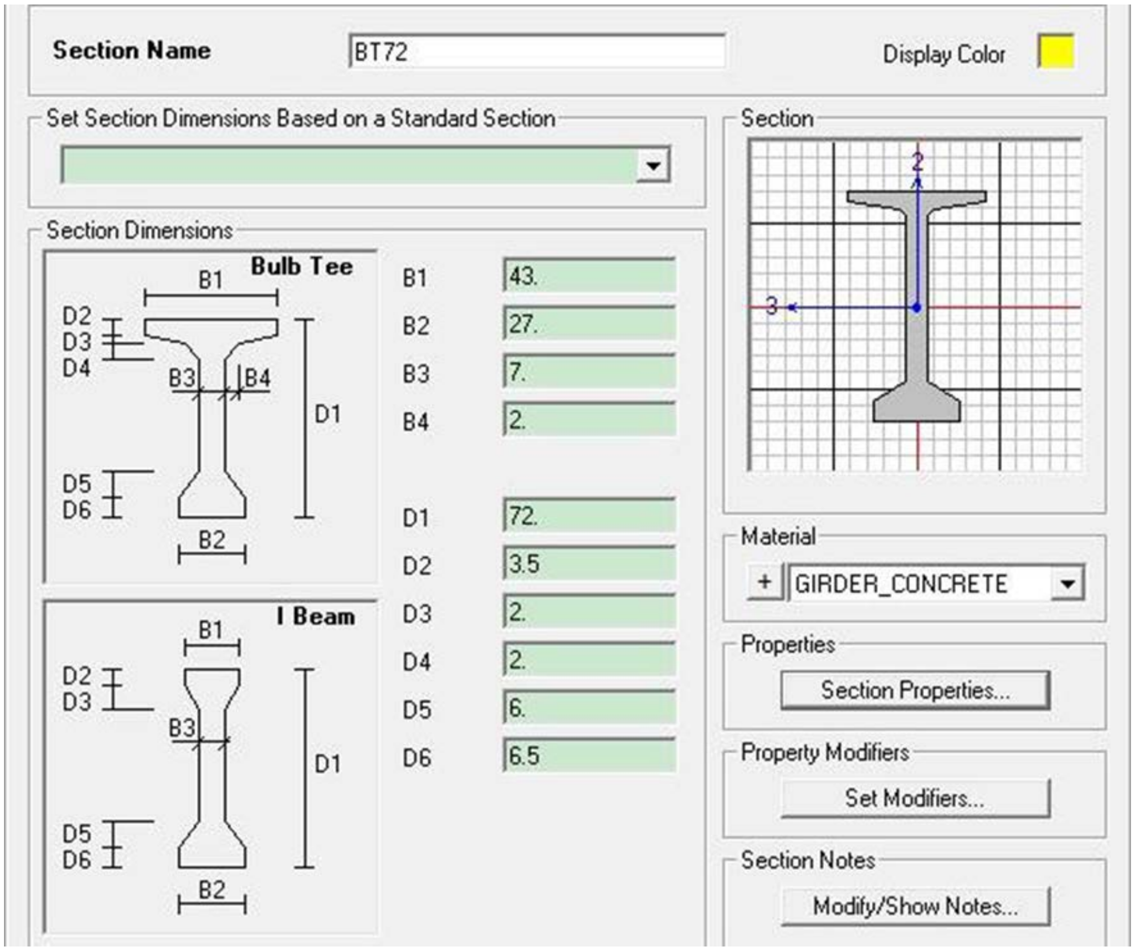
[✓ User Input] Dimensions Drawing

Figure 4-134. Definition of Bulb Tee Girder in LEAP Conspan

4) Bulb Tee Girder Section Parameters in SAP2000 (CSiBridge) [10]

The parameters of Bulb Tee girder section in the SAP2000 (CSiBridge) [10] girder definition window are shown below. The parameters have their meanings defined in Figure 4-135.

- Vertical: D1, D2, D3, D4, D5, D6.
- Horizontal: B1, B2, B3, B4.
- Section Properties: property modifiers.
- Other: Section Name, Section Notes.


Section Name BT72

Set Section Dimensions Based on a Standard Section

B1	43.
B2	27.
B3	7.
B4	2.
D1	72.
D2	3.5
D3	2.
D4	2.
D5	6.
D6	6.5

Material GIRDER_CONCRETE

Figure 4-135. Definition of Bulb Tee Girder in SAP2000 (CSiBridge) [10]

5) Bulb Tee Girder Section Parameters Proposed for ISM

From the parameters retrieved from these two software applications, the parameters proposed for ISM schema are listed below. Their meanings can be found in Figure 4-136.

- Vertical: TotHeight, H1, H2, H3, H4, H5.
- Horizontal: TopFlangeWidth, BotFlangeWidth, WebWidth, W1.
- Other: Use, SystemKind, SectionDescription.
- Fillet: RadiusFilletIndicator, TopFlangeFilletRadiusFlange, TopFlangeFilletRadiusWeb, BotFlangeFilletRadiusFlange, BotFlangeFilletRadiusWeb.

Parameters for fillet could be used; e.g., for New England Bulb Tee girders.

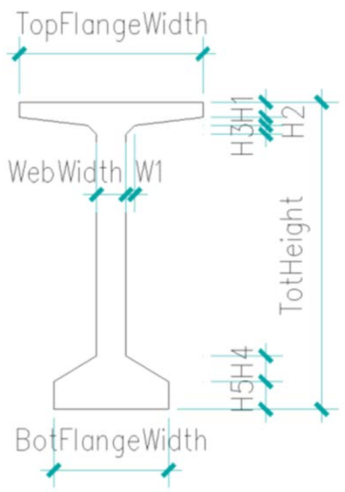
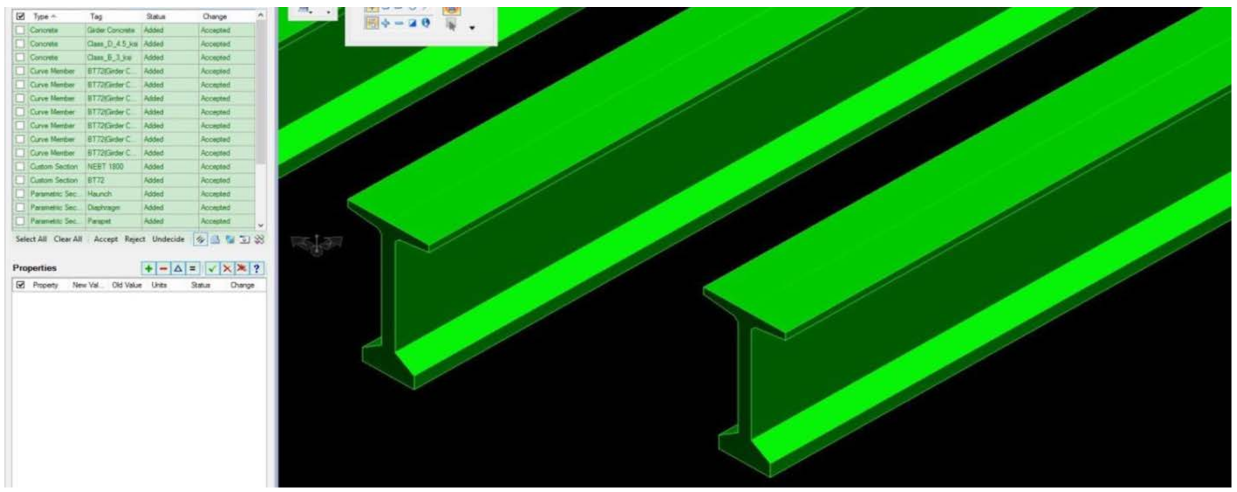


Figure 4-136. Definition of Bulb Tee Girder for ISM

The cross section of Bulb Tee girders can also be defined using IIsmCustomSection interface. Figure 4-137 shows ISM models of Bulb Tee girders.



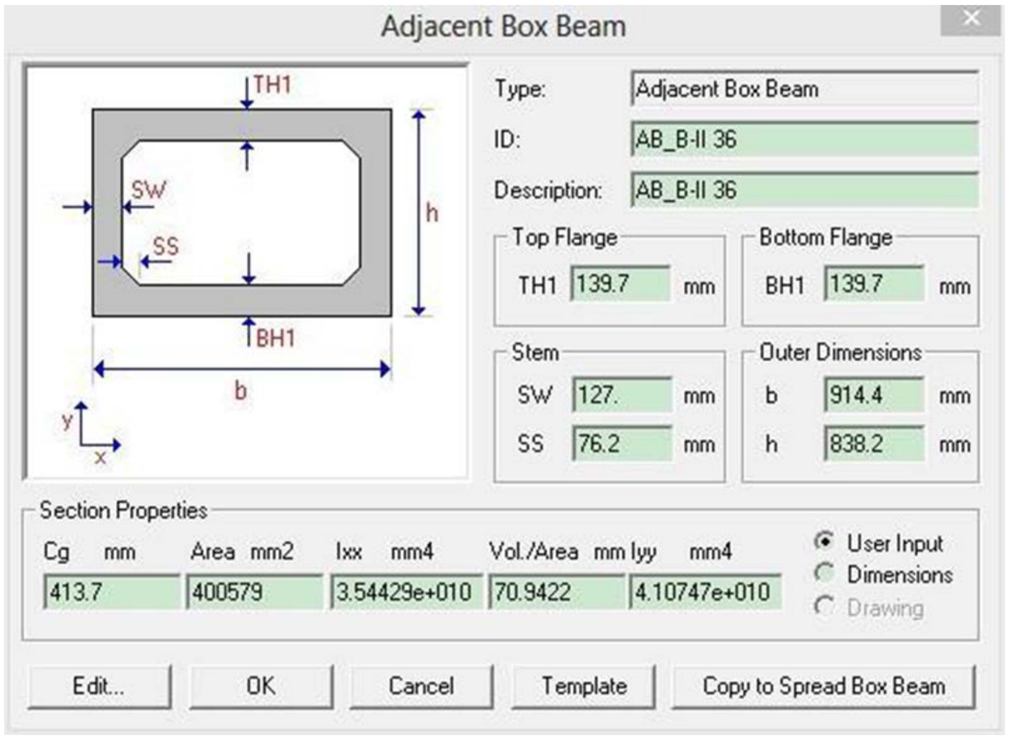
Type	Tag	Status	Change
Concrete	Girder Concrete	Added	Accepted
Concrete	Class_D_4.5_ksi	Added	Accepted
Concrete	Class_B_3_ksi	Added	Accepted
Curve Member	BT72(Girder C...	Added	Accepted
Curve Member	BT72(Girder C...	Added	Accepted
Curve Member	BT72(Girder C...	Added	Accepted
Curve Member	BT72(Girder C...	Added	Accepted
Curve Member	BT72(Girder C...	Added	Accepted
Curve Member	BT72(Girder C...	Added	Accepted
Curve Member	BT72(Girder C...	Added	Accepted
Custom Section	NEBT 1800	Added	Accepted
Custom Section	BT72	Added	Accepted
Parametric Sec...	Haunch	Added	Accepted
Parametric Sec...	Diaphragm	Added	Accepted
Parametric Sec...	Parapet	Added	Accepted

Figure 4-137. ISM Models of Bulb Tee Girders

6) Adjacent Box Beam Section Parameters in LEAP Conspan

The parameters of adjacent box beam section defined in the LEAP Conspan girder section library are shown below. The parameters have their meanings defined in Figure 4-138.

- Vertical: TH1, BH1, h.
- Horizontal: SW, SS, b.
- Section Properties: Cg, Area, Ixx, Vol./Area, Iyy.
- Other: Type, ID, Description.


Adjacent Box Beam

Type:	Adjacent Box Beam
ID:	AB_B-II 36
Description:	AB_B-II 36

Top Flange: TH1 139.7 mm

Bottom Flange: BH1 139.7 mm

Stem: SW 127. mm; SS 76.2 mm

Outer Dimensions: b 914.4 mm; h 838.2 mm

Section Properties

Cg mm	Area mm2	Ixx mm4	Vol./Area mm	Iyy mm4
413.7	400579	3.54429e+010	70.9422	4.10747e+010

[✓ User Input] Dimensions Drawing

Figure 4-138. Definition of Adjacent Box Beam in LEAP Conspan

7) Adjacent Box Beam Section Parameters Proposed for ISM

From the parameters retrieved from LEAP Conspan and CSiBridge, the parameters proposed for ISM schema are listed below. Their meanings can be found in Figure 4-139.

- Vertical: TotH, h1, h2, h3, h4.
- Horizontal: TotW, w1.
- Fillet: fh1, fw1, fh2, fw2, fh3, fw3.
- Other: Use, SystemKind, SectionDescription.

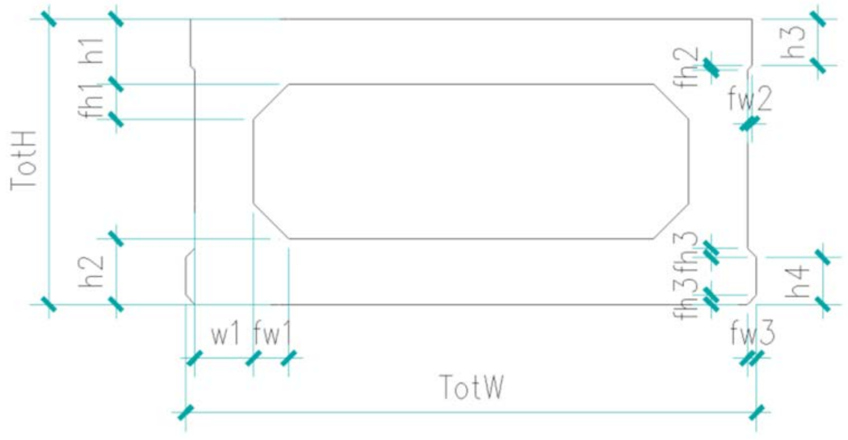
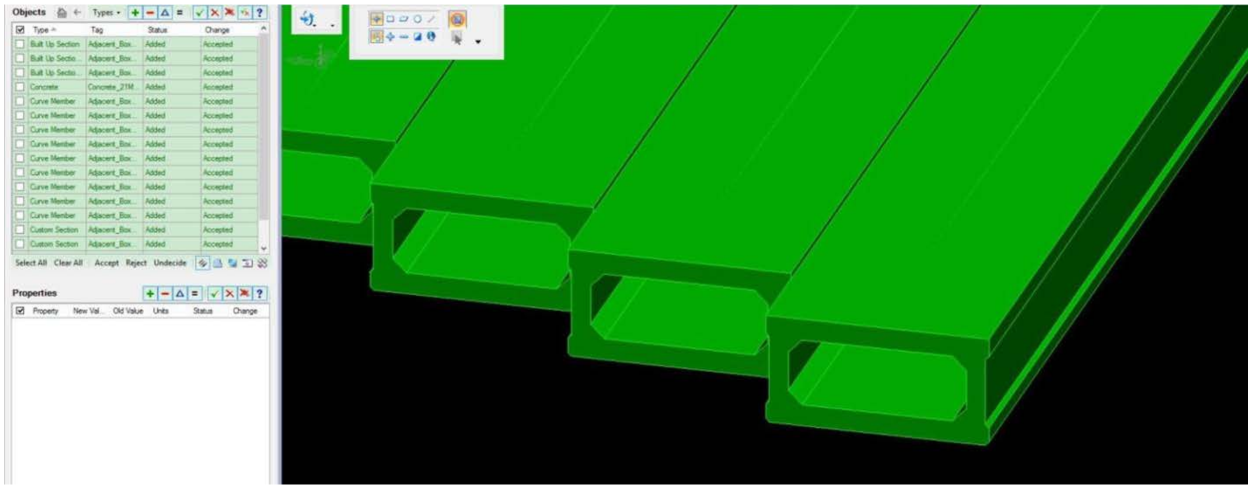


Figure 4-139. Definition of Adjacent Box Beam for ISM

Because there is a cutout in the cross section of the adjacent box beam, the section was defined as two segments by using IIsmCustomSection and then grouped by using IIsmBuiltUpSection. Figure 4-140 shows ISM models of adjacent box beams.



Type	Tag	Status	Change
Built Up Section	Adjacent_Box...	Added	Accepted
Built Up Sectio...	Adjacent_Box...	Added	Accepted
Built Up Sectio...	Adjacent_Box...	Added	Accepted
Concrete	Concrete_21M...	Added	Accepted
Curve Member	Adjacent_Box...	Added	Accepted
Curve Member	Adjacent_Box...	Added	Accepted
Curve Member	Adjacent_Box...	Added	Accepted
Curve Member	Adjacent_Box...	Added	Accepted
Curve Member	Adjacent_Box...	Added	Accepted
Curve Member	Adjacent_Box...	Added	Accepted
Curve Member	Adjacent_Box...	Added	Accepted
Curve Member	Adjacent_Box...	Added	Accepted
Curve Member	Adjacent_Box...	Added	Accepted
Custom Section	Adjacent_Box...	Added	Accepted
Custom Section	Adjacent_Box...	Added	Accepted

Property	New Val...	Old Value	Units	Status	Change

Figure 4-140. ISM Models of Adjacent Box Beams

8) NEXT Beam Section Parameters Proposed for ISM

In order to make definition of NEXT beam available using IIsmParametricSection, the following parameters are proposed by the authors to be added to ISM section library, as follows. Their meanings can be found in Figure 4-141.

- Vertical: TotH, h1.
- Horizontal: TotW, w1, w2, w3.
- Fillet: fh1, fw1, s1, r1.
- Other: Use, SystemKind, SectionDescription.

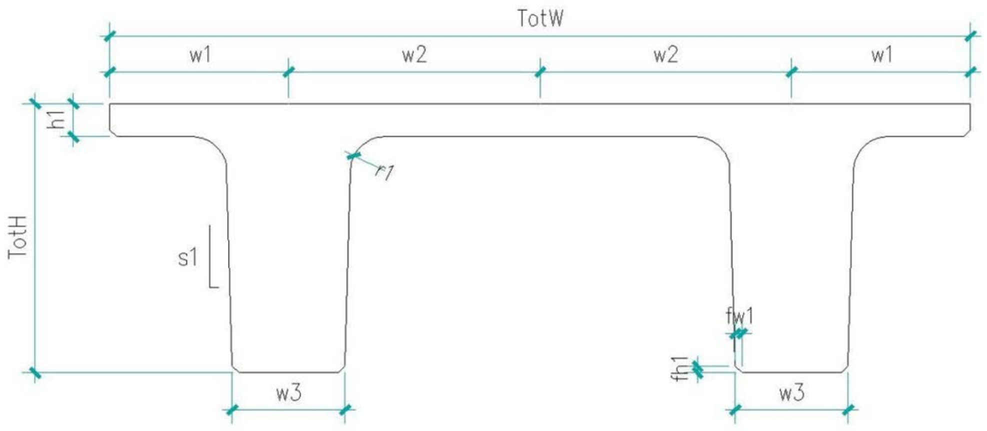
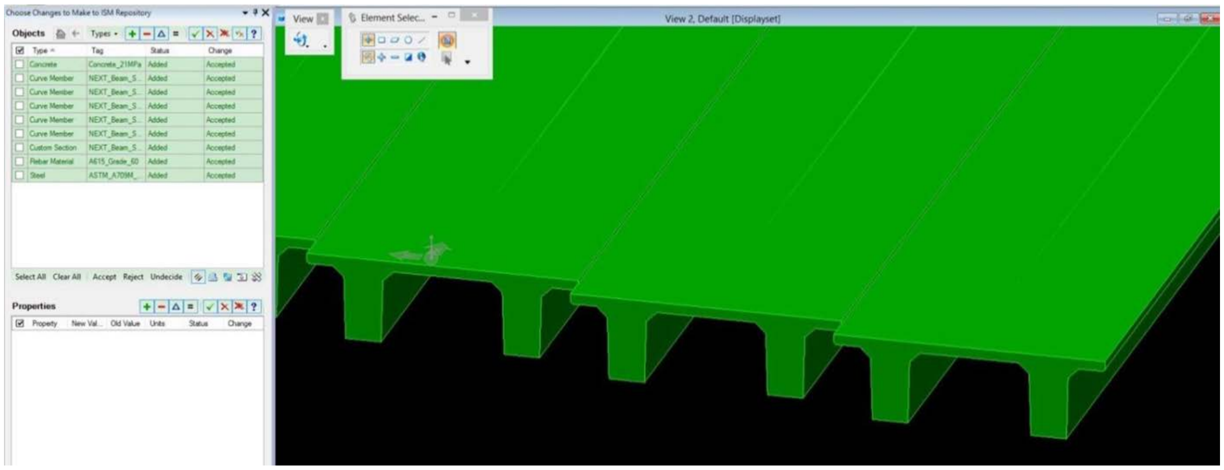


Figure 4-141. Definition of NEXT Beam for ISM

The cross section of NEXT beam can also be defined using IIsmCustomSection interface. Figure 4-142 shows ISM models of NEXT beams.



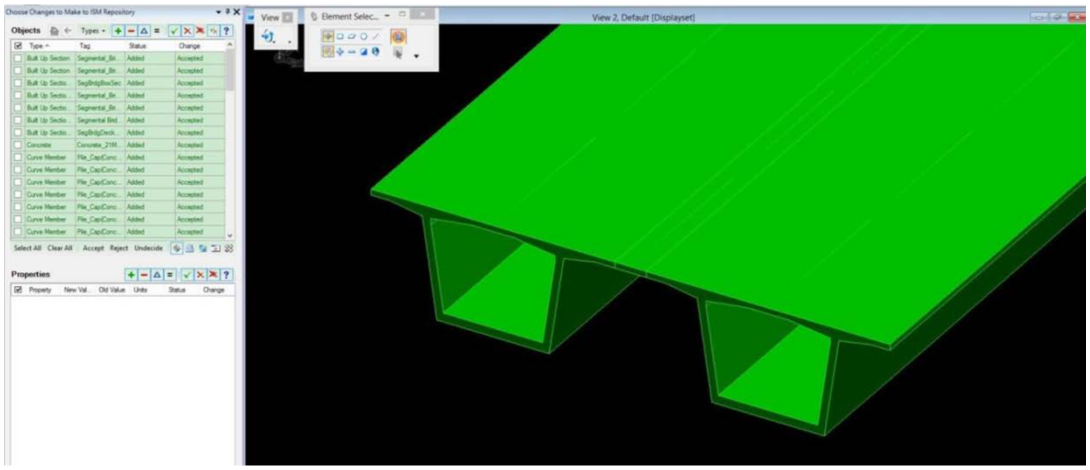
Type	Tag	Status	Change
Concrete	Concrete_21MPa	Added	Accepted
Curve Member	NEXT_Beam_S...	Added	Accepted
Curve Member	NEXT_Beam_S...	Added	Accepted
Curve Member	NEXT_Beam_S...	Added	Accepted
Curve Member	NEXT_Beam_S...	Added	Accepted
Curve Member	NEXT_Beam_S...	Added	Accepted
Custom Section	NEXT_Beam_S...	Added	Accepted
Rebar Material	A615_Grade_60	Added	Accepted
Steel	ASTM_A709M_...	Added	Accepted

Property	New Val...	Old Value	Units	Status	Change

Figure 4-142. ISM Models of NEXT Beams

9) California Box Beam Section Parameters Proposed for ISM

Because there is a cutout in the cross section of the California box beam, the section was defined as two segments by using IIsmCustomSection and then grouped by using IIsmBuiltUpSection. Figure 4-143 shows ISM models of California box beams.



Type	Tag	Status	Change
Built Up Section	Segmental_Bri...	Added	Accepted
Built Up Section	Segmental_Bri...	Added	Accepted
Built Up Sectio...	SegBrdgBoxSec	Added	Accepted
Built Up Sectio...	Segmental_Bri...	Added	Accepted
Built Up Sectio...	Segmental_Bri...	Added	Accepted
Built Up Sectio...	Segmental Brid...	Added	Accepted
Built Up Sectio...	SegBrdgDeck...	Added	Accepted
Concrete	Concrete_21M...	Added	Accepted
Curve Member	Pile_CapConc...	Added	Accepted
Curve Member	Pile_CapConc...	Added	Accepted
Curve Member	Pile_CapConc...	Added	Accepted
Curve Member	Pile_CapConc...	Added	Accepted
Curve Member	Pile_CapConc...	Added	Accepted
Curve Member	Pile_CapConc...	Added	Accepted
Curve Member	Pile_CapConc...	Added	Accepted

Property	New Val...	Old Value	Units	Status	Change

Figure 4-143. ISM Models of California Box Beams

4.11 Substructure

4.11.1 Abutments

Abutments support the superstructure at the ends of a bridge. Basic types of abutments can be broadly classified as:

- Conventional.
- Integeral.
- Semi-Intergal.

4.11.1.1 Conventional Abutment with Wing Walls

Abutment with wing walls can be defined as an assembly of extruded solids and B-Rep solids. In this example template, face wall, back wall and footings are defined as extrusion solids while the wing walls are defined as B-Rep solids. Rebar definition is not included in this example.

Parameters required for defining the abutment are shown in Figure 4-144 and Figure 4-145. Following parameters are required for definining abutment with wing walls:

- Length of face wall, l_w Height of face wall, h_fWall.
- Thickness of face wall, t_fWall.
- Height of back wall, h_bWall.
- Thickness of back wall, t_bWall.
- Length of wing wall-1, l_w1Wall.
- Height of wing wall-1, h_w1Wall.
- Thickness of wing wall-1, t_w1Wall.
- Skew angle of wing wall-1, angle_w1Wall.
- Stub length of wing wall-1, s_w1Wall.
- Length of wing wall-2, l_w2Wall.
- Height of wing wall-2, h_w2Wall.
- Thickness of wing wall-2, t_w2Wall.
- Skew angle of wing wall-2, angle_w2Wall.
- Stub length of wing wall-2, s_w2Wall.
- Length of face/back wall footing, l_w1Footing.
- Depth of face/back footing, d_w1Footing.
- Width segment A of face/back footing, a_w1Footing.
- Width segment B of face/back footing, b_w1Footing.
- Length of wing wall-1 footing, l_w1Footing.
- Depth of wing wall-1 footing, d_w1Footing.
- Width segment A of wing wall-1 footing, a_w1Footing.
- Width segment B of wing wall-1 footing, b_w1Footing.
- Length of wing wall-2 footing, l_w2Footing.
- Depth of wing wall-2 footing, d_w2Footing.
- Width segment A of wing wall-2 footing, a_w2Footing.
- Width segment B of wing wall-2 footing, b_w2Footing.
- Material Properties, material.

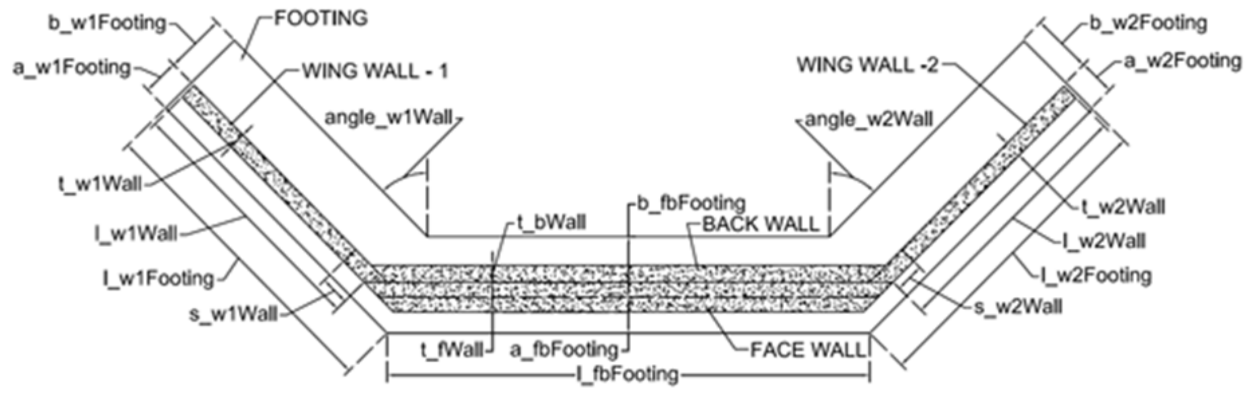


Figure 4-144. Plan View

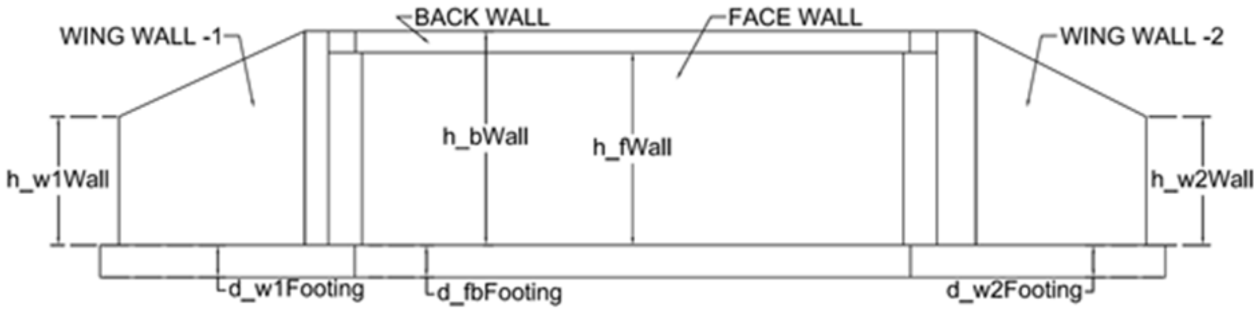
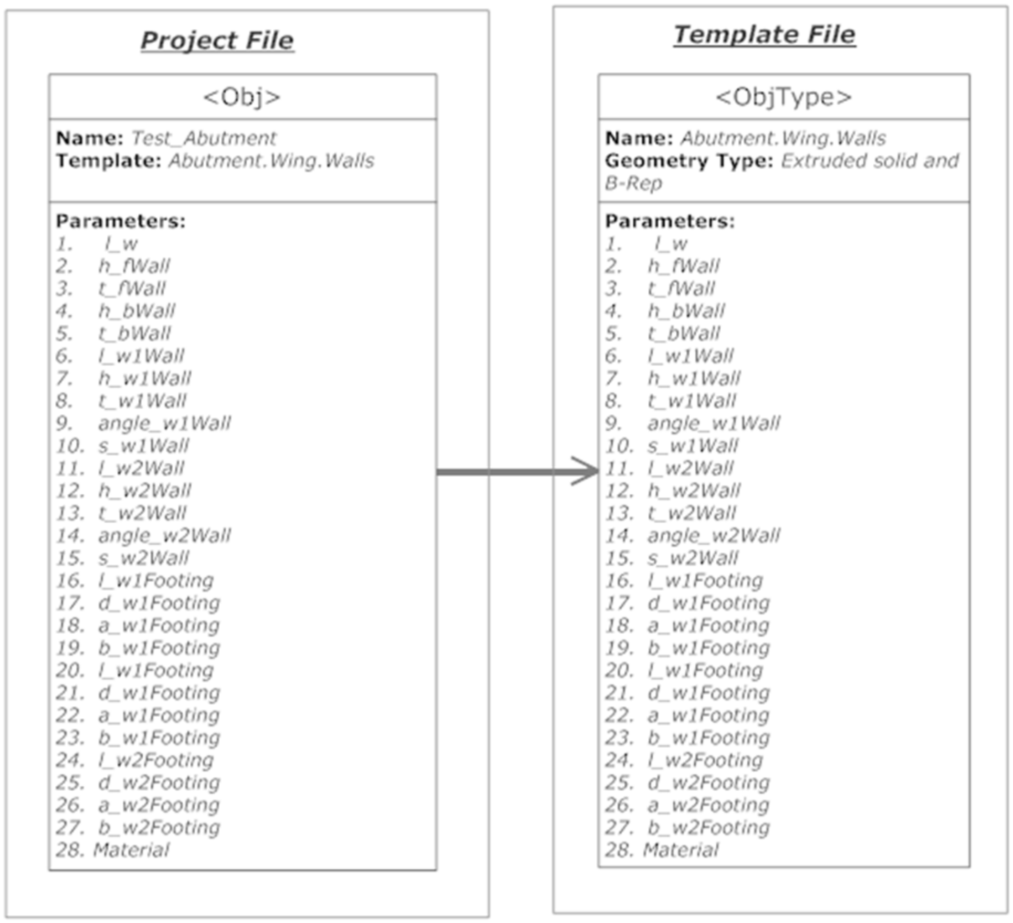


Figure 4-145. Elevation View

Figure 4-146 shows the rendering of “Test_Abutment_1” model generated using “Abutment.Wing.Walls” template. Schema Implementation 1 shows the object template file for “Abutment.Wing.Walls.” Schema Implementation 2 shows the object instance definition in the project file that refers to the “Abutment.Wing.Walls” object template for generating model view of the abutment (Figure 4-147). In the project file under the object instance definition, Sta, TOff and Elev key words can be used to set the location and rotation of the abutment object in the overall bridge project.



Project File

<Obj>

Name: *Test_Abutment*
Template: *Abutment.Wing.Walls*

Parameters:
1. *l_w*
2. *h_fWall*
3. *t_fWall*
4. *h_bWall*
5. *t_bWall*
6. *l_w1Wall*
7. *h_w1Wall*
8. *t_w1Wall*
9. *angle_w1Wall*
10. *s_w1Wall*
11. *l_w2Wall*
12. *h_w2Wall*
13. *t_w2Wall*
14. *angle_w2Wall*
15. *s_w2Wall*
16. *l_w1Footing*
17. *d_w1Footing*
18. *a_w1Footing*
19. *b_w1Footing*
20. *l_w1Footing*
21. *d_w1Footing*
22. *a_w1Footing*
23. *b_w1Footing*
24. *l_w2Footing*
25. *d_w2Footing*
26. *a_w2Footing*
27. *b_w2Footing*
28. *Material*

Template File

<ObjType>

Name: *Abutment.Wing.Walls*
Geometry Type: *Extruded solid and B-Rep*

Parameters:
1. *l_w*
2. *h_fWall*
3. *t_fWall*
4. *h_bWall*
5. *t_bWall*
6. *l_w1Wall*
7. *h_w1Wall*
8. *t_w1Wall*
9. *angle_w1Wall*
10. *s_w1Wall*
11. *l_w2Wall*
12. *h_w2Wall*
13. *t_w2Wall*
14. *angle_w2Wall*
15. *s_w2Wall*
16. *l_w1Footing*
17. *d_w1Footing*
18. *a_w1Footing*
19. *b_w1Footing*
20. *l_w1Footing*
21. *d_w1Footing*
22. *a_w1Footing*
23. *b_w1Footing*
24. *l_w2Footing*
25. *d_w2Footing*
26. *a_w2Footing*
27. *b_w2Footing*
28. *Material*

Figure 4-146. Rendering of "Test_Abutment_1" Model

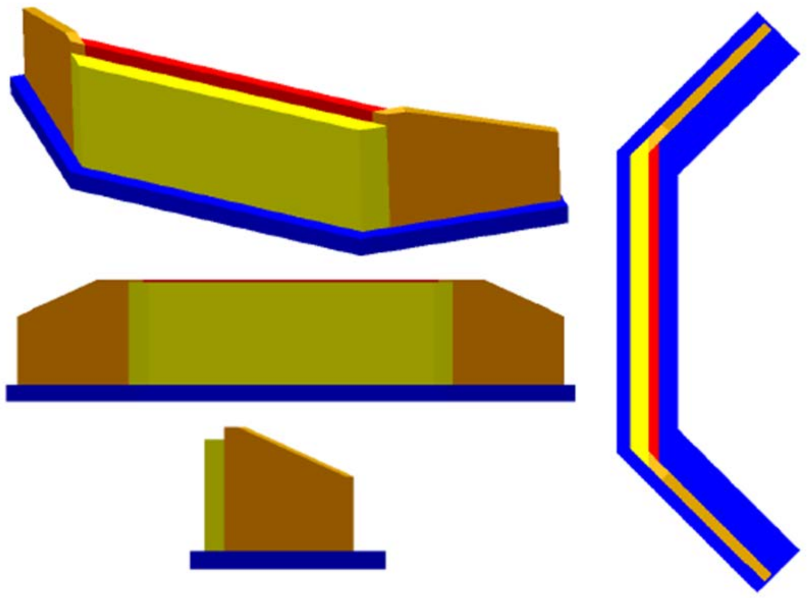


Figure 4-147. Rendering of Abutment with Wing Walls in OpenBrIM Viewer

Schema Implementation 5: Object Template for Abutment with Wing Walls

```
  <ObjType Name="Abutment.Wing.Walls" Label="Abutment with Wing Walls"
                Category="Abutments">
//Define Params using <Param>
<Param Name="material" Label="Material Properties" Value="NULL" Type="Material"/>
<!--Face Wall-->
<Param Name="h_fWall" Label="Height of face wall" Value="160" Type="Length"/>
<Param Name="l_w" Label="Length" Value="160" Type="Length"/>
 <Param Name="t_fWall" Label="Thickness of face wall" Value="7.0"
          Type="Length"/>
<!--Back Wall-->
     <Param Name="h_bWall" Label="Height of back wall" Value="160" Type="Length"/>
     <Param Name="t_bWall" Label="Thickness of back wall" Value="7.0"
     Type="Length"/>
<!--Wing Wall-1-->
<Param Name="h_w1Wall" Label="Height of back wall" Value="160" Type="Length"/>

<Param Name="l_w1Wall" Label="Length" Value="160" Type="Length"/>
<Param Name="s_w1Wall" Label="Length" Value="160" Type="Length"/>
<Param Name="t_w1Wall" Label="Thickness" Value="7.0" Type="Length"/>
<Param Name="angle_w1Wall" Label="Angle of wall-1" Value="7.0" Type="Angle"/>
<!--Wing Wall-2-->
<Param Name="h_w2Wall" Label="Height of back wall" Value="160" Type="Length"/>
 <Param Name="l_w2Wall" Label="Length" Value="160" Type="Length"/>
<Param Name="s_w2Wall" Label="Length" Value="160" Type="Length"/>
<Param Name="t_w2Wall" Label="Thickness of back wall" Value="7.0"
          Type="Length"/>
<Param Name="angle_w2Wall" Label="Angle of wall-2" Value="7.0" Type="Angle"/>
<!--Footing face/back wall-->
<Param Name="d_fbFooting" Label="depth" Value="160" Type="Length"/>
<Param Name="l_fbFooting" Label="Length" Value="160" Type="Length"/>
<Param Name="a_fbFooting" Label="width a" Value="7.0" Type="Length"/>
<Param Name="b_fbFooting" Label="width b" Value="7.0" Type="Length"/>
<!--Footing wing wall-1-->
<Param Name="d_w1Footing" Label="depth" Value="160" Type="Length"/>
 <Param Name="l_w1Footing" Label="Length" Value="160" Type="Length"/>
<Param Name="a_w1Footing" Label="width a" Value="7.0" Type="Length"/>
<Param Name="b_w1Footing" Label="width b" Value="7.0" Type="Length"/>
<!--Footing wing wall-2-->
<Param Name="d_w2Footing" Label="depth" Value="160" Type="Length"/>
<Param Name="l_w2Footing" Label="Length" Value="160" Type="Length"/>
<Param Name="a_w2Footing" Label="width a" Value="7.0" Type="Length"/>
 <Param Name="b_w2Footing" Label="width b" Value="7.0" Type="Length"/>
<!--Face Wall-->
<Line Color="yellow" Type="Linear">
```

```
<Orientation X="1" Y="0" Z="0" />
   //Define extrusion path using <Point>
<Point X="0" Y="0" Z="0"/>
<Point X="0" Y="0" Z="h_fWall"/>
//Define section to be extruded under <Polygon> using <Point>
  <Polygon>
  <Point X="-l_w/2" Y="t_fWall/2"/>
  <Point X="-l_w/2+t_fWall*tan(angle_w1Wall)" Y="-t_fWall/2"/>
  <Point X="l_w/2+t_fWall*tan(angle_w2Wall)" Y="-t_fWall/2"/>
  <Point X="l_w/2" Y="t_fWall/2"/>
  </Polygon>
  </Line>
<!--Back Wall-->
<Line Color="red" Type="Linear">
<Orientation X="1" Y="0" Z="0" />
   //Define extrusion path using <Point>
  <Point X="0" Y="0" Z="0"/>
  <Point X="0" Y="0" Z="h_bWall"/>
//Define section to be extruded under <Polygon> using <Point>
     <Polygon>
<Point X="-l_w/2+t_w1Wall/cos(angle_w1Wall)-t_w1Wall*tan(angle_w1Wall)"
        Y="t_fWall/2+t_bWall"/>
  <Point X="-l_w/2+t_w1Wall/cos(angle_w1Wall)" Y="t_fWall/2"/>
  <Point X="l_w/2-t_w1Wall/cos(angle_w2Wall)" Y="t_fWall/2"/>
  <Point X="l_w/2-t_w1Wall/cos(angle_w2Wall)-t_w1Wall*tan(angle_w2Wall)"
        Y="t_fWall/2+t_bWall"/>
 </Polygon>
</Line>
<!--Wing Wall-1-->
<Line Color="#ffc31f" Type="Linear">
  <Orientation X="1" Y="0" Z="0" />
   //Define extrusion path using <Point>
  <Point X="0" Y="0" Z="0"/>
  <Point X="0" Y="0" Z="h_bWall"/>
     //Define section to be extruded under <Polygon> using <Point>
  <Polygon>
     <Point X="-l_w/2" Y="t_fWall/2"/>
     <Point X="-l_w/2+t_w1Wall/cos(angle_w1Wall)" Y="t_fWall/2"/>
     <Point X="-l_w/2-s_w1Wall*sin(angle_w1Wall)+t_w1Wall*cos(angle_w1Wall)"
         Y="(t_fWall/2+s_w1Wall*cos(angle_w1Wall)+t_w1Wall*sin(angle_w1Wall))"/>
     <Point X="-l_w/2-s_w1Wall*sin(angle_w1Wall)"
      Y="(t_fWall/2+s_w1Wall*cos(angle_w1Wall))"/>
  </Polygon>
</Line>
//Define B-rep solid by defining <surface>(atleast two) using <Point> under <Volume>
<Volume Color="#ffc31f">
```

```
  <Surface Color="#ffc31f">
    <Point Y="-l_w/2-s_w1Wall*sin(angle_w1Wall)+t_w1Wall*cos(angle_w1Wall)"
        X="(t_fWall/2+s_w1Wall*cos(angle_w1Wall)
          +t_w1Wall*sin(angle_w1Wall))" Z="0"/>
    <Point Y="-l_w/2-l_w1Wall*sin(angle_w1Wall)+t_w1Wall*cos(angle_w1Wall)"
        X="(t_fWall/2+l_w1Wall*cos(angle_w1Wall)
          +t_w1Wall*sin(angle_w1Wall))" Z="0"/>
    <Point Y="-l_w/2-l_w1Wall*sin(angle_w1Wall)+t_w1Wall*cos(angle_w1Wall)"
       X="(t_fWall/2+l_w1Wall*cos(angle_w1Wall)
        +t_w1Wall*sin(angle_w1Wall))" Z="h_w1Wall"/>
    <Point Y="-l_w/2-s_w1Wall*sin(angle_w1Wall)+t_w1Wall*cos(angle_w1Wall)"
        X="(t_fWall/2+s_w1Wall*cos(angle_w1Wall)
          +t_w1Wall*sin(angle_w1Wall))" Z="h_bWall"/>
  </Surface>
  <Surface Color="#ffc31f">
     <Point Y="-l_w/2-s_w1Wall*sin(angle_w1Wall)"
         X="(t_fWall/2+s_w1Wall*cos(angle_w1Wall))" Z="0"/>
     <Point Y="-l_w/2-l_w1Wall*sin(angle_w1Wall)"
          X="(t_fWall/2+l_w1Wall*cos(angle_w1Wall))" Z="0"/>
     <Point Y="-l_w/2-l_w1Wall*sin(angle_w1Wall)"
          X="(t_fWall/2+l_w1Wall*cos(angle_w1Wall))" Z="h_w1Wall"/>
     <Point Y="-l_w/2-s_w1Wall*sin(angle_w1Wall)"
          X="(t_fWall/2+s_w1Wall*cos(angle_w1Wall))" Z="h_bWall"/>
    </Surface>
  </Volume>
  <!--Wing Wall-2-->
  <Line Color="#ffc31f" Type="Linear">
    <Orientation X="1" Y="0" Z="0" />
     //Define extrusion path using <Point>
    <Point X="0" Y="0" Z="0"/>
    <Point X="0" Y="0" Z="h_bWall"/>
     //Define section to be extruded under <Polygon> using <Point>
    <Polygon>
      <Point X="l_w/2-t_w2Wall/cos(angle_w2Wall)" Y="t_fWall/2"/>
      <Point X="l_w/2" Y="t_fWall/2"/>
      <Point X="l_w/2-s_w2Wall*sin(angle_w2Wall)"
             Y="t_fWall/2+s_w2Wall*cos(angle_w2Wall)"/>
      <Point X="l_w/2-s_w2Wall*sin(angle_w2Wall)-t_w2Wall*cos(angle_w2Wall)"
             Y="t_fWall/2+s_w2Wall*cos(angle_w2Wall)-t_w2Wall*sin(angle_w2Wall)"/>
   </Polygon>
 </Line>
//Define B-rep solid by defining <surface>(atleast two) using <Point> under <Volume>
<Volume Color="#ffc31f">
  <Surface Color="blue">
  <Point Y="l_w/2-s_w2Wall*sin(angle_w2Wall)-
         t_w2Wall*cos(angle_w2Wall)"
        X="(t_fWall/2+s_w2Wall*cos(angle_w2Wall) -
        t_w2Wall*sin(angle_w2Wall))" Z=“0”/>
  <Point Y="l_w/2-l_w2Wall*sin(angle_w2Wall)-
        t_w2Wall*cos(angle_w2Wall)"
        X="(t_fWall/2+l_w2Wall*cos(angle_w2Wall) -
        t_w2Wall*sin(angle_w2Wall))" Z=“0”/>
  <Point Y="l_w/2-l_w2Wall*sin(angle_w2Wall)-
        t_w2Wall*cos(angle_w2Wall)"
        X="(t_fWall/2+l_w2Wall*cos(angle_w2Wall) -
        t_w2Wall*sin(angle_w2Wall))" Z="h_w2Wall"/>
  <Point Y="l_w/2-s_w2Wall*sin(angle_w2Wall)-
        t_w2Wall*cos(angle_w2Wall)"
        X="(t_fWall/2+s_w2Wall*cos(angle_w2Wall)
          -t_w2Wall*sin(angle_w2Wall))" Z="h_bWall"/>
 </Surface>
  <Surface Color="#ffc31f">
  <Point Y="l_w/2-s_w2Wall*sin(angle_w2Wall)" X="(t_fWall/2+s_w2Wall*cos(angle_w2Wall))"
        Z=“0”/>
   <Point Y="l_w/2-l_w2Wall*sin(angle_w2Wall)" X="(t_fWall/2+l_w2Wall*cos(angle_w2Wall))"
        Z=“0”/>
  <Point Y="l_w/2-l_w2Wall*sin(angle_w2Wall)" X="(t_fWall/2+l_w2Wall*cos(angle_w2Wall))"
        Z="h_w2Wall"/>
   <Point Y="l_w/2-s_w2Wall*sin(angle_w2Wall)" X="(t_fWall/2+s_w2Wall*cos(angle_w2Wall))"
        Z="h_bWall"/> </Surface> </Volume>
```

<!--Footing-Face/back-->

```
<Line Color="blue" Type="Linear"> <Orientation X="1" Y="0" Z="0" />
```

//Define extrusion path using <Point>

```
<Point X=“0” Y=“0” Z=“0”/> <Point X=“0” Y=“0” Z="-d_fbFooting"/>
```

//Define section to be extruded under <Polygon> using <Point>

```
  <Polygon>
   <Point X="-l_fbFooting/2+(a_fbFooting+b_fbFooting)/tan(angle_w1Wall/2+0.7855)"
       Y="b_fbFooting"/>
    <Point X="-l_fbFooting/2" Y="-a_fbFooting"/>
    <Point X="l_fbFooting/2" Y="-a_fbFooting"/>
   <Point X="l_fbFooting/2+(a_fbFooting+b_fbFooting)/tan(angle_w2Wall/2-0.7855)"
       Y="b_fbFooting"/>
   </Polygon>
</Line>
```

<!--Footing-wing-1-->

```
<Line Color="blue" Type="Linear">
<Orientation X="1" Y="0" Z="0" />
```

//Define extrusion path using <Point>

```
<Point X=“0” Y=“0” Z=“0”/>
<Point X=“0” Y=“0” Z="-d_w1Footing"/>
```

//Define section to be extruded under <Polygon> using <Point>

```
  <Polygon>
```

```
    <Point X="-l_fbFooting/2" Y="-a_fbFooting"/>
    <Point X="-l_fbFooting/2+(a_fbFooting+b_fbFooting)/tan(angle_w1Wall/2+0.7855)"
          Y="b_fbFooting"/>
     <Point X="-l_fbFooting/2-l_w1Footing*sin(angle_w1Wall)
              +(a_w1Footing+b_w1Footing)*cos(angle_w1Wall)"
          Y="-a_fbFooting+l_w1Footing*cos(angle_w1Wall)
              +(a_w1Footing+b_w1Footing)*sin(angle_w1Wall)"/>
     <Point X="-l_fbFooting/2-l_w1Footing*sin(angle_w1Wall)"
          Y="-a_fbFooting+l_w1Footing*cos(angle_w1Wall)"/>
     </Polygon>
  </Line>
 <!--Footing-wing-2-->
 <Line Color="blue" Type="Linear">
   <Orientation X="1" Y="0" Z="0" />
       //Define extrusion path using <Point>
      <Point X="0" Y="0" Z="0"/>
      <Point X="0" Y="0" Z="-d_w2Footing"/>
       //Define section to be extruded under <Polygon> using <Point>
    <Polygon>
      <Point X="l_fbFooting/2" Y="-a_fbFooting"/>
       <Point X="l_fbFooting/2+(a_fbFooting+b_fbFooting)/tan(angle_w2Wall/2-0.7855)"
            Y="b_fbFooting"/>
      <Point X="l_fbFooting/2-l_w2Footing*sin(angle_w2Wall) -
             (a_w2Footing+b_w2Footing)*cos(angle_w2Wall)"
         Y="-a_fbFooting+l_w2Footing*cos(angle_w2Wall)-(a_w2Footing+b_w2Footing)
            *sin(angle_w2Wall)"/>
      <Point X="l_fbFooting/2-l_w2Footing*sin(angle_w2Wall)" Y="-
             a_fbFooting+l_w2Footing*cos(angle_w2Wall)"/>
     </Polygon>
   </Line>
 </ObjType>
```

Schema Implementation 6: Object Instance for Abutment with Wing Walls

```
//Assign placement location and orientation using "Sta", "TOff", "Elev", "RX", "RY", "RZ"
<Obj Name="Test_Abutment" RefObj="Abutment.Wing.Walls" Sta="0" TOff="0" Elev="0" RX="0"
RY="0" RZ="0">
    //Define and assign values to parameters corresponding to template parameters using <Param>
     <Param Name="material" Value="C60"/>
    <!--Face Wall-->
     <Param Name="l_w" Value="14176"/>
     <Param Name="h_fWall" Value="4500"/>
     <Param Name="t_fWall" Value="800"/>
    <!--Back Wall-->
     <Param Name="h_bWall" Value="5000"/>
     <Param Name="t_bWall" Value="450"/>
    <!--Wing Wall-1-->
    <Param Name="l_w1Wall" Value="6986"/>
    <Param Name="h_w1Wall" Value="3000"/>
    <Param Name="t_w1Wall" Value="450"/>
    <Param Name="angle_w1Wall" Value="0.785"/>
    <Param Name="s_w1Wall" Value="786"/>
    <!--Wing Wall-2-->
    <Param Name="l_w2Wall" Value="6986"/>
    <Param Name="h_w2Wall" Value="3000"/> <Param Name="t_w2Wall" Value="450"/>
    <Param Name="angle_w2Wall" Value="-0.785"/> <Param Name="s_w2Wall" Value="786"/>
    <!--Footing face/back wall-->
    <Param Name="d_fbFooting" Value="750" Type="Length"/>
    <Param Name="l_fbFooting" Value="13019" Type="Length"/>
    <Param Name="a_fbFooting" Value="975" Type="Length"/>
    <Param Name="b_fbFooting" Value="1625" Type="Length"/>
    <!--Footing wing wall-1-->
    <Param Name="d_w1Footing" Value="750" Type="Length"/>
     <Param Name="l_w1Footing" Value="8500" Type="Length"/>
    <Param Name="a_w1Footing" Value="975" Type="Length"/>
    <Param Name="b_w1Footing" Value="1625" Type="Length"/>
    <!--Footing wing wall-2-->
    <Param Name="d_w2Footing" Value="750" Type="Length"/>
    <Param Name="l_w2Footing" Value="8500" Type="Length"/>
     <Param Name="a_w2Footing" Value="975" Type="Length"/>
     <Param Name="b_w2Footing" Value="1625" Type="Length"/>
</Obj>
```

4.11.2 Multi-column Piers

Multi-Column piers can be defined as an assembly of extruded solids. In this example template, five RC columns, five caissons/shafts and a cap beam are considered as assembly parts for this multi-column pier. Rebar definition is not included in this example.

Parameters required for defining the multi-column pier are shown in Figure 4-148. Following parameters are required for definining the multi-column pier.

- Column diameter, colDia Shaft diameter, shaftDia.
- Column length, colleen.
- Shaft length, shaftLen.
- Cap beam depth, capBmD.
- Cap beam width, capBmW.
- Cap beam length, capBmLen.
- Column/Shaft spacing, colSpc.
- Material Properties, material.

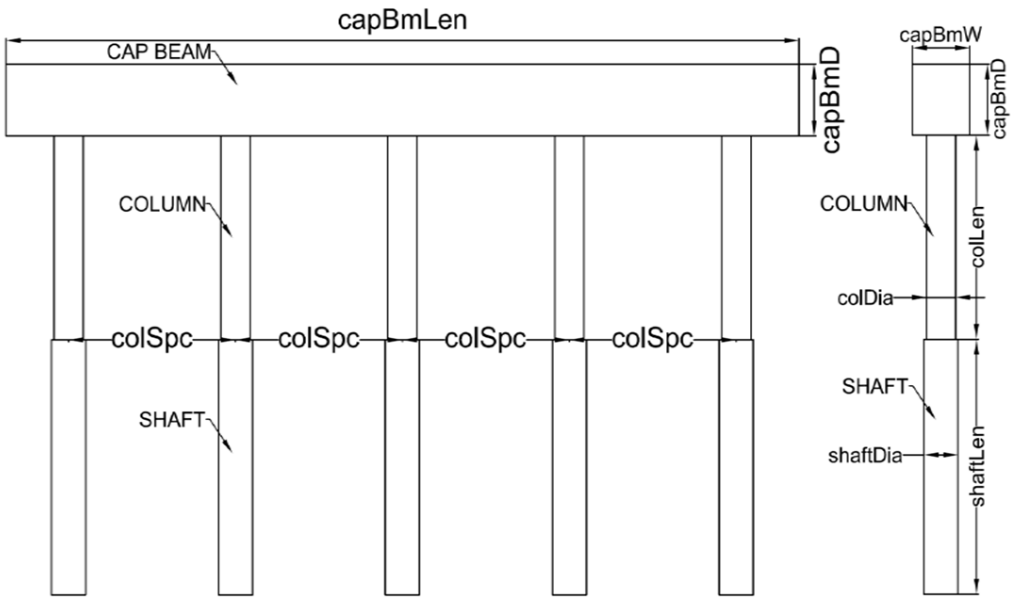


Figure 4-148. Parameters

Figures 4-149 and 4-150 show the rendering of "Test_MultiColumnPier_1" model generated using "Multi.Column.Pier" template. **Schema Implementation 7** shows the object template file for "Multi.Column.Pier." **Schema Implementation 8** shows the object instance definition in the project file that refers to the "Multi.Column.Pier" object template for generating model view of the pier. In the project file under the object instance definition, Sta, TOff and Elev key words can be used to set the location and rotation of the pier object in the overall bridge project.

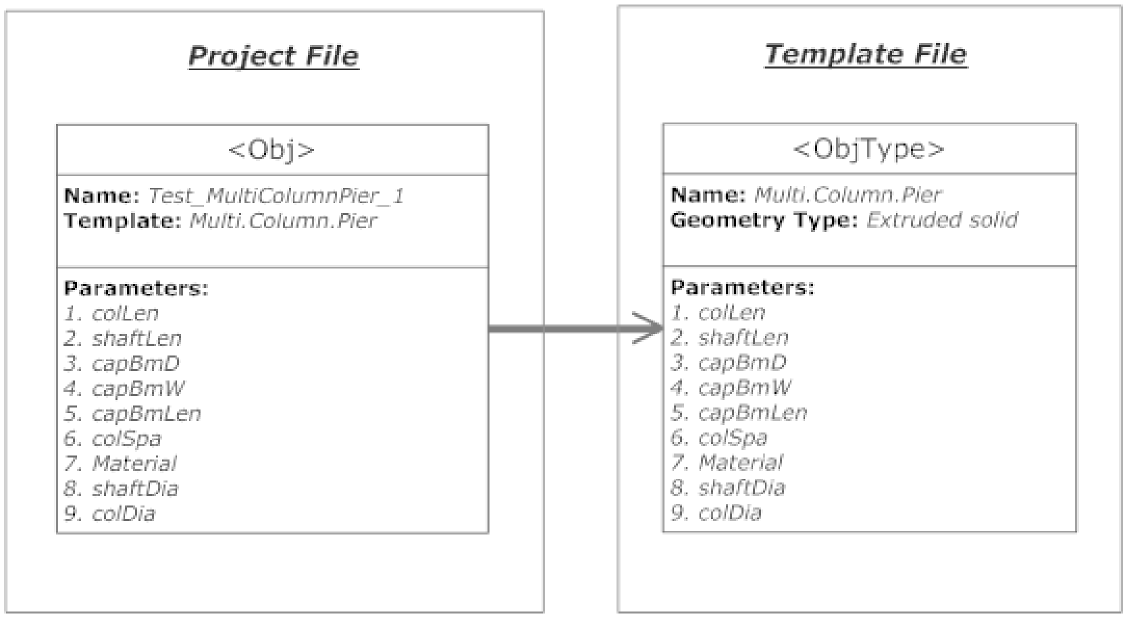


Figure 4-149. Block Diagram

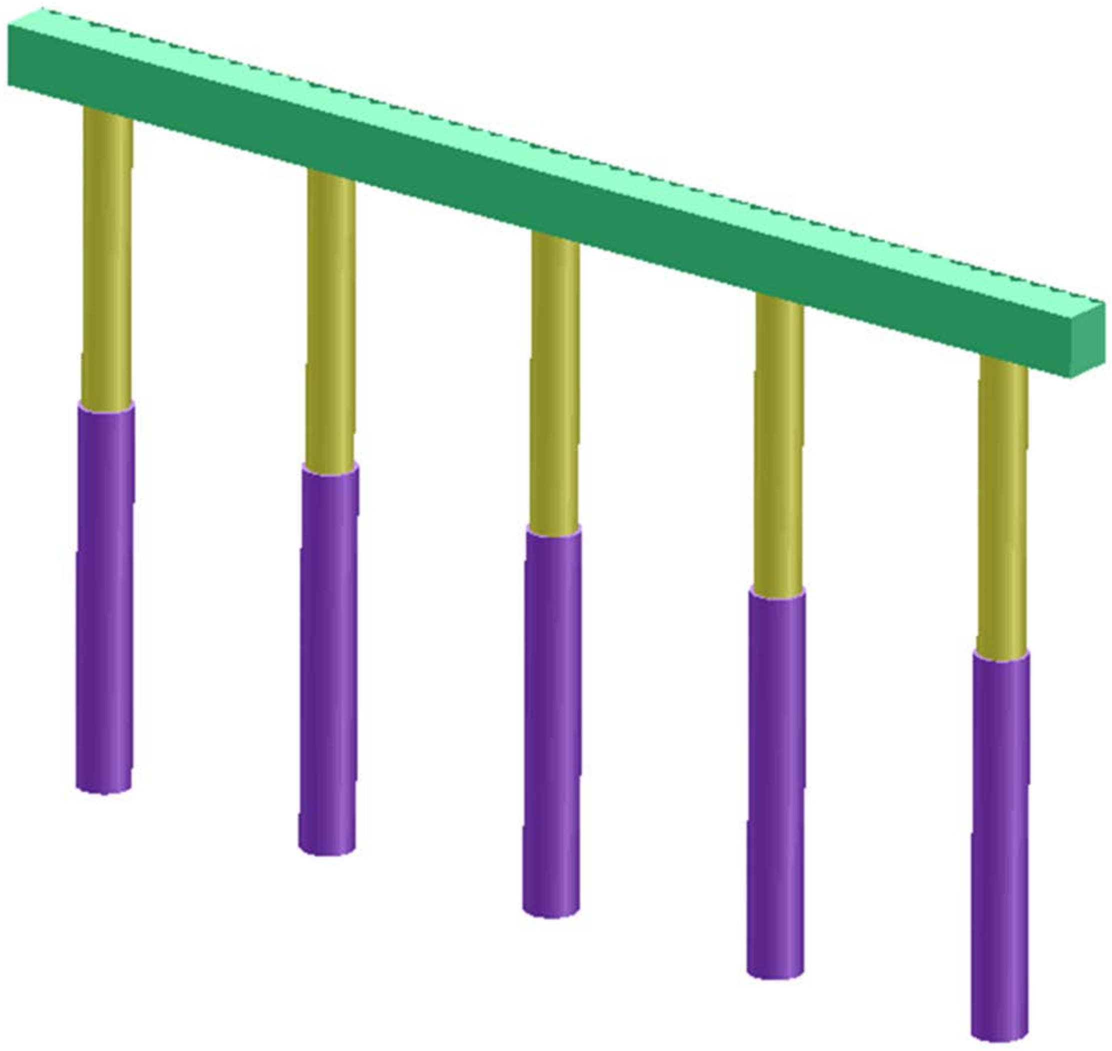


Figure 4-150. Rendering of Multi-Column Pier in OpenBrIM Viewer

Schema Implementation 7: Object Template for Multi-Column Pier with 5 Columns

```
<ObjType Name="Multi.Column.Pier" Label="Multi-column Pier" Category="Substructure">
      //Define parameters using <Param>
    <Param Name="capBmD" Label="Depth" Value="4" Type="Length" />
    <Param Name="capBmW" Label="width" Value="3.5" Type="Length" />
    <Param Name="capBmLen" Label="Cap Beam Length" Value="80" Type="Length" />
     <Param Name="colSpa" Label="Column Spacing" Value="17" Type="Length" />
    <Param Name="colDia" Label="Column Diameter" Value="3" Type="Length" />
    <Param Name="colLen" Label="Pier Column Length" Value="20" Type="Length" />
    <Param Name="shaftDia" Label="Shaft Diameter" Value="3.5" Type="Length" />
    <Param Name="shaftLen" Label="Drilled Shaft Height" Value="25" Type="Length" />
    <Param Name="numCol" Label="Number of Columns" Value="5" />
```

```
//Define cap beam geometry
  <Line Color="#99FFCC" Type="Linear">
    <Orientation X="0" Y="1" Z="0" />
    <Point X="0" Y="-capBmLen/2" Z="0" />
    <Point X="0" Y="capBmLen/2" Z="0" />
    <Polygon>
       <Point X="-capBmW/2" Y="capBmD/2" />
       <Point X="-capBmW/2" Y="-capBmD/2" />
       <Point X="capBmW/2" Y="-capBmD/2" />
       <Point X="capBmW/2" Y="capBmD/2" />
    </Polygon>
  </Line>
 //Define column geometry
  <Repeat Param="r" StartValue="-2" EndValue="2" Increment="1">
    <Line Color="#FFFF99" Type="Linear" Y="r*colSpa">
      <Orientation X="0" Y="0" Z="1" /> <Point X="0" Y="0" Z="-capBmD/2" />
      <Point X="0" Y="0" Z="-capBmD/2-colLen" />
      <Circle X="0" Y="0" Radius="colDia/2" />
    </Line>
   </Repeat>
 //Define shaft geometry
  <Repeat Param="r" StartValue="-2" EndValue="2" Increment="1">
    <Line Color="blue" Type="Linear" Y="r*colSpa">
     <Orientation X="0" Y="0" Z="1" />
     <Point X="0" Y="0" Z="-capBmD/2-colLen" />
    <Point X="0" Y="0" Z="-capBmD/2-colLen-shaftLen" />
     <Circle X="0" Y="0" Radius="shaftDia/2" />
    </Line>
  </Repeat>
</ObjType>
```

Schema Implementation 8: Object Instance for Multi-Column Pier with 5 Columns

```
   <Obj Name=" Test_MultiColumnPier_1" RefObj="Multi.Column.Pier" Sta="0" TOff="0"
Elev="0" RX="0" RY="0" RZ="0">
    <Param Name="capBmD" Value="4" />
    <Param Name="capBmW" Value="3.5" />
    <Param Name="capBmLen" Value="81.54" />
     <Param Name="colSpc" Value="17.166" />
    <Param Name="colLen" Value="20" />
    <Param Name="columnsec" Value="Circular Column Section" />
    <Param Name="shaftLen" Value="25" />
     <Param Name="shaftsec" Value="Circular Shaft Section" />
     <Param Name="material" Value="C60"/>
</Obj>
```

4.12 Parametric Dependency

Parametric dependency is one of the basis of the composition of bridge member assemblies. Besides modeling the bridge components individually, relationships among them should also be identified to form the whole bridge models. The model of the case study bridges consists of multiple linear members and surface members. The locations of the members are defined with member lines and placement points. Therefore, the members' relative locations can be defined by specifying the dependency of the member lines and the placement points.

The coordinates for bridge models are defined in the Section 4.1.1 and shown in Figure 4-151:

- Longitudinal axis: As defined along the highway alignment, longitudinal direction
- Transverse axis: Transverse direction which is perpendicular to longitudinal direction
- Vertical axis: Vertical direction

The control member line of the deck, which is the highway alignment line, is located at the top center of the deck, as shown in Figure 4-151. Angle A1 and angle A2 can be unequal, as they are in the case study bridge.

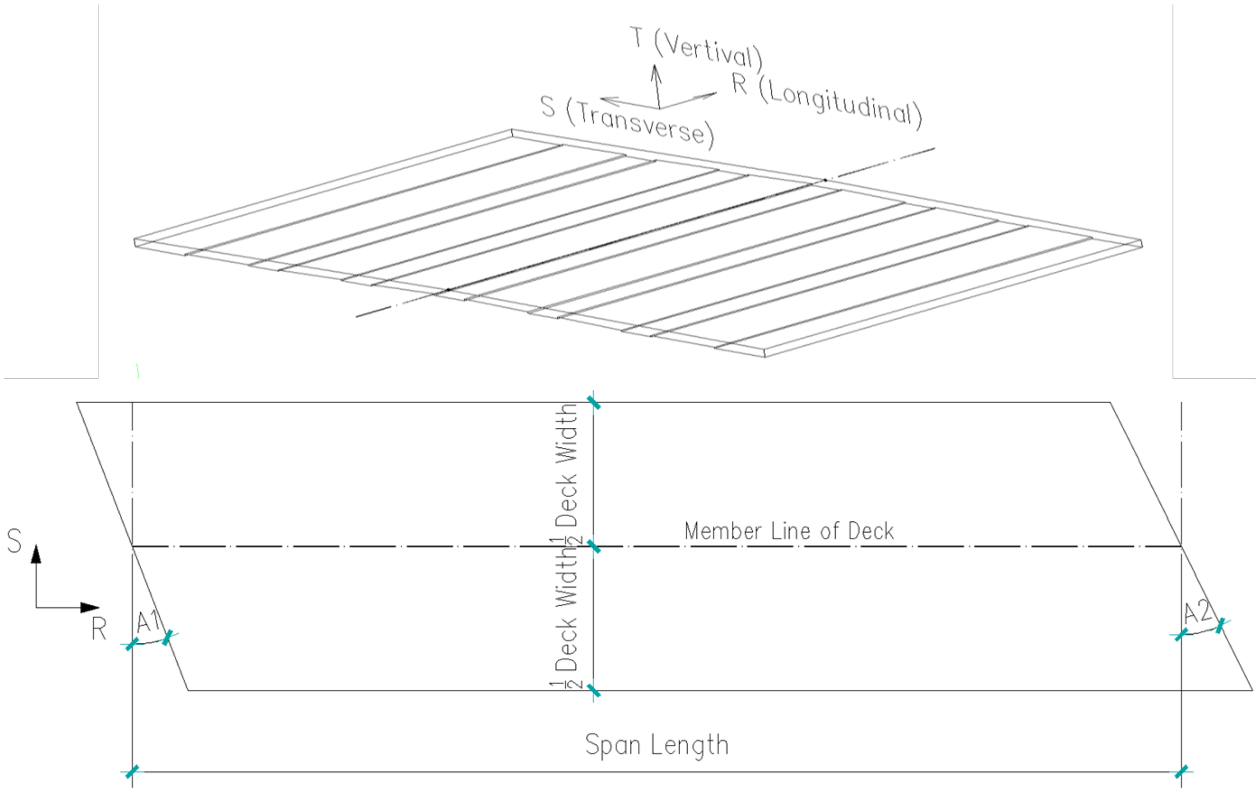


Figure 4-151. Member Line of the Deck and the Coordinates for Bridge Models

The relationship between the deck and the parapets is shown in Figure 4-152.

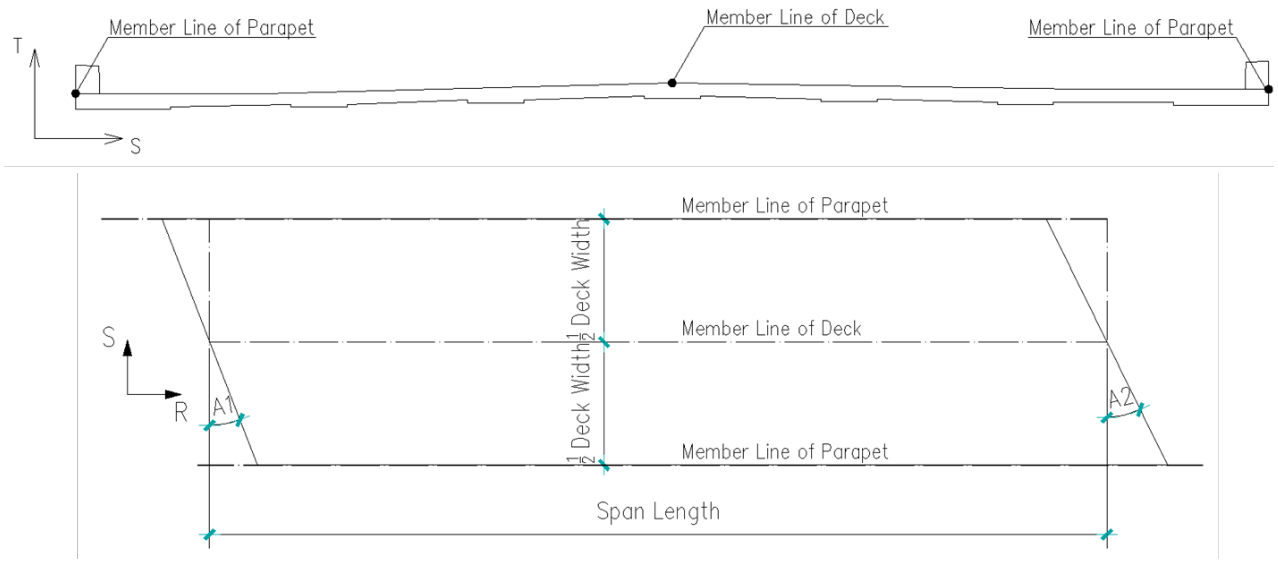


Figure 4-152. Relationship between the Deck and the Parapets

The relationship between the deck, the parapets and the sidewalks is shown in Figure 4-153.

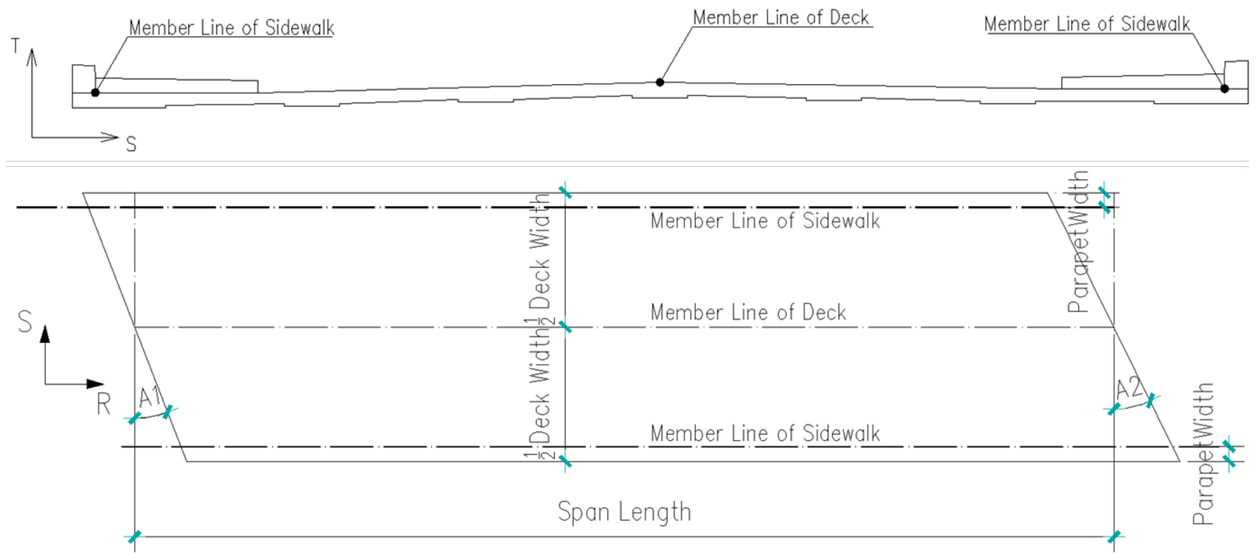


Figure 4-153. Relationship between the Deck, the Parapets and the Sidewalks

The relationship between the deck and the girders is shown in Figure 4-154.

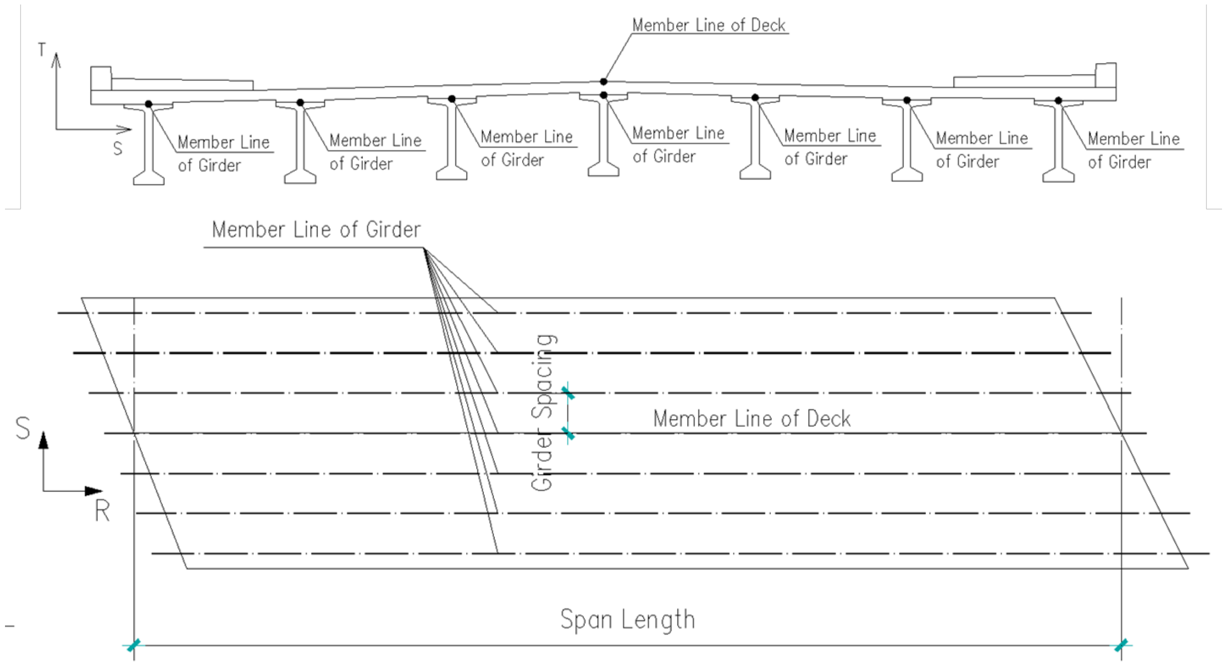


Figure 4-154. Relationship between the Deck and the Girders

The relationship between the deck, the girders and the cap beams is shown in Figure 4-155.

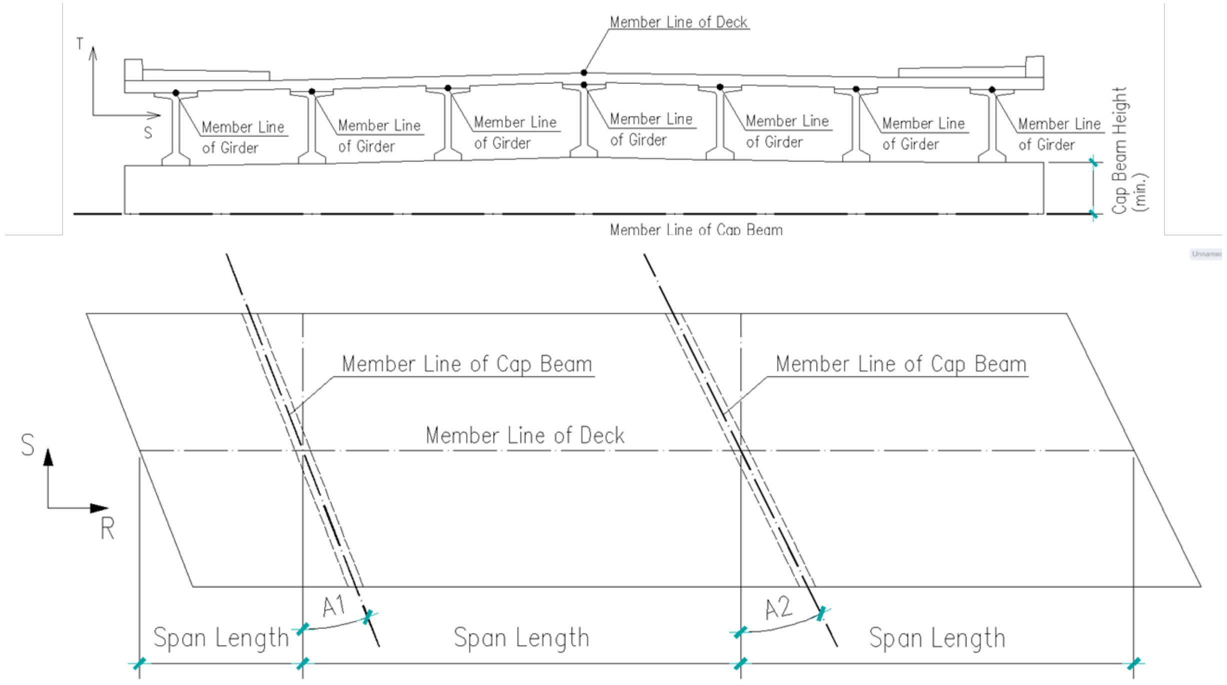


Figure 4-155. Relationship between the Deck, the Girders and the Cap Beam

The relationship between the superstructure and the columns is shown in Figure 4-156.

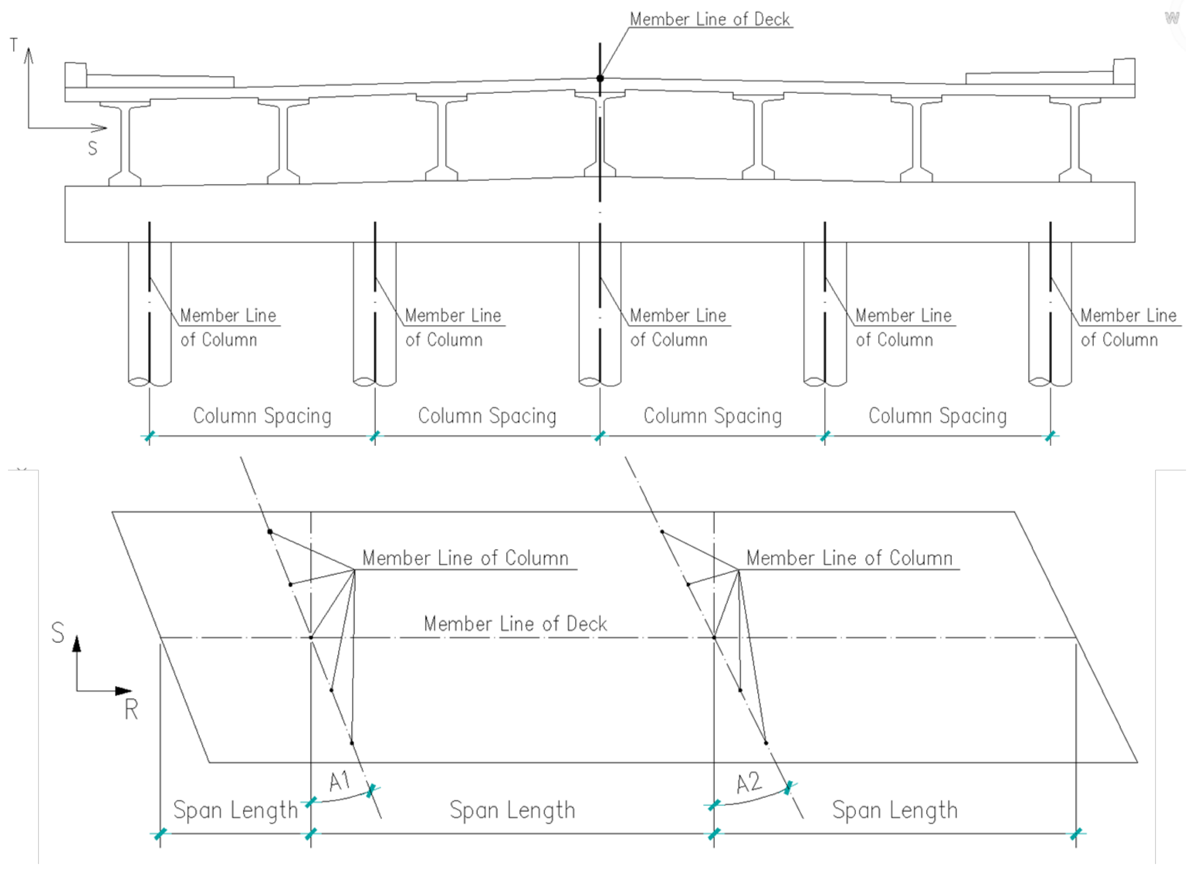


Figure 4-156. Relationship between the Superstructure and the Columns

The relationship between the superstructure and the drilled shafts is shown in Figure 4-157.

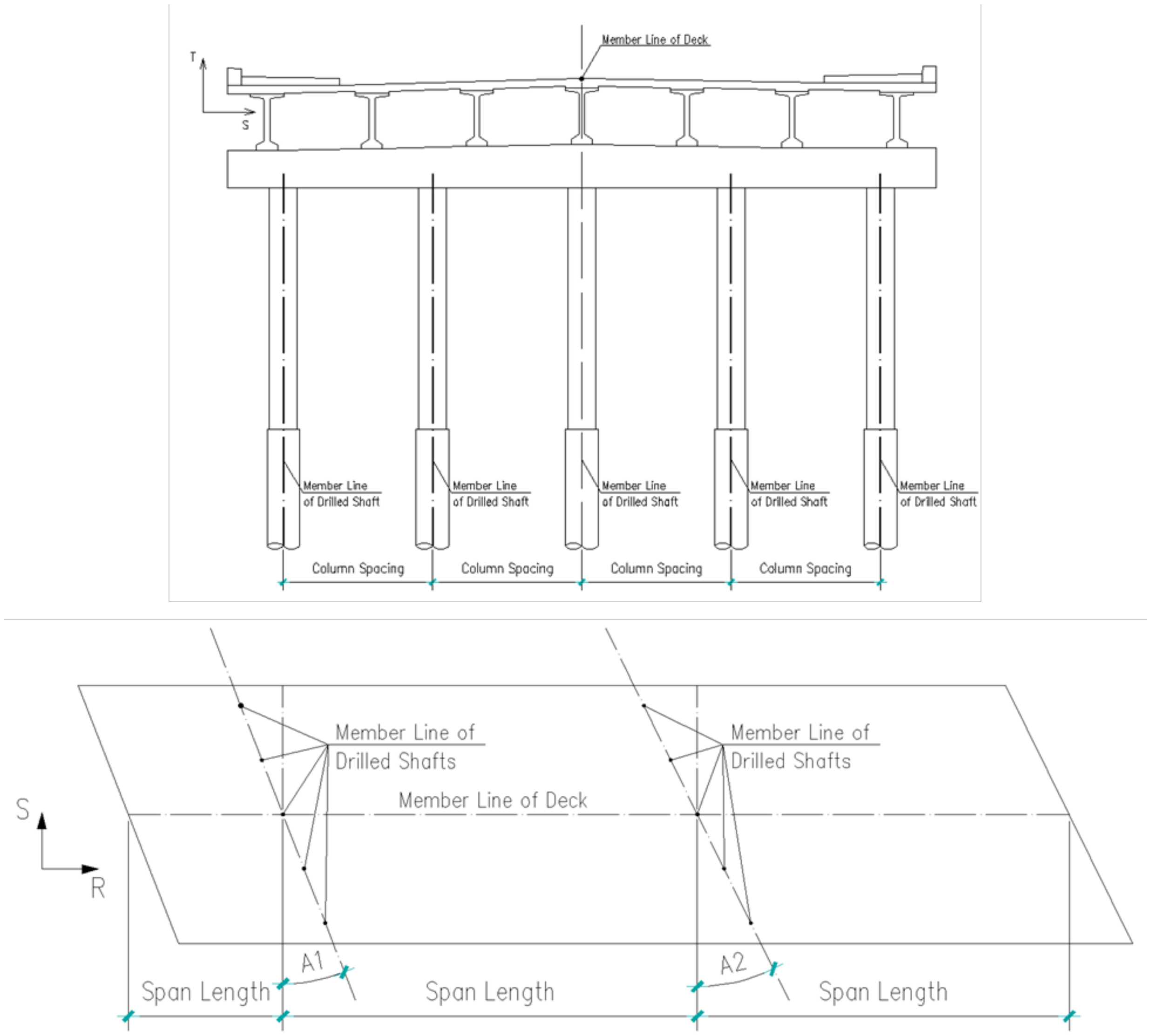
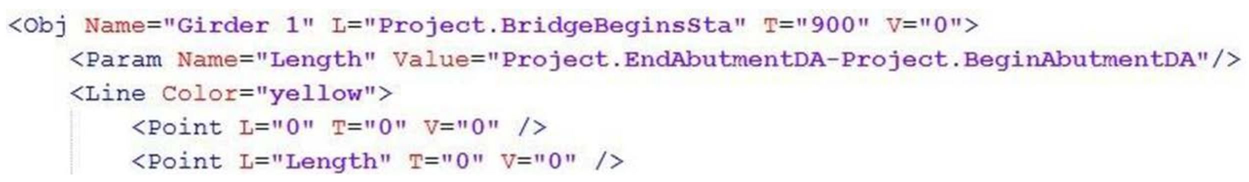

Member Line of Deck
T
S
Member Line of Drilled Shaft
Member Line of Drilled Shaft
Member Line of Drilled Shaft
Member Line of Drilled Shaft
Member Line of Drilled Shaft
Column Spacing
Column Spacing
Column Spacing
Column Spacing
Member Line of Drilled Shafts
Member Line of Drilled Shafts
Member Line of Deck
S
R
A1
A2
Span Length
Span Length
Span Length

Figure 4-157. Relationship between the Superstructure and the Drilled Shafts

The major geometric data in this exchange model consists of the geometries of bridge components, e.g., girders, stiffeners, cross frames, etc. These can be defined dependent on the roadway alignment described in Section 4.1. The alignment and bridge control information (ProjectParameters) including stations and skew angles at supports (i.e., abutments and piers) are specified in Section 4.1.4.

Figure 4-158 demonstrates the XML code for defining girder layout line. Because in most cases girders are parallel to roadway alignment, girder layout line is defined by specifying the station at where the girder begins (L=Project.BridgeBeginsSta), the transverse distance between girder layout line and roadway alignment (T=900), and length of girder (Length). Elevation of girder layout line is determined based on vertical profile and cross section. The Girder model can then be expressed by assigning girder cross section along the girder layout line. Figure 4-158 shows the underlying schema for the viewer-generated model of the girders shown in Figure 4-159.



```
<Obj Name="Girder 1" L="Project.BridgeBeginsSta" T="900" V="0">
    <Param Name="Length" Value="Project.EndAbutmentDA-Project.BeginAbutmentDA"/>
    <Line Color="yellow">
        <Point L="0" T="0" V="0" />
        <Point L="Length" T="0" V="0" />
```

Figure 4-158. XML Code for Girder Layout Line

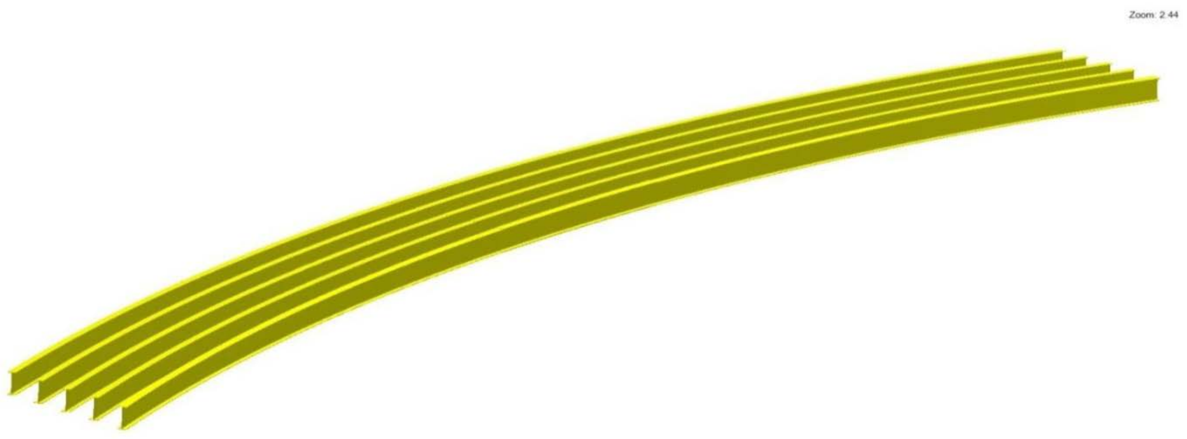
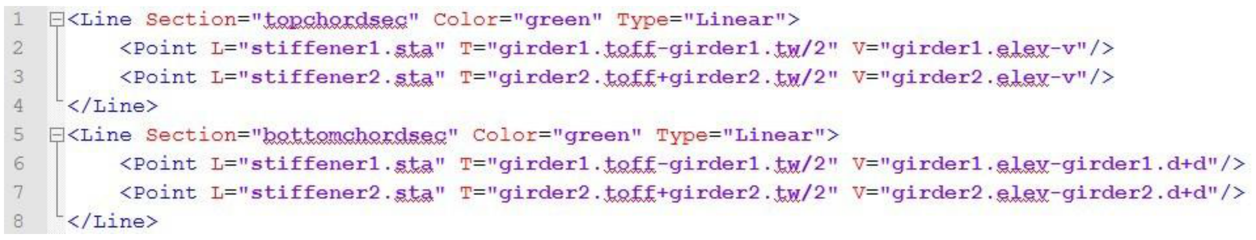


Figure 4-159. Model of Girders

Some other components in the bridge superstructure, such as stiffeners, can be treated as girder accessories. Their geometries can be defined based on girder geometry: parametric formulae which consist of dimensional parameters of girders and arithmetic operators are used to define their location and geometry, as shown in line 2, 3, 6, and 7 in Figure 4-160. This parametric dependent definition ensures that if one driving dimensional parameter changes, the driven expressions will be modified, as well as the entire model.



```
1  <Line Section="topchordsec" Color="green" Type="Linear">
2      <Point L="stiffener1.sta" T="girder1.toff-girder1.tw/2" V="girder1.elev-v"/>
3      <Point L="stiffener2.sta" T="girder2.toff+girder2.tw/2" V="girder2.elev-v"/>
4  </Line>
5  <Line Section="bottomchordsec" Color="green" Type="Linear">
6      <Point L="stiffener1.sta" T="girder1.toff-girder1.tw/2" V="girder1.elev-girder1.d+d"/>
7      <Point L="stiffener2.sta" T="girder2.toff+girder2.tw/2" V="girder2.elev-girder2.d+d"/>
8  </Line>
```

Figure 4-160. XML Code for Location and Geometry of Cross Frame

For example, the left part of Figure 4-161 shows two steel girders with a particular spacing. If that spacing is increased, changes occur not only in the girder locations, but also in the location of the stiffeners, and the location and geometry of cross frames, as shown in the right part of Figure 4-161.

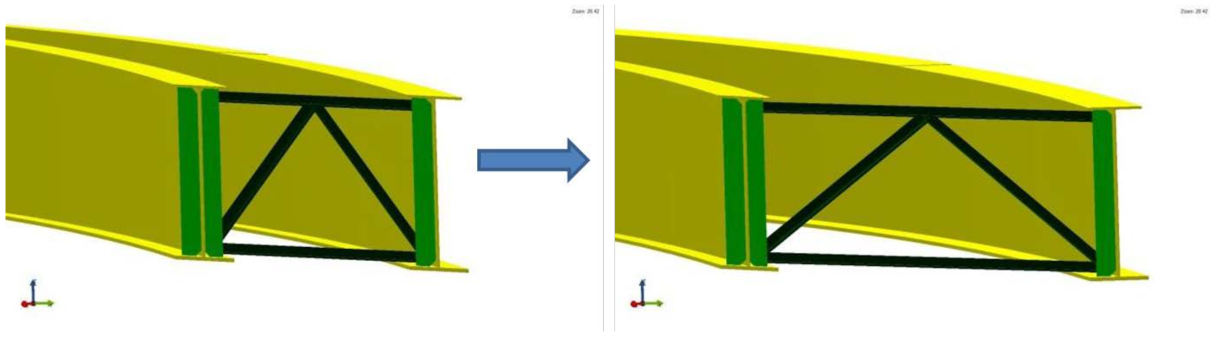


Figure 4-161. Automatic Update of Parametric Model

4.13 Semantic Exchange Modules

4.13.1 Introduction to Semantic Exchange Module

Current methodologies for developing Model View Definitions (MVD) operate in static and non-reusable manner. Therefore, duplicate work needs to be done by different companies in preparing translators between data schema and their software applications. This makes the translator development time-consuming and expensive for both creation and testing.

This situation is partially caused by the fact that data schemas, e.g., IFC and OpenBrIM schema, are semantically rich and flexible in order to satisfy a wide spectrum of demands in data exchange. Thereby, effective exchanges require a layer of specificity over the top of these schemas. Once a certain level of exchange functionality has been defined, it should be reusable as a module. This module enables users to select and specify the appropriate information entities, attributes and rules that govern their possible values for particular uses based on SEMs without worrying about the underlying data schema. This objective is potentially fulfilled by using Semantic Exchange Modules (SEM) proposed by Venugopal et al. [33], [34], [35].

Venugopal et al. define SEM as "a structured, modular subset of the objects and relationships in each of the multiple BIM exchange model definitions." SEM has two usages. First, it enables software companies to code translators in modular fashion, such that a function written to export or import model objects according to any given SEM can be tested and certified once, and then re-used to fulfill multiple exchange models without modification. Second, it allows end users who are not IT savvy to compose a MVD at run-time by defining it in terms of SEMs.

4.13.2 Structure of Semantic Exchange Module

The structure of SEM proposed by Venugopal et al. contains "a binding to a set of IFC entities, attributes, relations, and functions and a corresponding binding to a set of native model structures that carry the information associated with the IFC SEM definition". However, this structure needs to be modified for BrIM because 1) the data exchange standard for BrIM is based on OpenBrIM XSD instead of IFC, and 2) IDM should be involved in the SEM in order to make it more user friendly to end users who are not IT savvy.

Figure 4-162 shows the revised SEM structure proposed for this study. A SEM should carry a binding to a set of information items and attributes defined in the Data Dictionary, a binding to a set of objects and parameters defined in OpenBrIM schema, and a corresponding binding to a set of native model structures created by the commercial software.

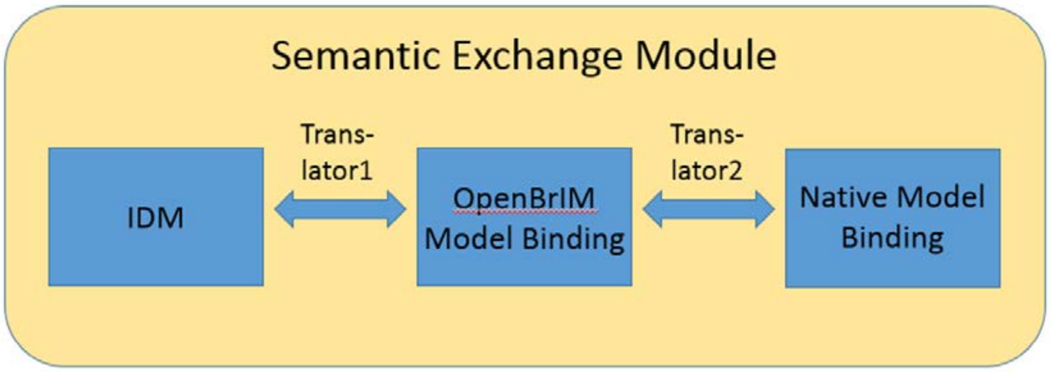


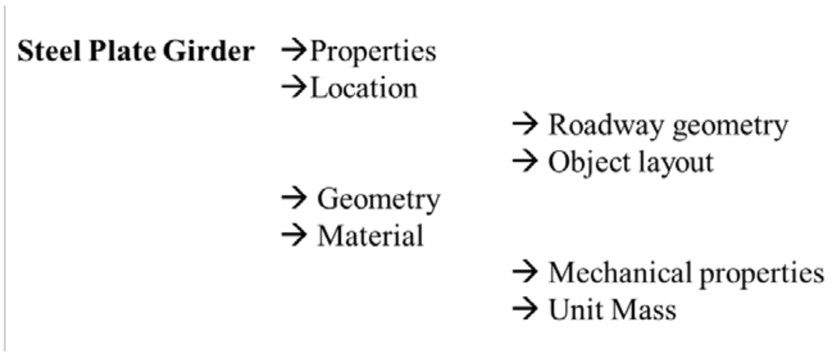
Figure 4-162. Modified Structure of Semantic Exchange Module

SEMs are defined in the form of modules binding together entities addressing a similar aspect. For example, geometry is a set of SEMs, and material is another set of SEMs.

4.13.3 Process of Creating SEM and Test Case

The process of generating a model view exchange using SEMs can be demonstrated by using a test case, exchange of information of steel plate girder for use case 9 (final bridge design).

Step 1: the process begins by bridge engineer selects SEMs from the Data Dictionary for steel plate girder. Because this exchange model will be used for initial load rating and engineer's estimate, the selection should at least contain the following items.



Steel Plate Girder →Properties
- →Location
  - → Roadway geometry
  - → Object layout
- → Geometry
- → Material
  - → Mechanical properties
  - → Unit Mass

Table 4-10 to Table 4-12 show the Bridge Layout SEM, Material SEM, and Plate Girder. SEM for modeling steel plate girder for initial load rating and engineer's estimate.

Table 4-10. Bridge Layout SEM

Information Groups	Information Items	Attribute Sets	Attributes	Value
Project information	Identification	Over roadway identities	State highway name and number	I-290 Ramp B Alg
Roadway geometry	Horizontal alignment	Types	Tangent	0
Roadway geometry	Horizontal alignment	Types	Circular	1
Roadway geometry	Horizontal alignment	Types	Spiral	2
Roadway geometry	Horizontal alignment	Lines	Horizontal Control Line (HCL)	TRUE
Roadway geometry	Horizontal alignment	Stations	Tangent to spiral point (T.S.)	1049139
Roadway geometry	Horizontal alignment	Azimuth	Azimuth	4.2195
Roadway geometry	Horizontal alignment	Directions	Direction	Left
Roadway geometry	Horizontal alignment	Lengths	Length of circular curve, Lc	367888
Roadway geometry	Horizontal alignment	Lengths	Length of spiral, Ls	63000
Roadway geometry	Horizontal alignment	Radii	Radius	230000
Roadway geometry	Vertical Profile	Types	Tangent	2
Roadway geometry	Vertical Profile	Types	Parabolic	1
Roadway geometry	Vertical Profile	Lines	Profile grade line (PGL)	TRUE
Roadway geometry	Vertical Profile	Stations	Point of vertical curvature (PVC)	1139000
Roadway geometry	Vertical Profile	Stations	Point of vertical Tangency (PVT)	1369683
Roadway geometry	Vertical Profile	Elevations	Elevation at PVC	192700
Roadway geometry	Vertical Profile	Elevations	Elevation at PVT	191530
Roadway geometry	Vertical Profile	Grades	G1	0.05
Roadway geometry	Vertical Profile	Grades	G2	-0.04908
Roadway geometry	Cross section	Geometries	Left edge to H.C.L./T.G.L.	-5557
Roadway geometry	Cross section	Stations	Station	1200649
Roadway geometry	Cross section	Slopes	Slope	-0.08
Roadway geometry	Cross section	Slopes	Slope	0.02
Roadway geometry	Cross section	Widths	Segment width	11257
Roadway geometry	Cross section	Widths	Segment width	1657

Table 4-11. Plate Girder SEM

Information Groups	Information Items	Attribute Sets	Attributes	Value
Bridge superstructure	Girder layout	Properties	Grider name	Grider1
Bridge superstructure	Girder layout	Properties	Girder material designation	Steel1
Bridge superstructure	Girder layout	Location	Station at start of span	1209549
Bridge superstructure	Girder layout	Location	Station at end of span	1330132
Bridge superstructure	Girder layout	Location	Skew angle at start of span	0
Bridge superstructure	Girder layout	Location	Skew angle at end of span	30
Bridge superstructure	Girder layout	Location	Centerline offset from HCL at start of span	-4800
Bridge superstructure	Girder layout	Location	Centerline offset from HCL at end of span	-4800
Bridge superstructure	Steel girder properties	Steel girder properties	Steel girder height	2090
Bridge superstructure	Steel girder properties	Steel girder properties	Steel girder length	120583
Bridge superstructure	Steel girder properties	Top flange properties	Top flange plate thickness	45
Bridge superstructure	Steel girder properties	Top flange properties	Top flange plate width	700
Bridge superstructure	Steel girder properties	Bottom flange properties	Bottom flange plate thickness	45
Bridge superstructure	Steel girder properties	Bottom flange properties	Bottom flange plate width	700
Bridge superstructure	Steel girder properties	Web properties	Web plate thickness	18
Bridge superstructure	Steel girder properties	Web properties	Web plate depth	2000

Table 4-12. Material SEM

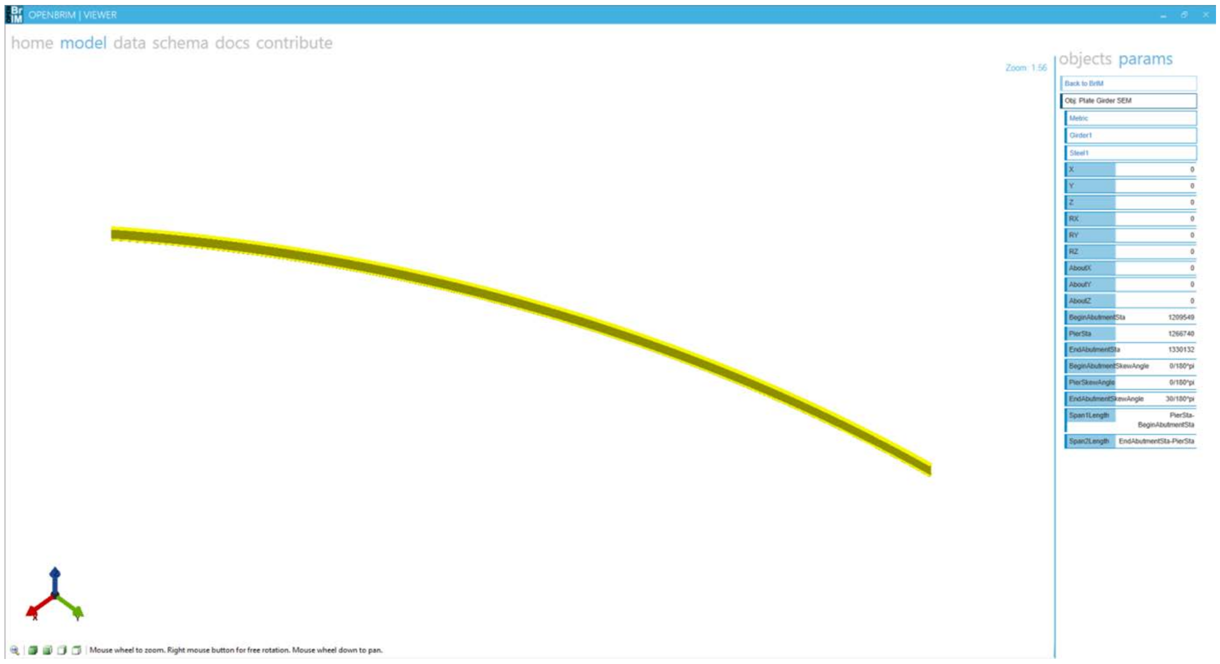
Information Groups	Information Items	Attribute Sets	Attributes	Value
Material library	Material properties	Structural steel	Material designation	ASTM A709M Grade 345W
Material library	Material properties	Structural steel	Unit mass	7l.85E-09
Material library	Material properties	Structural steel	Elastic modulus	2.00E+05
Material library	Material properties	Structural steel	Poisson's ratio	3.00E-01
Material library	Material properties	Structural steel	Thermal expansion coefficient	6.50E-06
Material library	Material properties	Structural steel	Yield Strength	3.45E+02
Material library	Material properties	Structural steel	Tensile Strength	5.00E+02

Step 2: The translator for converting plain English SEMs to OpenBrIM XML code has been created using C# language. A portion of the OpenBrIM XML code corresponding to the above three SEMs are shown in Code 4.5.

```
<BrIM Version="2">
    <Obj Name="Plate Girder SEM" Alignment="I-290 Ramp B Alg">
        <Obj Type="Units" Name="Metric" Length="Millimeter" Force="Newton"
        Mass="Kilogram" Angle="Degree" Temperature="Kelvin"/>

        <Param Name="BeginAbutmentSta" Value="1209549"/>
        <Param Name="PierSta" Value="1266740"/>
        <Param Name="EndAbutmentSta" Value="1330132"/>
        <Param Name="BeginAbutmentSkewAngle" Value="0/180*pi"/>
        <Param Name="PierSkewAngle" Value="0/180*pi"/>
        <Param Name="EndAbutmentSkewAngle" Value="30/180*pi"/>
        <Param Name="Span1Length" Value="PierSta-BeginAbutmentSta"/>
        <Param Name="Span2Length" Value="EndAbutmentSta-PierSta"/>

        <RoadwayGeometry Name="I-290 Ramp B Alg">
            <HorizontalAlignment StartStation="1049139" StartAzimuth=
            "4.2195">
```

Code 4.5 OpenBrIM XML Code for Three SEMs

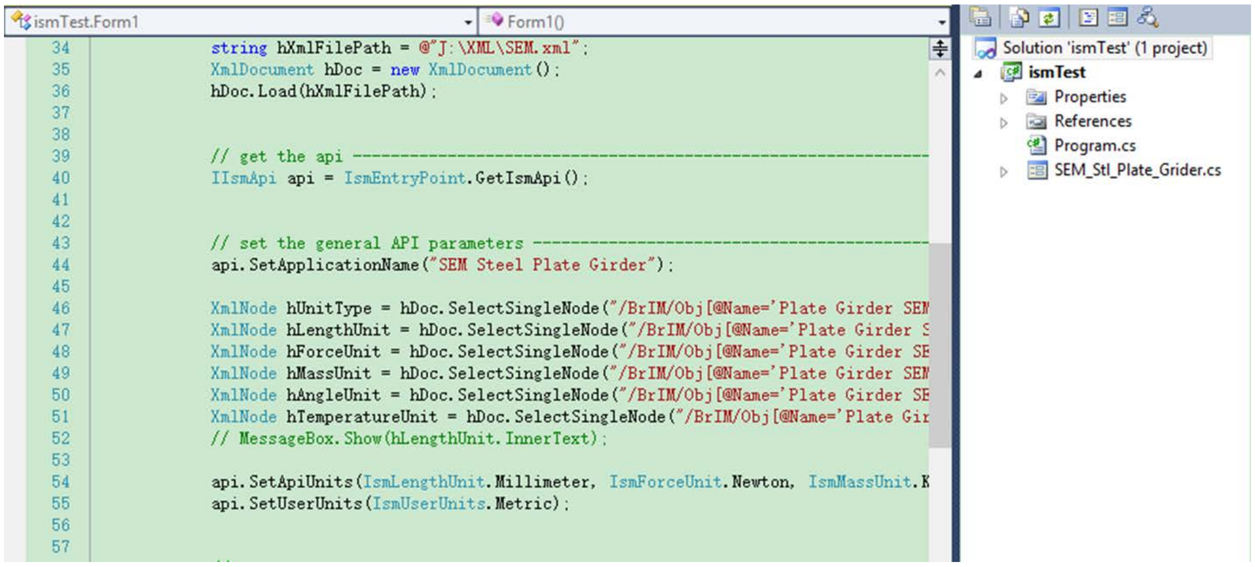
A viewer-generated 3D view of the steel plate girder model based on the OpenBrIM XML code are shown in Figure 4-163.


OPENBRIM | VIEWER
home model data schema docs contribute
Zoom: 1.56
objects params
Back to BrIM
Obj: Plate Girder SEM
Metric
Girder1
Steel1
Mouse wheel to zoom. Right mouse button for free rotation. Mouse wheel down to pan.

X	0
Y	0
Z	0
RX	0
RY	0
RZ	0
AboutX	0
AboutY	0
AboutZ	0
BeginAbutmentSta	1209549
PierSta	1266740
EndAbutmentSta	1330132
BeginAbutmentSkewAngle	0/180*pi
PierSkewAngle	0/180*pi
EndAbutmentSkewAngle	30/180*pi
Span1Length	PierSta-BeginAbutmentSta
Span2Length	EndAbutmentSta-PierSta

Figure 4-163. 3D View of the Steel Plate Girder

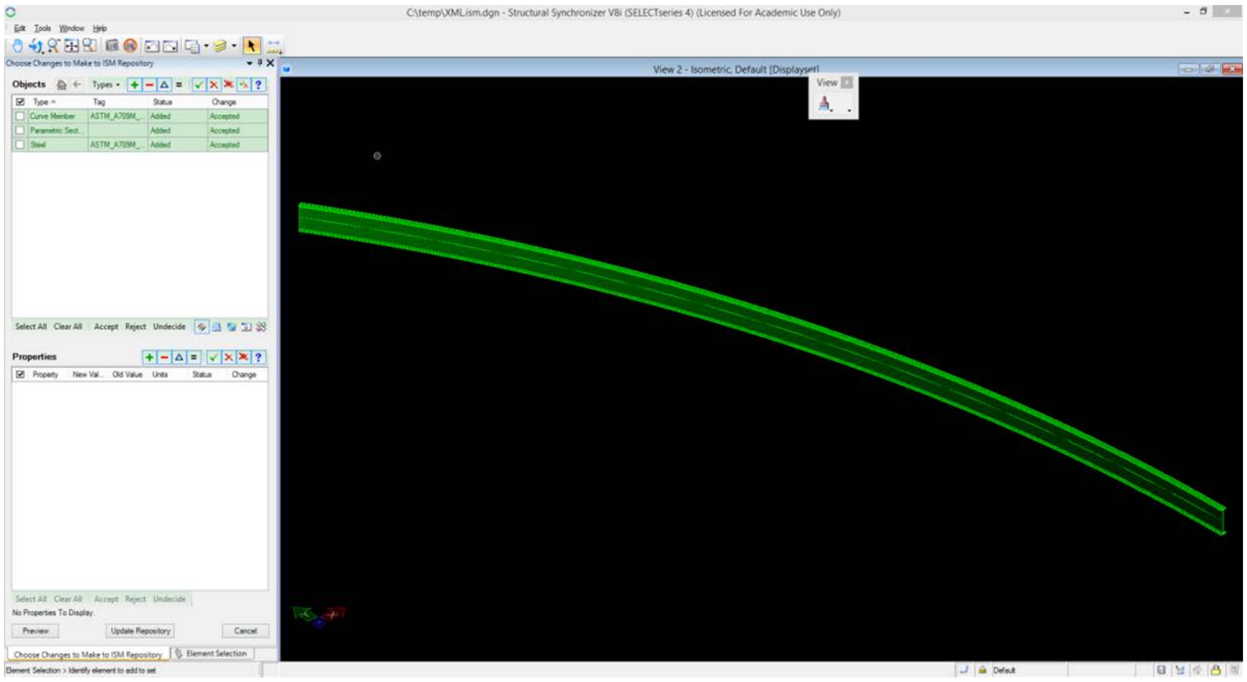
Step 3: The translator for converting OpenBrIM XML code to a native model, ISM model through its API has been created using C# language, as shown in Figure 4-164.



```
            string hXmlFilePath = @"J:\XML\SEM.xml";
            XmlDocument hDoc = new XmlDocument();
            hDoc.Load(hXmlFilePath);


            // get the api -------------------------------------------------------------
            IIsmApi api = IsmEntryPoint.GetIsmApi();


            // set the general API parameters ------------------------------------------
            api.SetApplicationName("SEM Steel Plate Girder");

            XmlNode hUnitType = hDoc.SelectSingleNode("/BrIM/Obj[@Name='Plate Girder SEM
            XmlNode hLengthUnit = hDoc.SelectSingleNode("/BrIM/Obj[@Name='Plate Girder S
            XmlNode hForceUnit = hDoc.SelectSingleNode("/BrIM/Obj[@Name='Plate Girder SE
            XmlNode hMassUnit = hDoc.SelectSingleNode("/BrIM/Obj[@Name='Plate Girder SEM
            XmlNode hAngleUnit = hDoc.SelectSingleNode("/BrIM/Obj[@Name='Plate Girder SE
            XmlNode hTemperatureUnit = hDoc.SelectSingleNode("/BrIM/Obj[@Name='Plate Gir
            // MessageBox.Show(hLengthUnit.InnerText);

            api.SetApiUnits(IsmLengthUnit.Millimeter, IsmForceUnit.Newton, IsmMassUnit.K
            api.SetUserUnits(IsmUserUnits.Metric);

```

Figure 4-164. Translator that Converts OpenBrIM XML to ISM

Figure 4-165 shows the 3D view of the ISM model of the steel plate girder based on those three SEMs.


C:\temp\XML.ism.dgn - Structural Synchronizer V8i (SELECTseries 4) (Licensed For Academic Use Only)
Edit Tools Window Help
Choose Changes to Make to ISM Repository
Objects
Types
View 2 - Isometric, Default [Displayset]
View
Select All Clear All Accept Reject Undecide
Properties
Property New Val... Old Value Units Status Change
Select All Clear All Accept Reject Undecide
Preview
Update Repository
Cancel
Choose Changes to Make to ISM Repository
Element Selection
Default

	Type	Tag	Status	Change
☐	Curve Member	ASTM_A709M_...	Added	Accepted
☐	Parametric Sect...		Added	Accepted
☐	Steel	ASTM_A709M_...	Added	Accepted

No Properties To Display.

Element Selection > Identify element to add to set

Figure 4-165. View of ISM Model of the Steel Plate Girder

5 References

1. AASHTO, (2007). *AASHTO Virtis and Opis Application Program Interface Guide*, 2007. p. 158.

2. BuildingSMART Alliance, (2012). *IFC Standard*. http://www.buildingsmart-tech.org/specifications/ifc-overview/ifc-overview-summary.

3. Amann, J., et al. (2013). "A Refined Product Model for Shield Tunnels Based on a Generalized Approach for Alignment Representation." Proc., International Conference on Civil and Building Engineering Informatics 2013.

4. Bentley, (2012). *Intergraded Structural Modeling-Programmer's Introduction to ISM*, Bentley System, Inc. p. 194.

5. Bentley, (2012). *LEAP Bridge Enterprise*, 2012: Bentley Systems, Inc.

6. Borrmann, A. (2013). "IFC Infrastructure Alignment representation." Technische Universitat Munchen.

7. buildingSMART. *Industry Foundation Classes Release 4 (IFC4)*. 2013, buildingSMART http://www.buildingsmart-tech.org/specifications/ifc-releases/ifc4release/ifc4-release-summary.

8. Chen, S.S., et al., *Information Delivery Manual Elements for Highway Bridge Interoperable Data Protocols*, Dec. 2013, The State University of New York at Buffalo: Buffalo, NY.

9. Crowley, A.J. and A.S. Watson, *CIMsteel Integration Standards Release 2 -The Logical Product Model (LPM/6)*, 2003, The Steel Construction Institute. p. 918.

10. CSI, *CSiBridge: Integrated 3-D Bridge Design Software*, 2013: Computers and Structures, Inc.

11. Ji, Y., A. Borrmann, and M. Obergrießer. *Toward the Exchange of Parametric Bridge Models Using a Neutral Data Format*. in *2011 ASCE International Workshop on Computing in Civil Engineering*. 2011. Miami, Florida, USA.

12. Ji, Y., et al. *Integration of Parametric Geometry into IFC-Bridge*. in *the 23th Forum Bauinformatik*. 2011. Cork, Ireland.

13. Ji, Y., et al., (2013). *Exchange of Parametric Bridge Models Using a Neutral Data Format*. Journal of Computing in Civil Engineering. **0**(ja): p. null.

14. Katz, C., (2008). *Parametric Description of Bridge Structures, in IABSE Symposium Report*: Helsinki. p. 17-27.

15. Kim, H. (2013). "Object Based 3D Intelligent Model for Construction Planning/Simulation in a Highway Construction." University of North Carolina at Charlotte, Korean Institute of Construction Technology (KICT).

16. Koch, A. (2013). *OpenBrIM: Community Driven Bridge Information Modeling*.

17. Koch, A., (2013). *OpenBrIM XML Schema Documentation*, Red Equation Corporation.

18. LandXML (2008). “LandXML-1.2Documentation.” LandXML12_er_designCoordination.pdf http://www.landxml.org/.

19. Lebegue, E., et al., (2012). *IFC-BRIDGE V3 Data Model -IFC 4*, BuildingSmart French Chapter, BuildingSmart German Chapter, BuildingSmart Japanese Chapter.

20. Liebich, T. and M. Weise, (2012). *ifcXML4 Specification Methodology*, Model Support Group (MSG) of buildingSMART International Ltd.

21. Liebich, T., (2009). *IFC 2x Model Implementation Guide*, 2009, buildingSMART International Modeling Support Group.

22. Nagarajan, K., (2010). *Development of Framework for Extending TransXML to Steel Bridge Construction*, in *Department of Civil, Structural and Environmental Engineering*, The State University of New York at Buffalo. p. 516.

23. NCHRP, (2007). *TransXML: XML Schemas for Exchange of Transportation Data (NCHRP Report 576)*, 2007, Transportation Research Board: Washington, D.C.

24. Nisbet, N. and T. Liebich, (2007). *ifcXML Implementation Guide*, 2007, International Alliance for Interoperability Model Support Group, June 28.

25. NYSDOT, (2012). *Selected Contract Plans and Erection Procedures*, 2012, New York State Department of Transportation.

26. OGC. (2013). *Geography Markup Language (GML).* http://www.opengeospatial.org/standards/gml.

27. Peak, R. S., et al. (2004). *STEP, XML, and UML: Complementary Technologies*, Journal of Computing and Information Science in Engineering, 59.

28. PennDOT, (2012). *Selected Contract Drawings and Erection Procedures*, Pennsylvania Department of Transportation.

29. Scarponcini, P., (2001). *Linear Reference System for Life-Cycle Integration.* Journal of Computing in Civil Engineering. **15**(1): p. 81-88.

30. Scarponcini, P., *Various linear reference systems explained*, 1997, Bentley Systems, Inc.

31. *SDS/2 Design Data*. (2013). Ferguson Structural Engineering Laboratory Cockrell School of Engineering, The University of Texas at Austin https://fsel.engr.utexas.edu/software/.

32. Tekla, *Tekla Structures*, (2012), Tekla, Inc.

33. Venugopal, M., et al. (2010). *Engineering Semantics of Model Views for Building Information Model Exchanges Using IFC*. in the *CIB W78 2010: 27th International Conference*. Cairo, Eygpt.

34. Venugopal, M., et al., (2012). *Semantics of Model Views for Information Exchanges Using the Industry Foundation Class Schema.* Advanced Engineering Informatics, **26**(2): p. 411-428.

35. Venugopal, M., (2011). *Formal Specification of Industry Foundation Class Concepts Using Engineering Ontologies*, in *School of Civil and Environmental Engineering*, Georgia Institute of Technology. p. 261.

36. W3C. *Extensible Markup Language (XML).* http://www.w3.org/XML/.

(This Page is Intentionally Left Blank)