IFC Release 2.0 Certification testing – IFC Schema Refinement

Version	1.0
Date	2001-05-23
Status	Released
Authors	Kari Karstila & Kalle Serén; Eurostepsys Oy, IAI Forum Finland / BLIS

Document history

Version	Date	Description	Comments
0.6	2001-04-18	Early draft for comments	
0.9	2001-05-16	Pre-release version	All schema refinements included
1.0	2001-05-23	Released version 1.0	Final version for the Certification on 2001-05-23

Contents

1 Introduction

The IFC software certification testing is a process for testing software's conformance with a given IFC Release specification, and its subsets defined as so-called Views. The aim of the certification testing is to promote quality in IFC implementations, and demonstrate to end-users that the software passing the certification implement the IFC specification in a consistent way, hence being able to exchange IFC product data with other certified software unambiguously.

The IFC certification is done through a public certification workshop, and quality assurance work preceding the workshop. In the workshop the candidate applications have to demonstrate successful IFC data exchange with the other candidates (or already certified) applications. The quality assurance work before the public workshop is done in co-operation among the candidate applications aiming to ensure consistent IFC implementations. The work also includes cross-checking IFC data exchange between the various candidate applications.

2 IFC product model data checking

2.1 Elements of the IFC Object Model

The IFC Object Model defines the representation of AEC/FM product data for data exchange and sharing purposes. The IFC Object Model is represented as a schema using a data specification language called EXPRESS, which is an ISO standard [ISO 10303-11]. The IFC schema defines the Entity Types (or object classes), their properties and the relationships between the Entity Types. The IFC schema defines the representation of IFC product model data in the exchange files to be exchanged between applications and disciplines. The syntax for the IFC exchange file is defined by the IFC schema together with the encoding rules of the ISO 10303-21 standard [ISO 10303-21].

In addition to the Entity Types, their properties and relationships the IFC schema also defines constraints on the product data, i.e. constraints that may define limitations to the allowable value space for the instantiations of the IFC schema. For example the constraints define the correct instantiation of the IFC Project model composition structure. A correct IFC product model has to conform to the constraints defined in the IFC schema.

The types of the constraints within the IFC schema include:

- UNIQUEness constraints defining that an attribute or combination of attributes of an instance shall have a unique value.
- Domain-rules (WHERE-rules) which define constraints on the Entity Type level that apply to every instance of that Entity Type.
- INVERSE-relationships which may constrain the cardinalities of the relationships.

One type of constraints are global RULEs, but IFC schemas do not currently include them.

One additional element of the IFC schema is so-called DERIVEd attributes whose values are defined as expressions from values of other attributes or by separate functions.

The constraints and DERIVEd attribute values do not have any representations in the IFC exchange files. Relevant elements of the schema are FUNCTIONs, which are often used in DERIVEd attributes and WHERE-rules.

2.2 Constraint checking in certification

One crucial aspect for high quality IFC data exchange is to ensure that the IFC product data conforms to the constraints defined in the IFC schema. In the certification process this is tested by an application that is capable of checking an IFC product model against the constraints of the IFC schema, and reports on any constraint violations.

The coverage of the constraints within the IFC schema is limited. Therefore even though the IFC product model would conform to the constraints, the model could still be more or less meaningless. It is in the end the end-users of the applications that have the responsibility to create meaningful models using the applications.

2.3 Need for IFC schema refinement

IFC schemas are not perfect; they are often developed with limited resources and a tight time schedule, some parts borrowed from other models, etc. An additional problem in the testing of the schemas and especially their constraints is that when the schema is published there are typically no implementations of it available yet, hence no application generated test data is available. Therefore the IFC schema may contain some constraints that do not make sense or have some mistakes in them, and they need to be corrected for certification purposes.

3 IFC schema refinement principles

3.1 Basic principles

The objective of the IFC schema refinement is to solve any issues that the schema might have with regard to the constraints and derived attributes. The resulting refined IFC schema is then used in the certification testing when testing the candidate application generated IFC product model against the schema constraints. The aim is that no errors would be reported when applying automated checking to an IFC product model that conforms to refined schema with correct constraints.

The following basic principles apply for the IFC schema refinement:

- The schema refinement may only change the constraints, derived attribute expressions and functions of the IFC schema. This does not change the syntax of the IFC data exchange files.
- No changes will be made that would change the syntax of the IFC data exchange files.
- The refinement of the IFC R2.0 schema is done consistently with those relevant refinements already done in the IFC R1.5.1 schema for the certification process; and with the IFC R2x schema constraints that are applicable and correct.
- No changes in cardinalities of aggregate attributes will be made in order to avoid unnecessary conflicts with existing implementations based on the unrefined schema.

  NOTE – The IFC R2.0 schema contains some cardinality constraints that do not make sense in certain use case scenarios. One example is **IfcSpace** which has a mandatory attribute **BoundedBy : LIST [1:?] OF IfcSpaceBoundary** which means that a space should always have at least one boundary. Some early design stage applications may deal with spaces only. This kind of formal schema violation will be recognised but consciously ignored.

3.2 WHERE-rule refinements

The possible types of refinements for the domain-rules or WHERE-rules include:

- Commenting out WHERE-rules that do not make sense
- Minor fixes in some boundary conditions of the rules
- Use of refined FUNCTIONs in the rules.

No new WHERE-rules will be added.

3.3 FUNCTION refinements

The main refinement type for the IFC R2.0 schema is the replacement of some of the FUNCTIONs with totally new versions. Many of these FUNCTIONs are shared with the refined IFC R1.5.1 schema; and a couple of the FUNCTIONs introduced in IFC R2.0 can also be found in the IFC R2x schema.

3.4 DERIVEd attribute refinements

The refinement principles for the DERIVEd attributes include:

- No new DERIVEd attributes will be introduced.
- No existing DERIVEd attributes will be deleted.
- The expressions of the DERIVEd attributes may be changed. One case of this is the use of refined FUNCTIONs in the DERIVEd attribute expressions.

3.5 Testing of refined schema constraints

In conjunction with refining the IFC schema the refinements will also be tested. The method for testing the refined IFC schema and its constrainst consists of testing each constraint with two minimal IFC product model instantiations: one conforming to the constraint and one not conforming to the constraint. This is done in order to ensure that there is (some) confidence in proper functioning of the constraints.

References

IFC Release 1.5.1 Conformance schema. International Alliance for Interoperability, 2000.

IFC Release 2.0 specification.International Alliance for Interoperability, 1999-03-15.

IFC Release 2x specification. International Alliance for Interoperability, 2000.

IFC Release 2.0 Certification testing – Certification procedure. BLIS, 2001.

ISO 10303-11. Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: EXPRESS language reference manual. ISO, Geneva, 1994.

ISO 10303-21. Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure. ISO, Geneva, 1994.

Appendix 1. Glossary

Term	Description
Certification process	In the IAI context: A process for ensuring software's conformance with a given IFC Release specification, and its subsets defined as so-called Views. The aim is to promote quality in IFC implementations, and demonstrate to end-users that the software passing the certification implement the IFC specification in a consistent way.
Certification testing	The tests conducted as a part of the certification process that provides the evidence that the tested software products are conformant with the IFC Release specification the software products claim to support.
Constraint	A limitation on the allowable value space for an attribute or combinations of attributes.
Constraint checking	The process of ensuringthat the (IFC) product data conforms to the constraints defined in the (IFC) schema. In the certification process this is tested by an application that is capable of checking an IFC product model against the constraints of the IFC schema, and reports on any constraint violations.
DERIVEd attribute	A derived attribute represents a property whose value is computed by evaluating an expression. Derived attributes are declared following the DERIVE keyword. [ISO 10303-11]
Domain rule	Domain rules constrain the values of individual attributes of combinations of attributes for every entity instance. All domain rules follow the WHERE keyword. [ISO 10303-11]
IFC Object Model	The IFC Object Model defines the representation of AEC/FM product data for data exchange and sharing purposes.
IFC schema	The IFC Object Model represented as a schema using a data specification language called EXPRESS, which is an ISO standard [ISO 10303-11]. The IFC schema defines the Entity Types (or object classes), their properties and the relationships between the Entity Types as well as the constraints.
IFC product model	The instantiation of data for a specific building or part of building that conforms to the IFC schema.
UNIQUE	A uniqueness constraint for individual attributes or combinations of attributes may be specified in a uniqueness rule. The uniqueness rule follow the UNIQUE keyword and specify either a single attribute name or a list of attribute names. [ISO 10303-11] Constraint defining that an attribute or combination of attributes of an instance shall have a unique value.
WHERE-rule	See Domain rule.

Appendix 2. Refined WHERE-rules.

KKa = Kari Karstila, Eurostepsys Oy

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

ENTITY IfcTrimmedCurve
  SUBTYPE OF (IfcBoundedCurve);
  BasisCurve : IfcCurve;
  Trim1 : SET [1:2] OF IfcTrimmingSelect;
  Trim2 : SET [1:2] OF IfcTrimmingSelect;
  SenseAgreement : BOOLEAN;
  MasterRepresentation : IfcTrimmingPreference;
  WHERE
  WR41: (HIINDEX(Trim1) = 1) XOR (TYPEOF(Trim1[1])
  <> TYPEOF(Trim1[2]));
  WR42: (HIINDEX(Trim2) = 1) XOR (TYPEOF(Trim2[1])
  <> TYPEOF(Trim2[2]));
  WR43: NOT('IFC20_LONGFORM.IFCBOUNDEDCURVE' IN
  TYPEOF(BasisCurve));
END_ENTITY;
	ENTITY IfcTrimmedCurve
	  SUBTYPE OF (IfcBoundedCurve);
	  BasisCurve : IfcCurve;
	  Trim1 : SET [1:2] OF IfcTrimmingSelect;
	  Trim2 : SET [1:2] OF IfcTrimmingSelect;
	  SenseAgreement : BOOLEAN;
	  MasterRepresentation : IfcTrimmingPreference;
	  WHERE
	  WR41: (HIINDEX(Trim1) = 1) OR (TYPEOF(Trim1[1])
	  <> TYPEOF(Trim1[2]));
	  WR42: (HIINDEX(Trim2) = 1) OR (TYPEOF(Trim2[1])
	  <> TYPEOF(Trim2[2]));
	  WR43: NOT('IFC20_LONGFORM.IFCBOUNDEDCURVE' IN
	  TYPEOF(BasisCurve));
	END_ENTITY;
		WR41 & WR42: *XOR*s changed to *OR*
		By: KKa
		Date: 11-May-2001

ENTITY IfcShapeRepresentation
  SUBTYPE OF (IfcRepresentation);
  Items : SET [1:?] OF IfcGeometricRepresentationItem;
  INVERSE
  OfProductDefinitionShape :
  SET [0:1] OF IfcProductDefinitionShape
  FOR ShapeRepresentations;
  OfShapeAspect : SET [0:1] OF IfcShapeAspect
  FOR ShapeRepresentations;
  WHERE
  WR22: (HIINDEX(OfProductDefinitionShape) = 1) XOR
  (HIINDEX(OfShapeAspect) = 1);
  WR23:
  'IFC20_LONGFORM.IFCGEOMETRICREPRESENTATIONCONTEXT'
  IN TYPEOF(SELF\IfcRepresentation.ContextOfItems);
  WR24: SELF\IfcRepresentation.RepresentationType IN
  ['BoundingBox','Standard','Advanced','Arbitrary',
  'Brep','UserDefined','NotDefined'];
  WR25: ((RepresentationType = 'BoundingBox') AND
  (HIINDEX(Items) = 1) AND
  ('IFC20_LONGFORM.IFCBOUNDINGBOX' IN
  TYPEOF(Items[1]))) OR
  (RepresentationType<>'BoundingBox');
END_ENTITY;
	ENTITY IfcShapeRepresentation
	  SUBTYPE OF (IfcRepresentation);
	  Items : SET [1:?] OF IfcGeometricRepresentationItem;
	  INVERSE
	  OfProductDefinitionShape :
	  SET [0:1] OF IfcProductDefinitionShape
	  FOR ShapeRepresentations;
	  OfShapeAspect : SET [0:1] OF IfcShapeAspect
	  FOR ShapeRepresentations;
	  WHERE
	  WR22: (HIINDEX(OfProductDefinitionShape) = 1) XOR
	  (HIINDEX(OfShapeAspect) = 1);
	  WR23:
	  'IFC20_LONGFORM.IFCGEOMETRICREPRESENTATIONCONTEXT'
	  IN TYPEOF(SELF\IfcRepresentation.ContextOfItems);
	  WR24: SELF\IfcRepresentation.RepresentationType IN
	  ['BoundingBox','Standard','Advanced','Arbitrary',
	  'Brep','2D_Geometry','UserDefined','NotDefined'];
	  WR25: ((RepresentationType = 'BoundingBox') AND
	  (HIINDEX(Items) = 1) AND
	  ('IFC20_LONGFORM.IFCBOUNDINGBOX' IN
	  TYPEOF(Items[1]))) OR
	  (RepresentationType <> 'BoundingBox');
	END_ENTITY;
		WR24: *'2D_Geometry'* added
		By: KKa
		Date: 11-May-2001

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

ENTITY IfcBuilding
  SUBTYPE OF (IfcProduct);
  BuildingReference : OPTIONAL STRING;
  BuildingName : OPTIONAL STRING;
  calcTotalHeight : OPTIONAL IfcLengthMeasure;
  calcSiteCoverage : OPTIONAL IfcAreaMeasure;
  calcTotalVolume : OPTIONAL IfcVolumeMeasure;
  ElevationOfRefHeight : OPTIONAL IfcLengthMeasure;
  ElevationOfTerrain : OPTIONAL IfcLengthMeasure;
  INVERSE
  ServicedBySystems :
  SET [0:?] OF IfcRelServicesBuildings
  FOR RelatedBuildings;
  WHERE
  WR41:
  SIZEOF(QUERY(Temp <* SELF\IfcObject.IsContainedBy |
  (Temp.RelationshipType = ProjectContainer) AND
  (Temp.ContainedOrReferenced = Referenced))) = 1;
  WR42:
  SIZEOF(QUERY(Temp <* SELF\IfcObject.IsContainedBy |
  (Temp.RelationshipType = SiteContainer) AND
  (Temp.ContainedOrReferenced = Contained))) <= 1;
END_ENTITY;
	ENTITY IfcBuilding
	  SUBTYPE OF (IfcProduct);
	  BuildingReference : OPTIONAL STRING;
	  BuildingName : OPTIONAL STRING;
	  calcTotalHeight : OPTIONAL IfcLengthMeasure;
	  calcSiteCoverage : OPTIONAL IfcAreaMeasure;
	  calcTotalVolume : OPTIONAL IfcVolumeMeasure;
	  ElevationOfRefHeight : OPTIONAL IfcLengthMeasure;
	  ElevationOfTerrain : OPTIONAL IfcLengthMeasure;
	  INVERSE
	  ServicedBySystems :
	  SET [0:?] OF IfcRelServicesBuildings
	  FOR RelatedBuildings;
	  WHERE
	  WR41:
	  SIZEOF(QUERY(Temp<*SELF\IfcObject.IsContainedBy |
	  (Temp.RelationshipType = ProjectContainer) AND
	  (Temp.ContainedOrReferenced = Referenced))) <= 1;
	  WR42:
	  SIZEOF(QUERY(Temp<*SELF\IfcObject.IsContainedBy |
	  (Temp.RelationshipType = SiteContainer) AND
	  (Temp.ContainedOrReferenced = Contained))) <= 1;
	END_ENTITY;
		WR41: SIZEOF query constraint <= *1* changed from = *1*.
		By: KKa
		Date: 11-May-2001

ENTITY IfcBuildingStorey
  SUBTYPE OF (IfcProduct);
  BuildingStoreyReference : OPTIONAL STRING;
  BuildingStoreyName : OPTIONAL STRING;
  Elevation : IfcLengthMeasure;
  calcTotalHeight : OPTIONAL IfcLengthMeasure;
  calcTotalArea : OPTIONAL IfcAreaMeasure;
  calcTotalVolume : OPTIONAL IfcVolumeMeasure;
  WHERE
  WR41:
  SIZEOF(QUERY(Temp <* SELF\IfcObject.IsContainedBy |
  (Temp.RelationshipType = ProjectContainer) AND
  (Temp.ContainedOrReferenced = Referenced))) = 1;
  WR42:
  SIZEOF(QUERY(Temp <* SELF\IfcObject.IsContainedBy |
  (Temp.RelationshipType = BuildingContainer) AND
  (Temp.ContainedOrReferenced = Contained))) <= 1;
END_ENTITY;
	ENTITY IfcBuildingStorey
	  SUBTYPE OF (IfcProduct);
	  BuildingStoreyReference : OPTIONAL STRING;
	  BuildingStoreyName : OPTIONAL STRING;
	  Elevation : IfcLengthMeasure;
	  calcTotalHeight : OPTIONAL IfcLengthMeasure;
	  calcTotalArea : OPTIONAL IfcAreaMeasure;
	  calcTotalVolume : OPTIONAL IfcVolumeMeasure;
	  WHERE
	  WR41:
	  SIZEOF(QUERY(Temp<* SELF\IfcObject.IsContainedBy |
	  (Temp.RelationshipType = ProjectContainer) AND
	  (Temp.ContainedOrReferenced = Referenced))) <= 1;
	  WR42:
	  SIZEOF(QUERY(Temp<* SELF\IfcObject.IsContainedBy |
	  (Temp.RelationshipType = BuildingContainer) AND
	  (Temp.ContainedOrReferenced = Contained))) <= 1;
	END_ENTITY;
		WR41: SIZEOF query constraint <=*1* changed from = *1*.
		By: KKa
		Date: 11-May-2001

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

ENTITY IfcElement
  ABSTRACT SUPERTYPE OF
  (ONEOF(IfcBuildingElement, IfcOpeningElement))
  SUBTYPE OF (IfcProduct);
  INVERSE
  ConnectedTo : SET [0:?] OF IfcRelConnectsElements
  FOR RelatingElement;
  ConnectedFrom : SET [0:?] OF IfcRelConnectsElements
  FOR RelatedElement;
  IsAssemblyThrough : SET [0:1] OF
  IfcRelAssemblesElements
  FOR RelatingElement;
  PartOfAssembly : SET [0:1] OF IfcRelAssemblesElements
  FOR RelatedElements;
  WHERE
  WR41:
  SIZEOF(QUERY(Temp<* SELF\IfcObject.IsContainedBy |
  (Temp.RelationshipType = ProjectContainer) AND
  (Temp.ContainedOrReferenced = Referenced))) = 1;
  WR42:
  SIZEOF(QUERY(Temp <* SELF\IfcObject.IsContainedBy |
  (((Temp.RelationshipType = SiteContainer) OR
  (Temp.RelationshipType = BuildingContainer) OR
  (Temp.RelationshipType = BuildingStoreyContainer)
  OR (Temp.RelationshipType = SpaceContainer)) AND
  (Temp.ContainedOrReferenced = Contained)))) <= 1;
END_ENTITY;
	ENTITY IfcElement
	  ABSTRACT SUPERTYPE OF (ONEOF(IfcBuildingElement
	  ,IfcOpeningElement))
	  SUBTYPE OF (IfcProduct);
	  INVERSE
	  ConnectedTo : SET [0:?] OF
	IfcRelConnectsElements
	  FOR RelatingElement;
	  ConnectedFrom : SET [0:?] OF
	IfcRelConnectsElements
	  FOR RelatedElement;
	  IsAssemblyThrough : SET [0:1] OF
	IfcRelAssemblesElements
	  FOR RelatingElement;
	  PartOfAssembly : SET [0:1] OF
	IfcRelAssemblesElements
	  FOR RelatedElements;
	END_ENTITY;
		WR41: Moved to IfcBuildingElement
		WR42: Moved to IfcBuildingElement
		
		By: KKa
		Date: 22-May-2001

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

ENTITY IfcBuildingElement
  SUPERTYPE OF (ONEOF(
  IfcBeam, IfcBuiltIn, IfcColumn, IfcCovering,
  IfcCurtainWall, IfcDiscreteElement,
  IfcDistributionElement, IfcDoor, IfcDoorLining,
  IfcDoorPanel, IfcElectricalAppliance, IfcEquipment,
  IfcFurniture, IfcPermeableCovering,IfcRailing,
  IfcRamp, IfcRampFlight, IfcRoof, IfcSlab, IfcStair,
  IfcStairFlight, IfcSystemFurnitureElement,
  IfcVisualScreen, IfcWall, IfcWindow, IfcWindowLining,
  IfcWindowPanel))
  SUBTYPE OF (IfcElement);
  HasMaterial : OPTIONAL IfcMaterialSelect;
  INVERSE
  ProvidesBoundaries : SET [0:?] OF
  IfcRelSeparatesSpaces
  FOR RelatingBuildingElement;
  HasOpenings : SET [0:?] OF IfcRelVoidsElement
  FOR RelatingBuildingElement;
  FillsVoids : SET [0:?] OF IfcRelFillsElement
  FOR RelatedBuildingElement;
END_ENTITY;
	ENTITY IfcBuildingElement
	  SUPERTYPE OF (ONEOF(
	  IfcBeam, IfcBuiltIn, IfcColumn, IfcCovering,
	  IfcCurtainWall, IfcDiscreteElement,
	  IfcDistributionElement, IfcDoor, IfcDoorLining,
	  IfcDoorPanel, IfcElectricalAppliance, IfcEquipment,
	  IfcFurniture, IfcPermeableCovering,IfcRailing,
	  IfcRamp, IfcRampFlight, IfcRoof, IfcSlab, IfcStair,
	  IfcStairFlight, IfcSystemFurnitureElement,
	  IfcVisualScreen, IfcWall, IfcWindow, IfcWindowLining,
	  IfcWindowPanel))
	  SUBTYPE OF (IfcElement);
	  HasMaterial : OPTIONAL IfcMaterialSelect;
	  INVERSE
	  ProvidesBoundaries : SET [0:?] OF
	  IfcRelSeparatesSpaces
	  FOR RelatingBuildingElement;
	  HasOpenings : SET [0:?] OF IfcRelVoidsElement
	  FOR RelatingBuildingElement;
	  FillsVoids : SET [0:?] OF IfcRelFillsElement
	  FOR RelatedBuildingElement;
	  WHERE
	  WR51: SIZEOF(QUERY(Temp <*
	  SELF\IfcObject.IsContainedBy |
	  (Temp.RelationshipType = ProjectContainer)
	  AND (Temp.ContainedOrReferenced = Referenced)))
	  <= 1;
	  WR52: SIZEOF(QUERY(Temp <*
	  SELF\IfcObject.IsContainedBy |
	  (((Temp.RelationshipType = SiteContainer)
	  OR (Temp.RelationshipType = BuildingContainer)
	  OR (Temp.RelationshipType =
	  BuildingStoreyContainer)
	  OR (Temp.RelationshipType = SpaceContainer))
	  AND (Temp.ContainedOrReferenced = Contained))))
	  = 1;
	END_ENTITY;
		WR41 & WR42 moved from IfcElement
		By: KKa
		Date: 22-May-2001
		
		WR41: SIZEOF query constraint <= *1* changed from = *1*.
		WR42: SIZEOF query constraint = *1* changed from <= *1*.
		By: KKa
		Date: 11-May-2001

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

ENTITY IfcSite
  SUBTYPE OF (IfcProduct);
  RefLatitude : OPTIONAL IfcCompoundPlaneAngleMeasure;
  RefLongitude : OPTIONAL IfcCompoundPlaneAngleMeasure;
  RefElevation : OPTIONAL IfcLengthMeasure;
  TrueNorth : OPTIONAL IfcDirection;
  calcSitePerimeter : OPTIONAL
  IfcPositiveLengthMeasure;
  calcSiteArea : OPTIONAL IfcAreaMeasure;
  WHERE
  WR41:
  SIZEOF(QUERY(Temp <* SELF\IfcObject.IsContainedBy |
  (Temp.RelationshipType = ProjectContainer) AND
  (Temp.ContainedOrReferenced = Referenced))) = 1;
END_ENTITY;
	ENTITY IfcSite
	  SUBTYPE OF (IfcProduct);
	  RefLatitude : OPTIONAL IfcCompoundPlaneAngleMeasure;
	  RefLongitude : OPTIONAL IfcCompoundPlaneAngleMeasure;
	  RefElevation : OPTIONAL IfcLengthMeasure;
	  TrueNorth : OPTIONAL IfcDirection;
	  calcSitePerimeter : OPTIONAL
	  IfcPositiveLengthMeasure;
	  calcSiteArea : OPTIONAL IfcAreaMeasure;
	  WHERE
	  WR41:
	  SIZEOF(QUERY(Temp<* SELF\IfcObject.IsContainedBy |
	  (Temp.RelationshipType = ProjectContainer) AND
	  (Temp.ContainedOrReferenced = Contained))) <= 1;
	END_ENTITY;
		WR41: *Referenced* changed to *Contained*. SIZEOF query constraint <= *1* changed from = *1*.
		By: KKa
		Date: 11-May-2001

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

ENTITY IfcSpace
  SUPERTYPE OF (ONEOF(
  IfcFireCompartment
  ,IfcWorkstation))
  SUBTYPE OF (IfcSpatialElement);
  BoundedBy : LIST [1:?] OF IfcSpaceBoundary;
  InteriorOrExteriorSpace : IfcInternalOrExternalEnum;
  SpaceReference : OPTIONAL STRING;
  SpaceName : OPTIONAL STRING;
  calcTotalPerimeter : OPTIONAL
  IfcPositiveLengthMeasure;
  calcTotalArea : OPTIONAL IfcAreaMeasure;
  calcTotalVolume : OPTIONAL IfcVolumeMeasure;
  calcAverageHeight : OPTIONAL
  IfcPositiveLengthMeasure;
  calcAverageGrossHeight : OPTIONAL
  IfcPositiveLengthMeasure;
  calcAverageClearHeight : OPTIONAL
  IfcPositiveLengthMeasure;
  calcElevationWithFlooring : OPTIONAL
  IfcLengthMeasure;
  INVERSE
  IsAssemblyThrough : SET [0:1] OF
  IfcRelAssemblesSpaces
  FOR RelatingSpace;
  PartOfAssembly : SET [0:1] OF
  IfcRelAssemblesSpaces
  FOR RelatedSpaces;
  WHERE
  WR52:
  SIZEOF(QUERY(Temp <* SELF\IfcObject.IsContainedBy |
  (Temp.RelationshipType = ProjectContainer) AND
  (Temp.ContainedOrReferenced = Referenced))) = 1;
  WR53:
  SIZEOF(QUERY(Temp <* SELF\IfcObject.IsContainedBy |
  (((Temp.RelationshipType = SiteContainer) OR
  (Temp.RelationshipType = BuildingStoreyContainer))
  AND
  (Temp.ContainedOrReferenced = Contained)))) <= 1;
END_ENTITY;
	ENTITY IfcSpace
	  SUPERTYPE OF (ONEOF(
	  IfcFireCompartment
	  ,IfcWorkstation))
	  SUBTYPE OF (IfcSpatialElement);
	  BoundedBy : LIST [1:?] OF IfcSpaceBoundary;
	  InteriorOrExteriorSpace : IfcInternalOrExternalEnum;
	  SpaceReference : OPTIONAL STRING;
	  SpaceName : OPTIONAL STRING;
	  calcTotalPerimeter : OPTIONAL
	  IfcPositiveLengthMeasure;
	  calcTotalArea : OPTIONAL IfcAreaMeasure;
	  calcTotalVolume : OPTIONAL IfcVolumeMeasure;
	  calcAverageHeight : OPTIONAL
	  IfcPositiveLengthMeasure;
	  calcAverageGrossHeight : OPTIONAL
	  IfcPositiveLengthMeasure;
	  calcAverageClearHeight : OPTIONAL
	  IfcPositiveLengthMeasure;
	  calcElevationWithFlooring : OPTIONAL
	  IfcLengthMeasure;
	  INVERSE
	  IsAssemblyThrough : SET [0:1] OF
	  IfcRelAssemblesSpaces
	  FOR RelatingSpace;
	  PartOfAssembly : SET [0:1] OF
	  IfcRelAssemblesSpaces
	  FOR RelatedSpaces;
	  WHERE
	  WR52:
	  SIZEOF(QUERY(Temp <* SELF\IfcObject.IsContainedBy |
	  (Temp.RelationshipType = ProjectContainer) AND
	  (Temp.ContainedOrReferenced = Referenced))) <= 1;
	  WR53:
	  SIZEOF(QUERY(Temp<* SELF\IfcObject.IsContainedBy |
	  (((Temp.RelationshipType = SiteContainer) OR
	  (Temp.RelationshipType = BuildingStoreyContainer))
	  AND
	  (Temp.ContainedOrReferenced = Contained)))) = 1;
	END_ENTITY;
		WR52: SIZEOF query constraint <= *1* changed from = *1*.
		WR53: SIZEOF query constraint = *1* changed from <= *1*.
		
		By: KKa
		Date: 11-May-2001

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

ENTITY IfcBeam
  SUBTYPE OF (IfcBuildingElement);
  calcBeamSectionArea : OPTIONAL IfcAreaMeasure;
  calcBeamVolume : OPTIONAL IfcVolumeMeasure;
  WHERE
  WR62: 'IFC20_LONGFORM.IFCMATERIALLIST' IN
  TYPEOF(SELF\IfcBuildingElement.HasMaterial);
END_ENTITY;
	ENTITY IfcBeam
	  SUBTYPE OF (IfcBuildingElement);
	  calcBeamSectionArea : OPTIONAL IfcAreaMeasure;
	  calcBeamVolume : OPTIONAL IfcVolumeMeasure;
	END_ENTITY;
		WR62: deleted
		By: KKa
		Date: 11-May-2001

ENTITY IfcColumn
  SUBTYPE OF (IfcBuildingElement);
  calcColumnSectionArea : OPTIONAL IfcAreaMeasure;
  calcColumnVolume : OPTIONAL IfcVolumeMeasure;
  WHERE
  WR62: 'IFC20_LONGFORM.IFCMATERIALLIST' IN
  TYPEOF(SELF\IfcBuildingElement.HasMaterial);
END_ENTITY;
	ENTITY IfcColumn
	  SUBTYPE OF (IfcBuildingElement);
	  calcColumnSectionArea : OPTIONAL IfcAreaMeasure;
	  calcColumnVolume : OPTIONAL IfcVolumeMeasure;
	END_ENTITY;
		WR62: deleted
		By: KKa
		Date: 11-May-2001

ENTITY IfcDoorPanel
  SUBTYPE OF (IfcBuildingElement);
  PredefinedType : IfcDoorPanelTypeEnum;
  WHERE
  WR61: 'IFC20_LONGFORM.IFCMATERIALLIST' IN
  TYPEOF(SELF\IfcBuildingElement.HasMaterial);
  WR62: ((PredefinedType =
  IfcDoorPanelTypeEnum.UserDefined) AND
  EXISTS(SELF\IfcObject.UserDefinedType)) OR
  ((PredefinedType <>
  IfcDoorPanelTypeEnum.UserDefined) AND
  NOT(EXISTS(SELF\IfcObject.UserDefinedType)));
END_ENTITY;
	ENTITY IfcDoorPanel
	  SUBTYPE OF (IfcBuildingElement);
	  PredefinedType : IfcDoorPanelTypeEnum;
	  WHERE
	  WR62: ((PredefinedType =
	  IfcDoorPanelTypeEnum.UserDefined) AND
	  EXISTS(SELF\IfcObject.UserDefinedType)) OR
	  ((PredefinedType <>
	  IfcDoorPanelTypeEnum.UserDefined) AND
	  NOT(EXISTS(SELF\IfcObject.UserDefinedType)));
	END_ENTITY;
		WR61: deleted
		By: KKa
		Date: 11-May-2001

Appendix 3. Refined CONSTANTS and FUNCTIONS.

KSn = Kalle Serén, Eurostepsys Oy

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description
	CONSTANTS	

<No constants in original specification>	CONSTANT
	  IfcDummyGri : IfcGeometricRepresentationItem :=
	  IfcGeometricRepresentationItem();
	END_CONSTANT;
		New constant added to resolve problem with geometry related functions
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

	FUNCTIONS	

FUNCTION IfcBuild2Axes
  (RefDirection : IfcDirection) : LIST [2:2] OF
  IfcDirection;
 	
  LOCAL
  U : LIST[2:2] OF IfcDirection
  := [IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0]),
  IfcGeometricRepresentationItem() ||
  IfcDirection([0.0,1.0])];
  END_LOCAL;
 	
  U[1] := NVL(IfcNormalise(RefDirection),
  IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0]));
  U[2] := IfcOrthogonalComplement(U[1]);
  RETURN(U);
 	
END_FUNCTION;
	FUNCTION IfcBuild2Axes
	  (RefDirection : IfcDirection) : LIST [2:2] OF
	  IfcDirection;
	
	  LOCAL
	  D : IfcDirection
	  := NVL(IfcNormalise(RefDirection),
	  IfcDummyGri || IfcDirection([1.0,0.0]));
	  END_LOCAL;
	
	  RETURN([D, IfcOrthogonalComplement(D)]);
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

FUNCTION IfcBuildAxes
  (Axis, RefDirection : IfcDirection)
  : LIST [3:3] OF IfcDirection;
 	
  LOCAL
  U : LIST [3:3] OF IfcDirection
  := [IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0,0.0]),
  IfcGeometricRepresentationItem() ||
  IfcDirection([0.0,1.0,0.0]),
  IfcGeometricRepresentationItem() ||
  IfcDirection([0.0,0.0,1.0])];
  END_LOCAL;
 	
  U[3] := NVL(IfcNormalise(Axis),
  IfcGeometricRepresentationItem() ||
  IfcDirection([0.0,0.0,1.0]));
  U[1] := IfcFirstProjAxis(U[3], RefDirection);
  U[2] := IfcNormalise(
  IfcCrossProduct(U[3],U[1])).Orientation;
  RETURN(U);
 	
END_FUNCTION;
	FUNCTION IfcBuildAxes
	  (Axis, RefDirection : IfcDirection)
	  : LIST [3:3] OF IfcDirection;
	
	  LOCAL
	  D1, D2 : IfcDirection;
	  END_LOCAL;
	
	  D1 := NVL(IfcNormalise(Axis), IfcDummyGri ||
	  IfcDirection([0.0,0.0,1.0]));
	  D2 := IfcFirstProjAxis(D1, RefDirection);
	  RETURN
	  ([D2,
	  IfcNormalise(IfcCrossProduct(D1,D2)).Orientation,
	  D1]);
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

FUNCTION IfcCircleProfileIntoCurve
  (ProfileDef : IfcCircleProfileDef) : IfcTrimmedCurve;
 	
  LOCAL
  Pos : IfcAxis2Placement2D;
  Circle : IfcCircle;
  ResCurve : IfcTrimmedCurve;
  END_LOCAL;
 	
  Pos := ProfileDef\IfcAttDrivenProfileDef.Position;
  Circle := IfcGeometricRepresentationItem() ||
  IfcCurve() || IfcConic(Pos) ||
  IfcCircle(ProfileDef.Radius);
  ResCurve := IfcGeometricRepresentationItem() ||
  IfcCurve() || IfcBoundedCurve() ||
  IfcTrimmedCurve(Circle, [0.0], [2*PI],
  TRUE, Parameter);
  RETURN (ResCurve);
 	
END_FUNCTION;
	FUNCTION IfcCircleProfileIntoCurve
	  (ProfileDef : IfcCircleProfileDef) : IfcTrimmedCurve;
	
	  LOCAL
	  Pos : IfcAxis2Placement2D;
	  Circle : IfcCircle;
	  ResCurve : IfcTrimmedCurve;
	  END_LOCAL;
	
	  Pos := ProfileDef\IfcAttDrivenProfileDef.Position;
	  Circle := IfcDummyGri || IfcCurve() ||
	  IfcConic(Pos) ||
	  IfcCircle(ProfileDef.Radius);
	  ResCurve := IfcDummyGri || IfcCurve() ||
	  IfcBoundedCurve() ||
	  IfcTrimmedCurve(Circle, [0.0], [2*PI],
	  TRUE, Parameter);
	  RETURN (ResCurve);
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

```
FUNCTION IfcCrossProduct
  (Arg1, Arg2 : IfcDirection) : IfcVector;

  LOCAL
    Mag    :  REAL := 0.0;
    V1,V2  :  LIST[3:3] OF REAL := [0.0:3];
    Res    :  IfcDirection
           := IfcGeometricRepresentationItem() ||
              IfcDirection([1.0,0.0,0.0]);
    Result :  IfcVector
           := IfcGeometricRepresentationItem() ||
              IfcVector(IfcGeometricRepresentationItem()
              || IfcDirection([1.0,0.0,0.0]), 1.0);
  END_LOCAL;

  IF (NOT EXISTS (Arg1) OR (Arg1.Dim = 2)) OR
     (NOT EXISTS (Arg2) OR (Arg2.Dim = 2)) THEN
  RETURN(?);
  ELSE
    BEGIN
      V1  := IfcNormalise(Arg1).DirectionRatios;
      V2  := IfcNormalise(Arg2).DirectionRatios;
      Res.DirectionRatios[1] :=
                           (V1[2]*V2[3] - v1[3]*v2[2]);
      Res.DirectionRatios[2] :=
                           (V1[3]*V2[1] - v1[1]*v2[3]);
      Res.DirectionRatios[3] :=
                           (V1[1]*V2[2] - v1[2]*v2[1]);
      Mag := 0.0;
      REPEAT i := 1 TO 3;
        Mag := Mag +
          Res.DirectionRatios[i]*Res.DirectionRatios[i];
      END_REPEAT;
      IF (Mag > 0.0) THEN
        Result.Orientation := Res;
        Result.Magnitude   := SQRT(Mag);
      ELSE
        Result.Orientation := Arg1;
        Result.Magnitude   := 0.0;
      END_IF;
      RETURN(Result);
    END;
  END_IF;

END_FUNCTION;
```

```
FUNCTION IfcCrossProduct
   (Arg1, Arg2 : IfcDirection) : IfcVector;

  LOCAL
    Mag : REAL;
    Res : IfcDirection;
    V1,V2  : LIST[3:3] OF REAL;
    Result : IfcVector;
  END_LOCAL;

  IF (NOT EXISTS (Arg1) OR (Arg1.Dim = 2)) OR
     (NOT EXISTS (Arg2) OR (Arg2.Dim = 2)) THEN
  RETURN(?);
  ELSE
    BEGIN
      V1  := IfcNormalise(Arg1).DirectionRatios;
      V2  := IfcNormalise(Arg2).DirectionRatios;
      Res := IfcDummyGri ||
             IfcDirection([(V1[2]*V2[3] - V1[3]*V2[2]),
                           (V1[3]*V2[1] - V1[1]*V2[3]),
                           (V1[1]*V2[2] - V1[2]*V2[1])]);
      Mag := 0.0;
      REPEAT i := 1 TO 3;
        Mag := Mag +
          Res.DirectionRatios[i]*Res.DirectionRatios[i];
      END_REPEAT;
      IF (Mag > 0.0) THEN
        Result := IfcDummyGri ||
                  IfcVector(Res, SQRT(Mag));
      ELSE
        Result := IfcDummyGri || IfcVector(Arg1, 0.0);
      END_IF;
      RETURN(Result);
    END;
  END_IF;

END_FUNCTION;
```

Function changed to resolve geometry related problems

By: KSn
Date: 18-Apr-2001
Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

FUNCTION IfcDotProduct
  (Arg1, Arg2 : IfcDirection) : REAL;
 	
  LOCAL
  Scalar : REAL := 0.0;
  Ndim : INTEGER := 0;
  Vec1,Vec2 : IfcDirection
  := IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0,0.0]);
  END_LOCAL;
 	
  IF NOT EXISTS (Arg1) OR NOT EXISTS (Arg2) THEN
  Scalar := ?;
  ELSE
  IF (Arg1.dim <> Arg2.dim) THEN
  Scalar := ?;
  ELSE
  BEGIN
  Vec1 := IfcNormalise(Arg1);
  Vec2 := IfcNormalise(Arg2);
  Ndim := Arg1.Dim;
  Scalar := 0.0;
  REPEAT i := 1 TO Ndim;
  Scalar := Scalar +
  Vec1.DirectionRatios[i] *
  Vec2.DirectionRatios[i];
  END_REPEAT;
  END;
  END_IF;
  END_IF;
  RETURN (Scalar);
 	
END_FUNCTION;
	FUNCTION IfcDotProduct
	  (Arg1, Arg2 : IfcDirection) : REAL;
	
	  LOCAL
	  Scalar : REAL;
	  Vec1, Vec2 : IfcDirection;
	  Ndim : INTEGER;
	  END_LOCAL;
	
	  IF NOT EXISTS (Arg1) OR NOT EXISTS (Arg2) THEN
	  Scalar := ?;
	  ELSE
	  IF (Arg1.Dim <> Arg2.Dim) THEN
	  Scalar := ?;
	  ELSE
	  BEGIN
	  Vec1 := IfcNormalise(Arg1);
	  Vec2 := IfcNormalise(Arg2);
	  Ndim := Arg1.Dim;
	  Scalar := 0.0;
	  REPEAT i := 1 TO Ndim;
	  Scalar := Scalar +
	  Vec1.DirectionRatios[i] *
	  Vec2.DirectionRatios[i];
	  END_REPEAT;
	  END;
	  END_IF;
	  END_IF;
	  RETURN (Scalar);
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

FUNCTION IfcExtrusionPath
  (Solid : IfcAttDrivenExtrudedSolid) : IfcPolyline;
 	
  LOCAL
  Path : IfcPolyline;
  Depth : IfcPositiveLengthMeasure
  := Solid.Segments[1].Depth;
  NDim : INTEGER := HIINDEX(Solid.Segments);
  END_LOCAL;
 	
  IF NDim > 1 THEN
  REPEAT i := 2 TO NDim;
  Depth := Depth + Solid.Segments[i].Depth;
  END_REPEAT;
  END_IF;
 	
  Path := IfcGeometricRepresentationItem() ||
  IfcCurve() ||
  IfcBoundedCurve() ||
  IfcPolyline(
  [Solid.Segments[1].Position.Location,
  IfcPointTranslation(
  Solid.Segments[1].Position.Location,
  IfcGeometricRepresentationItem() ||
  IfcVector(
  Solid.Segments[1].Position.P[3],
  Depth))]);
  RETURN(Path);
 	
END_FUNCTION;
	FUNCTION IfcExtrusionPath
	  (Solid : IfcAttDrivenExtrudedSolid) : IfcPolyline;
	
	  LOCAL
	  Path : IfcPolyline;
	  Depth : IfcPositiveLengthMeasure := 0;
	  NDim : INTEGER := HIINDEX(Solid.Segments);
	  END_LOCAL;
	
	  REPEAT i := 1 TO NDim;
	  Depth := Depth + Solid.Segments[i].Depth;
	  END_REPEAT;
	
	  Path := IfcDummyGri || IfcCurve() ||
	  IfcBoundedCurve() ||
	  IfcPolyline(
	  [Solid.Segments[1].Position.Location,
	  IfcPointTranslation(
	  Solid.Segments[1].Position.Location,
	  IfcDummyGri ||
	  IfcVector(
	  Solid.Segments[1].Position.P[3],
	  Depth))]);
	  RETURN(Path);
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

```
FUNCTION IfcFirstProjAxis
  (ZAxis, Arg : IfcDirection) : IfcDirection;

  LOCAL
    XAxis, V :  IfcDirection
             := IfcGeometricRepresentationItem() ||
                IfcDirection([1.0,0.0,0.0]);
    XVec     :  IfcVector
             := IfcGeometricRepresentationItem() ||
                IfcVector(
                   IfcGeometricRepresentationItem() ||
                   IfcDirection([1.0,0.0,0.0]), 1.0);
  END_LOCAL;

  IF (NOT EXISTS(ZAxis)) OR
     ((EXISTS(Arg)) AND (Arg.Dim <> 3)) THEN
    XAxis := ?;
  ELSE
    ZAxis := IfcNormalise(ZAxis);
    IF NOT EXISTS(Arg) THEN
      IF (ZAxis <> IfcDirection([1.0,0.0,0.0])) THEN
        V := IfcGeometricRepresentationItem() ||
             IfcDirection([1.0,0.0,0.0]);
      ELSE
        V := IfcGeometricRepresentationItem() ||
             IfcDirection([0.0,1.0,0.0]);
      END_IF;
    ELSE
      IF ((IfcCrossProduct(Arg,ZAxis).Magnitude) = 0.0)
      THEN
        RETURN (?);
      ELSE
        V := IfcNormalise(Arg);
      END_IF;
    END_IF;
    XVec  := IfcScalarTimesVector(
                        IfcDotProduct(V, ZAxis), ZAxis);
    XAxis := IfcVectorDifference(V, XVec).Orientation;
    XAxis := IfcNormalise(XAxis);
  END_IF;
  RETURN(XAxis);

END_FUNCTION;
```

```
FUNCTION IfcFirstProjAxis
  (ZAxis, Arg : IfcDirection) : IfcDirection;

  LOCAL
    XAxis : IfcDirection;
    V     : IfcDirection;
    Z     : IfcDirection;
    XVec  : IfcVector;
  END_LOCAL;

  IF (NOT EXISTS(ZAxis)) THEN
    RETURN (?) ;
  ELSE
    Z := IfcNormalise(ZAxis);
    IF NOT EXISTS(Arg) THEN
      IF (Z.DirectionRatios <> [1.0,0.0,0.0]) THEN
        V := IfcDummyGri || IfcDirection([1.0,0.0,0.0]);
      ELSE
        V := IfcDummyGri || IfcDirection([0.0,1.0,0.0]);
      END_IF;
    ELSE
      IF (Arg.Dim <> 3) THEN
        RETURN (?) ;
      END_IF;
      IF ((IfcCrossProduct(Arg,Z).Magnitude) = 0.0) THEN
        RETURN (?);
      ELSE
        V := IfcNormalise(Arg);
      END_IF;
    END_IF;
    XVec  := IfcScalarTimesVector(
                          IfcDotProduct(V, Z), Z);
    XAxis := IfcVectorDifference(V, XVec).Orientation;
    XAxis := IfcNormalise(XAxis);
  END_IF;
  RETURN(XAxis);

END_FUNCTION;
```

Function changed to resolve geometry related problems

By: KSn
Date: 18-Apr-2001
Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

```
FUNCTION IfcNormalise
  (Arg : IfcVectorOrDirection) : IfcVectorOrDirection;

  LOCAL
    Ndim   :  INTEGER := 0;
    Mag    :  REAL    := 0.0;
    V      :  IfcDirection
           := IfcGeometricRepresentationItem() ||
               IfcDirection([1.0,0.0,0.0]);
    Vec    :  IfcVector
           := IfcGeometricRepresentationItem() ||
              IfcVector(IfcGeometricRepresentationItem()
              || IfcDirection([1.0,0.0,0.0]), 1.0);
    Result :  IfcVectorOrDirection := V;
  END_LOCAL;

  IF NOT EXISTS (Arg) THEN
    Result := ?;
  ELSE
    Ndim := Arg.Dim;
    IF 'IFC20_LONGFORM.IFCVECTOR' IN TYPEOF(Arg) THEN
      BEGIN
        Vec := Arg;
        V   := Arg.Orientation;
        IF Arg.Magnitude = 0.0 THEN
          RETURN(?);
        ELSE
          Vec.Magnitude := 1.0;
        END_IF;
      END;
    ELSE
      V := Arg;
    END_IF;
    Mag  := 0.0;
    REPEAT  i := 1 TO Ndim;
      Mag := Mag +
            (V.DirectionRatios[i]*V.DirectionRatios[i]);
    END_REPEAT;
    IF Mag > 0.0 THEN
      Mag := SQRT(Mag);
      REPEAT  i := 1 TO Ndim;
        V.DirectionRatios[i] :=
              V.DirectionRatios[i]/Mag;
      END_REPEAT;
```

```
FUNCTION IfcNormalise
  (Arg : IfcVectorOrDirection) : IfcVectorOrDirection;

  LOCAL
    Ndim   : INTEGER;
    V      : IfcDirection
           := IfcDummyGri || IfcDirection([1.,0.]);
    Vec    : IfcVector
           := IfcDummyGri ||
              IfcVector(IfcDummyGri ||
                        IfcDirection([1.,0.]), 1.);
    Mag    : REAL;
    Result : IfcVectorOrDirection := V;
  END_LOCAL;

  IF NOT EXISTS (Arg) THEN
    RETURN (?);
  ELSE
    Ndim := Arg.Dim;
    IF 'IFC20_LONGFORM.IFCVECTOR' IN TYPEOF(Arg) THEN
      BEGIN
        Vec := Arg;
        V := Arg.Orientation;
        IF Arg.Magnitude = 0.0 THEN
          RETURN(?);
        ELSE
          Vec.Magnitude := 1.0;
        END_IF;
      END;
    ELSE
      V := Arg;
    END_IF;
    Mag := 0.0;
    REPEAT i := 1 TO Ndim;
      Mag := Mag +
             V.DirectionRatios[i]*V.DirectionRatios[i];
    END_REPEAT;
    IF Mag > 0.0 THEN
      Mag := SQRT(Mag);
      REPEAT i := 1 TO Ndim;
        V.DirectionRatios[i] :=
                    V.DirectionRatios[i]/Mag;
      END_REPEAT;
```

Function changed to resolve geometry related problems

By: KSn
Date: 18-Apr-2001
Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

IF 'IFC20_LONGFORM.IFCVECTOR' IN TYPEOF(Arg) THEN
Vec.Orientation := V;
Result := Vec;
ELSE
Result := V;
END_IF;
ELSE
RETURN(?);
END_IF;
END_IF;
RETURN(Result);
 	
END_FUNCTION;
	IF 'IFC20_LONGFORM.IFCVECTOR' IN TYPEOF(Arg) THEN
	Vec.Orientation := V;
	Result := Vec;
	ELSE
	Result := V;
	END_IF;
	ELSE
	RETURN(?);
	END_IF;
	END_IF;
	RETURN(Result);
	
	END_FUNCTION;	

FUNCTION IfcOrthogonalComplement
(Vec : IfcDirection) : IfcDirection;
 	
LOCAL
Result : IfcDirection
:= IfcGeometricRepresentationItem() ||
IfcDirection([1.0,0.0]);
END_LOCAL;
 	
IF (Vec.Dim <> 2) OR NOT EXISTS (Vec) THEN
RETURN(?);
ELSE
Result.DirectionRatios[1] := -Vec.DirectionRatios[2];
Result.DirectionRatios[2] := Vec.DirectionRatios[1];
RETURN(result);
END_IF;
 	
END_FUNCTION;
	FUNCTION IfcOrthogonalComplement
	(Vec : IfcDirection) : IfcDirection;
	
	LOCAL
	Result : IfcDirection ;
	END_LOCAL;
	
	IF NOT EXISTS (Vec) OR (Vec.Dim <> 2) THEN
	RETURN(?);
	ELSE
	Result := IfcDummyGri ||
	IfcDirection([-Vec.DirectionRatios[2],
	Vec.DirectionRatios[1]]);
	RETURN(Result);
	END_IF;
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

FUNCTION IfcPointTranslation
  (Origin : IfcCartesianPoint;
  Vec : IfcVector) : IfcCartesianPoint;
 	
  LOCAL
  NDim : INTEGER
  := HIINDEX(Origin.Coordinates);
  Point : IfcCartesianPoint
  := IfcGeometricRepresentationItem() ||
  IfcPoint() ||
  IfcCartesianPoint(Origin.Coordinates);
  END_LOCAL;
 	
  IF (Origin.Dim <> Vec.Dim) OR (NOT EXISTS (Vec)) OR
  (NOT EXISTS (Origin)) THEN
  RETURN(?);
  END_IF;
  REPEAT i := 1 TO NDim;
  Point.Coordinates[i] :=
  Origin.Coordinates[i] +
  Vec.Magnitude * Vec.Orientation.DirectionRatios[i];
  END_REPEAT;
  RETURN(Point);
 	
END_FUNCTION;
	FUNCTION IfcPointTranslation
	  (Origin : IfcCartesianPoint;
	  Vec : IfcVector) : IfcCartesianPoint;
	
	  LOCAL
	  NDim : INTEGER
	  := HIINDEX(Origin.Coordinates);
	  Point : IfcCartesianPoint
	  := IfcDummyGri ||
	  IfcPoint() ||
	  IfcCartesianPoint(Origin.Coordinates);
	  END_LOCAL;
	
	  IF (Origin.Dim <> Vec.Dim) OR (NOT EXISTS (Vec)) OR
	  (NOT EXISTS (Origin)) THEN
	  RETURN(?);
	  END_IF;
	  REPEAT i := 1 TO NDim;
	  Point.Coordinates[i] :=
	  Origin.Coordinates[i] +
	  Vec.Magnitude * Vec.Orientation.DirectionRatios[i];
	  END_REPEAT;
	  RETURN(Point);
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

FUNCTION IfcProfileIntoArea
  (ProfileDef : IfcAttDrivenProfileDef)
  : IfcCurveBoundedPlane;
 	
  LOCAL
  Curve2D : Ifc2DCompositeCurve;
  ResSurface : IfcCurveBoundedPlane;
  END_LOCAL;
 	
  Curve2D := IfcGeometricRepresentationItem() ||
  IfcCurve() || IfcBoundedCurve() ||
  IfcCompositeCurve(
  [IfcCompositeCurveSegment(
  Continuous, TRUE,
  ProfileDef.CurveForSurface)],
  FALSE) ||
  Ifc2DCompositeCurve();
  ResSurface := IfcGeometricRepresentationItem() ||
  IfcSurface() ||
  IfcCurveBoundedPlane(
  IfcGeometricRepresentationItem() ||
  IfcSurface() ||
  IfcElementarySurface(
  IfcGeometricRepresentationItem() ||
  IfcPlacement(
  IfcGeometricRepresentationItem() ||
  IfcPoint () ||
  IfcCartesianPoint([0.0, 0.0, 0.0]))
  ||
  IfcAxis2Placement3D(
  IfcGeometricRepresentationItem() ||
  IfcDirection([0.0,0.0,1.0]),
  IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0,0.0]))) ||
  IfcPlane(),
  Curve2D, []);
  RETURN(ResSurface);
 	
END_FUNCTION;
	FUNCTION IfcProfileIntoArea
	  (ProfileDef : IfcAttDrivenProfileDef)
	  : IfcCurveBoundedPlane;
	
	  LOCAL
	  Curve2D : Ifc2DCompositeCurve;
	  ResSurface : IfcCurveBoundedPlane;
	  END_LOCAL;
	
	  Curve2D := IfcDummyGri ||
	  IfcCurve() || IfcBoundedCurve() ||
	  IfcCompositeCurve(
	  [IfcDummyGri ||
	  IfcCompositeCurveSegment(
	  Continuous, TRUE,
	  ProfileDef.CurveForSurface)],
	  FALSE) ||
	  Ifc2DCompositeCurve();
	  ResSurface := IfcDummyGri ||
	  IfcSurface() ||
	  IfcCurveBoundedPlane(
	  IfcDummyGri || IfcSurface() ||
	  IfcElementarySurface(
	  IfcDummyGri ||
	  IfcPlacement(
	  IfcDummyGri || IfcPoint () ||
	  IfcCartesianPoint([0.0, 0.0, 0.0]))
	  ||
	  IfcAxis2Placement3D(
	  IfcDummyGri ||
	  IfcDirection([0.0,0.0,1.0]),
	  IfcDummyGri ||
	  IfcDirection([1.0,0.0,0.0]))) ||
	  IfcPlane(),
	  Curve2D, []);
	  RETURN(ResSurface);
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

FUNCTION IfcRectangleProfileIntoCurve
  (ProfileDef : IfcRectangleProfileDef) : IfcPolyline;
 	
  LOCAL
  Points : LIST [4:4] OF IfcCartesianPoint
  := [IfcGeometricRepresentationItem() ||
  IfcPoint() ||
  IfcCartesianPoint([0.0,0.0]):4];
  TempDir : IfcDirection
  := IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0]);
  ResCurve : IfcPolyline;
  END_LOCAL;
 	
  Points[1] :=
  ProfileDef\IfcAttDrivenProfileDef.Position.Location;
  TempDir :=
  ProfileDef\IfcAttDrivenProfileDef.Position.P[1];
  Points[2] := IfcPointTranslation(Points[1],
  IfcGeometricRepresentationItem() ||
  IfcVector(TempDir, ProfileDef.XDim));
  TempDir := IfcOrthogonalComplement(TempDir);
  Points[3] := IfcPointTranslation(Points[2],
  IfcGeometricRepresentationItem() ||
  IfcVector(TempDir, ProfileDef.YDim));
  TempDir := IfcOrthogonalComplement(TempDir);
  Points[4] := IfcPointTranslation(Points[3],
  IfcGeometricRepresentationItem() ||
  IfcVector(TempDir, ProfileDef.XDim));
  ResCurve := IfcGeometricRepresentationItem() ||
  IfcCurve() || IfcBoundedCurve() ||
  IfcPolyline([Points[1], Points[2],
  Points[3],
  Points[4], Points[1]]);
  RETURN (ResCurve);
 	
END_FUNCTION;
	FUNCTION IfcRectangleProfileIntoCurve
	  (ProfileDef : IfcRectangleProfileDef) : IfcPolyline;
	
	  LOCAL
	  Points : LIST [4:4] OF IfcCartesianPoint
	  := [IfcDummyGri || IfcPoint() ||
	  IfcCartesianPoint([0.0,0.0]):4] ;
	  TempDir : IfcDirection
	  := IfcDummyGri || IfcDirection([1.0,0.0]);
	  ResCurve : IfcPolyline;
	  END_LOCAL;
	
	  Points[1] :=
	  ProfileDef\IfcAttDrivenProfileDef.Position.Location;
	  TempDir :=
	  ProfileDef\IfcAttDrivenProfileDef.Position.P[1];
	  Points[2] := IfcPointTranslation(
	  Points[1], IfcDummyGri ||
	  IfcVector(TempDir, ProfileDef.XDim));
	  TempDir := IfcOrthogonalComplement(TempDir);
	  Points[3] := IfcPointTranslation(
	  Points[2], IfcDummyGri ||
	  IfcVector(TempDir, ProfileDef.YDim));
	  TempDir := IfcOrthogonalComplement(TempDir);
	  Points[4] := IfcPointTranslation(
	  Points[3], IfcDummyGri ||
	  IfcVector(TempDir, ProfileDef.XDim));
	  ResCurve := IfcDummyGri || IfcCurve() ||
	  IfcBoundedCurve() ||
	  IfcPolyline([Points[1], Points[2],
	  Points[3],
	  Points[4], Points[1]]);
	  RETURN (ResCurve);
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

```
FUNCTION IfcRevolutionPath
  (Solid : IfcAttDrivenRevolvedSolid) : IfcTrimmedCurve;
  LOCAL
    Path   : IfcTrimmedCurve;
    Pos    : IfcAxis2Placement3D;
    Circle : IfcCircle;
    Angle  : IfcPlaneAngleMeasure := 0;
    NDim   : INTEGER := HIINDEX(Solid.Segments);
  END_LOCAL;

  REPEAT i := 1 TO NDim;
    Angle := Angle + Solid.Segments[i].Angle;
  END_REPEAT;
  Pos := IfcGeometricRepresentationItem() ||
         IfcPlacement(
           IfcPointTranslation(
             Solid.Segments[1].Position.Location,
             IfcGeometricRepresentationItem() ||
             IfcVector(
               IfcNormalise(
                 Solid.Segments[1].Position.P[1]),
          Solid.Segments[1].Axis.Location.Coordinates[1]
         ))) ||
         IfcAxis2Placement3D(
           Solid.Segments[1].Position.P[2],
           IfcGeometricRepresentationItem() ||
           IfcDirection([
      Solid.Segments[1].Position.P[1].DirectionRatios[1]
              * -1.0,
      Solid.Segments[1].Position.P[1].DirectionRatios[2]
              * -1.0,
      Solid.Segments[1].Position.P[1].DirectionRatios[3]
              * -1.0]));
  Circle := IfcGeometricRepresentationItem() ||
            IfcCurve() || IfcConic(Pos) ||
            IfcCircle(
       Solid.Segments[1].Axis.Location.Coordinates[1]);
  Path   := IfcGeometricRepresentationItem() ||
            IfcCurve() || IfcBoundedCurve() ||
            IfcTrimmedCurve(Circle, [0.0], [Angle],
                            TRUE, Parameter);
  RETURN (Path);

END_FUNCTION;
```

```
FUNCTION IfcRevolutionPath
  (Solid : IfcAttDrivenRevolvedSolid) : IfcTrimmedCurve;
  LOCAL
    Path   : IfcTrimmedCurve;
    Pos    : IfcAxis2Placement3D;
    Circle : IfcCircle;
    Angle  : IfcParameterValue := 0;
    NDim   : INTEGER := HIINDEX(Solid.Segments);
  END_LOCAL;

  REPEAT i := 1 TO NDim;
    Angle  := Angle + Solid.Segments[i].Angle;
  END_REPEAT;
  Pos := IfcDummyGri ||
         IfcPlacement(
           IfcPointTranslation(
             Solid.Segments[1].Position.Location,
             IfcDummyGri ||
             IfcVector(
               IfcNormalise(
                 Solid.Segments[1].Position.P[1]),
           Solid.Segments[1].Axis.Location.Coordinates[1]
         ))) ||
         IfcAxis2Placement3D(
           Solid.Segments[1].Position.P[2],
           IfcDummyGri ||
           IfcDirection([
      Solid.Segments[1].Position.P[1].DirectionRatios[1]
              * -1.0,
      Solid.Segments[1].Position.P[1].DirectionRatios[2]
              * -1.0,
      Solid.Segments[1].Position.P[1].DirectionRatios[3]
              * -1.0]));
  Circle := IfcDummyGri || IfcCurve() || IfcConic(Pos) ||
            IfcCircle(
       Solid.Segments[1].Axis.Location.Coordinates[1]);
  Path   := IfcDummyGri || IfcCurve() ||
            IfcBoundedCurve() ||
            IfcTrimmedCurve(Circle, [0.0], [Angle],
                            TRUE, Parameter);
  RETURN (Path);

END_FUNCTION;
```

Function changed to resolve geometry related problems

By: KSn
Date: 18-Apr-2001
Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

FUNCTION IfcScalarTimesVector
  (Scalar : REAL;
  Vec : IfcVectorOrDirection) : IfcVector;
 	
  LOCAL
  Mag : REAL := 0.0;
  V : IfcDirection
  := IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0,0.0]);
  Result : IfcVector
  := IfcGeometricRepresentationItem() ||
  IfcVector(
  IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0,0.0]), 1.0);
  END_LOCAL;
 	
  IF NOT EXISTS (Scalar) OR NOT EXISTS (Vec) THEN
  Result := ?;
  ELSE
  IF 'IFC20_LONGFORM.IFCVECTOR' IN TYPEOF (Vec) THEN
  V := Vec.Orientation;
  Mag := Scalar * Vec.Magnitude;
  ELSE
  V := Vec;
  Mag := Scalar;
  END_IF;
  IF (Mag < 0.0) THEN
  REPEAT i := 1 TO SIZEOF(V.DirectionRatios);
  V.DirectionRatios[i] := -V.DirectionRatios[i];
  END_REPEAT;
  Mag := -Mag;
  END_IF;
  Result.Orientation := IfcNormalise(V);
  Result.Magnitude := Mag;
  END_IF;
  RETURN (Result);
 	
END_FUNCTION;
	FUNCTION IfcScalarTimesVector
	  (Scalar : REAL;
	  Vec : IfcVectorOrDirection) : IfcVector;
	
	  LOCAL
	  V : IfcDirection;
	  Mag : REAL;
	  Result : IfcVector;
	  END_LOCAL;
	
	  IF NOT EXISTS (Scalar) OR NOT EXISTS (Vec) THEN
	  RETURN (?) ;
	  ELSE
	  IF 'IFC20_LONGFORM.IFCVECTOR' IN TYPEOF (Vec) THEN
	  V := Vec.Orientation;
	  Mag := Scalar * Vec.Magnitude;
	  ELSE
	  V := Vec;
	  Mag := Scalar;
	  END_IF;
	  IF (Mag < 0.0) THEN
	  REPEAT i := 1 TO SIZEOF(V.DirectionRatios);
	  V.DirectionRatios[i] := -V.DirectionRatios[i];
	  END_REPEAT;
	  Mag := -Mag;
	  END_IF;
	  Result := IfcDummyGri ||
	  IfcVector(IfcNormalise(V), Mag);
	  END_IF;
	  RETURN (Result);
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

FUNCTION IfcTrapeziumProfileIntoCurve
  (ProfileDef : IfcTrapeziumProfileDef) : IfcPolyline;
  LOCAL
  Points : LIST [4:4] OF IfcCartesianPoint :=
  [IfcGeometricRepresentationItem() ||
  IfcPoint() || IfcCartesianPoint([0.0, 0.0]):4];
  TempDir : IfcDirection :=
  IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0]);
  TempPoint : IfcCartesianPoint :=
  IfcGeometricRepresentationItem() || IfcPoint()
  || IfcCartesianPoint([0.0,0.0]);
  ResCurve : IfcPolyline;
  END_LOCAL;
 	
  Points[1] :=
  ProfileDef\IfcAttDrivenProfileDef.Position.Location;
  TempDir :=
  ProfileDef\IfcAttDrivenProfileDef.Position.P[1];
  Points[2] :=
  IfcPointTranslation(
  Points[1], IfcGeometricRepresentationItem() ||
  IfcVector(TempDir, ProfileDef.BottomXDim));
  TempDir := IfcOrthogonalComplement(TempDir);
  TempPoint :=
  IfcPointTranslation(
  Points[2], IfcGeometricRepresentationItem() ||
  IfcVector(TempDir, ProfileDef.YDim));
  TempDir := IfcOrthogonalComplement(TempDir);
  Points[3] :=
  IfcPointTranslation(
  TempPoint, IfcGeometricRepresentationItem() ||
  IfcVector(TempDir, (ProfileDef.BottomXDim -
  ProfileDef.TopXDim - ProfileDef.TopXOffset)));
  Points[4] :=
  IfcPointTranslation(
  Points[3], IfcGeometricRepresentationItem() ||
  IfcVector(TempDir, ProfileDef.TopXDim));
  ResCurve :=
  IfcGeometricRepresentationItem() || IfcCurve() ||
  IfcBoundedCurve() || IfcPolyline(
  [Points[1], Points[2], Points[3], Points[4],
  Points[1]]);
  RETURN (ResCurve);
 	
END_FUNCTION;
	FUNCTION IfcTrapeziumProfileIntoCurve
	  (ProfileDef : IfcTrapeziumProfileDef) : IfcPolyline;
	
	  LOCAL
	  Points : LIST [4:4] OF IfcCartesianPoint
	  := [IfcDummyGri || IfcPoint() ||
	  IfcCartesianPoint([0.0,0.0]):4];
	  TempDir : IfcDirection
	  := IfcDummyGri || IfcDirection([1.0,0.0]);
	  TempPoint : IfcCartesianPoint
	  := IfcDummyGri || IfcPoint() ||
	  IfcCartesianPoint([0.0,0.0]);
	  ResCurve : IfcPolyline;
	  END_LOCAL;
	
	  Points[1] :=
	  ProfileDef\IfcAttDrivenProfileDef.Position.Location;
	  TempDir :=
	  ProfileDef\IfcAttDrivenProfileDef.Position.P[1];
	  Points[2] := IfcPointTranslation(Points[1],
	  IfcDummyGri ||
	  IfcVector(
	  TempDir, ProfileDef.BottomXDim));
	  TempDir := IfcOrthogonalComplement(TempDir);
	  TempPoint := IfcPointTranslation(Points[2],
	  IfcDummyGri ||
	  IfcVector(TempDir, ProfileDef.YDim));
	  TempDir := IfcOrthogonalComplement(TempDir);
	  Points[3] := IfcPointTranslation(TempPoint,
	  IfcDummyGri ||
	  IfcVector(TempDir,
	  (ProfileDef.BottomXDim -
	  ProfileDef.TopXDim -
	  ProfileDef.TopXOffset)));
	  Points[4] := IfcPointTranslation (Points[3],
	  IfcDummyGri ||
	  IfcVector(TempDir,
	  ProfileDef.TopXDim));
	  ResCurve := IfcDummyGri || IfcCurve() ||
	  IfcBoundedCurve() ||
	  IfcPolyline([Points[1], Points[2],
	  Points[3], Points[4], Points[1]]);
	  RETURN (ResCurve);
	
	END_FUNCTION;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

FUNCTION IfcVectorDifference
  (Arg1, Arg2 : IfcVectorOrDirection) : IfcVector;
 	
  LOCAL
  Mag, Mag1, Mag2 : REAL := 0.0;
  Ndim : INTEGER := 0;
  Res, Vec1, Vec2 : IfcDirection :=
  IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0,0.0]);
  Result : IfcVector :=
  IfcGeometricRepresentationItem() ||
  IfcVector(IfcGeometricRepresentationItem() ||
  IfcDirection([1.0,0.0,0.0]), 1.0);
  END_LOCAL;
 	
  IF ((NOT EXISTS (Arg1)) OR (NOT EXISTS (Arg2))) OR
  (Arg1.Dim <> Arg2.Dim) THEN
  Result := ?;
  ELSE
  BEGIN
  IF 'IFC20_LONGFORM.IFCVECTOR' IN TYPEOF(Arg1) THEN
  Mag1 := Arg1.Magnitude;
  Vec1 := Arg1.Orientation;
  ELSE
  Mag1 := 1.0;
  Vec1 := Arg1;
  END_IF;
  IF 'IFC20_LONGFORM.IFCVECTOR' IN TYPEOF(Arg2) THEN
  Mag2 := Arg2.Magnitude;
  Vec2 := Arg2.Orientation;
  ELSE
  Mag2 := 1.0;
  Vec2 := Arg2;
  END_IF;
  Vec1 := IfcNormalise(Vec1);
  Vec2 := IfcNormalise(Vec2);
  Ndim := SIZEOF(Vec1.DirectionRatios);
  Mag := 0.0;
  REPEAT i := 1 TO Ndim;
  Res.DirectionRatios[i] :=
  Mag1 * Vec1.DirectionRatios[i] -
  Mag2 * Vec2.DirectionRatios[i];
  Mag := Mag + (Res.DirectionRatios[i] *
  Res.DirectionRatios[i]);
  END_REPEAT;
	FUNCTION IfcVectorDifference
	  (Arg1, Arg2 : IfcVectorOrDirection) : IfcVector;
	
	  LOCAL
	  Result : IfcVector;
	  Res, Vec1, Vec2 : IfcDirection;
	  Mag, Mag1, Mag2 : REAL;
	  Ndim : INTEGER;
	  END_LOCAL;
	
	  IF ((NOT EXISTS (Arg1)) OR (NOT EXISTS (Arg2))) OR
	  (Arg1.Dim <> Arg2.Dim) THEN
	  RETURN (?);
	  ELSE
	  BEGIN
	  IF 'IFC20_LONGFORM.IFCVECTOR' IN TYPEOF(Arg1) THEN
	  Mag1 := Arg1.Magnitude;
	  Vec1 := Arg1.Orientation;
	  ELSE
	  Mag1 := 1.0;
	  Vec1 := Arg1;
	  END_IF;
	  IF 'IFC20_LONGFORM.IFCVECTOR' IN TYPEOF(Arg2) THEN
	  Mag2 := Arg2.Magnitude;
	  Vec2 := Arg2.Orientation;
	  ELSE
	  Mag2 := 1.0;
	  Vec2 := Arg2;
	  END_IF;
	  Vec1 := IfcNormalise (Vec1);
	  Vec2 := IfcNormalise (Vec2);
	  Ndim := SIZEOF(Vec1.DirectionRatios);
	  Mag := 0.0;
	  IF(NDim = 2) THEN
	  Res := IfcDummyGri || IfcDirection([1.,1.]);
	  ELSE
	  IF(NDim = 3) THEN
	  Res := IfcDummyGri || IfcDirection([1.,1.,1.]);
	  ELSE
	  RETURN (?);
	  END_IF;
	  END_IF;
		Function changed to resolve geometry related problems
		
		By: KSn
		Date: 18-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

IF (Mag > 0.0) THEN
Result.Magnitude := SQRT(Mag);
Result.Orientation := Res;
ELSE
Result.Magnitude := 0.0;
Result.Orientation := Vec1;
END_IF;
END;
END_IF;
RETURN (Result);
 	
END_FUNCTION;
	REPEAT i := 1 TO Ndim;
	Res.DirectionRatios[i] :=
	Mag1 * Vec1.DirectionRatios[i] +
	Mag2 * Vec2.DirectionRatios[i];
	Mag := Mag + (Res.DirectionRatios[i] *
	Res.DirectionRatios[i]);
	END_REPEAT;
	IF (Mag > 0.0) THEN
	Result := IfcDummyGri ||
	IfcVector(Res, SQRT(Mag));
	ELSE
	Result := IfcDummyGri || IfcVector(Vec1, 0.0);
	END_IF;
	END;
	END_IF;
	RETURN (Result);
	
	END_FUNCTION;	

Appendix 4. Refined DERIVEd attributes.

KSn = Kalle Serén, Eurostepsys Oy

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

ENTITY IfcArbitraryProfileDef
  SUBTYPE OF (IfcAttDrivenProfileDef);
  CurveForSurface : IfcBoundedCurve;
  DERIVE
  SELF\IfcAttDrivenProfileDef.Position :
  IfcAxis2Placement2D
  := IfcGeometricRepresentationItem() ||
  IfcPlacement(
  IfcGeometricRepresentationItem() ||
  IfcPoint() ||
  IfcCartesianPoint([0.,0.])) ||
  IfcAxis2Placement2D(
  IfcGeometricRepresentationItem() ||
  IfcDirection([1.,0.]));
  WHERE
  WR21: CurveForSurface.Dim = 2;
END_ENTITY;
	ENTITY IfcArbitraryProfileDef
	  SUBTYPE OF (IfcAttDrivenProfileDef);
	  CurveForSurface : IfcBoundedCurve;
	  DERIVE
	  SELF\IfcAttDrivenProfileDef.Position :
	  IfcAxis2Placement2D
	  := IfcDummyGri || IfcPlacement(
	  IfcDummyGri || IfcPoint() ||
	  IfcCartesianPoint([0.,0.])) ||
	  IfcAxis2Placement2D(
	  IfcDummyGri || IfcDirection([1.,0.]));
	  WHERE
	  WR21: CurveForSurface.Dim = 2;
	END_ENTITY;
		SELF\ IfcAttDrivenProfileDef. Position changed to resolve geometry related problems
		By: KSn
		Date: 20-Apr-2001
		Source: [IFC R1.5.1 Corr]

ENTITY IfcAttDrivenExtrudedSegment
  SUPERTYPE OF (ONEOF(
  IfcAttDrivenMorphedExtrudedSegment
  ,IfcAttDrivenTaperedExtrudedSegment))
  SUBTYPE OF (IfcExtrudedAreaSolid);
  Position : IfcAxis2Placement3D;
  ProfileDef : IfcAttDrivenProfileDef;
  DERIVE
  SELF\IfcSweptAreaSolid.SweptArea :
  IfcCurveBoundedPlane
  := IfcProfileIntoArea(ProfileDef);
  SELF\IfcExtrudedAreaSolid.ExtrudedDirection :
  IfcDirection :=
  IfcGeometricRepresentationItem() ||
  IfcDirection([0.0,0.0,1.0]);
  INVERSE
  PartOfSolid : IfcAttDrivenExtrudedSolid
  FOR Segments;
  WHERE
  WR51: ProfileDef.ProfileType = Area;
END_ENTITY;
	ENTITY IfcAttDrivenExtrudedSegment
	  SUPERTYPE OF (ONEOF(
	  IfcAttDrivenMorphedExtrudedSegment
	  ,IfcAttDrivenTaperedExtrudedSegment))
	  SUBTYPE OF (IfcExtrudedAreaSolid);
	  Position : IfcAxis2Placement3D;
	  ProfileDef : IfcAttDrivenProfileDef;
	  DERIVE
	  SELF\IfcSweptAreaSolid.SweptArea :
	  IfcCurveBoundedPlane
	  := IfcProfileIntoArea(ProfileDef);
	  SELF\IfcExtrudedAreaSolid.ExtrudedDirection :
	  IfcDirection
	  := IfcDummyGri ||
	  IfcDirection([0.0,0.0,1.0]);
	  INVERSE
	  PartOfSolid : IfcAttDrivenExtrudedSolid
	  FOR Segments;
	  WHERE
	  WR51: ProfileDef.ProfileType = Area;
	END_ENTITY;
		SELF\IfcSweptAreaSolid. SweptArea changed to resolve geometry related problems
		By: KSn
		Date: 23-Apr-2001
		Source: [IFC R1.5.1 Corr]

Original IFC R2.0 specification	Refined version for IFC R2.0 certification	Description

ENTITY IfcAxis1Placement
  SUBTYPE OF (IfcPlacement);
  Axis : OPTIONAL IfcDirection;
  DERIVE
  Z : IfcDirection
  := NVL (IfcNormalise(Axis)
  ,IfcGeometricRepresentationItem() ||
  IfcDirection([0.0,0.0,1.0]));
  WHERE
  WR31: (NOT (EXISTS (Axis))) OR (Axis.Dim = 3);
  WR32: SELF\IfcPlacement.Location.Dim = 3;
END_ENTITY;
	ENTITY IfcAxis1Placement
	  SUBTYPE OF (IfcPlacement);
	  Axis : OPTIONAL IfcDirection;
	  DERIVE
	  Z : IfcDirection
	  := NVL (IfcNormalise(Axis)
	  ,IfcDummyGri ||
	  IfcDirection([0.0,0.0,1.0]));
	  WHERE
	  WR31: (NOT (EXISTS (Axis))) OR (Axis.Dim = 3);
	  WR32: SELF\IfcPlacement.Location.Dim = 3;
	END_ENTITY;
		Z changed to resolve geometry related problems
		By: KSn
		Date: 23-Apr-2001
		Source: [IFC R1.5.1 Corr]

ENTITY IfcRevolvedAreaSolid
  SUPERTYPE OF (ONEOF(
  IfcAttDrivenRevolvedSegment))
  SUBTYPE OF (IfcSweptAreaSolid);
  Axis : IfcAxis1Placement;
  Angle : IfcPlaneAngleMeasure;
  DERIVE
  AxisLine : IfcLine
  := IfcGeometricRepresentationItem() ||
  IfcCurve() ||
  IfcLine(
  Axis.Location,
  IfcGeometricRepresentationItem() ||
  IfcVector(Axis.Z,1.0));
END_ENTITY;
	ENTITY IfcRevolvedAreaSolid
	  SUPERTYPE OF (ONEOF(
	  IfcAttDrivenRevolvedSegment))
	  SUBTYPE OF (IfcSweptAreaSolid);
	  Axis : IfcAxis1Placement;
	  Angle : IfcPlaneAngleMeasure;
	  DERIVE
	  AxisLine : IfcLine
	  := IfcDummyGri ||
	  IfcCurve() ||
	  IfcLine(
	  Axis.Location,
	  IfcDummyGri ||
	  IfcVector(Axis.Z,1.0));
	END_ENTITY;
		AxisLine changed to resolve geometry related problems
		By: KSn
		Date: 23-Apr-2001
		Source: [IFC R1.5.1 Corr]

Appendix 5. Example test instantiations as ISO 10303-21 exchange files.

A sample file conforming to the constraints:

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(
   /* Description          */  'IFC R2.0 Certification: IfcPerson / WR1 / OK',
   /* Implementation level */  '2;1');
FILE_NAME(
   /* Name                 */  'Test_IfcPerson_WR1_OK.stp',
   /* Time stamp           */  '2001-04-15T12:00',
   /* Author               */ ('KKa'),
   /* Organization         */ ('IAI Forum Finland'),
   /* Preprocessor Version */  'None',
   /* Originating system   */  'HandMade',
   /* Authorization        */  'KKa');
FILE_SCHEMA(('IFC20_LONGFORM'));
ENDSEC;

DATA;
#1=IFCPERSON('TestFamilyName', $, $, $, $, $, $);
ENDSEC;
END-ISO-10303-21;
```

A sample file not conforming to the constraints:

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(
   /* Description          */  'IFC R2.0 Certification: IfcPerson / WR1 / NOT OK',
   /* Implementation level */  '2;1');
FILE_NAME(
   /* Name                 */  'Test_IfcPerson_WR1_NotOK.stp',
   /* Time stamp           */  '2001-04-15T12:00',
   /* Author               */ ('KKa'),
   /* Organization         */ ('IAI Forum Finland'),
   /* Preprocessor Version */  'None',
   /* Originating system   */  'HandMade',
   /* Authorization        */  'KKa');
FILE_SCHEMA(('IFC20_LONGFORM'));
ENDSEC;

DATA;
#1=IFCPERSON($, $, $, $, $, $, $);
ENDSEC;
END-ISO-10303-21;
```