Appendix B: Precast Modifications to IFC 2.x4

Model View Definitions For Precast Concrete

Adaptations for Precast Concrete Exchanges

Version 2.0
April 2009

Rafael Sacks, Chuck Eastman, Ivan Panush

Contents

Introduction

This appendix outlines the adaptations that are proposed for IFC version 2x4 to accommodate the specific needs for precast concrete exchanges that are not addressed in the current version 2x3 of the IFC standard. All of these have been registered on the IFC 2x4 Alpha website, and are being considered by the international panel that maintains the IFC standard for inclusion in the 2x4 Beta version.
The MVD concepts defined in the main document all make use of these new entities, property sets and enumerated value lists.

Precast Specific Comments

Existing entities

Starting from our previous document, the following objects can already be represented in IFC 2x4 and therefore **do not** need any new entities:

- **Anchor plates, lifting hooks** and other similar discrete embeds can all be represented using the *IfcDiscreteAccessory* and *IfcDiscreteAccessoryType* entities. Some of the specific *ObjectType* values that are already listed for this entity include 'Wire lifting hook', 'Steel lifting hook', 'Lifting socket', 'Steel lifting anchor' and 'Lifting hole'; 'Standard fixing plate', 'Neoprene bearing plate' and many others. The list is open-ended. However, some of these may well require specific property sets to express their parameters uniquely for each type.
- **Epoxy glue, mortar and welded joints** are all provided in IFC 2x4 in the *IfcFastener* entity.
- **Recesses** are expressed using the *IfcOpeningElement* entity, with the inherited attribute *ObjectType* set to 'Recess'.
- **Pile caps** are represented using *IfcFooting* with *IfcFootingTypeEnum* set to “PILE_CAP”. The precast pile caps should be restricted to a swept solid representation.
- Precast **retaining walls** can be modeled using IfcWall. Where a wall is composed of individual precast wall sections, it can be modeled as an assembly as IfcWall with the individual parts modeled as instances of IfcBuildingElementPart.
- The PCI BIM advisory group has also specified a range of **beam** types: (Rectangular, Inverted Tees, L Beams, Transfer Beams, Structural Spandrels (Pocket, Ledge, Button Haunch), Deep Beams). However, the data exchange requirements and all other aspects of precast beams will not differ from that of other reinforced concrete beams, so that *IfcBeam* can be used. Furthermore, since all of these conform to the requirements for use of *IfcBeamStandardCase* (prismatic with constant profile along a linear extrusion), this entity should be preferred over *IfcBeam*. If the individual types are identified in the exporting software, they can be identified semantically in an IFC file using the inherited *ObjectType* property, provided that we establish a pre-set list of keywords. More specifically, if IfcBeamType entities are included, then the beam type enumerator should be extended to include the precast types. The current definition is:

```
TYPE IfcBeamTypeEnum = ENUMERATION OF
 (BEAM,
  JOIST,
  LINTEL,
  T_BEAM,
  USERDEFINED,
  NOTDEFINED);
```

```
END_TYPE;
```

The new definition should be:

```
TYPE IfcBeamTypeEnum = ENUMERATION OF
  (BEAM,
   JOIST,
   LINTEL,
   T_BEAM,
   INVERTED_T_BEAM,
   L_BEAM,
   PRECAST_SPANDREL,
   DOUBLE_TEE,
   HOLLOWCORE,
   USERDEFINED,
   NOTDEFINED);
END_TYPE;
```

Note: Why does IfcBeamType have a PredefinedType, whereas IfcBeam does not?

Missing entities

On the other hand, there are other precast object types that should be defined explicitly. These include prestressed slab components, volumetric building modules and some architectural precast components.

Prestressed slab components

Slabs with precast concrete components such as hollow core plank, flat prestressed plank, etc. generally have these components embedded within a slab that has a cast-in-place concrete topping. For hollow-core and flat planks in particular, the cast-in-place concrete fills cracks between the planks and forms beams where necessary. The topping concrete also often binds the planks to precast beams on which they rest using exposed reinforcing links. As such, the slab itself must have its own enveloping geometry and the planks are 'embedded' within it. This appears to violate the provision in IFC 2x4 that the assembly entity should not have independent geometry (i.e. its geometry should only be the sum of its parts). Therefore we propose extending the IfcSlab entity to represent the case where the slab has its own enveloping geometry and is comprised of components. In keeping with the policy of not creating specific material related entity types (such as *IfcPrecastSlab*) the entity proposed is *IfcSlabWithElements* This entity would have its own geometry and also contain embedded component parts.

```
ENTITY IfcSlabWithElements
     SUBTYPE OF (IfcSlab);
END_ENTITY;
```

Furthermore, although slabs on different levels of a building are rarely absolutely identical, as a convention, the *IfcSlab* should be used to represent instances of *IfcSlabTypes*, even where there is only one occurrence. For the precast slabs in our case, a *IfcSlabWithElementsType* is proposed. These slab types cannot be modeled as an *IfcSlabStandardCase*, because the topping and infill cast-in-place beam geometry will not generally conform to the geometry of any standard case, and so a standard entity type is not needed.

```
ENTITY IfcSlabWithElementsType
     SUBTYPE OF (IfcSlabType);
END_ENTITY;
     *a rule is needed to limit the components to be IfcBeamType
     entities only, with precast profile types
```

The *IfcSlabWithElementsType* will be used as an assembly containing *IfcBeam* objects to represent the double tee or plank components. The *IfcBeams* will be legitimate parts of the *IfcSlabWithElementsType* assembly because they inherit from *IfcBuildingElement.* Every *IfcBeam* object will be an instance of an *IfcBeamType* template beam. For each *IfcBeamType*, the specific type should be defined using the enumerated value *IfcBeamTypeEnum* to indicate the precast specific term for the module dealt with (double tees, hollow core, flat prestressed plank, etc.).

In addition, to express geometric parameters like the step distance between the individual parts and the angle opening between each part in the array, which are typical parameters for precast type slabs, a specific set of parameters is needed. For this purpose a specific precast slab property set is proposed:

```
     Pset_PrecastSlab
```

PropertySet Name	Pset_PrecastSlab
Applicable Entities	`IfcSlabWithElements , IfcSlabWithElementsType`
Applicable Type Value	
Definition	Definition from PCI BIM advisory committee: layout and component information defining how prestressed slab components are laid out in a precast slab assembly. The values are global defaults for the slab as a whole, but can be overridden by local

	placements of the individual components of the slab.

Property Definitions:

Name	Property Type	Data Type	Definition
TypeDesignator	IfcPropertySingleValue	IfcLabel	Type designator for the precast concrete slab, expressing mainly the component type. Possible values are “Hollow-core”, “Double-tee”, “Flat plank”, etc.
ToppingType	IfcPropertySingleValue	IfcLabel	Defines if a topping is applied and what kind. Values are “Full topping”, “Perimeter Wash”, “None”
EdgeDistanceToFirstAxis	IfcPropertySingleValue	IfcPositiveLengthMeasure / LENGTHUNIT	The distance from the left (‘West’) edge of the slab (in the direction of span of the components) to the axis of the first component.
DistanceBetweenComponentAxes	IfcPropertySingleValue	IfcPositiveLengthMeasure / LENGTHUNIT	The distance between the axes of the components, measured along the ‘South’ edge of the slab.
AngleToFirstAxis	IfcPropertySingleValue	IfcPlaneAngleMeasure	The angle of rotation of the axis of the first component relative to the ‘West’ edge of the slab.
AngleBetweenComponentAxes	IfcPropertySingleValue	IfcPlaneAngleMeasure	The angle between the axes of each pair of components.
NominalThickness	IfcPropertySingleValue	IfcPositiveLengthMeasure / LENGTHUNIT	The nominal overall thickness of the slab.
NominalToppingThickness	IfcPropertySingleValue	IfcPositiveLengthMeasure / LENGTHUNIT	The nominal thickness of the topping.

Precast modules

(three dimensional core units, prison cells, bathroom units, staircase sections, etc.). These are unique objects that are generally composed of slab and wall components but occupy 3D volumes, and they must have a separate identity as precast pieces for design, production and erection. They should not be represented simply as assemblies of other components by IfcAssembly or derived entities, because they often have geometry that cannot simply be expressed as an aggregation of other object types. Some examples can be seen at the following links:
School modules:
http://www.opmg.com/Oldcastle%20Bathroom%20Modules.htm?Product_ID=&location_ID=&location_Product_ID=&
Seawall Modules: http://www.tectonicsystems.com/Precast.html
Bridge and prison cell modules: www.pomeroycorp.com/pris_cell.php
We propose adding an ***IfcModule*** entity to the Shared Building Elements section of the IFC schema. Since there could be prefabricated modules made of other materials (e.g. metal stud framed bathroom units), a generic *IfcModule* should be considered rather than a precast specific entity. Like other precast pieces, which will be modeled as generic Ifc products with appropriate values for ObjectType and appropriate Property sets related to their precast properties, a precast module can be an IfcModule. The specific type ("Core unit", "Cell", "Bathroom unit", "Stair unit" etc. can be defined using the *ObjectType* property of the *IfcObject* to indicate the precast specific term for the module dealt with). The precast property sets would be added by relation to each instance.

```
ENTITY IfcModule
     SUBTYPE OF (IfcBuildingElement);
END_ENTITY;
```

As is done for all of the other building products, an *IfcModuleType* entity should also be added.

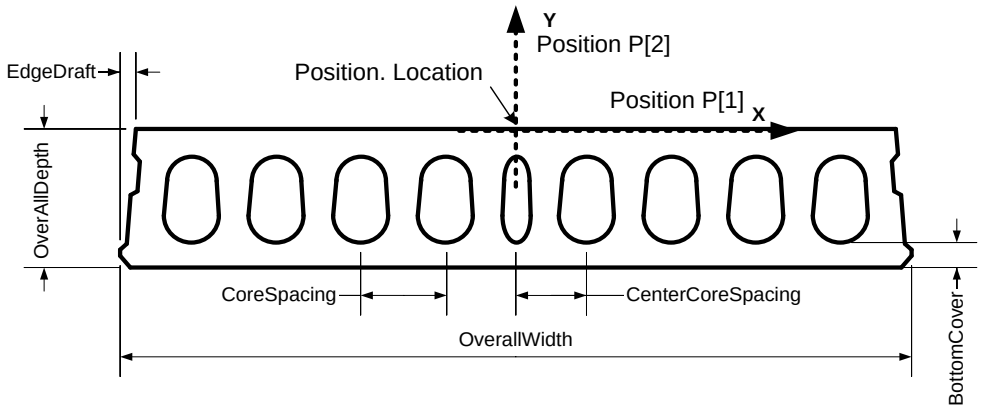
```
ENTITY IfcModuleType
     SUBTYPE OF (IfcBuildingElementType);
END_ENTITY;
```

Precast profiles

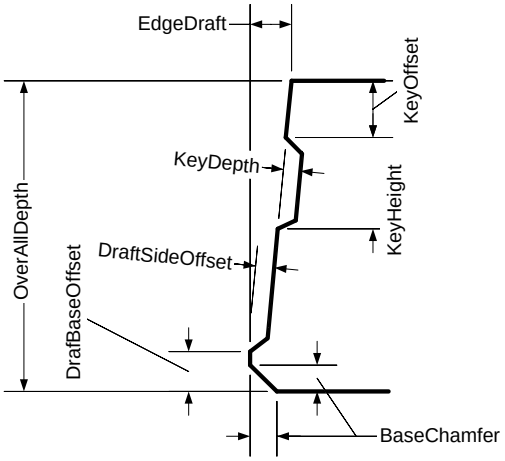
IFC 2x4 defines numerous basic parameterized profile types, such as *IfcLShapeProfileDef*, *IfcIShapeProfileDef* and others. While standard precast profile types such as hollow core planks and double tees can be represented using *IfcArbitraryClosedProfileDef*, it would be better to represent them as parameterized profiles wherever possible. The advantages include:

- IFC exchange files would be minimized,
- Information would be semantically meaningful and more useful. Different applications could rebuild them at appropriate levels of internal detail, without necessarily replicating all of the finer detail required for fabrication.
- Embedded components could be located relative to a parameterized profile definition.

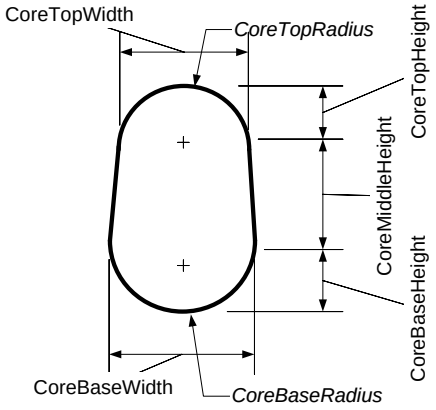
At least two new parameterized profiles are needed: ***IfcHollowCoreShapeProfileDef*** and ***IfcDoubleTeeShapeProfileDef***. Proposals for their EXPRESS definitions:

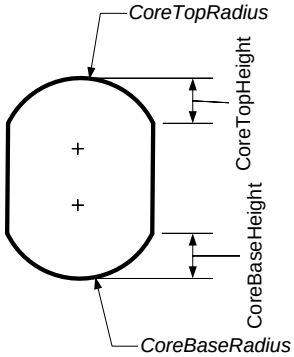


NOTE: In all cases, the cores are symmetrically distributed on either side of the plank center line, irrespective of whether the number of cores is odd or even. For planks with a center core with different geometry to that of the other cores, set CenterCoreSpacing > 0. When the number of cores is even, all Center Core parameters are ignored.



NOTE:
Key chamfers and draft chamfer are all 45 degree chamfers





NOTE: The CoreTopRadius and CoreBaseRadius parameters can be derived and are therefore not listed in the IfcHollowCoreShapeProfileDef definition. They are shown to define that the curves are arcs. The parameters for the center core are the same as above, but with the prefix “Center”.

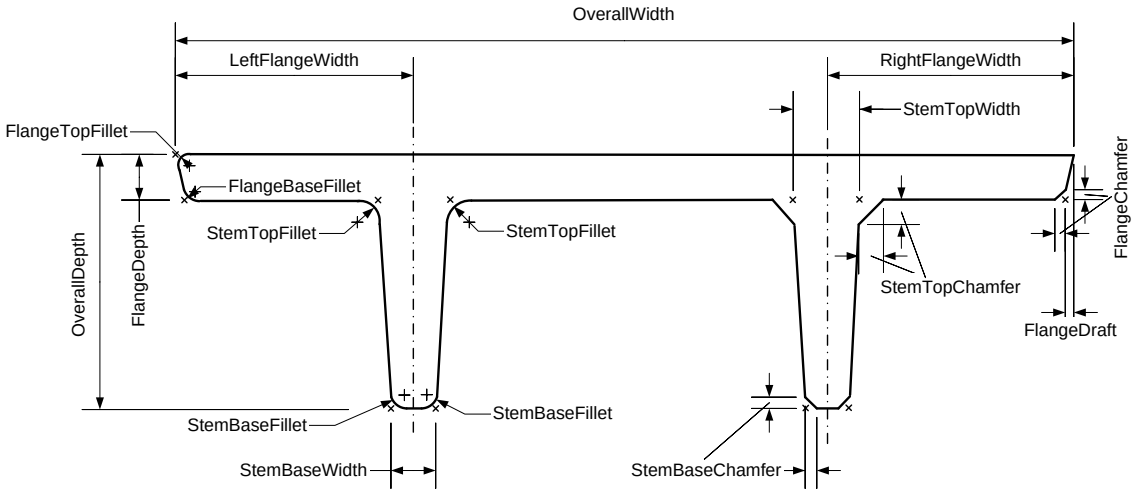
```
ENTITY IfcHollowCoreShapeProfileDef
     SUBTYPE OF ( IfcParameterizedProfileDef);
     OverallWidth        :     IfcPositiveLengthMeasure;
     OverallDepth        :     IfcPositiveLengthMeasure;
     EdgeDraft           :     IfcPositiveLengthMeasure;
     DraftBaseOffset     :     OPTIONAL IfcPositiveLengthMeasure;
     DraftSideOffset     :     OPTIONAL IfcPositiveLengthMeasure;
     BaseChamfer         :     IfcPositiveLengthMeasure;
     KeyDepth            :     IfcPositiveLengthMeasure;
     KeyHeight           :     IfcPositiveLengthMeasure;
     KeyOffset           :     IfcPositiveLengthMeasure;
     BottomCover         :     IfcPositiveLengthMeasure;
     CoreSpacing         :     IfcPositiveLengthMeasure;
     CoreBaseHeight      :     IfcPositiveLengthMeasure;
     CoreMiddleHeight    :     IfcPositiveLengthMeasure;
     CoreTopHeight       :     IfcPositiveLengthMeasure;
     CoreBaseWidth       :     IfcPositiveLengthMeasure;
     CoreTopWidth        :     IfcPositiveLengthMeasure;
     CenterCoreSpacing   :     OPTIONAL IfcPositiveLengthMeasure;
     CenterCoreBaseHeight:     OPTIONAL IfcPositiveLengthMeasure;
     CenterCoreMiddleHeight:   OPTIONAL IfcPositiveLengthMeasure;
     CenterCoreTopHeight :     OPTIONAL IfcPositiveLengthMeasure;
     CenterCoreBaseWidth :     OPTIONAL IfcPositiveLengthMeasure;
     CenterCoreTopWidth  :     OPTIONAL IfcPositiveLengthMeasure;
     NumberOfCores       :     IfcCountMeasure;

WHERE
        <Validation tests…>
END_ENTITY;
```

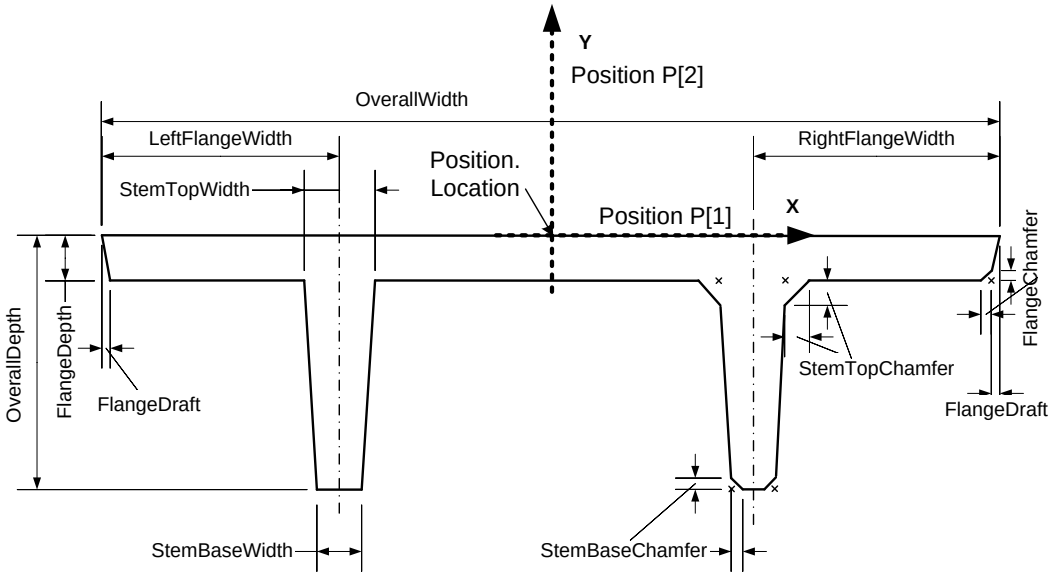
Inheritance graph

```
ENTITY IfcHollowCoreShapeProfileDef;
      ENTITY IfcProfileDef;
            ProfileType : IfcProfileTypeEnum;
            ProfileName : OPTIONAL IfcLabel;
      INVERSE
            HasProperties     : SET OF IfcProfileProperties FOR
                              ProfileDefinition;
      ENTITY IfcParameterizedProfileDef
            Position          : OPTIONAL IfcAxis2Placement2D;
      ENTITY IfcHollowCoreShapeProfileDef;
     OverallWidth        :     IfcPositiveLengthMeasure;
     OverallDepth        :     IfcPositiveLengthMeasure;
     EdgeDraft           :     IfcPositiveLengthMeasure;
     DraftBaseOffset     :     OPTIONAL IfcPositiveLengthMeasure;
     DraftSideOffset     :     OPTIONAL IfcPositiveLengthMeasure;
     BaseChamfer         :     IfcPositiveLengthMeasure;
     KeyDepth            :     IfcPositiveLengthMeasure;
     KeyHeight           :     IfcPositiveLengthMeasure;
     KeyOffset           :     IfcPositiveLengthMeasure;
     BottomCover         :     IfcPositiveLengthMeasure;
     CoreSpacing         :     IfcPositiveLengthMeasure;
     CoreBaseHeight      :     IfcPositiveLengthMeasure;
     CoreMiddleHeight    :     IfcPositiveLengthMeasure;
     CoreTopHeight       :     IfcPositiveLengthMeasure;
```

```
    CoreBaseWidth       :     IfcPositiveLengthMeasure;
    CoreTopWidth        :     IfcPositiveLengthMeasure;
    CenterCoreSpacing   :     OPTIONAL IfcPositiveLengthMeasure;
    CenterCoreBaseHeight:     OPTIONAL IfcPositiveLengthMeasure;
    CenterCoreMiddleHeight:   OPTIONAL IfcPositiveLengthMeasure;
    CenterCoreTopHeight :     OPTIONAL IfcPositiveLengthMeasure;
    CenterCoreBaseWidth :     OPTIONAL IfcPositiveLengthMeasure;
    CenterCoreTopWidth  :     OPTIONAL IfcPositiveLengthMeasure;
    NumberOfCores       :     IfcCountMeasure;
END_ENTITY;
```



NOTE: The left and right stems show alternate configurations; either can have chamfers or fillets at top or base. Where the values of Fillet parameters are grater than zero, Chamfer values will be ignored.




```
ENTITY IfcDoubleTeeShapeProfileDef
    SUBTYPE OF ( IfcParameterizedProfileDef);
       OverallWidth    :     IfcPositiveLengthMeasure;
       LeftFlangeWidth :     IfcPositiveLengthMeasure;
       RightFlangeWidth:     IfcPositiveLengthMeasure;
```

```
        OverallDepth    :     IfcPositiveLengthMeasure;
        FlangeDepth     :     IfcPositiveLengthMeasure;
        FlangeDraft     :     OPTIONAL IfcPositiveLengthMeasure;
        FlangeChamfer   :     OPTIONAL IfcPositiveLengthMeasure;
        FlangeBaseFillet:     OPTIONAL IfcPositiveLengthMeasure;
        FlangeTopFillet :     OPTIONAL IfcPositiveLengthMeasure;
        StemBaseWidth   :     IfcPositiveLengthMeasure;
        StemTopWidth    :     IfcPositiveLengthMeasure;
        StemBaseChamfer :     OPTIONAL IfcPositiveLengthMeasure;
        StemTopChamfer  :     OPTIONAL IfcPositiveLengthMeasure;
        StemBaseFillet  :     OPTIONAL IfcPositiveLengthMeasure;
        StemTopFillet   :     OPTIONAL IfcPositiveLengthMeasure;

WHERE
LeftFlangeWidth > 0.5 * StemTopWidth + StemTopChamfer
RightFlangeWidth > 0.5 * StemTopWidth + StemTopChamfer
StemTopWidth + StemTopChamfer
StemBaseFillet * 2 <= StemBaseWidth
StemBaseChamfer * 2 <= StemBaseWidth
IF StemBaseChamfer > 0 THEN SET StemBaseFillet = 0
IF StemTopChamfer > 0 THEN SET StemTopFillet = 0
FlangeBaseFillet + FlangeTopFillet <= FlangeDepth
IF FlangeChamfer > 0 THEN SET FlangeBaseFillet = 0
END_ENTITY;
```

Inheritance graph

```
ENTITY IfcDoubleTeeShapeProfileDef;
      ENTITY IfcProfileDef;
            ProfileType : IfcProfileTypeEnum;
            ProfileName : OPTIONAL IfcLabel;
      INVERSE
            HasProperties     : SET OF IfcProfileProperties FOR
                                ProfileDefinition;
      ENTITY IfcParameterizedProfileDef
            Position          : OPTIONAL IfcAxis2Placement2D;
      ENTITY IfcHollowCoreShapeProfileDef;
        OverallWidth    :     IfcPositiveLengthMeasure;
        LeftFlangeWidth :     IfcPositiveLengthMeasure;
        RightFlangeWidth:     IfcPositiveLengthMeasure;
        OverallDepth    :     IfcPositiveLengthMeasure;
        FlangeDepth     :     IfcPositiveLengthMeasure;
        FlangeDraft     :     OPTIONAL IfcPositiveLengthMeasure;
        FlangeChamfer   :     OPTIONAL IfcPositiveLengthMeasure;
        FlangeBaseFillet:     OPTIONAL IfcPositiveLengthMeasure;
        FlangeTopFillet :     OPTIONAL IfcPositiveLengthMeasure;
        StemBaseWidth   :     IfcPositiveLengthMeasure;
        StemTopWidth    :     IfcPositiveLengthMeasure;
        StemBaseChamfer :     OPTIONAL IfcPositiveLengthMeasure;
        StemTopChamfer  :     OPTIONAL IfcPositiveLengthMeasure;
        StemBaseFillet  :     OPTIONAL IfcPositiveLengthMeasure;
        StemTopFillet   :     OPTIONAL IfcPositiveLengthMeasure;
END_ENTITY;
```

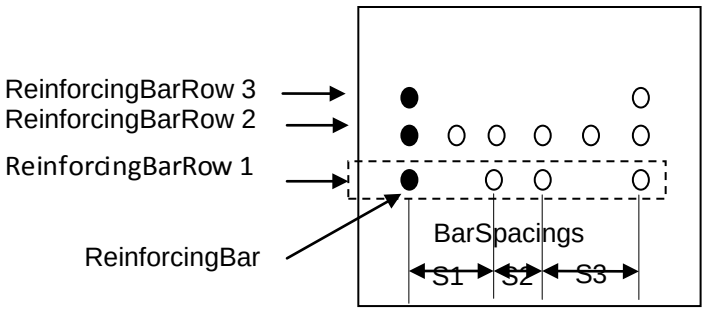
Additional standard parameterized profiles will be needed for AASHO beams and other complex standard types. Many simpler precast beam and column profiles, such as L and inverted T beams, have less

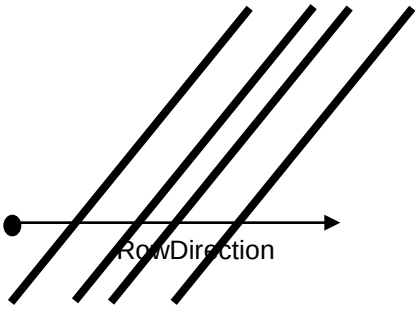
universally standardized profiles and so it is reasonable to represent them generically using *IfcArbitraryClosedProfileDef.*

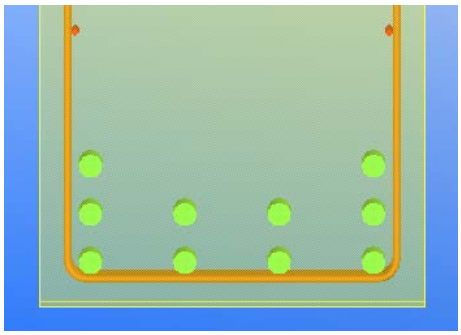
Reinforcement or Tendon Pattern

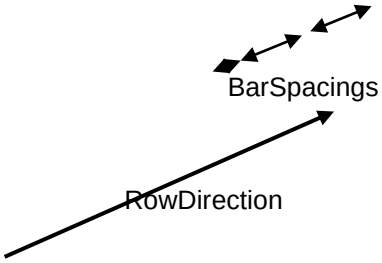
```
ENTITY IfcReinforcingBarRow
 SUBTYPE OF (IfcReinforcingBar);
      BarSpacing :      LIST OF [1:?] IfcPositiveLengthMeasure;
      RowDirection :    IfcDirection;

END_ENTITY;
```









Architectural decorative panels

These may be non-load bearing wall panels, column covers, sills, caps or other precast concrete pieces. Since buildings have similar parts made of other materials, such as GFRC panels (glass-fibre reinforced concrete), EIFS panels, and panels made from wood, glass, stone or other materials, we propose a collective building element called ***IfcFacadeElement***. They do not conform to the requirements for IfcWall entities, and so must have a separate semantic definition. The specific type should be carried as a value of the *ObjectType* property. Specific precast properties will be carried in the standard precast property sets.

```
ENTITY IfcFacadeElement
     SUBTYPE OF (IfcBuildingElement);
END_ENTITY;
```

As is done for all of the other building products, an *IfcFacadeElementType* entity should also be added.

```
ENTITY IfcFacadeElementType
     SUBTYPE OF (IfcBuildingElementType);
END_ENTITY;
```

Missing enumerators or property sets

Precast Concrete Connection Hardware and Embeds

The way that the hardware that is used in connections and joints, such as plates, shims, etc. are delivered and applied, must be identified. There are five possibilities: 1) the components are cast into the cast-in-place concrete structure or 2) provided as part of or welded onto the steel structural frame; 3) they are provided loose to the site and assembled when the precast piece is connected to the structure; 4) they are attached to the precast piece in the plant and delivered with it; and 5) they are cast into the precast piece. All of these should be made explicit. The issue is not unique to precast concrete, but relevant for construction of steel frame structures, wood structures and even parts of CIP structures.
In addition, there are other properties that must be defined for each of the hardware pieces, such as their treatment, which may be galvanized, stainless, painted or none. This is relevant not only for discrete accessories, but also for rebars and mesh in many applications.
Both of these issues can be done by adding a property set with two enumerated property types for the *IfcDiscreteAccessory* entity, as follows:
PropertySet Definition:

PropertySet Name	Pset_DiscreteAccessoryProductionRequirements
Applicable Entities	IfcDiscreteAccessory IfcDiscreteAccessoryType
Applicable Type Value	All types
Definition	Design requirements for determining production methods and tasks

Property Definitions:

Name	Property Type	Data Type	Definition
DeliveryType	IfcAccesoryDeliveryTypeEnum	IfcText	Determines how the accessory will be installed
CorrosionTreatment	IfcCorrosionTreatmentTypeEnum	IfcText	Determines corrosion treatment for metal accessories

The enumerated type definitions are:

```
TYPE IfcAccesoryDeliveryTypeEnum = ENUMERATION OF (
    CAST-IN-PLACE,
    WELDED_TO_STRUCTURE,
    LOOSE,
    ATTACHED_FOR_DELIVERY,
    PRECAST,
    NOTDEFINED);
END_TYPE;

 TYPE IfcCorrosionTreatmentTypeEnum = ENUMERATION OF (
    PAINTED,
    EPOXYCOATED,
    GALVANISED,
    STAINLESS,
    NONE,
    NOTDEFINED);
END_TYPE;
```

Concrete surface finishes are dealt with in IFC 2x4 using the *Pset_ConcreteElementSurfaceFinishQuantityGeneral* property set. The properties cover both formed surfaces and exposed surfaces and appear to be adequate, although the surface type definitions would be better communicated as preset enumerated types rather than text labels (IfcLabel type).

Rebar Bending Shapes According to ACI 315

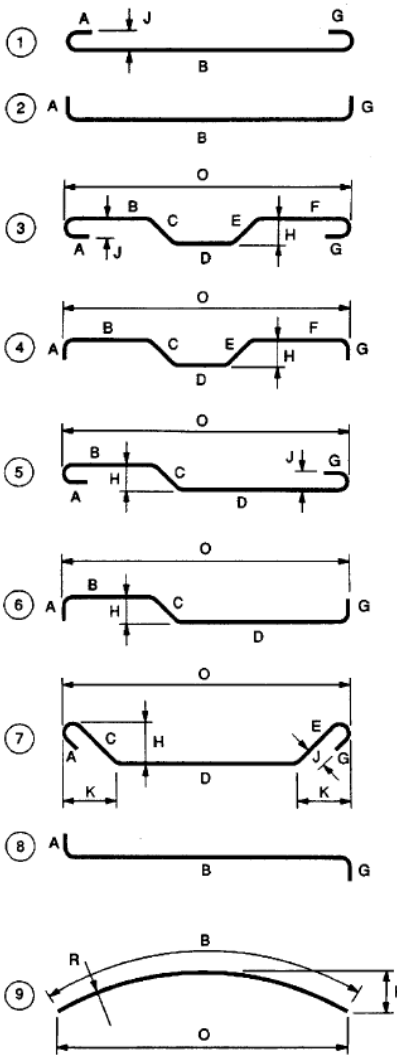
A new property set definition is needed for US standard rebar shapes. The primary resource for coding rebar bending shapes is the American Concrete Institute code 315 – ACI 315. The two pages from the code that define the rebar bending shapes are provided at the end of this section. As can be seen, the code defines a wide variety of bending shapes parametrically, in much the same way as do the British BS 8666, the German DIN135610, the Finnish BEC and the ISO CD3766. Because ACI 315 does not cover the complete range of rebar shapes that are possible (no code can predict all shapes), precasters may from time to time use shapes that are not defined in ACI 315. In those cases, the exceptional shapes can either be exchanged explicitly, or they can be parameterized by mutual agreement to define a new shape, which could use the same set of parameter definitions from the property set proposed below for ACI 315.

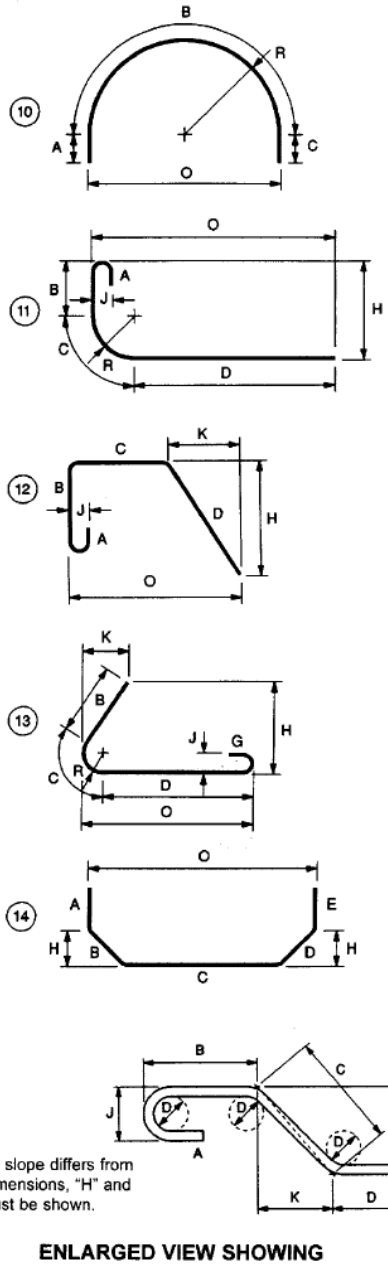
PropertySet Definition:

PropertySet Name	Pset_ReinforcingBarBendingsACI315Common
Applicable Entities	IfcReinforcingBar IfcReinforcingMesh
Applicable Type Value	
Definition	Definition from IAI: Properties expressing the bending information of non-prestressed reinforcing bars. The properties in this Pset are defined according to the local United States ACI 315 standard with minor adjustments (only bar bending information is included). The bending shape property definitions apply to both reinforcing bars (IfcReinforcingBar) and reinforcing meshes (IfcReinforcingMesh). It is presumed that a single standard for defining the bar bending is used throughout the project and that this standard is referenced from the IfcProject object through the IfcDocumentReference mechanism.

Property Definitions:

Name	Property Type	Data Type	Definition
ACI315BarShapeCode	IfcPropertySingleValue	IfcLabel	The bending type code for the specific bending shape as defined in the ACI 315 standard (Valid values are: 1 to 26, S1 to S11, T1 to T9).
ACI315CuttingLength	IfcPropertySingleValue	IfcPositiveLengthMeasure / LENGTHUNIT	Usually calculated from the sum of the partial length parameters with corrections for the bendings.
ACI315ShapeParameter_A	“	“	Bar shape parameter A.
ACI315ShapeParameter_B	“	“	Bar shape parameter B.
ACI315ShapeParameter_C	“	“	Bar shape parameter C.
ACI315ShapeParameter_D	“	“	Bar shape parameter D.
ACI315ShapeParameter_E	“	“	Bar shape parameter E.
ACI315ShapeParameter_F	“	“	Bar shape parameter F.
ACI315ShapeParameter_G	“	“	Bar shape parameter G.
ACI315ShapeParameter_H	“	“	Bar shape parameter H.
ACI315ShapeParameter_J	“	“	Bar shape parameter J.
ACI315ShapeParameter_K	“	“	Bar shape parameter K.
ACI315ShapeParameter_O	“	“	Bar shape parameter O.
ACI315ShapeParameter_R	“	“	Bar shape parameter R.
ACI315RollerDiameter	“	“	Diameter of bending roller.



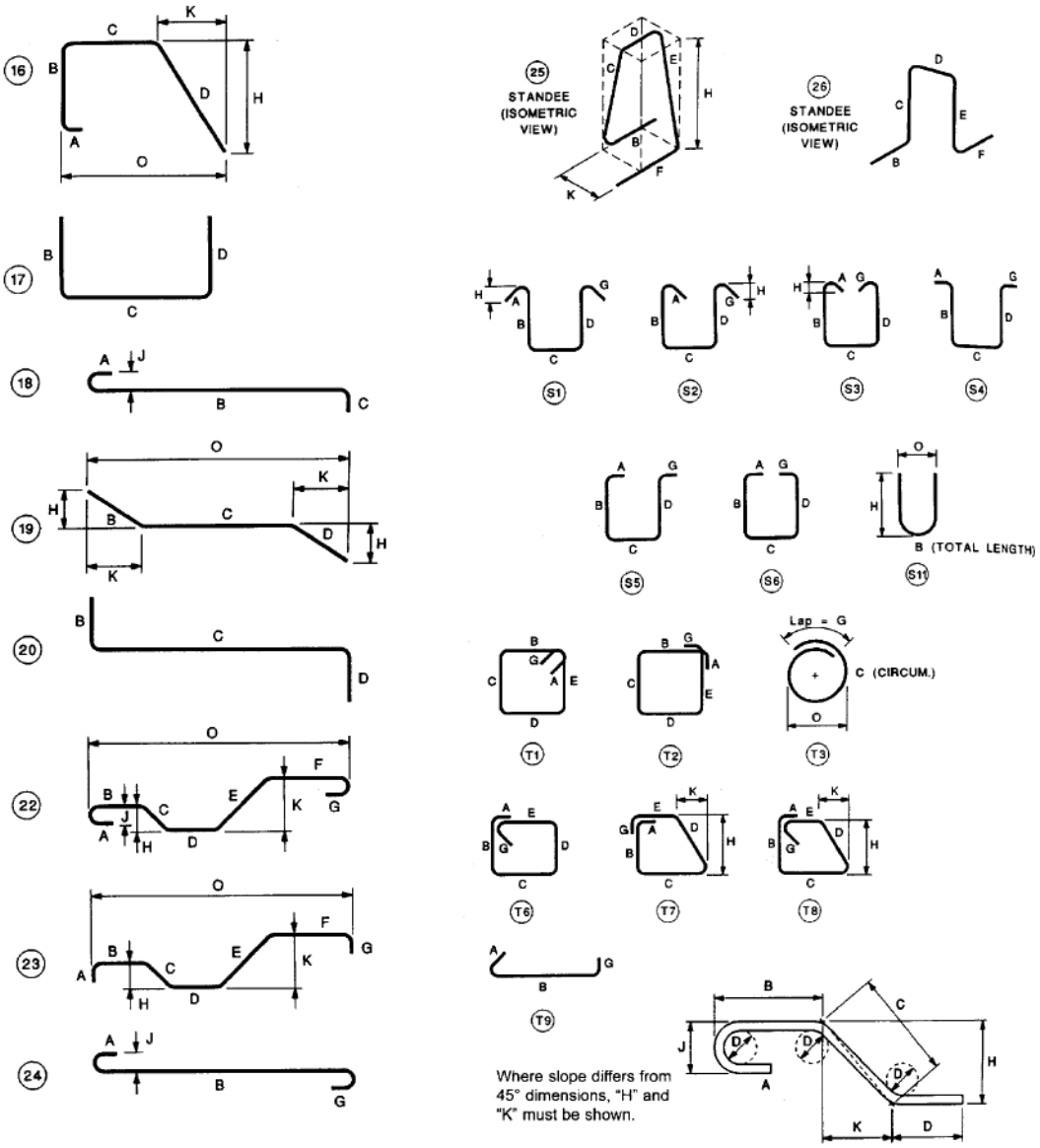


Where slope differs from 45° dimensions, "H" and "K" must be shown.

ENLARGED VIEW SHOWING BAR BENDING DETAILS

Notes:
1. All dimensions are out-to-out of bar except "A" and "G" on standard 180 and 135 degree hooks.
2. "J" dimensions on 180 degree hooks to be shown only where necessary to restrict hook size, otherwise ACI standard hooks are to be used.
3. Where "J" is not shown, "J" will be kept equal or less than "H" on Types 3, 5, and 22. Where "J" can exceed "H," it should be shown.
4. "H" dimension stirrups to be shown where necessary to fit within concrete.
5. Where bars are to be bent more accurately than standard fabricating tolerances, bending dimensions that require closer fabrication should have limits indicated.
6. Figures in circles show types.
7. For recommended diameter "D" of bends and hooks, see Section 3.7.1; for recommended hook dimensions, see Table 1.
8. Unless otherwise noted, diameter "D" is the same for all bends and hooks on a bar (except for Types 11 and 13).

Fig. 10—Typical bar bends.



ENLARGED VIEW SHOWING BAR BENDING DETAILS

Notes:
1. All dimensions are out-to-out of bar except "A" and "G" on standard 180 and 135 degree hooks.
2. "J" dimensions on 180 degree hooks to be shown only where necessary to restrict hook size, otherwise ACI standard hooks are to be used.
3. Where "J" is not shown, "J" will be kept equal or less than "H" on Types 3, 5, and 22. Where "J" can exceed "H," it should be shown.
4. "H" dimension stirrups to be shown where necessary to fit within concrete.
5. Where bars are to be bent more accurately than standard fabricating tolerances, bending dimensions that require closer fabrication should have limits indicated.
6. Figures in circles show types.
7. For recommended diameter "D" of bends and hooks, see Section 3.7.1; for recommended hook dimensions, see Table 1.
8. Type S1 through S6, S11, T1 through T3, T6 through T9: apply to bar sizes No. 3 through 8 (No. 10 through 25).
9. Unless otherwise noted, diameter "D" is the same for all bends and hooks on a bar (except for Types 11 and 13).

Fig. 10 (cont.)—Typical bar bends.

Precast Concrete Element Properties

IFC 2x4 Alpha has a single property set for various properties of precast pieces (*Pset_PrecastConcreteElementGeneral*). However, some properties are missing from this set, and the set mixes properties of designed pieces and manufactured pieces in a single set, which is semantically incorrect and will result in mistakes in their use. For example:

- the properties *TypeDesignator*, *ElementWeight* and *ElementGrossVolume* are properties that can be set during design and can apply to both building elements and building element types.
- the properties *ProductionLotId* and *SerialNumber* are only relevant for manufacturing purposes and can only be assigned values once a piece is fabricated. As such, they can apply only to building elements, but not to building element types.

The properties that are missing include:

1. geometric properties that define shape distortions subject to prestressing (**camber**), material shrinkage and creep (**shortening**) and from irregular support conditions (**twisting** - such as out of plane supports for double tee legs that induce warping), and;
2. classification properties that define grouping of identical or similar pieces for design and production (such as **piece mark** and **location number**), and;
3. design properties that define adjustments that should be made to forms to compensate for camber (**battering angles** at the start and end of each piece).

We propose replacing the single property set with two separate sets, one for designed pieces and one for manufactured pieces. The existing property set *Pset_PrecastConcreteElementGeneral* can be designated the property set for design, and a new property set *Pset_PrecastConcreteElementFabrication* can be added. The following table lists the properties that will be moved or added. The newly defined properties are shown in bold text. In addition, the *Pset_PrecastConcreteElementGeneral* set will have building element 'types' added to its list of applicable entities, as shown in the second table below.

Existing *PrecastConcreteElement-General*	**New *PrecastConcreteElementGeneral***	**New *PrecastConcreteElementFabrication***
TypeDesignator	TypeDesignator	TypeDesignator
ProductionLotId	~~ProductionLotId~~	ProductionLotId
SerialNumber	~~SerialNumber~~	SerialNumber
ElementWeight	ElementWeight*	ElementWeight*
ElementGrossVolume	ElementGrossVolume*	ElementGrossVolume*
ElementNetVolume	ElementNetVolume*	ElementNetVolume*
CornerChamfer	CornerChamfer	~~CornerChamfer~~
ManufacturingTolerance-Class	ManufacturingToleranceClass	~~ManufacturingToleranceClass~~
FormStrippingStrength	FormStrippingStrength	~~FormStrippingStrength~~
LiftingStrength	LiftingStrength	~~LiftingStrength~~
ReleaseStrength	ReleaseStrength	~~ReleaseStrength~~
MinimumAllowableSupport-Length	MinimumAllowableSupportLength	~~MinimumAllowableSupportLength~~
InitialTension	InitialTension	~~InitialTension~~
TendonRelaxation	TendonRelaxation	~~TendonRelaxation~~
TransportationStrength	TransportationStrength	TransportationStrength
	SupportDuringTransport-	SupportDuringTransportDescription

SupportDuringTransport-Description SupportDuringTransportDoc-Reference HollowCorePlugging	Description SupportDuringTransportDoc-Reference HollowCorePlugging **CamberAtMidspan** **BatterAtStart** **BatterAtEnd** **Twisting** **Shortening** **PieceMark** **DesignLocationNumber** **ElementSurfaceArea** **ElementFormedSurfaceArea**	SupportDuringTransportDoc-Reference ~~HollowCorePlugging~~ **PieceMark** **AsBuiltLocationNumber** **ActualProductionDate** **ActualErectionDate** **ElementSurfaceArea** **ElementFormedSurfaceArea**

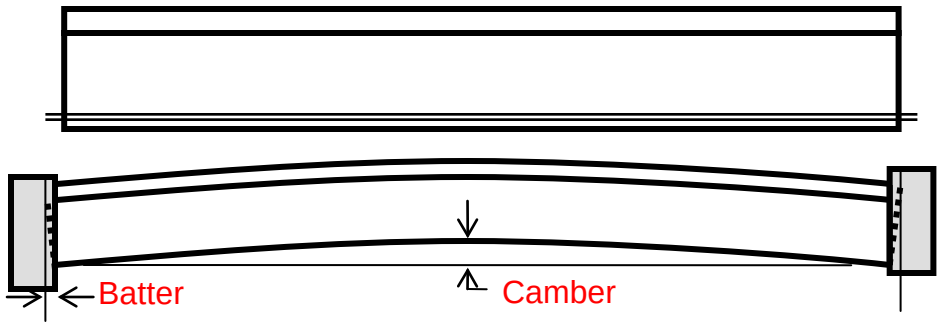
Property Set	**New PrecastConcreteElementGeneral**	**New PrecastConcreteElement-Fabrication**
Applicable Entities	IfcBeam IfcBeamType IfcBuildingElementPart IfcBuildingElementPartType IfcBuildingElementProxy IfcBuildingElementProxyType IfcColumn IfcColumnType IfcCovering IfcCoveringType IfcCurtainWall IfcCurtainWallType IfcFooting IfcFootingType IfcMember IfcMemberType IfcPile IfcPileType IfcRailing IfcRailingType IfcRamp IfcRampType IfcRampFlight IfcRampFlightType IfcRoof IfcRoofType IfcSlab IfcSlabType IfcStair IfcStairType IfcStairFlight IfcStairFlightType IfcWall IfcWallType IfcWallStandardCase	IfcBeam IfcBuildingElementPart IfcBuildingElementProxy IfcColumn IfcCovering IfcCurtainWall ~~IfcDoor~~ IfcFooting IfcMember IfcPile IfcRailing IfcRamp IfcRampFlight IfcRoof IfcSlab IfcStair IfcStairFlight IfcWall IfcWallStandardCase

* Note: these properties could also be provided using the IFCELEMENTQUANTITY entity with IFCQUANTITYVOLUME and IFCQUANTITYWEIGHT entities, but it is more cumbersome and not recommended.

The following table defines the properties that need to be added:

Name	Property Type	Data Type	Definition
CamberAtMidspan	IfcProperty SingleValue	IfcRatio Measure	The camber deflection, measured from the midpoint of a cambered face of a piece to the midpoint of the chord joining the ends of the same face, as shown in the figure below, divided by the original (nominal) straight length of the face of the piece.
BatterAtStart	IfcProperty SingleValue	IfcPlaneAngle Measure	The angle, in radians, by which the formwork at the starting face of a piece is to be rotated from the vertical in order to compensate for the rotation of the face that will occur once the piece is stripped from its form, inducing camber due to eccentric prestressing.
BatterAtEnd	IfcProperty SingleValue	IfcPlaneAngle Measure	The angle, in radians, by which the formwork at the ending face of a piece is to be rotated from the vertical in order to compensate for the rotation of the face that will occur once the piece is stripped from its form, inducing camber due to eccentric prestressing.

Name	Property Type	Data Type	Definition
Twisting	IfcProperty SingleValue	IfcPlaneAngle Measure	The angle, in radians, through which the end face of a precast piece is rotated with respect to its starting face, along its longitudinal axis, as a result of non-aligned supports. This measure is also termed the 'warping' angle.
Shortening	IfcProperty SingleValue	IfcRatio Measure	The ratio of the distance by which a precast piece is shortened after release from its form (due to compression induced by prestressing) to its original (nominal) length.
PieceMark	IfcProperty SingleValue	IfcLabel	Defines a unique piece for production purposes. All pieces with the same piece mark value are identical and interchangeable. The piece mark may be composed of sub-parts that have specific locally defined meaning (e.g. B-1A may denote a beam, of generic type '1' and specific shape 'A').
DesignLocationNumber	IfcProperty SingleValue	IfcLabel	Defines a unique location within a structure, the 'slot' for which the piece was designed.
AsBuiltLocationNumber	IfcProperty SingleValue	IfcLabel	Defines a unique location within a structure, the 'slot' into which the piece was installed. Where pieces share the same piece mark, they can be interchanged. The value is only known after erection.
ActualProductionDate	IfcProperty SingleValue	IfcTimeStamp	Production date (stripped from form).
ActualErectionDate	IfcProperty SingleValue	IfcTimeStamp	Date erected.
ElementSurfaceArea	IfcProperty SingleValue	IfcArea Measure	The gross surface area of the precast concrete element. Usually expressed in square metres (m^2).
ElementFormedSurface Area	IfcProperty SingleValue	IfcArea Measure	The net surface area of the precast concrete element that is in contact with a form. Usually expressed in square metres (m^2).



Additional property fields are needed for quality control purposes, but may be available in a generic IFC quality control entity with a property set (although there does not appear to be one yet). These include: stripping inspection date, result, action and inspector ID, pre-shipping inspection data, site arrival inspection data, and post-erection inspection data.

General Comments

Representation of nested components (rebar, embeds)

In general, the IFC schema does not determine the way in which entities are to behave within applications. It does not apply parametric constraints or fix behavior, such as cleaning up wall corners, etc; this is left to the internal logic of each application. The condition of rebars and other embeds within concrete elements is similarly not dealt with in any way that determines whether their volumes should be subtracted or not form the host element. The relationship between an *IfcBeam*, say, and an *IfcReinforcingBar* is implemented using *IfcRelAggregates.* The simplest approach is to accept that this is satisfactory, and to leave the question of whether the volume of the rebar should be subtracted from the volume of the concrete of the beam to the applications that will manipulate the data. As long as the basic condition of semantic identification of the parts is met, (i.e. the rebar is identified and recognized as a rebar) then each application can decide whether to subtract volume or not according to the task at hand and the degree of accuracy required. The same can be said for clash checking, for both soft and hard clashes.

CE: How can an application tell if some embedded element is subtracted from its host? Are there fixed conventions regarding which elements are subtracted, by type, and which are not? Your approach works if we know and can assume a standard case.

RS: We can add a property or a label and state whether the geometry of a RelAggregated entity is disjoint, overlapping or wholly embedded.

Improvements already implemented in IFC 2x4 Beta 1

The following is a list of the improvements we have requested that have already been implemented in IFC 2x4 Beta 1 (as of May 1st 2009):

1. Beam type enumerators: SPANDREL and HOLLOW_CORE types added.
2. Slabs with precast elements: IfcSlabElementedCase has been added.
3. Property set for rebar bending shapes according to ACI 315 has been added.
4. Pset_ComponentProductionRequirements has been added to provide production designations for embeds.
5. Precast module pieces catered for by enabling an IfcElementAssembly to have its own distinct geometry.