INTERLIS Versión 2 –

Manual de Referencia

Edición 2006-04-13 (V.1.0 de la traducción al español)

Informacion y contacto: www.interlis.ch, info@interlis.ch

Este manual de referencia se redactó inicialmente en alemán, los autores de la versión en inglés han tratado de respetar tanto como ha sido posible el texto original.

La versión en español (V.1.0) es una traducción del manual en inglés, aunque se ha realizado una verificación cruzada con la versión en alemán en aquellos puntos en los que la traducción inglesa dejaba dudas. La traducción fue hecha por el Proyecto Modernización de la Administración de Tierras en Colombia, con fondos de la Secretaria de Asuntos Económicos de Suiza (SECO)

Contenido

1 Principios básicos 9
1.1 Visión general 9
1.2 La utilización de modelos 10
1.3 Estructuración en modelos y temas 11
1.4 Concepto de objeto 12
1.4.1 Objetos y clases 12
1.4.2 Extensión de clases y el polimorfismo 13
1.4.3 Metamodelos y Metaobjetos 13
1.4.4 Las relaciones entre los objetos 14
1.4.5 Contenedores, replicación y transferencia de datos 15
1.5 Concepto de Vistas 17
1.6 Concepto de gráfico 17
1.8. Servicios, capacidades de las herramientas y conformidad 18
1.9 Un pequeño ejemplo como introducción 20
1.10 ¿Cómo está estructurado este documento? 20

2 Descripción del lenguaje 22
2.1 Sintaxis aplicada 22
2.2 Símbolos básicos del lenguaje 23
2.2.1 Los códigos de caracteres utilizados, espacios en blanco y finales de la línea 23
2.2.2 Nombres 23
2.2.3 Concatenaciones 23
2.2.4 Dígitos 24
2.2.5 Conjuntos de propiedades 24
2.2.6 Explicaciones 24
2.2.7 Caracteres especiales y palabras reservadas 25
2.2.8 COMENTARIOS 25
2.2.8.1 Comentario de línea 25
2.2.8.2 Comentario de bloque 26
2.3 Regla principal 26
2.4 Herencia 26
2.5 Modelos, temas (*topics*) y clases 26
2.5.1 Modelos 26
2.5.2 Temas 28
2.5.3 Las clases y las estructuras 29
2.5.4 Espacios de nombres 30
2.6 ATRIBUTOS 31
2.6.1 Observaciones generales relativas a los atributos 31
2.6.2 Atributos con el dominio como tipo 32
2.6.3 Atributos de referencia 32
2.6.4 Atributos de estructura 33
2.7 Relaciones propiamente dichas o reales 33
2.7.1 Descripción de las relaciones 33
2.7.2 La fuerza de la relación 35
2.7.3 Cardinalidad 36
2.7.4 Relaciones ordenadas 36
2.7.5 Acceso de la relación 36
2.8 Dominios y constantes 36
2.8.1 Cadenas de texto 38
2.8.2 Enumeraciones 39
2.8.3 Orientación del texto 40
2.8.4 BOOLEAN 41
2.8.5 Tipos de datos numéricos 41
2.8.6 Dominios con formato 43
2.8.7 Fecha y hora 43
2.8.8 Coordenadas 44
2.8.9 Dominios de identificación de objetos 45
2.8.10 Cajas negras 46

2.8.11 Dominios de clases y Rutas de atributo ... 46
2.8.12 Cadenas de línea ... 47
2.8.12.1 Geometría de las cadenas de línea ... 47
2.8.12.2 Cadenas de línea con segmentos de línea recta y arcos de círculo como segmentos de curva predefinidos ... 48
2.8.12.3 Otras formas de segmentos de curva ... 51
2.8.13 Las superficies y teselas ... 51
2.8.13.1 Geometría de superficies ... 51
2.8.13.2 Superficies.: ... 54
2.8.13.3 Superficies de una tesela ... 54
2.8.13.4 Extensibilidad ... 55
2.9 Unidades ... 55
2.9.1 Unidades de base ... 55
2.9.2 Unidades derivadas ... 56
2.9.3 Unidades combinadas ... 56
2.10 Tratamiento de metaobjetos ... 56
2.10.1 Observaciones generales relativas a metaobjetos ... 56
2.10.2 Parámetros. ... 58
2.10.2.1 Parámetros para sistemas de referencia y coordenadas ... 58
2.10.2.2 Parámetros de símbolos ... 58
2.10.3 Sistemas de referencia ... 58
2.11 Parámetros de tiempo de ejecución ... 59
2.12 Constricciones ... 59
2.13 Expresiones ... 61
2.14 Funciones ... 64
2.15 Vistas. ... 66
2.16 Descripciones gráficas ... 70

3 transferencia secuencial ... 76
3.1 Introducción ... 76
3.2 Reglas generales para la transferencia secuencial ... 76
3.2.1 Origen en el modelo de datos ... 76
3.2.2 Lectura de modelos extendidos ... 76
3.2.3 Organización de una transferencia: Preliminares ... 76
3.2.4 Objetos transferibles ... 76
3.2.5 Orden de los objetos dentro del dominio de datos ... 77
3.2.6 Codificación de los objetos ... 77
3.2.7 Tipos de transferencia ... 77
3.3 Codificación XML ... 78
3.3.1 Introducción ... 78
3.3.3 Estructura general de un archivo de transferencia ... 79
3.3.4 Sección de cabecera ... 80
3.3.4.1 Información relativa a la estructura de las identificaciones de objetos ... 81
3.3.4.2 Significado y contenido de la tabla Alias ... 81
3.3.5 Sección de datos ... 85
3.3.6 Codificación de los temas ... 85
3.3.7 Codificación de las clases ... 86
3.3.8 Codificación de vista ... 87
3.3.9 Codificación de relaciones ... 87
3.3.9.1 Relaciones integradas ... 88
3.3.9.2 Las relaciones no integradas ... 88
3.3.10 Codificación de las definiciones gráficas ... 88
3.3.11 Codificación de atributos ... 89
3.3.11.1 Reglas generales para la codificación de los atributos ... 89
3.3.11.2 Codificación de cadenas ... 89
3.3.11.3 Codificación de enumeraciones ... 89
3.3.11.4 Codificación de tipos de datos numéricos ... 89
3.3.11.5 Codificación de dominios con formato ... 89
3.3.11.6 Codificación de cajas negras ... 90
3.3.11.7 Codificación de los tipos de clase ... 90
3.3.11.8 Codificación de tipos de atributos de ruta ... 90

3.3.11.9 Codificación de atributos de la estructura ... 90
3.3.11.10 Codificación de las subestructuras ordenadas y no ordenadas ... 90
3.3.11.11 Codificación de coordenadas ... 90
3.3.11.12 Codificación de cadenas de líneas ... 91
3.3.11.13 Codificación de superficies y mosaicos ... 92
3.3.11.14 Codificación de referencias ... 93
3.3.11.15 Codificación de metaobjetos ... 93
3.3.11.16 Codificación del OIDType ... 93
3.4 Aplicación de herramientas XML ... 93

Apéndice A (normativo) El modelo de datos interno de INTERLIS ... 94

Apéndice B (normativo para CH) Tabla de símbolos ... 98

Apéndice C (informativo) un pequeño ejemplo de carreteras ... 102

Apéndice D (sugerencia de extensión estándar) Organización de los identificadores de objeto (OID) ... 125

Apéndice E (sugerencia de extensión estándar) Unicidad de las claves de usuario ... 128

Apéndice F (sugerencia de extensión estándar) Definición de unidades ... 131

Apéndice G (sugerencia extensión estándar) Las definiciones de tiempo ... 133

Apéndice H (sugerencia de extensión estándar) Definiciones de Color ... 137

Apéndice I (sugerencia de extensión estándar) Sistemas de coordenadas y sistemas de referencia de coordenadas ... 144

Apéndice J (sugerencia de extensión estándar) modelos de simbología ... 158

Apendice K (Informativo) Glosario ... 165

Apendice L (informativo) Indice ... 189

Lista de gráficos

Figura 1: Transferencia de datos entre varias bases de datos a través de un modelo de datos común (esquema de datos) descrito en un lenguage de descripción de datos común. ... 9
Figura 2: Especialización en el modelamiento de un concepto desde el nivel de cantón hasta un nivel de cantón (específico de país) y nivel local. ... 10
Figura 3: Jerarquía de herencia de direcciones, personas y edificios. ... 12
Figura 4: Actualización de una base de datos primaria y la posterior transferencia a bases de datos secundarias (una doble flecha significa actualización incremental). ... 15
Figura 5: Definiciones gráficas; por una parte, desarrollado con datos y visualizaciones y, por la otra, símbolos que permiten la generación de gráficos (diagrama de abstracto). ... 17
Figura 6: Los diversos rangos de aplicación de INTERLIS. Una doble fleca significa que los datos se pueden transferir de manera incremental. ... 19
Figura 7: Vías - un pequeño ejemplo. ... 20
Tabla 1: Palabras reservadas en INTERLIS 2. ... 25
Figura 8: Ejemplo de una enumeración. ... 39
Figura 9: Orientación de texto horizontal (HALIGNMENT) y vertical (VALIGNMENT). ... 41
Figura 10: Ejemplos de segmentos de curva plana. ... 47
Figura 11: Ejemplo de conjuntos planos que no representan segmentos curvos (un círculo doble indica "no liso" y un cuadro doble es "no inyectivo"). ... 47
Figura 12: Ejemplos de secuencias de línea. ... 48
Figura 13: Ejemplos de conjuntos planos que no son secuencias de línea (el círculo doble significa "no continuo" y el rombo doble "sin imagen de intervalo"). ... 48
Figura 14: Ejemplos de secuencias de línea simple (planos). ... 48
Figura 15: a) El parámetro de altura (de la flecha) no puede superar la tolerancia dada; b) traslape inadmisible de polilíneas, ya que hay otro vórtice entre el vórtice y la intersección; c) traslape inadmisible de polilíneas, ya que no hay un vórtice común. ... 50
Figura 16: Ejemplos de elementos de superficie. ... 52
Figura 17: Ejemplos de puntos en el espacio, que no representan elementos de superficie (aquí, un círculo doble significa "no liso"). ... 52
Figura 18: Ejemplos de superficies en el espacio. ... 52
Figura 19: Ejemplos de puntos planos que no representan espacios (un círculo doble marca un "punto único"). ... 52
Figura 20: Superficie plana con límites y enclavamientos. ... 53
Figura 21: a) Ejemplos de superficies generales planas; b) Ejemplos de conjuntos planos que no son superficies generales, ya que su interior no está conectado. Pero estos conjuntos planos se pueden subdividir en superficies generales ("---" muestra la subdivisión en elementos de superficie y "===" la subdivisión en superficies generales). ... 53
Figura 22: Diferentes subdivisiones posibles del límite de una superficie general. ... 53
Figura 23: Configuraciones de límites no permitidos para teselado. ... 53
Figura 24: Superficies individules (SURFACE). ... 54
Figura 25: Teselado (AREA). ... 54
Table 2: Símbolos de Unicode permitidos en INTERLIS y su codificación. ... 100
Figura 26: Diagrama de clase UML para modelos de datos. ... 102
Figura 27: Gráfico generado de descripciones de gráficos y datos. ... 124
Figura H.1: Idoneidad de los espacios de colores diferentes para fines de INTERLIS. ... 138
Figura H.2: La conversión de XYZ a L*a*b*. ... 138
Figura H.3: Conversión del espacio cartesiano L*a*b*-al formato polar $L^*C^*_{ab}h^*_{ab}$ (de acuerdo con [Sangwine/Horne, 1998]). ... 139
Figura H.4: El espacio de color $L^*C^*_{ab}h^*_{ab}$ funciona con coordenadas polares L*a*b*. ... 139
Figura H.5: Cálculo de diferencias de color en el espacio Cartesiano L*a*b*-. ... 140
Figura H.6: Coordenadas cartesianas y polares de un color, lejanas del punto cero (conversión, véase la figura H.3). ... 140
Figura H.7: Coordenadas cartesianas y polares de algunos colores. ... 142
Figura I.1: Cómo transformar la superficie de tierra en coordenadas horizontales 2D. ... 147

Prefacio

Este manual de referencia se dirige a los expertos en sistemas de información, especialmente en sistemas de información geográfica o sistemas de información de tierras. Sobre todo, debería ser de interés para las autoridades cuyo principal objetivo sea el tratamiento cuidadoso de los datos.

"INTERLIS – ein Datenaustausch-Mechanismus für Land-Informationssysteme" (INTERLIS – un mecanismo de intercambio de datos para los Sistemas de Información de Tierras), fue publicado por primera vez en 1991. El propósito de INTERLIS es lograr una descripción más precisa de los datos. El mecanismo consiste en un lenguaje de descripción conceptual y un formato de transferencia secuencial con especial atención a los datos espaciales (geodatos), permitiendo así la compatibilidad entre varios sistemas y la disponibilidad a largo plazo, es decir, archivado y documentación de datos. Hacer uso de INTERLIS en la toma de decisiones, la planificación o en la administración de procesos puede generar grandes beneficios. Muy a menudo - por ejemplo, mediante la aplicación múltiple y la producción uniforme de datos documentados y verificados - se pueden lograr ahorros económicos.

Cinco años después de su publicación, INTERLIS, retrospectivamente considerado como versión 1, ó INTERLIS 1, ha salido de su "existencia de la Bella Durmiente". Entretanto, se ha puesto a disposición del usuario una gama considerable de herramientas de software que permiten procesar los datos geográficos descritos y codificados en INTERLIS. INTERLIS se ha creado a partir de los requerimientos del “Levantamiento Catastral”, pero su gama de aplicaciones es considerablemente más amplia, como lo prueban más de un centenar de modelos y proyectos de datos que trabajan con INTERLIS diez años después de su publicación. La versión estándar "INTERLIS versión 1", en la forma adoptada en la Norma Suiza SN612030, seguirá en uso durante algún tiempo, en paralelo con sus versiones sucesoras.

Para satisfacer la demanda creciente de nuestros usuarios, se han hecho necesarias varias extensiones de INTERLIS 1, por ejemplo, la reexportación incremental, la orientación estructural de objetos o la descripción formal de la representación gráfica de objetos. En 1998 se inició un proceso que iba a durar varios años e implicaba los esfuerzos conjuntos de media docena de expertos en investigación, administración, asesoría e industria del software. Su resultado es un producto que puede considerarse una extensión de INTERLIS 1 y, al mismo tiempo, una síntesis de todos los conceptos más recientes.

En el manual de referencia INTERLIS Versión 2, nos hemos esforzado por establecer sólo las necesidades absolutas; con ejemplos y figuras que sólo aparecen allí donde puedan complementar al texto. De esta manera la especificación está claramente estructurada y es fácil de implementar. Si algunos elementos del lenguaje, tales como las vistas o las descripciones gráficas, parecen ambiciosos y exigentes, es posible que no se deba al propio INTERLIS, sino a la complejidad del campo de aplicación en si. Para resolver este problema nos basamos en los siguientes métodos: un buen ejemplo, formación básica y continua y los llamados "perfiles", es decir, subgrupos de capacidades de herramientas bien definidas de INTERLIS

Para una comprensión general de los conceptos de INTERLIS 2, se aconseja al lector examinar al menos el capítulo 1: Principios Básicos.

Extensiones de INTERLIS 2 comparadas con INTERLIS 1

Con algunas excepciones, el lenguaje descriptivo de INTERLIS 1 sólo se ha complementado y no ha sido alterado. De esta manera, hemos ampliado las posibilidades para describir relaciones entre objetos; relaciones reales como clase de asociación y atributos de referencia con REFERENCE TO (Nota: Al "->" de la sintaxis del atributo de relación de INTERLIS 1, se le ha dado un significado diferente); teniendo en cuenta que la transición de INTERLIS 1 a INTERLIS 2 se deberá lograr tan fácil como sea posible (véase el capítulo 2.7: Adecuación de las relaciones). A partir de ahora, las relaciones y los atributos de referencia pueden, bajo ciertas condiciones, referirse a objetos de otros contenedores. El Contenedor es un nuevo término para la organización conocida de objetos (es decir, datos que describen la realidad, también llamados instancias de objetos o instancias) en una base de datos. Hemos revisado el término tabla

(TABLE) que se ha convertido en clase (CLASS), conforme al cambio del formalismo relacional al orientado a objetos. Sin más especificaciones, cualquier atributo se considera opcional (se omite OPTIONAL) teniéndose que indicar como MANDATORY para que sea obligatorio. Por otra parte, la palabra clave única IDENT se ha cambiado a UNIQUE. Los nuevos conceptos orientados a objetos incluyen la herencia, entre otros, de temas, de clases, de vistas, de descripciones gráficas y de rangos de valores de atributos. Otras extensiones de gran importancia son los tipos de datos fijos (LIST, BAG), constricciones, vistas de datos, descripciones gráficas, descripciones de unidades, descripción de metaobjetos (sistemas de coordenadas y símbolos gráficos) y reexportación incremental. Además, se pueden definir extensiones específicas de usuario, como funciones y geometrías lineales. Sin embargo, esto hará que los contratos, es decir, los acuerdos con proveedores de herramientas, sean necesarios.

A partir de ahora, el lenguaje eXtensible Markup Language (XML) se encargará de la codificación de nuestro formato de transferencia INTERLIS 2. Esperamos que XML se vuelva internacionalmente extendido y universalmente aceptado y que cuente con un gran número de productos de software compatibles que puedan obtenerse en un futuro próximo.

Cualquier usuario familiarizado con INTERLIS 1 no tendrá que enfrentarse a muchos cambios, siempre y cuando renuncie al uso de los nuevos conceptos, tales como la orientación a objetos o las posibilidades de representación gráfica: el conocimiento que tiene actualmente también será aplicable en INTERLIS 2. Varias herramientas, como el compilador INTERLIS 2 disponible de forma normal, facilitarán la adaptación a la nueva versión. Los productores, que ya pensaban en posibilidades de configuración flexibles cuando se implementó INTERLIS 1, además de tener en cuenta las reglas y el arte del desarrollo de software (por ejemplo, la modularización y la abstracción), se encontrarán con que sus inversiones anteriores en desarrollo seguirán siendo de valor. Gracias a las bibliotecas de programas accesibles universalmente, los fabricantes de software podrán concentrarse completamente en la integración de sus sistemas con INTERLIS 2.

Perspectiva

INTERLIS 1 apareció en un momento en el que el lenguaje de definición de datos relacionales SQL-92 aún no se había normalizado y no se había hablado de la orientación a objetos. Fue gracias a Joseph Dorfschmied -creador de INTERLS 1- actuando con una previsión poco común, como algunos de estos conceptos, bien asentados hoy en día, fueron introducidos. Ahora que INTERLIS ha sido revisado y orientado a objetos y se han incluido conceptos específicos de informática, puede decirse que ha alcanzado un nuevo grado de madurez. De este modo, INTERLIS 2 puede utilizarse, ya hoy, como una herramienta eficaz.

Sin embargo, somos conscientes del hecho de que, incluso con la versión INTERLIS 2, nuestra búsqueda de un lenguaje de descripción de datos universal no ha terminado. Cualquier vacilación podría conducir a consecuencias similares a las producidas por la manipulación miope de los recursos de nuestra tierra. INTERLIS merece ser aceptado no sólo como un formato de intercambio de datos, sino también como una herramienta sostenible: Gracias a INTERLIS se ha dado nombre a la petición de una gestión sostenible de la tecnología.

Cada lenguaje tiene que ser estudiado e integrado con su propio método. Por lo tanto, es obvio que este manual de referencia tendrá que ser seguido por varios manuales de usuario adicionales.

Agradecimientos

En su posición de jefe de un equipo encargado de la evolución de INTERLIS y editor principal de este documento, Stefan Keller (*Hochschule für Technik Rapperswil*, anteriormente en la Dirección Federal de Levantamientos Catastrales) ha supervisado completamente todo el trabajo en INTERLIS 2 desde su inicio y, en gran medida, incluso después de su salida del Instituto Tecnológico de Rapperswil (ITR). Nos gustaría expresar nuestro agradecimiento tanto a él como a todo el "equipo núcleo de INTERLIS 2" por sus

excelentes e inigualables esfuerzos. Los miembros de este equipo son Joseph Dorfschmied (Adasys AG), Michael Germann (infoGrips Gmbh), Hans Rudolf Gnägi (Instituto Tecnológico Federal de Zürich), Jürg Kaufmann (Kaufmann Consulting), René L'Eplattenier (Departamento de Ordenamiento Territorial y Catastro, Cantón de Zúrich), Hugo Thalmann (a/m/t AG), así como Sascha Brawer (Adasys AG), Claude Eisenhut (Eisenhut Informatik AG) y para Rolf Zürcher (KOGIS) por su coordinación.

Desde principios de 2002 toda la responsabilidad de la edición de este manual de referencia recae en KOGIS, Departamento de coordinación de la Infraestructura de Datos Espaciales de la Confederación Suiza, en estrecha cooperación con la Dirección Federal de Catastro Oficial. Poco después de este cambio, tuvo lugar una revisión pública de la Asociación de Normalización Suiza (SNV), relativa a INTERLIS 2. Fueron recibidos 109, los cuales fueron revisados y, según el caso, implementados por el equipo núcleo de INTERLIS 2 en los meses siguientes. Esto ha llevado a cambios considerables en el lenguaje en comparación con la versión 2.1 del 17 de octubre de 2001. Consecuentemente, la versión de INTERLIS 2 ha sido numerada como 2.2. A finales de noviembre de 2002, tuvo lugar la votación final del INB/TK 151 de la SNV, donde finalmente INTERLIS 2 fue declarado como norma SN 612031. Después de completar algunas revisiones textuales finales, este manual de referencia podría ser publicado.

Ahora, dos años más tarde, tenemos que atender a los primeros ajustes a la norma. El desarrollo de herramientas (sobre todo, Compiler, Checker y UML-Editor), así como la implementación de algunos proyectos importantes (por ejemplo geocat.ch) han sacado a la luz varios errores y cuestiones pendientes dentro del lenguaje, intentado corregirlos o especificarlos con la presente versión 2.3.

Mediante la financiación de expertos y mediante el suministro de herramientas básicas de software, KOGIS ha contribuido en el desarrollo de la versión actual. Esperamos con interés el desarrollo de nuevas herramientas y productos innovadores basados en INTERLIS.

No existe una norma que pueda ser definida de forma individual, muy al contrario, se necesita de la participación y contribución de muchos especialistas. ¡Deseamos expresar nuestro agradecimiento a todos estos profesionales!

Wabern, abril 2006

Rolf Zürcher

1 Principios básicos

1.1 Visión general

INTERLIS permite la cooperación entre sistemas de información, en particular sistemas de información geográfica o sistemas de información territorial. Como sugiere su nombre, INTERLIS se sitúa entre (**INTER**) los sistemas de información de la tierra (**L**and **I**nformation **S**ystems). Es crucial que todos los sistemas implicados tengan una noción muy clara de estos conceptos, que son de gran importancia para su cooperación.

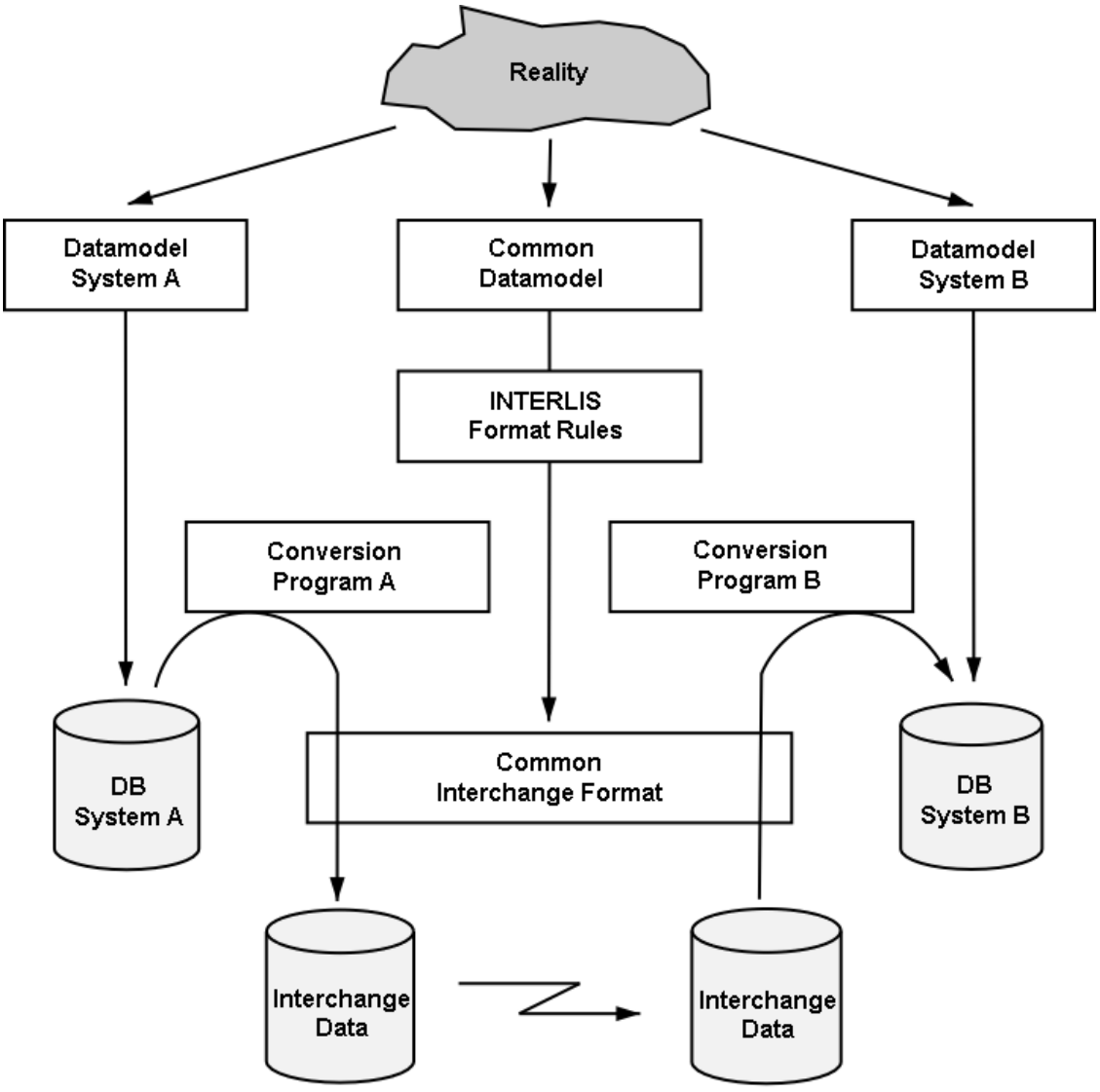

Figura 1: Transferencia de datos entre varias bases de datos a través de un modelo de datos común (esquema de datos) descrito mediante un lenguaje de descripción de datos común.

Por eso INTERLIS incluye un lenguaje descriptivo conceptual. Gracias a este lenguaje se puede describir el mundo real con el detalle que sea de interés para una aplicación determinada. Tal descripción (conceptual) se denomina modelo de aplicación o esquema de aplicación, o de forma simple, modelo o esquema, respectivamente. Unos pocos conceptos con significado estrictamente definido describen (modelizan) clases de objetos con sus respectivas características y relaciones. Además, el lenguaje de descripción de INTERLIS permite la introducción de clases de objetos derivados, haciendo posible definir

estas como vistas sobre otras clases de objetos. Tanto las clases de objetos primarias como las derivadas pueden servir como base de descripciones gráficas, sin embargo INTERLIS asegura una estricta separación entre descripciones gráficas (definición de representación) y descripción de la estructura de datos subyacente (definición de datos).

INTERLIS no apunta a ninguna aplicación específica. Sin embargo, en el diseño, dirigido hacia principios generales de la orientación a objetos, se ha prestado atención al apoyo de aquellos conceptos que son particularmente importantes para los sistemas de información geográfica. Por lo tanto, las coordenadas, líneas y superficies como tipos de datos son construcciones básicas de INTERLIS, y hay elementos del lenguaje disponibles para describir precisiones y unidades de medida. Sin embargo, las aplicaciones no geográficas también pueden ser procesadas con INTERLIS.

Los aspectos que subyacen en un campo de aplicación se pueden describir en un modelo básico. Posteriormente, este modelo se especializará de acuerdo con las necesidades específicas de un país, en etapas sucesivas, de acuerdo con las de un área determinada (condado, región o comunidad) (ver figura 2). INTERLIS 2 ofrece dos conceptos orientados a objetos como posibles herramientas para esta especialización: la herencia y el polimorfismo, asegurando así que las definiciones ya establecidas no necesiten ser repetidas o llevar a confusión de forma accidental.

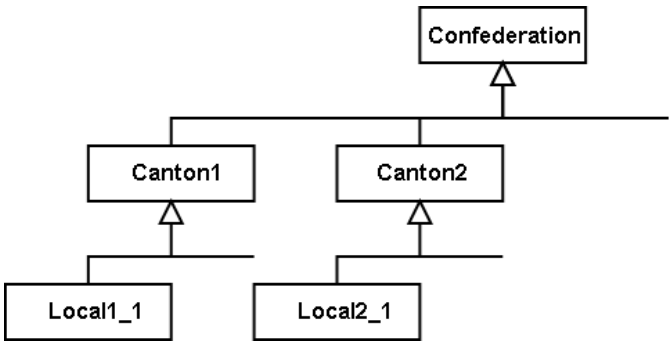

Figura 2: Especialización de la modelización de un concepto desde el nivel federal o central, a los cantones (específico del país) y el nivel local.

Las aplicaciones grandes no necesitan ser definidas en una sola descripción. Por el contrario, pueden dividirse en varias unidades descriptivas (modelos, esquemas). Una unidad de descripción puede comprender varios temas. En aras de la legibilidad de tales modelos también es posible definir un modelo como una simple traducción de otros modelos.

1.2 La utilización de modelos

En primer lugar, un modelo o esquema de INTERLIS representa un medio de comunicación para los usuarios. Su lenguaje está diseñado de tal manera que sea legible para las personas. No obstante, los modelos INTERLIS son precisos, inequívocos y no pueden dar lugar a ningún malentendido. Por lo tanto, el lenguaje textual INTERLIS se ofrece como un complemento necesario al lenguaje gráfico descriptivo *Unified Modeling Language* (UML, www.omg.org/uml).

Pero INTERLIS no se detiene aquí: Puesto que un modelo posee una significación formal y claramente definida, permite que la implementación de un servicio en un sistema informático pueda derivar automáticamente de este modelo. Por ejemplo, INTERLIS incluye un servicio de transferencia basado en XML, cuyas definiciones se generan basándose en las reglas de los respectivos modelos. La utilización del modelado de datos en estrecha relación con los servicios neutros de interfaz del sistema, se denomina Arquitectura Dirigida por el Modelo (ver "Model Driven Architecture" del OMG, www.omg.org/mda/).

Los modelos pueden basarse en conceptos básicos comunes. Dado que INTERLIS permite una descripción explícita mediante el uso de la herencia y el polimorfismo, está claro en todo momento qué conceptos son comunes y cuáles son específicos en cada caso. Esto es de gran importancia de cara a la interoperabilidad semántica buscada. Por ejemplo, el archivo de transferencia de un municipio puede ser fácilmente interpretado por una unidad administrativa de orden superior (cantón, gobierno federal), sin que las partes tengan que ponerse de acuerdo sobre un modelo único. Es suficiente con que cada nivel construya su propio modelo sobre el empleado por su unidad superior.

Es concebible y además deseable que se desarrollen nuevos servicios basados en INTERLIS, una tarea facilitada en gran medida por el uso de un compilador de modelos. Este programa lee y escribe modelos INTERLIS, permite cambios internos necesarios y examina si los modelos están de acuerdo con las condiciones sintácticas y semánticas de INTERLIS. Entre otros, este compilador puede generar automáticamente - de acuerdo con el presente servicio de transferencia INTERLIS con XML - documentos de esquema XML (www.w3org./XML/Schema) derivados de los modelos INTERLIS. Mediante la introducción de herramientas XML generales adecuadas es posible hacer que los archivos concretos de INTERLIS / XML estén disponibles para un rango aún mayor de aplicación. Mientras no se violen las condiciones de uso, este compilador INTERLIS está disponible para la producción de nuevas herramientas.

1.3 Estructuración en modelos y temas

Un modelo (o esquema) describe una imagen de nuestro mundo, ya que puede ser importante para una aplicación específica. Un modelo es una unidad autónoma, que también puede usar o extender partes de otros modelos. En cierta medida, un modelo INTERLIS puede ser comparado con módulos o paquetes de algunos lenguajes de programación.

Principalmente, se distingue entre los modelos que sólo contienen definiciones de tipo (unidades, rangos de valores, estructuras) y los modelos que incluyen datos de la realidad. Además de su nombre, los modelos también contienen información sobre el editor y la versión. Todas las descripciones con datos de la realidad se dividen en temas (*topics*). Esta división es el resultado de la propia concepción que se tiene de cuáles son las unidades de organización existentes y por quién son controlados y utilizados estos datos.

Los datos que normalmente son administrados o utilizados por diferentes entidades, también deben definirse en varios temas (*topics*). Estas interdependencias deben limitarse a lo estrictamente necesario. En los temas cuyos datos son gestionados por varias autoridades, deben omitirse las relaciones siempre que sea posible, ya que los esfuerzos especiales para mantener la consistencia son inevitables. En cualquier caso, se excluye la dependencia circular. Además de las definiciones de datos como tales, los temas (*topics*) pueden incluir también definiciones de vistas y gráficos.

Un tema puede extender a otro. De esta manera todos los conceptos definidos por el tema básico se transmiten y se pueden complementar o especializar.

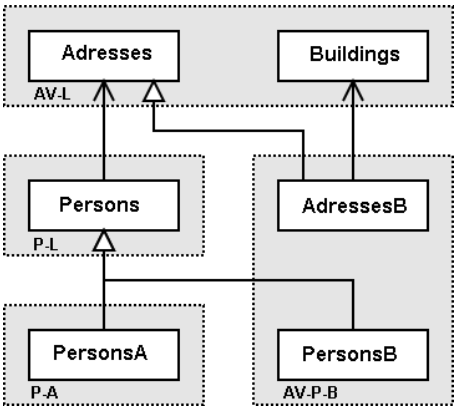

Figura 3: Jerarquía de herencia de direcciones, personas y edificios.

Por ejemplo, un modelo catastral de un país (AV-L) podría incluir los temas de "direcciones" y "edificios" (ver figura 3). Estos conceptos son independientes entre sí; la referencia se hace algorítmicamente por coordenadas. Las personas tienen direcciones, de modo que el modelo de persona de este país se basa en el modelo de tierra de catastro, por lo que el sujeto "personas" depende del tema "direcciones" del modelo de catastro.

Digamos que se planea describir a los habitantes de la región A con mayor precisión que en el modelo "Persons" del modelo nacional. Así, esta región A crea su propio modelo, tomando el tema (TOPIC) "Persons" del modelo nacional y extendiéndolo.

En la región B, por una parte se ha previsto establecer una relación explícita entre los edificios y las direcciones. Por otra, se considera inexacto el modelo nacional de personas. Una vez más, los temas (*topics*) respectivos son tomados (importados) y especializados. Ambas extensiones se combinan en un único modelo: la vista global del área B.

1.4 Concepto de objeto

1.4.1 Objetos y clases

Un objeto (también llamado instancia de objeto o simplemente instancia) consiste en los datos relativos a un elemento del mundo real y se puede identificar de forma inequívoca. Obviamente, hay numerosos objetos que poseen características similares y, por lo tanto, se pueden integrar. Esta serie de objetos (conjunto de objetos) con características similares se llama clase. A cada propiedad de una clase de objetos le corresponde al menos un atributo. En INTERLIS 1, se utilizaba el término tabla en lugar del término clase. Otros términos para clase son: conjunto de entidades, tipo de entidades, característica, etc.

Al describir una clase registramos, entre otra información, las cualidades y características que cada objeto posee. Estos son llamados atributos. Los valores de los atributos de los objetos no pueden ser elegidos al azar, sino que deben cumplir con ciertas condiciones estipuladas por la descripción de un atributo.

Respecto a esto, INTERLIS ofrece una serie de tipos básicos de datos (tipos de datos base: cadenas de texto, tipo de dato numérico, enumeraciones, coordenadas cartesianas y elípticas en 2D o 3D), en base a las cuales se pueden construir estructuras de datos nuevas y más complejas. Para hacer esta declaración más precisa aún, los atributos numéricos pueden ser suministrados con una unidad de medida y relacionados con un sistema de referencia, o las coordenadas pueden referirse a un sistema de referencia de coordenadas.

La fecha y hora son en realidad aplicaciones de dominio de valor con formato. Dado que las indicaciones de tiempo son bastante comunes, se ha definido una estructura independiente para la fecha y la hora (XMLTime, XMLDate y XMLDateTime).

Además de estos tipos de datos básicos, un atributo puede incluir también subestructuras. Cada uno de los elementos de una subestructura debe ser tratado como elemento estructural que sólo puede existir en relación con su objeto principal, no pudiéndose encontrar de otra manera que pasando por éste. La organización de los elementos estructurales se describe de manera similar a las clases.

Además del hecho de que todos los valores de los atributos deben corresponder a su respectivo tipo, pueden definirse otras condiciones. INTERLIS distingue entre las siguientes condiciones de constricción:

- Las constricciones que se refieren a un solo objeto. Estas constricciones se subdividen en requisitos que deben cumplir, obligatoriamente, cada objeto de la clase (constricciones "duras") y restricciones cuyo incumplimiento es posible, aunque en raros casos (constricciones "blandas").
- Constricciones que exigen la claridad de las combinaciones de atributos de todos los objetos de una clase.
- Constricciones que exigen la existencia de un valor de atributo en una instancia de otra clase.
- Condiciones más complejas, que se refieren a un conjunto de objetos y que tienen que ser definidas por medio de vistas.

1.4.2 Extensión de clases y el polimorfismo

Las clases son o autónomas o extienden (se especializan a partir de o heredan de) una superclase. Es decir, la descripción de una clase es independiente o contiene extensiones de otra descripción heredada. Una extensión de clase (también llamado subclase) puede proporcionar atributos y restricciones adicionales, así como definir condiciones más estrictas a las heredadas (tipos de datos, constricciones).

Cada objeto individual pertenece a exactamente una clase (en otras palabras, es instancia de objeto o instancia de una clase). Al mismo tiempo, siempre cumple con todos los requerimientos de su clase base (superclase). Por lo tanto, se puede encontrar, correspondiente a cada clase, un conjunto de objetos que son instancias de la propia clase o de una de sus extensiones. En el caso de las clases concretas, normalmente existe un subconjunto más pequeño de instancias, que pertenece exactamente a esta clase.

Una clase extendida es polimorfa en relación con sus clases base. Dondequiera que se puedan esperar instancias de una clase base, pueden aparecer instancias de una extensión (denominado polimorfismo parcial o principio de sustitución). INTERLIS ha sido diseñado de tal manera que la lectura polimorfa sea siempre posible. Por ejemplo, si se define una relación con una clase (véase el capítulo 1.4.4 Relaciones entre objetos), los objetos de una extensión tienen estas mismas relaciones. La escritura polimorfa completa no es parte de los objetivos de esta versión de INTERLIS.

Los elementos de las subestructuras no son objetos independientes, sino elementos estructurales y, por lo tanto, no forman parte del conjunto de instancias de una clase dada.

1.4.3 Metamodelos y Metaobjetos

Los sistemas de referencia y de coordenadas, así como los símbolos gráficos, desde el punto de vista de la aplicación, se representan como elementos del modelo (elementos del esquema) que pueden ser utilizados en la definición de aplicación. Puesto que diferentes sistemas de coordenadas y sistemas de referencia de coordenadas y, sobre todo, diferentes símbolos gráficos, pueden describirse de la misma manera, tiene sentido describir también sus características dentro de los modelos por medio de clases. Así, cada sistema o cada símbolo gráfico (por ejemplo, un símbolo de un punto o de tipo de línea) se corresponde con un objeto.

Los metaobjetos deben estar definidos en un metamodelo. Los objetos aplicables deben identificarse explícitamente como metaobjetos (extensiones de la clase predefinida METAOBJECT) y por lo tanto pueden ser referenciados a través de sus nombres. Para ello, es necesario que los metaobjetos estén a disposición de la herramienta, que trata la definición de aplicación por medio de contenedores (ver capítulo 1.4.5 contenedores, replicación y transferencia de datos).

1.4.4 Las relaciones entre los objetos

INTERLIS 2 (a diferencia de INTERLIS 1) distingue dos tipos de relaciones entre objetos: relaciones propiamente dichas o reales y atributo de referencia.

El término *relación real* se refiere a un conjunto de pares de objetos (en general, objetos de n tuplas). El primer objeto de cada par pertenece a una primera clase A, el segundo a una segunda clase B. Dado que la atribución de los objetos a cada par debe estar predefinida, sólo debe ser descrita, es decir, modelizada. Como se muestra más adelante en el capítulo 1.5, sobre el concepto de *Vista*, también es posible computar esta asignación mediante algoritmos, por ejemplo sobre la base de valores de atributo.

Las *Relaciones propiamente dichas* se describen como construcciones independientes, denominadas clases de relación (clases de asociación) que a su vez son extensibles. INTERLIS 2 no sólo es compatible con las relaciones uno a uno, sino que también permite relaciones múltiples y relaciones con sus propios atributos. Por lo tanto, una clase de relación es también en si misma una clase de objeto.

Las características importantes de las relaciones son:

- *Cardinalidad*: ¿Cuántos objetos de clase B (o clase A) se pueden atribuir a un objeto de clase A (o B) dentro de la relación?
- *Fuerza* - INTERLIS 2 distingue entre asociaciones, agregaciones y composiciones. En todos los casos los objetos en cuestión se pueden aplicar independientemente. En el caso de la agregación y de asociación los objetos existen independientes unos de otros. En el caso de la agregación y la composición encontramos asimetría entre las clases en cuestión. Los objetos de una clase (superclase) se describen como entidades (superobjetos), los objetos de la otra clase (subclase) son partes (subobjetos). En el caso de las asociaciones todos los objetos son iguales y sólo ligeramente conectados. Tanto la agregación como la composición son asociaciones conceptualmente dirigidas: A una entidad (superobjeto de la superclase) se le atribuyen varias partes (subobjetos de la subclase). Cuando se trata de una **agregación**, todas las partes atribuidas se copian automáticamente cuando se copia la entidad, sin embargo, cuando se elimina la entidad, las partes correspondientes permanecen intactas. En comparación con una agregación, verá que una **composición** implica además que al eliminar la entidad todas sus partes son eliminadas al mismo tiempo. Cómo se relacionan los efectos de copiado con otros objetos se describe en el capítulo 2.7.2 Fuerza de relación. Nota: Los subobjetos (partes) de composiciones son objetos identificables en oposición a elementos estructurales de subestructuras.
- *Función* - ¿Qué significado tienen las clases involucradas desde el punto de vista de la relación? Esto se determina para cada clase implicada por medio de su papel en la misma. INTERLIS 2 (a diferencia de INTERLIS 1) también admite relaciones que exceden los límites de un sujeto. Esto es así, sin embargo, con la condición de que el atributo de relación se defina dentro de una clase, según pertenezca a un sujeto, dependiendo de la clase a la que hace referencia.

Mediante el uso de un atributo de referencia creamos una relación entre un objeto, o un elemento estructural, con otro objeto. Sin embargo, esta relación sólo es conocida por el objeto de referencia y no por el objeto referido. Por lo tanto, es unilateral.

Sin violar la independencia de los temas (*topics*), es posible también definir las relaciones (es decir, relaciones propiamente dichas o reales y las de atributos de referencia) por medio de una marca especial (EXTERNAL), que creará una relación con los objetos de un contenedor del mismo o de un tema (TOPIC) diferente. Sin embargo, sólo con la condición de que la estructura dentro de la cual se define la relación pertenezca a un tema que dependa del tema a cuya clase se está refiriendo.

En interés de una estructura clara, las relaciones sólo pueden hacer referencia a clases ya conocidas en la definición de las clases de relación (o atributos de referencia).

1.4.5 Contenedores, replicación y transferencia de datos

Un contenedor es un conjunto compacto de objetos, que forman parte de un tema o de su extensión. Compacto significa que un contenedor contiene todos los objetos relacionados entre sí dentro de un determinado tema. Un ejemplo típico puede ser cierta región (ciudad, departamento o incluso un país entero) cuyos objetos, en su totalidad, están dentro de un contenedor. Un contenedor puede tener datos suministrados por diversas extensiones (por ejemplo, diferentes cantones con sus propias extensiones de temas (*topics*)), siempre que dentro de dicha comunidad de transferencia todas las etiquetas de modelos, temas y extensiones de temas sean inequívocas.

En el ámbito de las constricciones, a menudo se habla de "todos" los objetos de una clase. Básicamente esto incluye todos los objetos que tienen la calidad deseada y que realmente existen, esto es, lo que abarca el contenedor. Por diversas razones (eficiencia, disponibilidad, derechos de acceso, etc.), es evidente que el control sólo es posible dentro de contenedores localmente accesibles. En el caso de contenedores con metaobjetos, se declara explícitamente que las constricciones sólo se aplican dentro de un contenedor, ya que se supone que los metadatos son de carácter descriptivo y tienen que estar a disposición del usuario dentro del contenedor (de forma similar a lo que ocurre en una biblioteca).

Se distinguen diferentes tipos de contenedores:

- Contenedores de datos - comprende todas las instancias de clases de un tema (TOPIC).
- Contenedores de vistas - comprende todas las instancias de vistas de un tema (TOPIC).
- Contenedores con los datos básicos para gráficos - comprende las instancias de todos los datos o vistas necesarios para los gráficos de un tema (TOPIC). Permite el uso de aplicaciones de software gráfico.
- Contenedores con elementos gráficos - comprende las instancias de todos los objetos gráficos (= símbolos), necesarias para los gráficos de un tema (TOPIC). Permite el uso de un software de representación gráfica (Renderer - ver figura 5).

INTERLIS no establece reglas sobre cómo los objetos deben mantenerse dentro de sus sistemas. Las reglas sólo se aplican a la intersección entre sistemas. En la actualidad, se define la intersección para la transferencia de contenedores mediante archivos XML. No solo permite la transferencia completa de todo el contenedor, sino también la transferencia incremental de datos. Se asume que, para el caso de una transferencia completa, el receptor crea copias nuevas e independientes de los objetos, sin ninguna conexión inmediata con el objeto original. En el transcurso de una transferencia completa, los objetos solo deben estar marcados con una identificación de transferencia temporal (TID). Los TID se utilizan para transferir relaciones.

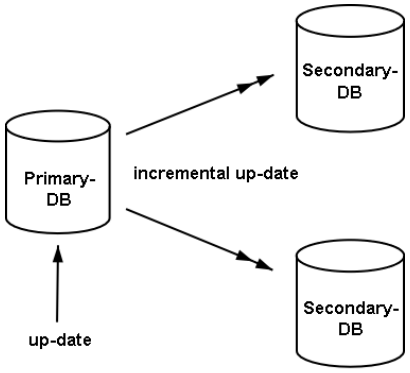

Figura 4: La actualización de una base de datos primaria y posterior traslado a las bases de datos secundarias (una flecha doble significa la actualización incremental).

En el caso de la transferencia incremental, se supone que, para empezar, el remitente proporciona un estado inicial de un contenedor de datos (se excluyen todos los demás tipos de contenedores) y, posteriormente, proporciona actualizaciones al destinatario, lo que permitirá actualizar sus datos (ver figura 4). Así, los objetos se replican y además mantienen su relación con el objeto original, es decir, no pueden ser alterados de manera independiente del original y no se les dará una nueva identificación.

Se supone que entre los gestores de bancos de datos primarios y de bancos de datos secundarios existen acuerdos contractuales (alcance, frecuencia de las actualizaciones, etc.) que en la actualidad no pueden ser cubiertos por las funcionalidades proporcionadas por INTERLIS. INTERLIS ofrece las herramientas necesarias para la actualización. Los nuevos objetos se visualizan de la misma forma que cuando se hace por primera vez una transferencia y se les asigna una identificación inequívoca de objeto (OID), que debe mantenerse en todo momento. Cada vez que se producen cambios, este OID inequívoco se usa como un punto de referencia y se actualizan todos los atributos del objeto (incluyendo todos los elementos estructurales de los atributos de estructura). De la misma manera, se pone en conocimiento del usuario los objetos que se han eliminado. Asegurar la consistencia de los objetos es responsabilidad, principalmente, del remitente (por ejemplo, observancia de las constricciones, corrección y cardinalidad de las relaciones). A tal fin, notifica al receptor cualquier alteración: objetos eliminados, actualizados, o recién creados. En interés de la independencia de los temas (*topics*), no se asume garantía de integridad en todo momento cuando se trata de varios temas. Corresponde al receptor hacer frente a las inconsistencias temporales entre los temas básicos y los temas dependientes, es decir, cuando un objeto referenciado no existe.

INTERLIS no determina los límites dentro de los cuales se garantiza la claridad del OID (ver Apéndice D *Organización de los identificadores de objeto (OID)* para un ejemplo de cómo un OID puede estructurarse). Sin embargo, pueden concebirse otras posibilidades. En cualquier caso, es de primordial importancia que todos los registradores de datos de una comunidad de transferencia apliquen correctamente las reglas que rigen la estructura de un OID, asegurando de este modo que, en todo momento, el OID permanece inequívoco dentro del ámbito de la comunidad de transferencia. Dependiendo de la estructura del OID, es posible realizar el intercambio de datos incrementales, dentro de un círculo más amplio (por ejemplo, en todo el mundo) o más pequeño (por ejemplo, dentro de una organización). Por lo tanto, el método utilizado para determinar un OID también define la comunidad de transferencia potencial.

Cualquier alteración a un objeto que no sea dentro de su contenedor original depende estrictamente del permiso de su administrador. Todos los demás contenedores secundarios pueden alterar un objeto sólo como resultado de una actualización. Por eso INTERLIS exige, dentro del marco de la actualización incremental, que no sólo los objetos sino también los contenedores deben ser identificables de manera inequívoca y permanente. Por eso, un OID también se asigna a un contenedor. Del mismo modo, en el caso de una transferencia completa, el contenedor sólo necesita una identificación de transferencia (TID). Siempre que sea importante distinguir claramente entre OID y OID de un contenedor, se habla de BOID (o Identificación del contenedor -BID *basket identification*-).

Se debe asumir que diversos objetos son registrados y gestionados dentro de un contenedor, por ejemplo, de un municipio, y que a continuación, estos contenedores en su conjunto se transmiten a una instancia superior (cantón, distrito, departamento) y, posteriormente, se integran a contenedores que contienen temas (topics) del todo el cantón (división administrativa superior), que podrían ser transferidas posteriormente a, por ejemplo, entidades al nivel de Estado. Con el fin de tener en cualquier momento garantía definitiva en cuanto al contenedor original, se suministra el BOID original con cada réplica del objeto. Esto permitirá que el receptor cree su propia administración de contenedores, indicando en cuál de sus contenedores se almacenan los objetos replicados y qué contenedor original los ha proporcionado. INTERLIS 2 proporciona las herramientas necesarias para describir ese tipo de contenedores e intercambiarlos como si fueran objetos normales. Una de las características de INTERLIS 2 es que, cuando se trata de relaciones referentes a varios temas, no sólo el OID del objeto de referencia, sino también el

BOID de sus contenedores originales. Si el receptor hace uso pleno de esta funcionalidad de INTERLIS 2, que en el caso de las relaciones que abarcan varios temas permite no sólo transferir el OID de los objetos, sino también el BOID de los contenedores originales, dispondrá de un medio eficaz para determinar qué contenedores tienen el objeto de referencia.

1.5 Concepto de Vistas

INTERLIS 2 permite al usuario no sólo modelar objetos, sino también vistas. Las vistas son, en realidad, clases virtuales cuyas instancias no son datos que provienen de un objeto real, sino datos derivados matemáticamente a partir de otros objetos.

Una definición de vista se compone de las siguientes partes:

- *Conjuntos de base*: ¿De qué clases u otras vistas, son los objetos utilizados para el cálculo de los objetos de la vista? En el caso de las clases, no sólo son las instancias respectivas las que se tienen en cuenta sino también, en el sentido de polimorfismo, todas las instancias de extensiones. INTERLIS no define los contenedores que un sistema debe tener en cuenta al calcular una vista.
- *Reglas de interrelación*: INTERLIS 2 distingue enlaces (*joins*), uniones (*unions*), agregaciones e inspecciones de subestructuras. En términos de teoría de conjuntos, los "*joins*" representan productos cruzados y unen la combinación de conjuntos de base. La agregación permite combinar elementos del conjunto base en un nuevo objeto de vista, siempre que cumplan criterios definibles. La inspección de subestructuras permite ver los elementos estructurales de una subestructura como un conjunto de elementos estructurales. Los "*joins*" y las agregaciones también pueden concebirse como asociaciones virtuales.
- *Selección* - ¿Cuáles de los objetos calculados deberían formar parte de la vista? INTERLIS permite definir condiciones complejas sobre este aspecto.

1.6 Concepto de gráfico

Las descripciones gráficas se basan en clases o vistas, usando una determinada proyección, y declaran en las denominadas definiciones gráficas (ver figura 5 y el apéndice K *Glosario*), qué símbolos gráficos necesitan ser asignados a los objetos de una vista (por ejemplo, símbolo de un punto, línea, símbolo de área o etiqueta de texto), permitiendo al usuario gráfico crear objetos gráficos presentables. Los símbolos gráficos en sí han sido definidos en un modelo adicional de simbología, en el cual se pueden encontrar las características del símbolo.

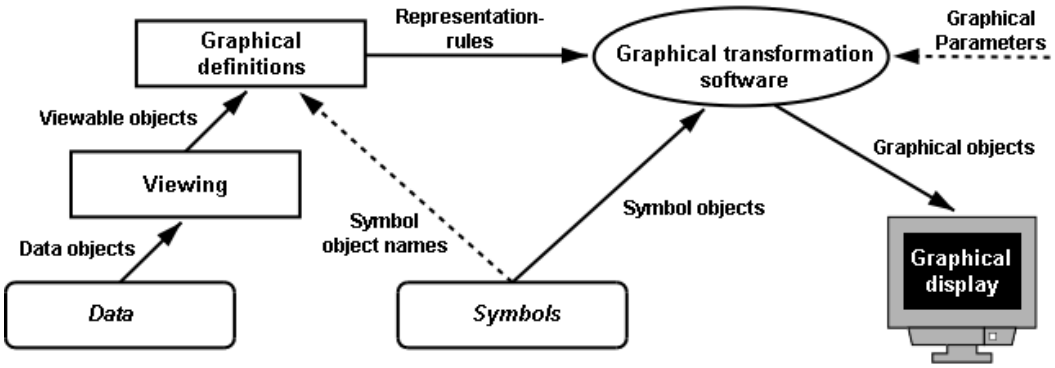

Figura 5: Definiciones de Gráfico, por una parte, basadas en datos y vistas, y por otro lado en los símbolos que permiten la generación de gráficos (diagrama abstracto).

La referencia a los símbolos gráficos en cuestión se realiza mediante el uso de nombres (ver nombres de objeto de simbología en la figura 5). Los símbolos gráficos como tales (también llamados objetos de

simbología) están contenidos como objetos (datos) en sus respectivos contenedores. Un contenedor con tales objetos de simbología también se conoce como una librería de símbolos.

En el modelo de simbología se determina, para cada tipo de símbolo, a través de la respectiva clase de símbolo, qué parámetros adicionales son necesarios para su representación (por ejemplo, posición y orientación de un símbolo). Por lo tanto, la interfaz con el subsistema gráfico (el software de aplicación gráfica) de los sistemas respectivos no se determina por INTERLIS mismo, sino por los modelos de simbología. Hasta este punto es posible declarar parámetros globales (por ejemplo, la escala de representación), momento en el que son conocidos dentro del subsistema gráfico siendo posible su influencia en la decisión de qué símbolos se van a utilizar en la representación. Puesto que tales definiciones están estrechamente relacionadas con las posibilidades reales de los sistemas, estos parámetros, así como las características de los símbolos, deben ser objeto de un acuerdo contractual con los proveedores (véase el capítulo 1.7 Contratos).

Una descripción gráfica no necesita regular la implementación en símbolos de una manera definitiva. Por el contrario, puede ser heredada de una descripción gráfica diferente. Entonces es posible complementar los parámetros que aún están sin definir en las definiciones base, o se pueden reemplazar instrucciones de representación existentes.

1.7. Contratos

Para que INTERLIS 2 sea capaz de satisfacer todo tipo de requerimientos, contiene también construcciones cuya implementación no está regulada por la definición; por ejemplo funciones, tipos de líneas, símbolos (los detalles se tratarán en el capítulo 2: Lenguaje de Descripción). Sin más esfuerzo, no se puede lograr el objetivo de que una descripción de INTERLIS 2 se convierta en un servicio de forma automática. En aras de la simplicidad, sin embargo, INTERLIS no ofrece posibilidades de definiciones adicionales para estos casos (como, por ejemplo, un lenguaje de programación propiamente dicho para la definición de funciones).

Con el fin de hacer factible la conversión automática dentro de al menos ciertos límites (por ejemplo, un país o un determinado rango de aplicación), estas construcciones sólo deberían ser admisibles dentro de los modelos cubiertos por contratos.

Los contratos son acuerdos entre partes que definen modelos y partes que suministran herramientas basadas en INTERLIS. Por lo general son sólo modelos básicos de un sector temático o de un país los que son objeto de un acuerdo de este tipo. Con el fin de lograr la definición de estos modelos básicos, los modeladores y los proveedores de herramientas deben trabajar juntos y firmar acuerdos. A partir de entonces los proveedores de herramientas deben ser capaces de realizar los elementos exigidos por los modelos (por ejemplo, funciones) independientemente de la aplicación concreta. En consecuencia, los modelos concretos se pueden implementar de forma automática (es decir, sin la ayuda del proveedor de la herramienta).

Es de gran importancia que dichas construcciones complementarias sean discutidas en general y neutrales respecto a cualquier sistema específico, así como que estén a disposición del público de la misma manera que el propio modelo. De lo contrario, uno de los objetivos más importantes de INTERLIS, esto es, la apertura y la interoperabilidad, no se podrían garantizar.

1.8. Servicios, capacidades de las herramientas y conformidad

INTERLIS 2 permite la descripción conceptual de los datos y define una transferencia neutral respecto al sistema. INTERLIS 2 renuncia explícitamente a todas las directivas relativas a la implementación con el fin de seguir siendo independiente de cualquier sistema. Por lo tanto, a menudo para el funcionamiento en la práctica se planteará la cuestión de si una determinada herramienta o servicio está en conformidad con INTERLIS o no.

INTERLIS no estipula que sólo sean concebibles los extremos la conformidad o de la no conformidad. Por el contrario, un servicio se podrá ajustar a INTERLIS en algunos aspectos, aunque no reúna los demás requisitos.

En el más simple de los casos un determinado sistema cumple con las especificaciones INTERLIS en un caso concreto (para un conjunto determinado de modelos, sólo lectura o sólo escritura o ambos, etc.).

Lo ideal sería que un conjunto de modelos INTERLIS pueda ser entregado a una herramienta de INTERLIS, con lo que la herramienta se adapta automáticamente a sus servicios y capacidades de acuerdo con la situación definida por los modelos. En muchos casos, sin embargo, esta adaptación exigirá esfuerzos manuales adicionales, como la configuración del sistema o incluso la programación. Sin duda, un requisito esencial en este tipo de herramientas es su capacidad para leer modelos de INTERLIS . Por encima de todo, esto significa que las capacidades del sistema se ofrecen correctamente de acuerdo con las construcciones de INTERLIS , sobre todo en lo que respecta a las construcciones de la herencia.

Desde el punto de vista de INTERLIS, las capacidades de dichas herramientas se pueden clasificar de la siguiente manera (capacidades, funcionalidades o servicios):

- Leer datos, incluyendo vistas almacenadas, sin generación de objetos de vista.
- Leer datos, incluyendo vistas almacenadas y la generación de objetos de vista.
- Examinar consistencias.
- Escribir vistas.
- Generar gráficos, incluyendo la lectura de vistas.
- Tratar y escribir datos.
- Generar identificadores de objeto (OID).
- Leer actualizaciones incrementales.
- Escribir actualizaciones incrementales.

Es perfectamente concebible que una determinada herramienta o servicio tengan ciertas capacidades (por ejemplo, la actualización incremental) para un modelo o tema (datos o vista), pero no los soportarán con otros modelos o temas.

Por otra parte, se plantea la cuestión de qué contratos (es decir, qué modelos básicos) deben ser soportados por una herramienta.

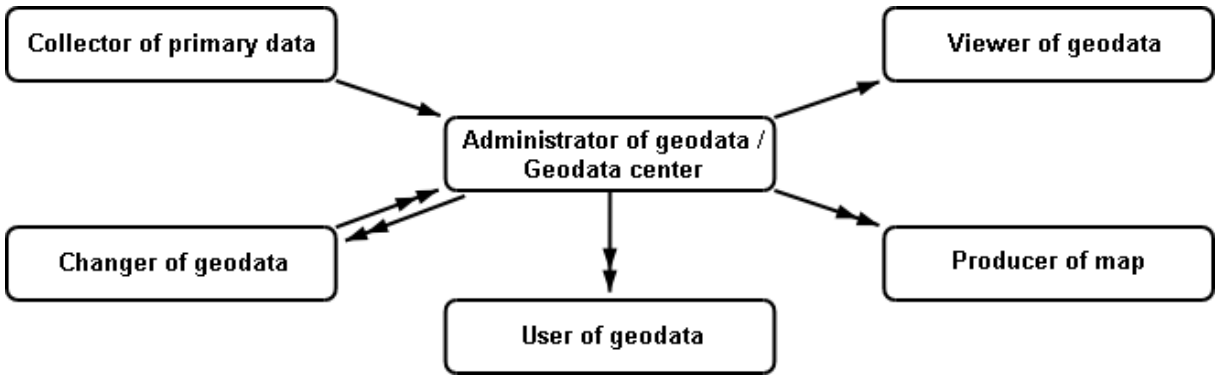

Figura 6: Los diferentes rangos de aplicación de INTERLIS. Una flecha doble significa que los datos pueden ser transferidos de forma incremental.

Al observar un ejemplo de los esfuerzos conjuntos de distintas partes interesadas, se hace evidente que se necesitarán diferentes capacidades o servicios de INTERLIS 2, en función del área operativa y de la función del usuario (ver figura 6). Dentro de una aplicación (es decir, un modelo dr INTERLIS) con diferentes temas (TOPICS), un único usuario puede asumir varias funciones:

- *Recolector de datos primarios*: trabajar con datos y escribirlos, generar OID, si se presenta la ocasión.
- *Cambiador de geodatos* (actualización, completar): recoger datos, trabajar con datos, escribir datos, generar OID, producir incrementos, leer incrementos, validar consistencias (localmente).
- *Administrador de geodatos, centro de geodatos*: recopilar datos, escribir y trabajar con datos, producir OID, leer incrementos, validar consistencias (a nivel global).
- *Usuario de geodatos*: recoger datos, leer incrementos.
- *Visor de geodatos:* recoger datos, producir objetos de vistas y representaciones gráficas.
- *Productores de mapas*: recoger datos, leer incrementos; leer vistas y producir representaciones gráficas.

1.9 Un pequeño ejemplo como introducción

En el Apéndice C: *Caminos, un pequeño ejemplo*, se provee un ejemplo que presenta los elementos más importantes de INTERLIS en el marco de una sencilla aplicación.

Figura 7: Caminos. Un pequeño ejemplo.

1.10 ¿Cómo está estructurado este documento?

Este manual de referencia se subdivide de la siguiente manera: En el capítulo 2 se define formalmente el lenguaje de descripción. En el capítulo 3 se especifica la transferencia secuencial de datos geográficos. Este capítulo se divide en una parte más general que trata de los actuales y futuros servicios de transferencia secuencial de INTERLIS 2, y una parte más específica sobre el servicio de transferencia INTERLIS 2 con XML.

Los apéndices están dispuestos en dos normativos, A y B, complementados con un apéndice informativo C, varias sugerencias de extensión del estándar, desde apéndice D hasta el J; así como un glosario (apéndice K) y un índice (Anexo L).

Los apéndices normativos son parte integral de esta especificación. Se recomienda encarecidamente las sugerencias de la extensión al estándar (anexos del D al J) para su implementación. Estos son apéndices

informativos (no normativos), de carácter autónomo. Las implementaciones pueden encontrar diferentes respuestas a distintas preguntas, permanenciendo conformes a INTERLIS 2. En tal caso les corresponde a las partes involucradas en la transferencia llegar a un acuerdo sobre la especificación. La mayoría de estos campos de aplicación serán objeto de un contrato.

2 Descripción del lenguaje

2.1 Sintaxis aplicada

Con el fin de determinar un modelo conceptual de datos (esquema de datos) y los parámetros de una transferencia de datos, en los siguientes capítulos se definirá un lenguaje formal. Este lenguaje se ha definido formalmente. Las reglas de sintaxis establecen la secuencia admisible de símbolos.

De esta manera, esta descripción es análoga a la empleada generalmente en lenguajes de programación modernos. De ahí que solamente se presenta un breve esquema necesario para la comprensión básica. Le recomendamos consultar la literatura técnica para más detalles (por ejemplo, se puede encontrar una breve introducción en “Programming in Modula-2” por Niklaus Wirth).

En el sentido de la notación extendida Backus-Naur (EBNF), una formula se estructura de la siguiente manera:

```
Nombre-Formula = Expresión-Formula.
```

La expresión de la fórmula es una combinación de:

- Palabras fijas del lenguaje (incluyendo caracteres especiales). Están entre apóstrofos, por ejemplo, ‘BEGIN’
- Referencias de otras fórmulas, con indicación del nombre de la fórmula.

Las combinaciones válidas son las siguientes

Composición secuencial

```
a b c          primero a, luego b, luego c.
```

Agrupación

```
( a )          los paréntesis agrupan expresiones de formula.
```

Elección

```
a | b | c      a, b o c.
```

Opción

```
[ a ]          a o nada (vació).
```

Repetición Opcional

```
{a}            cualquier secuencia de a o nada (vacío).
```

La repetición obligatoria (como complemento a EBNF)

```
(* a *)        cualquier secuencia de a, pero por lo menos una
```

Ejemplos:

```
(a|b)(c|d)     ac, ad, bc o bd
a[b]c          abc o ac
a{ba}          a, aba, ababa, abababa, ...
{a|b}c         c, ac, bc, aac, abc, bbc, bac, ...
a(*b*)         ab, abb, abbb, abbbb, ...
(*ab|[c]d*)    ab, d, cd, abd, dab, cdab, ababddd, cdababcddcd, ...
```

A menudo, a uno le gustaría utilizar una fórmula sintácticamente idéntica en diferentes contextos, para diferentes propósitos. Con el fin de expresar esta correlación, habría que escribir una fórmula adicional:

```
Example = 'CLASS' Classname '=' Classdef.
Classname = Name.
```

Con el fin de evitar este desvío, es preferible usar la siguiente notación abreviada:

```
Example = 'CLASS' Class-Name '=' Classdef.
```

La fórmula *Class-Name* no está definida. Sintácticamente la regla "*Name*" se aplica directamente (véase el capítulo 2.2.2 Nombres). Teniendo en cuenta su significado, sin embargo, el nombre es un *class-name*. Por lo tanto "class" prácticamente se convierte en un comentario.

2.2 Símbolos básicos del lenguaje

El lenguaje de descripción presenta las siguientes clases de símbolos: nombres, concatenaciones, figuras, explicaciones, símbolos especiales, palabras reservadas y comentarios.

2.2.1 Los códigos de caracteres utilizados, espacios en blanco y finales de la línea

El lenguaje en sí sólo utiliza los caracteres imprimibles US-ASCII (32 a 126). Qué otros símbolos, aparte del espacio en blanco, se consideran intercalados, tiene que ser determinado por la implementación actual del compilador. También depende de esta implementación determinar qué símbolos o combinación de símbolos marcan el final de una línea. Del mismo modo, depende de la implementación del compilador la memorización de los caracteres (juego de caracteres). Esto puede variar dependiendo de la plataforma utilizada.

En lo que se refiere a los comentarios, también es admisible usar caracteres tales como las diéresis, los acentos, etc.

2.2.2 Nombres

Un nombre se define como una secuencia de un máximo de 255 letras, dígitos y guiones bajos, en el que el primer carácter tiene que ser una letra. Se distingue entre mayúsculas y minúsculas. Los nombres que coinciden con palabras reservadas del lenguaje no se admiten (véase el capítulo 2.2.7 Caracteres especiales y palabras reservadas).

Reglas de Sintaxis:

```
Name = Letter { Letter | Digit | '_' }.
Letter = ( 'A' | .. | 'Z' | 'a' | .. | 'z' ).
Digit = ( '0' | '1' | .. | '9' ).
HexDigit = ( Digit | 'A' | .. | 'F' | 'a' | .. | 'f' ).
```

Más información sobre la unicidad y la validez de dominio de nombres se encuentra en el capítulo 2.5.4 Espacio de nombres.

2.2.3 Concatenaciones

Las concatenaciones aparecen en relación con constantes. Comienzan y terminan con comillas y no podrán rebasar una línea. \" representa una comilla y \\ representa una barra invertida, dentro de una cadena de texto.

Una secuencia de \u, inmediatamente seguida por exactamente cuatro hexadígitos, representa cualquier signo elegido de Unicode. Se deben marcar los símbolos más allá de U+10000, como en la codificación UTF-16, con dos códigos de sustitución (ver www.unicode.org).

Regla de sintaxis:

```
String = '"' { <any character except '\' or '"'>
             | '\"'
             | '\\'
             | '\u' HexDigit HexDigit HexDigit HexDigit
             } '"'.
```

2.2.4 Dígitos

Los dígitos pueden aparecer en diferentes formas: números enteros positivos incluyendo el 0 (PosNumber), números enteros (Number), números decimales (Dec) y números estructurados. Con números decimales, la escala puede indicarse mediante potencias de diez (por ejemplo, 1E2 es 100, 1E-1 es 0,1). Los números estructurados sólo tienen sentido en relación con las unidades y rangos de valores (por ejemplo, la hora). Sólo el valor de un número es importante y no su representación. Por ejemplo, 007 es lo mismo que 7.

Reglas de Sintaxis:

```
PosNumber = (* Digit *).
Number = [ '+' | '-' ] PosNumber.
Dec = ( Number [ '.' PosNumber ] | Float ).
Float = [ '+' | '-' ] '0.' ( ( '1' | '2' | .. | '9' ) [ PosNumber ]
                           | (* '0' *) ) Scaling.
Scaling = ( 'e' | 'E' ) Number.
```

Ejemplos:

```
PosNumber:    5134523             1             23
Número:           123          -435          +5769
Dec:          123.456    0.123456e4      -0.123e-2
Float:          0.1e7  -0.123456E+4    0.987e-100
```

2.2.5 Conjuntos de propiedades

Es necesario asignar propiedades a una temática por diversos motivos. Para ello, se usa la sintaxis general:

Regla de sintaxis:

```
Properties = [ '(' Property { ',' Property } ')' ].
```

Con el fin de definir que en un cierto lugar dentro de una regla de sintaxis tales propiedades deben ser definidas, se inserta en la sintaxis la siguiente construcción:

```
'Properties' '<' Property-Keyword { ',' Property-Keyword } '>'
```

Por lo tanto, se escribe "Properties" y se definen entre paréntesis angulares (< y >) las palabras clave admisibles. Si se toma por ejemplo la regla ClassDef (véase el capítulo 2.5.3 Clases y Estructuras), con "Properties<ABSTRACT, EXTENDED, FINAL>" las palabras claves "ABSTRACT", "EXTENDED", "FINAL" serán aceptados para la descripción de las propiedades. En una definición INTERLIS 2, entre otras, serían posibles las siguientes definiciones:

```
CLASS A (ABSTRACT) = .....
CLASS A (EXTENDED, FINAL) = .....
```

2.2.6 Explicaciones

Se requieren explicaciones siempre que las circunstancias requieran una descripción más cercana. Desde el punto de vista del mecanismo estándar, la explicación no será interpretada más allá, es decir, se considera un comentario. Sin embargo, es perfectamente posible formalizar explicaciones con más detalle, con el fin de prepararlos para su posterior procesamiento automatizado. El uso de // marca el comienzo y el final de una explicación. Dentro de una explicación nunca debe haber dos trazos diagonales consecutivos.

Regla de sintaxis:

```
Explanation = '//' any character except // '//'.
```

2.2.7 Caracteres especiales y palabras reservadas

En relación con las reglas de sintaxis del lenguaje; aunque, no en lo que respecta a la descripción de datos; los símbolos especiales, así como las palabras reservadas, siempre aparecen entre apóstrofos, como por ejemplo 'MODEL'. En principio, todas las palabras reservadas están escritas en mayúsculas. Con el fin de evitar conflictos entre nombres y palabras reservadas, aconsejamos no crear nombres en mayúsculas.

Se han usado las siguientes palabras reservadas. Algunas de ellas, representadas en cursiva no negrita, ya se usaban en INTERLIS 1, permaneciendo reservadas por razones de compatibilidad:

ABSTRACT	**ACCORDING**	**AGGREGATES**	**AGGREGATION**
ALL	**AND**	**ANY**	**ANYCLASS**
ANYSTRUCTURE	**ARCS**	**AREA**	**AS**
ASSOCIATION	**AT**	**ATTRIBUTE**	**ATTRIBUTES**
BAG	**BASE**	**BASED**	**CONTENEDOR**
BINARY	**BLACKBOX**	*BLANK*	**BOOLEAN**
BY	**CARDINALITY**	**CIRCULAR**	**CLASS**
CLOCKWISE	*CODE*	**CONSTRAINT**	**CONSTRAINTS**
CONTINUE	**CONTINUOUS**	*CONTOUR*	**CONTRACTED**
COORD	*COORD2*	*COORD3*	**COUNTERCLOCKWISE**
DATE	*DEFAULT*	**DEFINED**	*DEGREES*
DEPENDS	*DERIVATIVES*	**DERIVED**	*DIM1*
DIM2	**DIRECTED**	**DOMAIN**	**END**
ENUMTREEVAL	**ENUMVAL**	**EQUAL**	**EXISTENCE**
EXTENDED	**EXTENDS**	**EXTERNAL**	**FINAL**
FIRST	*FIX*	*FONT*	**FORM**
FORMAT	*FREE*	**FROM**	**FUNCTION**
GRADS	**GRAPHIC**	**HALIGNMENT**	**HIDING**
I16	*I32*	*IDENT*	**IMPORTS**
IN	**INHERITANCE**	**INSPECTION**	**INTERLIS**
JOIN	**LAST**	**LINE**	*LINEATTR*
LINESIZE	**LIST**	**LNBASE**	**LOCAL**
MANDATORY	**METAOBJECT**	**MODEL**	**MTEXT**
NAME	**NO**	**NOT**	**NULL**
NUMERIC	**OBJECT**	**OBJECTS**	**OF**
OID	**ON**	*OPTIONAL*	**OR**
ORDERED	**OTHERS**	**OVERLAPS**	**PARAMETER**
PARENT	*PERIPHERY*	**PI**	**POLYLINE**
PROJECTION	*RADIANS*	**REFERENCE**	**REFSYSTEM**
REQUIRED	**RESTRICTION**	**ROTATION**	**SET**
SIGN	**STRAIGHTS**	**STRUCTURE**	**SUBDIVISION**
SURFACE	**SYMBOLOGY**	*TABLE*	**TEXT**
THATAREA	**THIS**	**THISAREA**	*TID*
TIDSIZE	**TO**	**TOPIC**	*TRANSFER*
TRANSIENT	**TRANSLATION**	**TYPE**	**UNDEFINED**
UNION	**UNIQUE**	**UNIT**	**UNQUALIFIED**
URI	**VALIGNMENT**	**VERSION**	**VERTEX**
VERTEXINFO	**VIEW**	**WHEN**	**WHERE**
WITH	**WITHOUT**		

Tabla 1: Palabras reservadas en INTERLIS 2

2.2.8 COMENTARIOS

Se dispone de dos tipos de comentarios:

2.2.8.1 Comentario de línea

Dos signos de admiración, uno inmediatamente después del otro, marcan el comienzo de una línea de comentario. El final de línea cierra el comentario de línea.

Regla de sintaxis:

```
!! Comentario de línea; va hasta el final de la línea
```

2.2.8.2 Comentario de bloque

Un trazo diagonal y un asterisco introducen un comentario de bloque; un asterisco y un trazo diagonal marcan su fin. Puede extenderse sobre varias líneas y puede contener en sí comentarios de línea; no permitiéndose comentarios de bloque intercalados.

Regla de sintaxis:

```
/* Block comment,
    additional line comment * /
```

2.3 Regla principal

Cada unidad de descripción comienza indicando la versión del lenguaje. Así, se sientan las bases para los suplementos del lenguaje en una fecha posterior. En este documento se describe la versión 2.3 de INTERLIS.

Posteriormente se encontrarán las descripciones de los modelos.

Regla de sintaxis:

```
INTERLIS2Def = 'INTERLIS' Version-Dec ';'
               { ModelDef }.
```

2.4 Herencia

Las diferentes construcciones de INTERLIS se pueden extender, en el sentido de la forma de pensar orientada a objetos: una primera definición crea la base a partir de cual se pueden realizar especializaciones en varios pasos.

Los temas (*topics*), las clases, las vistas, las definiciones gráficas, las unidades y los dominios pueden extender sus construcciones básicas correspondientes (palabra clave EXTENDS, también EXTENDED), con lo que heredan todas sus propiedades. En ciertos casos, cabe la posibilidad de que, indicando EXTENDED, se extienda una construcción definida como principal, manteniendo su nombre.

FINAL impide la extensión de una definición. Algunas construcciones se pueden definir de una forma aún incompleta (palabra clave ABSTRACT), completándose posteriormente a través de una extensión para convertirse en una definición concreta.

Con el fin de indicar todas las palabras clave admisibles dentro de un determinado contexto, se utiliza la notación de propiedad general (véase el capítulo 2.2.5 conjuntos de propiedades).

2.5 Modelos, temas (*topics*) y clases

2.5.1 Modelos

El término "modelo" significa ‘una definición completa autosuficiente’. Según el tipo de modelo, puede contener varias construcciones.

Un modelo de tipo puro (TYPE MODEL) sólo puede declarar unidades de medida, dominios, funciones y tipos de líneas.

Un modelo de sistema de referencia (REFSYSTEM MODEL) debería declarar las definiciones de un modelo de tipo, así como los temas (*topics*) y las clases relacionados con las extensiones de las clases predefinidas AXIS, o REFSYSTEM (ver capítulo 2.10.3 Sistemas de Referencia). El lenguaje no puede reforzar la observancia de esta regla. Es responsabilidad del usuario cumplir con ella.

Un modelo de simbología (SYMBOLOGY MODEL) debe declarar definiciones de un modelo de tipo, así como los temas (*topics*) y las clases relacionadas con las extensiones de la clase predefinida SIGN, y

además los parámetros de tiempo de ejecución (véase el capítulo 2.16 Descripciones gráficas y el capítulo 2.11 parámetros de tiempo de ejecución). Es responsabilidad del usuario el observar esta regla. Los modelos de simbología sólo pueden ser permitidos en relación con los contratos, ya que tienen que ser adaptados para su tratamiento por el sistema.

Cuando no se definen tales propiedades restrictivas del modelo, un modelo puede contener cualquier construcción concebible.

A continuación del nombre del modelo, se puede indicar el idioma de forma opcional. Siempre que sea posible, esto debe hacerse de acuerdo con la norma ISO-639 es decir, dos letras y usando minúsculas (ver www.iso.ch/), donde: "es" representa español, "fr" francés, "it" italiano, "en" inglés. De acuerdo con la norma ISO-3166 se puede añadir un código de país, separado por un guion bajo para indicar una variedad del idioma utilizado en un país específico: por ejemplo "es_CO" sería para el español empleado en Colombia. Esta indicación tiene un valor documental, en relación con la posibilidad de declarar una traducción de otro modelo (TRANSLATION OF). Ambos modelos deben ser exactamente idénticos en su estructura, pudiendo sólo diferir en los nombres empleados. Sin embargo, la declaración "TRANSLATION" no está ligada a la indicación del idioma. Por ejemplo, con el fin de apoyar el uso local del lenguaje o vocabulario comercial particular, es admisible agregar traducciones en el idioma original.

Después de esto el autor del modelo se identifica mediante la indicación de la correspondiente URI (véase capítulo 2.8.1.). Se espera que el nombre del modelo sea inequívoco dentro de este contexto.

Regla de sintaxis:

```
ModelDef = [ 'CONTRACTED' ] [ 'TYPE' | 'REFSYSTEM' | 'SYMBOLOGY' ]
           'MODEL' Model-Name [ '(' Language-Name ')' ]
             'AT' URI-String
             'VERSION' ModelVersion-String [ Explanation ]
           [ 'TRANSLATION' 'OF' Model-Name '[' ModelVersion-String ']' ]
             '='
               { 'IMPORTS' [ 'UNQUALIFIED' ] Model-Name
               { ',' [ 'UNQUALIFIED' ] Model-Name } ';' }
               { MetaDataBasketDef
               | UnitDef
               | FunctionDef
               | LineFormTypeDef
               | DomainDef
               | RunTimeParameterDef
               | ClassDef
               | StructureDef
               | TopicDef }
           'END' Model-Name '.'.
```

Por medio de la versión del modelo se pueden distinguir diferentes versiones (sobre todo diferentes niveles de desarrollo). En las explicaciones es posible añadir información adicional como las observaciones relativas a la compatibilidad con versiones anteriores. Sin embargo, en determinado momento sólo debe existir una versión del modelo. Es por ello que no se indicará ninguna versión al importar modelos. Si un modelo es una traducción de otro, su versión tiene que ser indicada entre paréntesis. Con la indicación de una versión dentro del alcance de una definición de traducción (TRANSLATION OF) sólo se mostrará cuál es la versión básica utilizada para esta traducción, es decir, qué versión básica tendrá exactamente las mismas estructuras.

Cada vez que un modelo utiliza elementos del lenguaje donde es necesario exigir un contrato, este modelo tiene que ser señalizado específicamente (palabra clave CONTRACTED).

Cada vez que una construcción INTERLIS se refiere a una definición indicada en otro modelo, este modelo tiene que ser importado (palabra clave IMPORTS). De esta manera, los temas pueden ser extendidos y

creadas las referencias a clases. IMPORTS sólo ofrece la posibilidad de su uso. Cuando se utilizan las definiciones importadas estas todavía tienen que ser referidas con un nombre cualificado (modelo, tema), a menos que se aplique la palabra clave UNQUALIFIED. Los temas (*topics*) sólo pertenecerán a un modelo (y pueden ser transferidos de acuerdo con el capítulo 3.3.6 Codificación de temas), si se han definido dentro de este modelo (regla TopicDef).

Un modelo de base predefinido "INTERLIS" (véase Apéndice A: *El modelo interno de datos INTERLIS*) está conectado al lenguaje. No tiene que ser importado. No obstante, sus elementos sólo están disponibles con los nombres no cualificados, si el modelo se introduce con IMPORTS UNQUALIFIED INTERLIS.

2.5.2 Temas

Un tema (palabra reservada TOPIC) contiene todas las definiciones necesarias para describir una parte específica de la realidad. Un tema también puede definir tipos, como unidades de medida, dominios o estructuras o puede utilizar los que pertenecen al modelo envolvente o a un modelo importado.

Pueden definirse entre paréntesis () las propiedades de herencia. Dado que una extensión de un tema siempre se refiere a un tema de un nombre diferente, EXTENDED no tendría sentido y por lo tanto no es admisible.

Reglas de Sintaxis:

```
TopicDef = [ 'VIEW' ] 'TOPIC' Topic-Name
               Properties<ABSTRACT,FINAL>
               [ 'EXTENDS' TopicRef ] '='
                   [ 'BASKET' 'OID' 'AS' OID-DomainRef ';' ]
                   [ 'OID' 'AS' OID-DomainRef ';' ]
                   { 'DEPENDS' 'ON' TopicRef { ',' TopicRef } ';' }
                   Definitions
           'END' Topic-Name ';'.

Definitions =  { MetaDataBasketDef
               | UnitDef
               | FunctionDef
               | DomainDef
               | ClassDef
               | StructureDef
               | AssociationDef
               | ConstraintsDef
               | ViewDef
               | GraphicDef }.

TopicRef = [ Model-Name '.' ] Topic-Name.
```

Con respecto a un tema determinado, que contiene clases concretas, pueden existir un número indefinido de contenedores (bases de datos, etc.) Todos ellos poseen una estructura correspondiente al tema, pero contienen diferentes objetos.

Un contenedor de datos sólo presenta instancias de clases (y sus subestructuras). Si un tema contiene descripciones gráficas, se pueden configurar tres tipos de contenedores.

- Contenedores de datos.
- Contenedores con los datos básicos para los gráficos. Tales contenedores contienen las instancias de clases o vistas que sientan las bases para las descripciones gráficas.
- Contenedores de gráficos. Estos contenedores contienen objetos gráficos que han sido generados (según la definición de los parámetros de los símbolos empleados).

Como regla los contenedores y objetos disponen de una identificación de objetos. Sus dominios son el resultado de la definición correspondiente: BASKET OID AS para las identificaciones de objetos de

cualquier contenedor, OID AS para las identificaciones de objetos de cualquier objeto, siempre que ninguna definición específica se haya hecho con la clase correspondiente. Una vez que un dominio de OID se ha asignado a un tema, esto ya no se puede modificar en extensiones. En muchos casos será adecuado utilizar el dominio estándar STANDARDOID (véase el capítulo 2.8.9: Dominios de identificaciones de objetos, así como los apéndices A y D). Las definiciones relativas a las identificaciones de objeto (BASKET OID AS, OID AS) sólo son admisibles dentro del alcance de los contratos. Si no existe una definición OID para un tema o una determinada clase, no se proporcionará una identificación de objetos estable para estos contenedores u objetos y por lo tanto no será posible el intercambio de datos incremental.

Con excepción de indicaciones específicas, un tema es, en materia de datos, independiente de otros temas. Las dependencias relacionadas con datos, surgen como consecuencia de las relaciones, o atributos de referencia, que se refieren a diferentes contenedores, a través de restricciones especiales o por medio de la construcción de vistas o definiciones gráficas sobre clases u otras vistas, pero no como consecuencia de la utilización de metaobjetos (véase el capítulo 2.10.1: Observaciones generales en referencia a metaobjetos). En aras de un reconocimiento claro de dichas dependencias, estas deben ser declaradas explícitamente en el encabezado del tema (palabra reservada DEPENDS ON). Las definiciones detalladas (por ejemplo, las definiciones de relaciones) no pueden violar las declaraciones de dependencia. Las dependencias cíclicas no son admisibles. Sin una declaración adicional, un tema extendido presenta las mismas dependencias que su tema de base.

2.5.3 Las clases y las estructuras

Una definición de clase (palabra reservada CLASS) declara las propiedades de todos sus objetos pertinentes. Las definiciones de clases pueden extenderse, por lo que una extensión hereda todos los atributos, principalmente, de su clase base. Su dominio puede ser limitado y se pueden definir atributos adicionales.

El dominio de las identificaciones de objetos de todos los objetos de una clase se puede determinar de forma explícita (OID AS). Sin esta definición, se aplica la definición del tema, a menos que se haya indicado explícitamente que no se exigirán identificaciones de objeto (NO OID). Es imposible ampliar una definición OID existente, con la única excepción de sustituir una ANY heredada por una definición concreta. Sin embargo, esta ANY heredada no puede ser reemplazada por NO OID (véase el capítulo 2.8.9 Dominios de identificaciones de objetos).

Como parte de una definición de clase se pueden indicar constricciones. Estas condiciones representan reglas adicionales que todos los objetos tienen que cumplir. Las constricciones heredadas nunca pueden ser suspendidas, sino que siempre están en vigor, además de las declaradas localmente.

Los objetos de una clase son siempre independientes e identificables individualmente. Las estructuras (palabra clave STRUCTURE) formalmente se definen de la misma manera que las clases, sin embargo, sus elementos estructurales son dependientes y no pueden ser identificados individualmente. O bien se generan dentro de subestructuras de objetos (cf. capítulo 2.6 Atributos) o sólo existen temporalmente como resultado de funciones.

Las clases especiales como las de los sistemas de referencia, los ejes de coordenadas y los símbolos gráficos (en otras palabras, extensiones de la clase predefinida METAOBJECT) serán tratadas en el capítulo 2.10 .

Entre paréntesis (propiedades de regla) se pueden definir las características de la herencia. Todas las posibilidades son admisibles. Siempre que una clase o una estructura tengan atributos abstractos, tienen que ser declaradas como ABSTRACT. Los atributos abstractos deben poner en términos concretos dentro de la extensión concreta de la clase. Sin embargo, también es admisible declarar clases como abstractas a pesar de que sus atributos estén completamente definidos. Las instancias de objetos sólo pueden existir para las clases concretas que se han definido dentro de un tema. Las clases que se han definido fuera de

temas (es decir, directamente dentro del modelo) no pueden contener atributos de referencia. Tampoco es admisible definir asociaciones a dichas clases.

Si sólo se hereda una clase individual y no la totalidad de un tema (*topic*), no se pueden definir relaciones (ver capítulo 2.7: Relaciones verdaderas).

Si un *topic* extiende a otro, todas las clases del *topic* heredado se transfieren. De este modo se convierten en clases del *topic* actual y tienen el mismo nombre que en el *topic* heredado. Una clase de este tipo puede extenderse incluso manteniendo su nombre (EXTENDED). Por ejemplo, si en un *topic* T2 extiende el *topic* T1 que contiene la clase C, sólo hay una clase C (EXTENDED) dentro de T2, que es C. Las nuevas clases, que deben diferir en nombre de las heredadas, también pueden extender a clases heredadas. En consecuencia, con C2 EXTENDS C hay dos clases a partir de C en T2 (C y C2). Dado que INTERLIS, buscando la simplicidad y la claridad, sólo reconoce la herencia simple, EXTENDED es sólo admisible cuando ni en el tema base ni en el actual se ha extendido la clase base con EXTENDS. EXTENDED y EXTENDS se excluyen uno al otro en la misma definición de clase.

Las clases y estructuras que no se basan en una clase o estructura ya definida no necesitan EXTENDS.

Reglas de Sintaxis:

```
ClassDef = 'CLASS' Class-Name
             Properties<ABSTRACT,EXTENDED,FINAL>
               [ 'EXTENDS' ClassOrStructureRef ] '='
             [ ( 'OID' 'AS' OID-DomainRef | 'NO' 'OID' ) ';' ]
             ClassOrStructureDef
           'END' Class-Name ';'.

StructureDef = 'STRUCTURE' Structure-Name
                 Properties<ABSTRACT,EXTENDED,FINAL>
                   [ 'EXTENDS' StructureRef ] '='
                 ClassOrStructureDef
               'END' Structure-Name ';'.

ClassOrStructureDef = [ 'ATTRIBUTE' ] { AttributeDef }
                      { ConstraintDef }
                      [ 'PARAMETER' { ParameterDef } ].

ClassRef = [ Model-Name '.' [ Topic-Name '.' ] ] Class-Name.

StructureRef = [ Model-Name '.' [ Topic-Name '.' ] ] Structure-Name.

ClassOrStructureRef = ( ClassRef | StructureRef ).
```

Los nombres que deben calificarse (con Nombre-Modelo, Nombre-Modelo-Nombre-Tema) se explicarán al final del siguiente párrafo (véase el capítulo 2.5.4 Espacios de nombres).

2.5.4 Espacios de nombres

Con el término *"Espacio de nombres"* se entiende un conjunto de nombres (inequívocos). Cada elemento modelado (modelo de datos, *topic*, elemento de clase), así como todos los contenedores de metadatos proporcionan sus respectivos espacios de nombres para sus categorías de nombre (nombre de tipo, nombre de parte, nombre de metaobjeto).

Existen elementos de modelado en tres niveles jerárquicos:

- Modelo (MODEL es el único elemento de modelización en el nivel superior).
- Tema (TOPIC es un elemento de modelado único su este nivel).
- Elementos de clase como CLASS, STRUCTURE, ASSOCIATION, VIEW y GRAPHIC

Los nombres de metacontenedor de datos dan acceso a metaobjetos (véase el capítulo 2.10 Tratamiento de metaobjetos).

Hay tres categorías de nombres que contienen los siguientes nombres:

- Los nombres de tipo son las abreviaturas (nombres) de las unidades y los nombres de funciones, tipos de línea, dominios, estructuras, *topics*, clases, asociaciones, vistas, gráficos, contenedores.
- Nombre de partes son los nombres de los parámetros de tiempo de ejecución, los atributos, las reglas para dibujar, parámetros, roles y vistas básicas.
- Nombres Meta-objeto son los nombres de metaobjetos. Sólo existen dentro de contenedores de metadatos.

Cada vez que un elemento de modelado se extiende sobre otro, todos los nombres del elemento base del modelo serán añadidos a sus espacios de nombres. Con el fin de evitar malos entendidos, los elementos de modelado toman el nombre de los elementos de modelado superior en función de la categoría nombre. Un nombre definido localmente dentro del elemento de modelado, no puede colisionar con un nombre que se ha transmitido, a menos que específicamente se denomine como una extensión (EXTENDED).

Si desea referenciar elementos de descripción del modelo de datos, normalmente sus nombres deben ser cualificados, es decir, tienen que ser indicados con referencia al modelo anterior el nombre del *topic*. Siendo No cualificado es posible utilizar los nombres de los espacios de nombres de los respectivos elementos de modelado.

2.6 ATRIBUTOS

2.6.1 Observaciones generales relativas a los atributos

Un atributo es definido por su nombre y su tipo. Entre paréntesis (propiedades de la norma) pueden definirse las características de herencia. Cada vez que un atributo es una extensión de un atributo heredado, se deberá indicar de forma explícita con EXTENDED. Si el dominio de este atributo es abstracto, el atributo debe ser declarado como ABSTRACT. Un atributo numérico (véase el capítulo 2.8.5 Tipos de datos numéricos) se puede definir como una subdivisión (palabra clave SUBDIVISION) de su atributo predecesor también numérico (como, por ejemplo, minutos como una subdivisión de horas). Este atributo predecesor debe ser un número entero y el rango de valor de la subdivisión debe ser positiva. Si la subdivisión es continua (palabra clave CONTINUOS), entonces las diferencias de los límites del dominio deben estar en consonancia con el factor entre la unidad del atributo y la unidad del atributo precedente. Si un sistema de referencia ha sido definido con respecto a una subdivisión, este debe coincidir con el sistema de un atributo precedente directa o indirectamente. Sin embargo, ningún compilador de INTERLIS o sistema de tiempo de ejecución tendrán que comprobar este hecho.

Regla de sintaxis:

```
AttributeDef = [ [ 'CONTINUOUS' ] 'SUBDIVISION' ]
               Attribute-Name Properties<ABSTRACT,EXTENDED,FINAL,TRANSIENT>
                 ':' AttrTypeDef
                     [ ':=' Factor { ',' Factor } ] ';'.
```

Si el valor del atributo se ha determinado por medio de un factor (véase el capítulo 2.13 Expresiones), su tipo de resultado debe ser compatible con el atributo definido, es decir, debe consistir en ya sea el mismo dominio o uno extendido, es decir, un dominio especializado. Dentro del alcance de las vistas, sobre todo en el caso de las uniones y extensiones de vistas (véase capítulo 2.15: Vistas), es posible determinar varios factores y en las extensiones de vista adicionales se pueden añadir factores adicionales. Es el último factor (primero la base, luego la extensión) con un valor definido que es el válido. Los atributos, que han sido determinados por medio de un factor y que sólo son significativos dentro de otros factores, deben excluirse de cualquier transferencia de datos y deben ser señalados como transitorios.

En las extensiones es posible sobrescribir un atributo por los siguientes medios:

- Dominio limitado.
- Con una constante del dominio requerido. Tal definición es implícitamente final, es decir, ya no puede ser anulada.
- Un factor, si el tipo de resultado sería admisible como extensión de un atributo. Todavía podría ser anulada.

Reglas de Sintaxis:

```
AttrTypeDef = ( 'MANDATORY' [ AttrType ]
              | AttrType
              | ( ( 'BAG' | 'LIST' ) [ Cardinality ]
                  'OF' RestrictedStructureRef ) ).

AttrType = ( Type
           | DomainRef
           | ReferenceAttr
           | RestrictedStructureRef ).

ReferenceAttr = 'REFERENCE' 'TO'
                  Properties<EXTERNAL> RestrictedClassOrAssRef.

RestrictedClassOrAssRef = ( ClassOrAssociationRef | 'ANYCLASS' )
                            [ 'RESTRICTION' '(' ClassOrAssociationRef
                                { ';' ClassOrAssociationRef } ')' ].

ClassOrAssociationRef = ( ClassRef | AssociationRef ).

RestrictedStructureRef = ( StructureRef | 'ANYSTRUCTURE' )
                           [ 'RESTRICTION' '(' StructureRef
                               { ';' StructureRef } ')' ].

RestrictedClassOrStructureRef = ( ClassOrStructureRef | 'ANYSTRUCTURE' )
                                  [ 'RESTRICTION' '(' ClassOrStructureRef
                                    { ';' ClassOrStructureRef } ')' ].
```

Dentro del alcance de las extensiones es permisible indicar solo MANDATORY. En este caso, el tipo de atributo ya definido es válido. Sin embargo, se requiere estrictamente que se defina el valor.

2.6.2 Atributos con el dominio como tipo

Una definición de tipo directa (regla Type) y el uso de dominios ya definidos (regla DomainRef) son concebibles como tipos de atributo. Las diferentes posibilidades se enumeran en el capítulo 2.8 Dominios y constantes.

2.6.3 Atributos de referencia

Mediante el uso de un atributo de referencia, se crea una referencia a otro objeto. Los atributos de referencia sólo son admisibles dentro de las estructuras. Una estructura, que contiene directa o indirectamente (a través de subestructuras) atributos de referencia, no se puede extender a una clase. Los vínculos entre los objetos independientes tienen que ser definidos por medio de relaciones apropiadas (véase el capítulo 2.7: relaciones apropiadas).

Las clases, cuyos elementos están en consideración para la referencia, pueden ser clases de objetos concretos o abstractos o de relación, pero no estructuras (ya que son objetos dependientes). Todas las clases concretas son candidatas para ello, siempre que correspondan a la clase principal que aparece, o a una de las clases restringidas (RESTRICTION TO) (la clase misma o una de sus subclases). En todos los niveles de restricción (definición inicial o pasos para la extensión) deben ser enumeradas todas las clases que aún son admisibles. Cada clase definida como una restricción, debe ser una subclase de una

clase hasta este momento admisible. Sin embargo, una clase de este tipo sólo es admisible si pertenece al mismo *topic* que el atributo de referencia o a un *topic*, del cual depende, a su vez, el referenciado (DEPENDS ON). Si se requiere que la referencia pueda apuntar al objeto de un contenedor diferente del mismo o de un tema diferente (prerrequisito: DEPENDS ON), se debe indicar la propiedad EXTERNAL. En las extensiones es posible omitir y por lo tanto excluir esta propiedad, sin embargo, no se puede añadir. A menos que el atributo de referencia haya sido declarado abstracto, debe existir como mínimo una subclase concreta admisible.

2.6.4 Atributos de estructura

Los valores de los atributos de estructura se componen de uno o varios (número admisible en función de la cardinalidad indicada) elementos estructurales ordenados (LIST) o desordenados (BAG), (ni LIST ni BAG son requisitos). Los elementos estructurales no tienen OID, sólo existen en relación con su objeto principal y sólo se pueden encontrar mediante este objeto principal. Su composición es consecuencia de la estructura indicada (regla RestrictedStructureRef).

Los atributos de estructura se pueden definir en estructuras concretas o abstractas. En principio, todas las estructuras (pero no todas las clases) son adecuados para elementos de estructura concreta, siempre que correspondan o amplíen las estructuras que figuran como primarias o restrictivas (RESTRICTION TO). No se necesita información adicional para las estructuras del *topic* en uso. Las estructuras de todos los demás temas sólo se tendrán en cuenta si una clase básica también se define dentro de otro *topic*, y se enumeran explícitamente en la definición del atributo de estructura como estructura primaria o restricción. En cada nivel de restricción (definición primaria o pasos de ampliación) todas las estructuras admisibles deberán ser enumeradas. Cada estructura definida como una extensión debe ser una extensión de una estructura admisible hasta este momento.

Si la estructura de un atributo de estructura es arbitraria (ANYSTRUCTURE) o si no se puede encontrar ninguna estructura que cumpla con la definición, entonces el atributo de estructura debe ser declarado como abstracto, siempre que sea obligatorio o si su cardinalidad mínima es mayor que cero. Si las estructuras se definen como argumentos formales de una función (ver capítulo 2.14: Funciones), las rutas para estructurar elementos u objetos son válidos como argumentos actuales. Por otra parte, a través de ANYSTRUCTURE todos los elementos de la estructura y todos los objetos son compatibles.

Una subestructura ordenada (LIST) no podrá extenderse mediante una subestructura desordenada (BAG). Las mismas reglas que se aplican a las relaciones se aplican a la cardinalidad (véase el capítulo 2.7.3: cardinalidad).

2.7 Relaciones propiamente dichas o reales

2.7.1 Descripción de las relaciones

Las relaciones propiamente dichas (por oposición a los atributos de referencia, véase capítulo 2.6: Atributos) se describen como construcciones independientes. Sin embargo, en gran medida tienen las mismas propiedades que las clases. Por ejemplo, pueden contener atributos locales, así como restricciones de consistencia. El nombre de la asociación puede ser omitido. En este caso se compone implícitamente de los nombres de los roles (empezando por el primero, luego el segundo, etc.). Sin embargo, la propiedad más importante de una relación consiste en la enumeración de al menos dos roles con sus clases o relaciones asignadas (las mismas reglas que se aplican a los atributos de referencia, (ver el capítulo 2.6.3. Atributos de referencia) y detalles como la fuerza de la relación y cardinalidad. Los nombres de los roles deben ser típicamente sustantivos. Pueden, pero no tienen porqué coincidir con los nombres de las clases o relaciones asignadas. Sin embargo, la relación por definir puede no ser una extensión de una relación asignada. También es posible asignar alternativamente diferentes clases o relaciones a un papel (rol). Dicha clase alternativa o relación, no puede ser una extensión de otra clase o relación alternativa con el mismo papel.

Ejemplo de una relación entre la clase K, por un lado, y de las clases K o L por el otro:

```
ASSOCIATION A =
  K -- K;
  KL -- K OR L;
END A;
```

En principio, las relaciones pueden extenderse de la misma manera que las clases. Con el fin de asegurar que la significación de la relación sea clara y constante, no pueden aparecer papeles o funciones (roles) adicionales en las extensiones. Sin embargo, es posible restringir las clases o relaciones asignadas, así como la cardinalidad. Los roles sin cambios no deben ser listados.

Ejemplo de cómo se puede especializar la relación A con A1, cuando sólo por una parte se hace referencia a K1 (una subclase de K) y a K, L1, L2 (subclases de L), por otro:

```
ASSOCIATION A1 EXTENDS A =
  K (EXTENDED) -- K1;
  KL (EXTENDED) -- K OR L RESTRICTION (L1, L2);
END A1;
```

Por lo tanto, un objeto de la clase K1 puede estar relacionado a través de una relación A con objetos de las clases K o L (admisible, ya que K1 es una subclase de K) o por medio de una relación A1 con objetos de las clases K, L1 o L2. Si, además, quisiéramos asegurar que un objeto K1 dentro del rol K sólo pueda tener una relación especializada A1 (en contraposición a la relación general A), entonces el rol K tiene que ser señalado como oculto (HIDING).

```
ASSOCIATION A1 EXTENDS A =
  K (EXTENDED, HIDING) -- K1;
  KL (EXTENDED) -- K OR L RESTRICTION (L1, L2);
END A1;
```

Sin embargo, esto sólo es admisible si para K1 no se han definido otras relaciones extendidas de A, en las cuales el papel K no se ha definido como escondido.

Reglas de Sintaxis:

```
AssociationDef = 'ASSOCIATION' [ Association-Name ]
                   Properties<ABSTRACT,EXTENDED,FINAL,OID>
                   [ 'EXTENDS' AssociationRef ]
                   [ 'DERIVED' 'FROM' RenamedViewableRef ] '='
                   [ ( 'OID' 'AS' OID-DomainRef | 'NO' 'OID' ) ';' ]
                   { RoleDef }
                   [ 'ATTRIBUTE' ] { AttributeDef }
                   [ 'CARDINALITY' '=' Cardinality ';' ]
                   { ConstraintDef }
                 'END' [ Association-Name ] ';'.

AssociationRef = [ Model-Name '.' [ Topic-Name '.' ] ] Association-Name.

RoleDef = Role-Name Properties<ABSTRACT,EXTENDED,FINAL,HIDING,
                               ORDERED,EXTERNAL>
            ( '--' | '-<>' | '-<#>' ) [ Cardinality ]
            RestrictedClassOrAssRef { 'OR' RestrictedClassOrAssRef }
            [ ':=' Role-Factor ] ';'.

Cardinality = '{' ( '*' | PosNumber [ '..' ( PosNumber | '*' ) ] ) '}'.
```

La referencia de asociación entre objetos puede considerarse como un objeto independiente (referencia de asociación). En primer lugar, todos los roles para las referencias a objetos de referencia se indicarán en esta referencia de asociación (¡tienen que definirse todos!). Sin más información, se identificará esta instancia de asociación a través de las referencias a los objetos conectados por la referencia de asociación.

Entre estos objetos, sólo será admisible una referencia de asociación. Entre la misma combinación de objetos se admiten múltiples instancias de asociación sólo si, para la relación, se requiere explícitamente una cardinalidad con un límite superior mayor a uno (CARDINALITY). En este caso también se requiere una identificación de objeto (por medio de la propiedad OID). Si se requiere una identificación de objeto propia, entonces la relación misma puede ser utilizada como referencia de roles.

Si se busca la compatibilidad con INTERLIS 1, sólo se definen relaciones con dos roles, sin exceder la cardinalidad máxima de un rol mayor que.

Normalmente las relaciones concretas entre los objetos se deben establecer explícitamente por medio de una aplicación y posteriormente fijarse como una instancia en el sistema de procesamiento. Una relación, sin embargo, también puede derivar de una vista sin ser instanciado (DERIVED FROM). Tal relación puede ser una extensión de una relación abstracta, pero no puede ser abstracta en sí misma. Si se extiende, la extensión debe basarse en la misma vista o en una de sus extensiones. A todos los roles y atributos se debe asignar una correspondiente ruta de objeto o de atributo de la vista. Debe indicarse un camino de objeto (véase el capítulo 2.13) que designa una clase o asociación correspondiente al rol. La cardinalidad debe estar acuerdo con el resultado de la vista. Sin embargo, esto sólo puede comprobarse en tiempo de ejecución.

Un caso típico de una aplicación podría ser la derivación de una relación a partir de sus condiciones geométricas: dentro de una vista (unión) a la que se hace referencia en la asociación y basada en la geometría de la propiedad inmobiliaria donde estén ubicados, por ejemplo, los edificios se traen en relación a estos predios (véase capítulo 2.15: Vistas).

2.7.2 La fuerza de la relación

De acuerdo con UML se distingue diferentes fuerzas de relación. Para explicarlas, se describe principalmente la influencia que la fuerza de la relación posee al copiar o borrar objetos. Para INTERLIS 2 sin embargo, sólo el borrado de objetos es de importancia (debido a la actualización incremental). Además, hay otras consideraciones que influyen en la fuerza de la relación. Sobre todo, depende del software establecer fuerzas de relación más refinadas o, incluso, otros criterios para el tratamiento en determinadas operaciones.

- Asociación: los objetos en cuestión ligeramente conectados. Si uno de los objetos se copia, la copia está conectada a los mismos objetos que el original. Si se elimina un objeto, la relación se elimina al mismo tiempo, sin embargo, el objeto en sí mismo permanece. Sintácticamente se indica '--' con todos los roles.
- Agregación: existe una relación débil entre la totalidad y sus partes. Las agregaciones sólo están permitidas en las relaciones con dos roles. Sintácticamente el rol o función que lleva a la totalidad se debe indicar con un rombo (- <>). El rol que lleva a la parte se define con "--". Una clase de objeto puede aparecer en varias agregaciones en el rol de una parte. Por lo tanto, a un cierto objeto de una parte se le pueden asignar diferentes objetos de la totalidad. A diferencia de las asociaciones, al hacer una copia de la totalidad se establecen copias de las partes correspondientes. Ya que dentro del ámbito de INTERLIS 2 la copia de objetos no es importante, INTERLIS 2 trata las agregaciones como asociaciones.
- Composición: Existe una fuerte relación entre la totalidad y sus partes. Una clase de objeto puede aparecer en más de una composición, en el papel o rol parcial. sólo se puede asignar una totalidad a un determinado objeto parcial. Al borrar la totalidad también se eliminan sus partes. El rol que lleva a la totalidad se indica con un rombo lleno (- <#>).

Las asociaciones pueden ser extendidas a agregaciones, y éstas pueden extenderse a composiciones; lo contrario no es admisible.

2.7.3 Cardinalidad

La cardinalidad define el número mínimo y máximo de objetos permitidos. Donde sólo se indica un valor, el mínimo y el máximo son iguales. Si para el máximo un asterisco sustituye al número, no hay límite superior para el número de subobjetos. La indicación de cardinalidad con {*} es el equivalente a {0..*}. Si no hay ninguna indicación de cardinalidad entonces se aplica {0..*}. Con roles de composición solo es admisible {0..1} ó {1} (una parte sólo puede pertenecer a una totalidad). Donde no hay ninguna indicación, se aplica {0..1}.

En las extensiones, la cardinalidad sólo puede ser restringida, pero no extendida. Por lo tanto, si en el primer lugar se ha indicado una cardinalidad {2..4}, una extensión no puede declarar {2..5}, {7} o {*}. Si en los atributos extendidos se omite la indicación de cardinalidad, se asume que se aplica el valor heredado.

Dependiendo de la forma de aplicación, la cardinalidad tiene el siguiente significado:

- Con subestructuras: número de elementos admisibles.
- Con roles de relaciones: Número de objetos correspondientes a un rol que, a través de la relación, pueden ser asignados a una combinación arbitraria de los objetos que corresponden a los otros papeles.
- Con la relación en su conjunto: Número de referencias de asociación para una combinación arbitraria de objetos, de acuerdo con todas las funciones (roles) de la relación.

2.7.4 Relaciones ordenadas

Si se quiere lograr que una relación, desde el punto de vista de una cierta clase de referencia, se establezca en un orden determinado, esta propiedad (ORDERED) tiene que ser requerida dentro del rol. Esta orden se define a la hora de establecer la relación y tiene que ser mantenida en todas las transferencias.

2.7.5 Acceso de la relación

El acceso de la relación (AssociationAccess) significa la posibilidad, desde el punto de vista de un objeto, de navegar a las referencias de la asociación y más adelante sobre los objetos de referencia de acuerdo con una relación. Los Accesos de la relación no necesitan ser definidos; se originan en la definición de una relación para todas las clases asignadas a través de roles, que se han definido en el mismo tema que la relación. Si una clase involucrada en una relación ha sido definida en un tema diferente (relación de *topic*), o si debería permitirse que un objeto de referencia correspondiente a esta función pueda encontrarse en un contenedor distinto al de la referencia de asociación, debe especificarse el hecho específicamente con el rol (EXTERNAL, véase el capítulo 2.7.1 Descripción de las relaciones y el capítulo 2.6.3 Atributos de referencia). Entonces esta clase no será suministrada con accesos de relación. Esta propiedad se definirá dentro de la definición básica de una relación y no puede ser modificada dentro de una extensión. Si una función (rol) se refiere a una clase heredada del tema heredado, los accesos de la relación de esta clase sólo se permiten si esta clase ha sido extendida dentro del tema actual del mismo nombre (EXTENDED). Esta restricción impide que se modifique posteriormente (es decir, fuera del alcance dentro del cual se había definido originalmente).

El acceso de relación se transmitirá a las subclases, a menos que esto se descarte para un rol de una relación extendida por medio del requerimiento (HIDING).

Los accesos de relaciones son de importancia para todas las descripciones de la ruta (capítulo 2.13: Expresiones).

2.8 Dominios y constantes

Existen diferentes aspectos que están relacionados con la idea de un dominio. Principalmente un tipo de datos tiene que ser definido. Los Tipos de datos de INTERLIS son independientes de su implementación. Es por eso que no se habla de entero o real, por ejemplo, sino simplemente de tipos de datos numéricos (capítulo 2.8.5 Tipos de datos numéricos).

Una vez que se ha determinado el tipo de datos, pueden ser necesarias o posibles más especificaciones, dependiendo del tipo de datos en cuestión. Si una definición de dominio está incompleta (por ejemplo, la longitud de un dominio de cadena de texto), tiene que ser declararse como abstracta (palabra clave ABSTRACT, regla *Properties*).

Los dominios pueden ser, como otras construcciones, heredado y, posteriormente ampliados, siempre y cuando no se hayan definido mediante FINAL. En tal caso, el principio básico de que las definiciones ampliadas deben ser siempre compatibles con su definición básica es de gran importancia. Así, en los dominios, las extensiones (palabra clave EXTENDS) son realmente especificaciones, o restricciones. La palabra clave EXTENDED (regla *Properties*) no es admisible. En interés de la legibilidad, se sugiere que las partes de las definiciones en los dominios de base (tales como unidades de medida) se repitan en la extensión, incluso si no se han modificado. Ejemplo:

```
DOMINIO
  Text (ABSTRACT) = TEXT;              !! abstract domain
  GenName EXTENDS Text = TEXT*12;      !! concrete extension
  SpezName EXTENDS GenName = TEXT*10; !! ok
  SpezName EXTENDS GenName = TEXT*14; !! false, since incompatible
```

Una cuestión importante en la definición de dominios es si el valor "indefinido" pertenece al dominio o no. Sin ninguna indicación específica forma parte, sin embargo, es posible pedir su exclusión, es decir, siempre debe definirse un atributo con este dominio (palabra clave MANDATORY). MANDATORY es admisible sólo en las extensiones.

Al definir el atributo de una clase o estructura (y sólo entonces), MANDATORY puede darse si el dominio ha sido declarado FINAL y consecuentemente, no debe ser restringirse más.

Reglas de Sintaxis:

```
DomainDef = 'DOMAIN'
               { Domain-Name Properties<ABSTRACT,FINAL>
                   [ 'EXTENDS' DomainRef ] '='
               ( 'MANDATORY' [ Type ] | Type ) ';' }.

Type = ( BaseType | LineType ).

DomainRef = [ Model-Name '.' [ Topic-Name '.' ] ] Domain-Name.

BaseType = ( TextType
           | EnumerationType
           | EnumTreeValueType
           | AlignmentType
           | BooleanType
           | NumericType
           | FormattedType
           | CoordinateType
           | OIDType
           | BlackboxType
           | ClassType
           | AttributePathType ).
```

En las operaciones de comparación (véase el capítulo 2.13: Expresiones), los valores de atributos también pueden compararse con constantes. Estos se definen como sigue:

```
Constant = ( 'UNDEFINED'
           | NumericConst
           | TextConst
           | FormattedConst
           | EnumerationConst
           | ClassConst
```

```
              | AttributePathConst ).
```

Cada tipo de datos define sus propias constantes específicas.

2.8.1 Cadenas de texto

El término, cadena de texto (*string*) representa a una serie de símbolos de longitud máxima. Todos los símbolos suministrados deben estar claramente definidos (véase el apéndice B- Tabla de símbolos).

Dentro del tipo de datos MTEXT se encuentran los símbolos "retorno de carro" (#xD), “salto de línea” (#xA) y 'tabulador' (#x9), en comparación con el dominio del tipo TEXT.

Con los tipos de datos cadena de texto (TEXT), es sobre todo la longitud de la cadena lo que interesa. Dependiendo de la forma de definición, se indica de manera explícita o implícita. En su forma explícita (TEXT* ...), la longitud máxima es determinada por el número de símbolos (mayor a 0). Si se indican sólo las palabras claves (TEXT o MTEXT), el número de símbolos es ilimitado. En el ámbito de una extensión, su longitud se puede acortar, pero el alargamiento conduciría a un dominio que no sería compatible con el dominio básico.

Una longitud de cadena en INTERLIS muestra el número de símbolos que percibe el usuario, pero no el número máximo de espacios de almacenamiento de memoria que un sistema necesita para la representación de dicha cadena. Una cadena cuya longitud sea cero se considera indefinida.

Nota: En relación con INTERLIS, la longitud de una cadena se define como el número de símbolos que, de acuerdo con el estándar Unicode, posee la combinación canónica n° 0, después de que la cadena haya sido puesta en su forma canónica descompuesta de Unicode (ver www.unicode/org./unicode/reports/tr15/). Así, una cadena consiste en <LATIN CAPITAL LETTER C WITH CIRCUMFLEX><COMBINING CEDILLA> posee la longitud 1, la misma que la cadena equivalente <LATIN CAPITAL LETTER C> COMBINING CIRCUMFLEX ACCENT><COMBINING CEDILLA>. De acuerdo con la definición anterior, las ligaduras para "fl" o "ffi" poseen la longitud 1. Sin embargo, se recomienda no usar dichas formas de representación para los atributos de cadena.

El tipo de cadena de nombres (palabra clave NAME) define un dominio que corresponde exactamente al de los nombres de INTERLIS (ver capítulo 2.2.2: Nombres). Se aplica en la clase predefinida METAOBJECT y, sobre todo, en las clases de sistemas de referencia, así como los símbolos (capítulo 2.10.3: Sistemas de referencia y capítulo 2.16: Descripciones gráficas), porque es allí donde los atributos de datos tienen que coincidir con la descripción de los elementos de los modelos.

URI (*Uniform Resource Identifier)*; Identificador uniforme de recursos) representa un tipo de cadena más, como son las direcciones http, ftp y mailto (véase el apartado 1.2 de la norma internet IETF RFC 2396 en www.w3.org). La longitud de un URI en INTERLIS se limita a 1023 caracteres y por lo tanto corresponde a la siguiente definición:

```
DOMINIO
  URI (FINAL) = TEXT*1023;  !! ATTENTION: according to IETF RFC 2396
  NAME (FINAL) = TEXT*255;  !! ATTENTION: according to chapter 2.2.2 Names
```

Reglas de Sintaxis:

```
TextType = ( 'MTEXT' [ '*' MaxLength-PosNumber ]
           | 'TEXT' [ '*' MaxLength-PosNumber ]
           | 'NAME'
           | 'URI' ).

TextConst = String.
```

2.8.2 Enumeraciones

Por medio de una enumeración, se determinan todos los valores admisibles para un tipo. Sin embargo, una enumeración no es simplemente lineal, sino que cuenta con una estructura en forma de árbol. Las ramas de este árbol (pero no sus nodos) forman el conjunto de valores admisibles. Ejemplo:

```
DOMAIN
  Colors = (red (dark_red, orange, crimson),
            yellow
            green (light_green, dark_green));
```

Produce los siguientes valores admisibles, descrito por medio de constantes:

```
#red.dark_red #red.orange #red.crimson #yellow #green.light_green
#green.dark_green
```

Un nivel de subdivisión se indica entre paréntesis (). Los nombres de los elementos de cada nivel de subdivisión deben ser inequívocos. Hay libertad para elegir el nivel de profundidad del árbol.

En una enumeración ordenada (palabra clave ORDERED), la secuencia de elementos está claramente definida. Si la enumeración es circular (palabra clave CIRCULAR), la secuencia de los elementos se define como si la enumeración se hubiese ordenado. Además, se indica que el último elemento de nuevo será seguido por el primero.

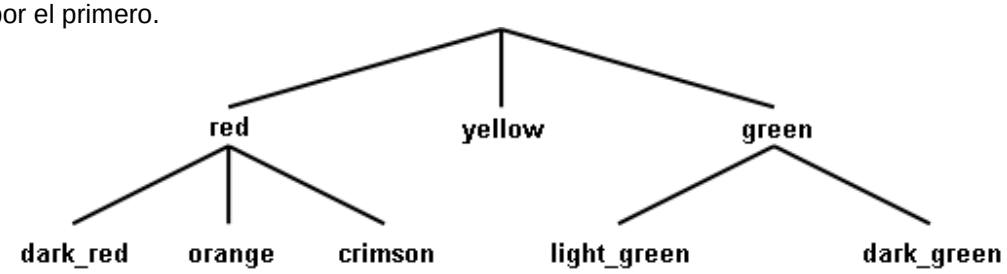

Figure 8: Ejemplo de una enumeración.

Además de la definición de enumeración en su sentido más estricto, también es posible definir un dominio que comprenderá, como valores admisibles, todas las hojas y nodos de una definición de la enumeración (ALL). Por lo tanto, es posible asignar a un atributo el valor de un atributo de su definición fundamental de enumeración.

Ejemplos

```
DOMINIO
  Position = (bottom, center, top) ORDERED;
  Weekdays = (Working days (Monday, Tuesday, Wednesday,
                            Thursday, Friday, Saturday),
              Sunday) CIRCULAR;
  ValuesOfWeekdays = = ALL OF Weekdays;
```

Reglas de Sintaxis:

```
EnumerationType = Enumeration [ 'ORDERED' | 'CIRCULAR' ].

EnumTreeValueType = 'ALL' 'OF' Enumeration-DomainRef.

Enumeration = '(' EnumElement { ',' EnumElement } [ ':' 'FINAL' ]
               | 'FINAL' ')'.

EnumElement = EnumElement-Name { '.' EnumElement-Name } [Sub-Enumeration].

EnumerationConst = '#' ( EnumElement-Name { '.' EnumElement-Name }
                                          [ '.' 'OTHERS' ]
                       | 'OTHERS' ).
```

Dentro del ámbito de las nuevas definiciones de enumeraciones (definiciones primarias, y elementos complementarios de extensiones), el EnumElement solo puede consistir en un nombre. Varios nombres sólo son admisibles con el fin de identificar un elemento de la enumeración que prevalece para una extensión.

Por otro lado, las enumeraciones se pueden extender mediante la definición de subentradas para las hojas de enumeraciones existentes hasta ese momento (en otras palabras, los elementos de enumeración que no cuentan con subenumeraciones). Al extender la definición, las hojas anteriores se convierten en nudos para los que no se pueden definir ningún valor.

Por otra parte, cada segmento de la enumeración individual en las extensiones se puede complementar con otros elementos (nudos u hojas). Así, la enumeración básica comprende, además de los elementos mencionados, más elementos potenciales que sólo se definirán en las extensiones. A nivel básico, tales valores posibles pueden ser abordados a través de OTHERS en las expresiones, los argumentos de las funciones y asignaciones de símbolos (véase el capítulo 2.13 expresiones, capítulo 2.14 Funciones y capítulo 2.16 Descripciones gráficas). Sin embargo, OTHERS no es ningún valor admisible dentro del ámbito de la clase al que pertenece el objeto. La posibilidad de añadir elementos suplementarios de la enumeración de las extensiones puede ser restringida declarando la enumeración parcial como final (FINAL). Esto se hace después del último de los elementos enumerados dentro del alcance de una extensión, incluso sin agregar nuevos elementos.

Las enumeraciones circulares (palabra clave CIRCULAR) no pueden extenderse.

```
DOMINIO
  Color = (red,
           yellow
           green);
  ColorPlus EXTENDS Color = (red (dark_red, orange, crimson),
                             green (light_green, dark_green: FINAL.
                             Blue);
  ColorPlusPlus EXTENDS ColorPlus = (red (FINAL),
                                     blue (light_blue, dark_blue));
```

Para Color Plus esto se traduce en los siguientes valores admisibles, descritos a través de constantes:

```
#red.dark_red #red.orange #red.crimson #yellow #green.light_green #green.dark_green
#blue
```

y para ColorPlusPlus:

```
#blue.light_blue #blue.dark_blue instead of #blue
```

Al indicar FINAL con los tonos de verde en ColorPlus no es admisible definir otros tonos de verde en ColorPlusPlus. FINAL en la subdivisión de color rojo en ColorPlusPlus evita la adición de más variedades de color rojo en las posibles ampliaciones de ColourPlusPlus.

2.8.3 Orientación del texto

Para el procesamiento de planos y mapas, deben determinarse las posiciones del texto. Debe indicarse a qué punto del texto corresponde la posición. La alineación horizontal determina si la posición está situada en el margen izquierdo, en el derecho o en el centro del texto. La alineación vertical determina la posición en la dirección del tamaño del texto.

La distancia entre la parte de arriba y la base corresponde con el tamaño de las letras mayúsculas. Las astas descendentes se colocan en el área inferior de la base.

Se puede entender que la alineación horizontal y vertical significa las siguientes enumeraciones predefinidas:

```
DOMINIO
```

```
  HALIGNMENT (FINAL) = (Left, Center, Right) ORDERED;
  VALIGNMENT (FINAL) = (Top, Cap, Half, Base, Bottom) ORDERED;
```

Regla de sintaxis:

```
AlignmentType = ( 'HALIGNMENT' | 'VALIGNMENT' ).
```

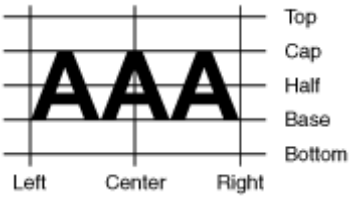

Figura 9: Orientación del texto horizontal (HALIGNMENT) y verticalmente (VALIGNMENT).

2.8.4 BOOLEAN

El tipo booleano cuenta los valores verdadero y falso. Se puede interpretar con la siguiente enumeración predefinida:

```
DOMINIO
  BOOLEAN (FINAL) = (false, true) ORDERED;
```

Regla de sintaxis:

```
BooleanType = 'BOOLEAN'.
```

2.8.5 Tipos de datos numéricos

La información más importante en relación con los tipos de datos numéricos son el valor mínimo y el máximo, incluyendo el número de decimales, así como el factor de escala. Además, se puede indicar que el tipo es circular (palabra clave CIRCULAR), es decir, que en el último dígito significativo el valor máximo incrementado en 1 y el valor mínimo técnicamente tiene el mismo significado (por ejemplo, con ángulos de 0 .. 359 grados). Si el atributo se ha definido como una subdivisión continua del atributo predecesor (ver el capítulo 2.6.1: Observaciones generales relativas a los atributos), el tipo tiene que ser definido como circular. Si la indicación del valor mínimo y el máximo debe faltar (palabra clave NUMERIC), el dominio es considerado como abstracto.

```
DOMINIO
  Angle1 = 0.00 .. 359.99 CIRCULAR [degree];  !! correct
  Angle2 = 0.00 .. 360.00 CIRCULAR [degree];  !! syntactically correct, but
                                              !! technically false, since 360.01
                                              !! subsequently corresponds to the
                                              !!  valor mínimo 0.00
```

El número de dígitos debe coincidir en el valor mínimo y el máximo. Por medio del escalado, es posible describir números flotantes, pero entonces el valor mínimo, así como el valor máximo tienen que estar indicados en la representación numérica del tipo de mantisa, es decir, partiendo de cero (0) y seguido por un punto decimal (.) El primer dígito después del punto decimal debe ser diferente de cero (0). La escala del valor mínimo debe ser inferior a la escala del valor máximo. Sin embargo, la notación de valor mínimo y máximo de ninguna manera califica como una indicación sobre cómo transferir estos valores (si el dominio se define con 000 .. 999 esto no quiere decir que el valor 7 se transfiera como 007). Los números Flotantes son la excepción a esta regla. Éstos se van a transferir en la representación numérica del tipo mantisa y con escala.

Con las extensiones de los mínimos, los valores máximos sólo pueden ser restringidos. Así, el rango numérico se hace más pequeño. Observe la siguiente situación:

```
DOMINIO
  Normal = 0.00 .. 7.99
  Exact EXTENDS Normal = 0.0000 .. 7.9949;    !! correcto, ya que es Normal
                                              !! también es representable
  Exact EXTENDS Normal = 0.0000 .. 7.9999;    !! falso, dado que redondea
                                              !! fuera de lo definido como Normal
```

Con el fin de explicar claramente el significado de un valor, se puede indicar una unidad de medida (ver el capítulo: 2.9 unidades). Las unidades de medida abstracta sólo son admisibles siempre y cuando el propio dominio aún no esté definido (palabra clave NUMERIC).

Las siguientes reglas se aplican a las extensiones:

- Si un dominio de base concreto no cuenta con ninguna unidad de medida, entonces ninguna puede aparecer en extensiones.
- Cuando el dominio de base emplea una unidad de medida abstracta, únicamente las unidades de medida pueden ser utilizadas en las extensiones, así como las extensiones de unidad de medida antes mencionada.
- Cuando el dominio base emplea unidades de medidas concretas, no se pueden sustituir en las extensiones.

Ejemplos:

```
UNIT
  foot [ft] = 0.3048 [m];

DOMAIN
  Distance (ABSTRACT) = NUMERIC [Length];
  MeterDist (ABSTRACT) EXTENDS Distance = NUMERIC [m];
  FootDist (ABSTRACT) EXTENDS Distance = NUMERIC [ft];
  ShortMeters EXTENDS MeterDist = 0.00 .. 100.00 [m],
  ShortFeet EXTENDS FussDist = 0.00 .. 100.00 [ft];
  ShortFeet2 (ABSTRACT) EXTENDS ShortMeters = NUMERIC [ft]; !! false: m vs. ft
```

Un sistema escalar (ver el capítulo 2.10.3: Sistemas de referencia) puede ser atribuido a un dominio numérico. A partir de entonces los valores se refieren al punto cero determinado por el sistema escalar. En consecuencia, son valores absolutos en este sistema escalar. Si en la clase del sistema escalar la unidad no es ANYUNIT, una unidad de este tipo tiene que ser indicada con el tipo de datos numérico será compatible con el sistema de referencia. Con respecto a un sistema de referencia puede indicar el eje de referencia de los valores. La unidad indicada debe ser compatible a la unidad del eje correspondiente. Cuando se omite esta información, no se especifica la referencia, si no que procede del sujeto (por ejemplo, si se refiere a una elipse al indicar una altura, se entiende como alturas elípticas). Si se refiere a un dominio diferente, el mismo sistema de referencia debe ser aplicable al igual que en la primera. En este caso, la indicación del eje sólo se podrá omitir cuando se refiera a dominios numéricos. Con un dominio de coordenadas, la indicación del eje es obligatoria. La indicación de los sistemas de referencia no se puede cambiar en las extensiones.

Si el valor numérico representa un ángulo, se puede determinar su orientación. En el caso de direcciones se puede especificar a qué sistema de coordenadas se refieren (definido por su dominio de coordenadas). De este modo se conocen tanto en la dirección cero (azimut) como el sentido de rotación (véase el capítulo 2.8.8: Coordenadas). Esta indicación no se puede cambiar en las extensiones.

Además de los números decimales, dos números más se definen como constantes numéricas: pi (palabra clave PI) y e, base del logaritmo natural (palabra clave LNBASE).

Reglas de Sintaxis:

```
NumericType = ( Min-Dec '..' Max-Dec | 'NUMERIC' ) [ 'CIRCULAR' ]
```

```
                [ '[' UnitRef ']' ]
                [ 'CLOCKWISE' | 'COUNTERCLOCKWISE' | RefSys ].

  RefSys = ( '{' RefSys-MetaObjectRef [ '[' Axis-PosNumber ']' ] '}'
           | '<' Coord-DomainRef [ '[' Axis-PosNumber ']' ] '>' ).

  DecConst = ( Dec | 'PI' | 'LNBASE' ).

  NumericConst = DecConst [ '[' UnitRef ']' ].
```

2.8.6 Dominios con formato

Los dominios con formato se basan en estructuras y utilizan sus atributos numéricos y definidos en un formato. Por un lado, este formato sirve para el intercambio de datos (véase el capítulo 3.3.11.5: Codificación de dominios con formato). Por otro lado, para la definición de los límites inferior y superior del dominio.

Reglas de Sintaxis:

```
  FormattedType = [ 'FORMAT' 'BASED' 'ON' StructureRef FormatDef ]
                  [ Min-String '..' Max-String ]
                  | 'FORMAT' FormattedType-DomainRef
                    Min-String '..' Max-String.

  FormatDef = '(' [ 'INHERITANCE' ]
                    [ NonNum-String ] { BaseAttrRef NonNum-String }
                                        BaseAttrRef [ NonNum-String ] ')'.

  BaseAttrRef = ( NumericAttribute-Name [ '/' IntPos-PosNumber ]
                | StructureAttribute-Name '/' Formatted-DomainRef ).

  FormattedConst = String.
```

Principalmente, una definición básica de un dominio con formato, define la estructura sobre la que se construye y el formato que está siendo utilizado. Además, es posible definir tanto el límite inferior como el superior del dominio. No se pueden extender los límites definidos por la estructura.

Dentro del alcance de una extensión, es posible hacer referencia a una extensión de la estructura original, para complementar el formato (la parte heredada debe figurar al principio y, buscando claridad, debe ser señalizada con la palabra clave INHERITANCE) y restringir el dominio.

Por un lado, la definición de formato puede contener cadenas constantes, que no comienzan con un número (al principio, al final o en medio de las referencias de atributos individuales), por otro lado, puede contener, directa o indirectamente (a través de los atributos de la estructura) referencias de atributos. La referencia de atributo debe designar un atributo numérico o un atributo de estructura. En el caso de un atributo numérico es posible definir la cantidad de dígitos a la izquierda del punto decimal. Como resultado de ello se introducirán ceros delante si es necesario. El número de decimales será el resultado del dominio numérico. Con atributos de estructura, se debe definir con qué dominio de formato tiene que ser formateada. La estructura debe concordar con la estructura básica del dominio o ser una extensión de la mismo.

2.8.7 Fecha y hora

Cuando las indicaciones de fecha o hora no consisten sólo en un valor (por ejemplo año, segundo), se tiende a utilizar dominios con formato.

Buscando la compatibilidad con XML, los elementos correspondientes son predefinidos por INTERLIS:

```
UNIT
  Minute [min] = 60 [INTERLIS.s];
```

```
  Hour   [h]   = 60 [min];
  Day    [d]   = 24 [h];
  Month [M] EXTENDS INTERLIS.TIME;
  Year [Y] EXTENDS INTERLIS.TIME;

REFSYSTEM BASKET BaseTimeSystems ~ TIMESYSTEMS
  OBJECTS OF CALENDAR: GregorianCalendar
  OBJECTS OF TIMEOFDAYSYS: UTC;

STRUCTURE TimeOfDay (ABSTRACT) =
  Hours: 0 .. 23 CIRCULAR [h];
  CONTINUOUS SUBDIVISION Minutes: 0 .. 59 CIRCULAR [min];
  CONTINUOUS SUBDIVISION Seconds: 0,0000.000 .. 59.999 CIRCULAR [INTERLIS.s];
END TimeOfDay;

STRUCTURE UTC EXTENDS TimeOfDay =
  Hours(EXTENDED): 0 .. 23 {UTC};
END UTC;

DOMAIN
  GregorianYear = 1582 .. 2999 [Y] {GregorianCalendar};

STRUCTURE GregorianDate =
  Year: GregorianYear;
  SUBDIVISION Month: 1 .. 12 [M];
  SUBDIVISION Day: 1 .. 31 [d];
END GregorianDate;

STRUCTURE GregorianDateTime EXTENDS GregorianDate =
  SUBDIVISION Hours: 0 .. 23 CIRCULAR [h] {UTC};
  CONTINUOUS SUBDIVISION Minutes: 0 .. 59 CIRCULAR [min];
  CONTINUOUS SUBDIVISION Seconds: 0,0000.000 .. 59.999 CIRCULAR [INTERLIS.s];
END GregorianDate;

DOMAIN XMLTime = FORMAT BASED ON UTC ( Hours/2 ":" Minutes ":" Seconds );
DOMAIN XMLDate = FORMAT BASED ON GregorianDate ( Year "-" Month "-" Day );
DOMAIN XMLDateTime EXTENDS XMLDate = FORMAT BASED ON GregorianDateTime
                                        ( INHERITANCE "T" Hours/2 ":" Minutes
                                                      ":" Seconds );
```

Ejemplo práctico:

```
CLASS Project =
  Start: FORMAT INTERLIS.XMLDateTime "2000-01-01T00:00:00.000" ..
                                     "2005-12-31T23:59:59.999";
  End: FORMAT INTERLIS.XMLDateTime "2002-01-01T00:00:00.000" ..
                                   "2007-12-31T23:59:59.999";
END Project;
```

2.8.8 Coordenadas

Las coordenadas se pueden definir en una, dos o tres dimensiones, y por lo tanto son un solo dígito, dos dígitos o tres dígitos. Está permitido añadir la segunda o tercera dimensión sólo en una extensión. Para cada dimensión debe indicarse el dominio numérico, así como tal vez una unidad de medida y un sistema de coordenadas (incluyendo número de ejes). Sólo se pueden indicar las unidades de medida concreta. Si se indica ningún sistema de referencia y si, además, las unidades de medida no parecen unidades de longitud, el sistema que implementa el modelo puede presumir que se trata de coordenadas cartesianas.

Siempre que se produce una definición de rotación (palabra clave ROTATION), es posible, dentro del alcance de las definiciones de cero direcciones (véase capítulo 2.8.5: Tipos de datos numéricos) referirse a dicho sistema de coordenadas de referencia. La definición de rotación determina qué eje del dominio de coordenadas corresponde a la dirección cero y cuál a la dirección de un ángulo recto positivo. Puede faltar en una definición concreta de coordenadas y ser añadida en una extensión.

Cuando se hagan extensiones, no se puede cambiar ninguna indicación relativa a la definición de eje y rotación del mismo.

```
DOMINIO
  CHCoord = COORD 480000.00 .. 850000.00 [m] {CHLV03[1]},
                   60000.00 .. 320000.00 [m] {CHLV03[2]},
                  ROTATION 2 -> 1;
```

En ambos ejes definidos se indica el dominio admisible, así como las unidades y el sistema de referencia incluyendo el número de eje al que se refieren las coordenadas. Los ejes reales se definen dentro del sistema de referencia. La definición de la rotación determina que la dirección cero conduce desde el eje nº2 al eje nº1, es decir, en el sistema Federal Suizo donde el valor nº1 corresponde a este y el valor nº2 al norte, la dirección cero muestra el norte y gira en sentido horario.

```
DOMINIO
  WGS84Coord = COORD  -90.00000 ..  90.00000          [Units.Angle_DMS] {WGS84[1]},
                    0,0000.000 .. 359.99999 CIRCULAR [Units.Angle_DMS] {WGS84[2]},
                      -1000.00 .. 9000.00            [m] {WGS84Alt[1]};
```

Típicamente las coordenadas geográficas se representan en grados y se refieren a un sistema de coordenadas elipsoidales (por ejemplo, CH1901). Entonces, la altitud se describe en metros. Se refiere a un sistema de altura de elipse especial con un eje.

Reglas de Sintaxis:

```
CoordinateType = 'COORD' NumericType
                   [ ',' NumericType [ ',' NumericType ]
                       [ ',' RotationDef ] ].

RotationDef = 'ROTATION' NullAxis-PosNumber '->' PiHalfAxis-PosNumber.
```

A menos que se defina un mínimo de un dominio numérico (con NumericType), el atributo correspondiente ha de ser definido como abstracto.

2.8.9 Dominios de identificación de objetos

Lo objetos identificables siempre están etiquetados con una identificación de objeto. Con el fin de dejar claro al sistema qué almacenamiento debe suministrarse y cómo deben ser generadas estas identificaciones de objetos, los dominios correspondientes se pueden definir y asignar a los temas y clases (ver los capítulos 2.5.2: Temas y 2.5.3: Clases y estructuras). Para la administración de identificadores de objeto, principalmente pero también de contenedores, tiene sentido tener atributos ordinarios con tales dominios.

Regla de sintaxis:

```
OIDType = 'OID' ( 'ANY' | NumericType | TextType ).
```

INTERLIS 2 en sí define los siguientes dominios OID-(véase el Apéndice A-El modelo de datos interno de INTERLIS):

```
DOMINIO
  ANYOID = OID ANY;
  I32OID = OID 0 .. 2147483647;     !! valores enteros positivos que necesitan 4 bytes
                                    !! de memoria
  STANDARDOID = OID TEXT*16;        !! de acuerdo con el apéndice D
                                    !! (solo se permiten letras y números)
  UUIDOID = OID TEXT*36;            !! de acuerdo con ISO 11578
```

Si se utiliza ANYOID para temas o clases abstractos, es necesario esperar una identificación de objeto cuya definición exacta sin embargo debe permanecer abierta. De lo contrario ANYOID sólo se puede

utilizar como un dominio de atributo. En consecuencia, no es sólo el propio OID el que pertenece al valor del atributo, sino también el dominio OID concreto.

Los valores OID de los dominios OID de texto, deben cumplir con las normas del tipo ID de XML: El primer símbolo debe ser una letra o un guion bajo, seguido de letras, números, puntos, guiones o guiones bajos; no hay dos puntos (!), Ver www.w3.org/TR/REC-xml.

2.8.10 Cajas negras

Al utilizar este tipo de datos es posible modelar atributos cuyo contenido no se puede especificar. La versión XML describe un atributo con contenido XML y la versión BINARY un contenido binario. Este tipo no se puede refinar en extensiones.

Regla de sintaxis:

```
BlackboxType = 'BLACKBOX' ( 'XML' | 'BINARY' ).
```

2.8.11 Dominios de clases y Rutas de atributo

Tal vez tenga sentido que los objetos de los datos contengan referencias a ciertas clases y atributos:

Reglas de Sintaxis:

```
ClassType = ( 'CLASS'
                [ 'RESTRICTION' '(' ViewableRef
                                    { ';' ViewableRef } ')' ]
            | 'STRUCTURE'
                [ 'RESTRICTION' '(' ClassOrStructureRef
                                    { ';' ClassOrStructureRef } ')' ] ).

AttributePathType = 'ATTRIBUTE'
                    [ 'OF' ( ClassType-AttributePath
                           | '@' Argument-Name ) ]
                    [ 'RESTRICTION' '(' AttrTypeDef
                                        { ';' AttrTypeDef } ')' ].

ClassConst = '>' ViewableRef.

AttributePathConst = '>>' [ ViewableRef '->' ] Attribute-Name.
```

Al indicar la estructura de cualquier estructura o clase, mediante la indicación de la clase (también admisible como extensión de STRUCTURE) es admitida cualquier clase (pero sin estructuras). Si sólo ciertas estructuras o clases y sus extensiones pueden ser admitidas, éstas deberán ser enumerados (RESTRICTION). En las extensiones todas las estructuras o clases admisibles, tienen que ser listadas de nuevo. No pueden contradecir la definición de la base. Tan pronto como se han definido estas restricciones, STRUCTURE ya no puede ser extendido por CLASS.

Mediante la indicación ATTRIBUTE se admite cierto tipo de ruta de atributo. Se puede afirmar que debe pertenecer a una clase (¡no a una subclase!), de acuerdo con otra definición (OF). Es posible hacer referencia a un atributo ClassType o como en el caso de la definición de una función (ver el capítulo 2.14: Funciones) a un argumento diferente. Además, todos los posibles tipos de atributos pueden ser restringidos (RESTRICTION). Los siguientes son adecuados como constantes: los nombres de los atributos de las clases, estructuras, asociaciones y vistas. El nombre de la clase correspondiente se puede expresar de forma explícita o derivarse del contexto o de la referencia a otro atributo u otro argumento (OF).

2.8.12 Cadenas de línea

2.8.12.1 Geometría de las cadenas de línea

En la práctica, un segmento de curva es una estructura unidimensional que no tiene divisiones, ni esquinas, ni de puntos dobles de ningún tipo (ver figuras 10 y 11). Los segmentos de curva suaves y únicos. Los segmentos de líneas rectas, arcos de círculo, segmentos de parábolas y clotóides son ejemplos de segmentos de curva. Cada segmento de curva tiene dos puntos límite (puntos de inicio y de final, que no se permite sean idénticos.) Los otros puntos de un segmento de curva se denominan puntos internos, que forman el interior del segmento de la curva.

> Definición exacta (los términos matemáticos que no se explican aquí, pero cuya definición se puede encontrar en los libros de texto, están escritos "*en cursiva y entre comillas*"): El segmento de curva significa un subconjunto del "*espacio euclidiano tridimensional*" (llamado espacio de aquí en adelante para abreviar), que es el "*conjunto de imágenes*" de un "*mapeo*" "*fluido*" e "*inyectivo*" de un *"intervalo*" (de la "*línea recta numérica*"). El punto de inicio y el punto final del segmento de curva son las imágenes del punto final del intervalo. El segmento de curva plana significa un segmento de curva en un plano (*"subespacio bidimensional*" del espacio).

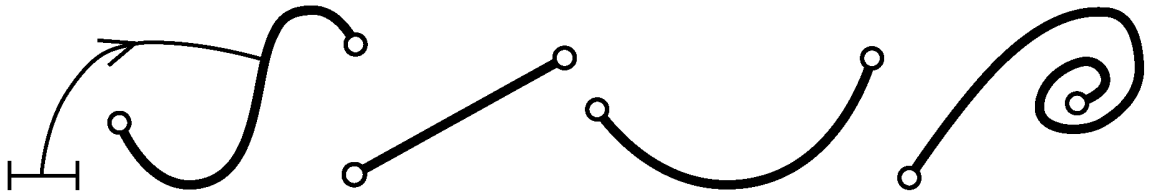

Figura 10: Ejemplos de segmentos de curva plana.

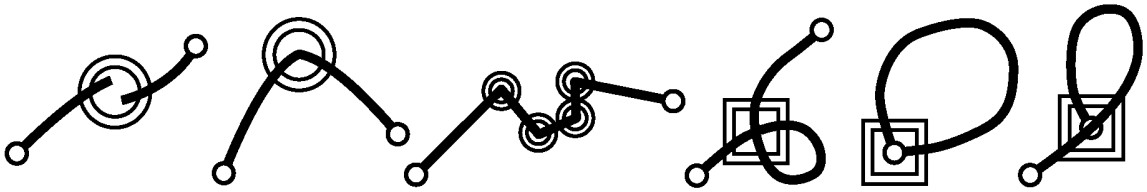

Figura 11: Ejemplos de conjuntos planos que no son segmentos de curva (un doble círculo indica "no fluida" y un doble cuadrado "no inyectiva").

Una cadena de líneas es una secuencia finita de segmentos de curva. Con excepción del primer segmento de curva, el punto de inicio de cada segmento de curva corresponde al punto final del segmento de curva anterior. Estos puntos son llamados puntos de control de la cadena de líneas. En la práctica, una línea poligonal puede utilizar múltiples segmentos de curva, segmentos de curva con puntos de base común, segmentos de curva que se intersecan y segmentos de curva que empiezan o terminan en el interior de otros segmentos de la curva (ver figuras 12 y 13). Una cadena de líneas simple, no contiene puntos de auto-intersección (ver figura 14). Además, para una línea simple cerrada, el punto inicial del primer segmento de curva y el punto final del último segmento de curva son idénticos.

> Definición exacta (los términos matemáticos que no se explican aquí, pero cuya definición se pueden encontrar en los libros de texto están escritas "*en cursiva y entre comillas*"): Cadenas de línea significa un subconjunto del espacio, que es el "*conjunto de imágenes*"

de un "*continuo*" y "*parcialmente fluido*" (pero no necesariamente "*inyectivo*") "*mapeo*" de un "*intervalo*" (denominado mapeo característico) y que contiene sólo un número finito de "*puntos no fluidos*". Un "*punto no fluido*" se llama vértice. Con una cadena de línea cerrada, el punto inicial y final son idénticos. Una cadena línea, cuya asignación de características también es "*inyectiva*", a excepción de su punto inicial y final que son idénticos, se llama cadena de línea simple.

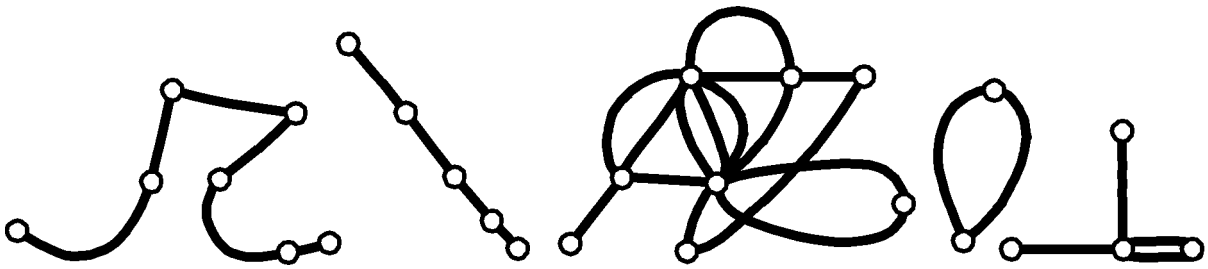

Figura 12: Ejemplos de cadenas de línea planas.

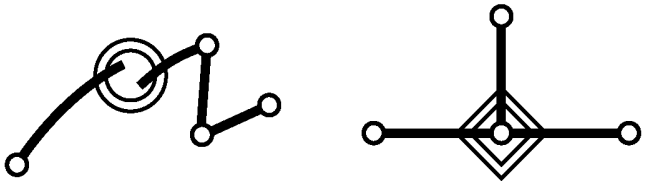

Figura 13: Ejemplos de conjuntos planos que no son cadenas de línea (el doble círculo significa "no continua" y el doble rombo "no es una imagen de intervalo").

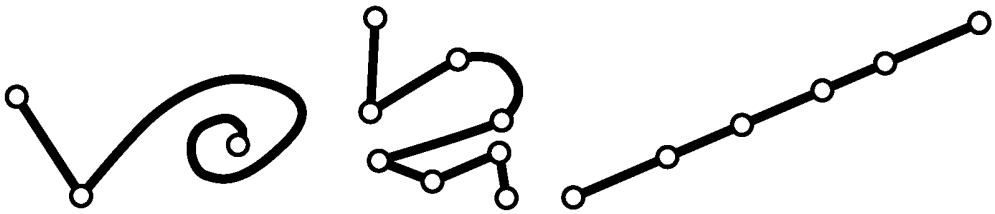

Figura 14: Ejemplos de (planos) cadenas de línea simples

2.8.12.2 Cadenas de línea con segmentos de línea recta y arcos de círculo como segmentos de curva predefinidos

Existen cadenas de línea según las directivas (DIRECTED POLYLINE) o cadenas de línea no dirigidas (POLYLINE) y también pueden usarse en el ámbito de las superficies y las teselas (véase el capítulo 2.8.13: Las superficies y las teselas).

La definición de un dominio de valor concreto de una cadena de línea, siempre requiere la especificación de las formas admisibles de segmentos de curva por medio de una enumeración, por ejemplo, segmentos de línea rectos (palabra clave STRAIGHTS), arcos de círculo (palabra clave ARCS) u otras posibilidades (ver el. capítulo 2.8.12.3: Otras formas de segmentos de curva) y además la indicación del dominio de los vértices. Dentro de un dominio de valor abstracto de una cadena de líneas, estas especificaciones se pueden omitir. Las siguientes reglas se aplican a las extensiones de dominio:

- Una línea sólo puede ser reducida, pero no completada con nuevos tipos.
- El dominio de coordenadas indicado dentro del ámbito de un dominio de valores de cadena de líneas, debe ser una restricción del dominio de coordenadas del dominio de valores de cadena de línea original, siempre que se haya definido este último.

Los segmentos de curva siempre se consideran una extensión de la estructura básica 'segmento de línea'. El dominio de coordenadas aplicado es el mismo que el definido en la definición de la cadena de líneas.

```
STRUCTURE LineSegment (ABSTRACT) =
  SegmentEndPoint: MANDATORY LineCoord;
END LineSegment;

STRUCTURE StartSegment (FINAL) EXTENDS LineSegment =
END StartSegment;

STRUCTURE StraightSegment (FINAL) EXTENDS LineSegment =
END StraightSegment;

STRUCTURE ArcSegment (FINAL) EXTENDS LineSegment =
  ArcPoint: MANDATORY LineCoord;
  Radius: NUMERIC [LENGTH];
END ArcSegment;
```

El primer segmento de curva de una cadena de línea es siempre un segmento de inicio. El segmento de inicio consiste sólo en el propio punto de inicio, que al mismo tiempo es el punto final del segmento de inicio. El segmento de línea recta tiene un punto final y por lo tanto determina una recta desde el punto final del segmento de curva predecesor a su punto final. Tanto el segmento inicial como el segmento de línea recta no requieren especificaciones adicionales. Por lo tanto, las extensiones correspondientes del LineSegment son nulas. Dos vértices sucesivos (SegmentEndPoints) pueden no coincidir en la proyección.

Un segmento de arco de círculo describe un segmento de curva que aparece como un segmento de arco de círculo verdadero en la proyección. Además del punto final un punto intermedio describe el segmento de arco de círculo. Sólo tiene importancia en relación con la geometría. Con coordenadas de tridimensionales usamos interpolación lineal para la altura en el segmento de arco de círculo. Se puede imaginar esta curva como la rosca de un tornillo cilíndrico en posición perpendicular al plano de proyección. El punto intermedio no es un vértice de la cadena de líneas. Se debe colocar lo más exactamente posible en el centro, entre el inicio y el punto final. Dado que el punto intermedio se indica con la misma precisión que los vértices, el radio calculado y el radio efectivo pueden diferir ampliamente. Siempre que se indique el radio efectivo, es relevante para la definición del arco de círculo. Entonces, el punto intermedio sólo determinará cuál de los cuatro posibles arcos de círculo es el deseado. Sin embargo, incluso en este caso, el punto intermedio solamente puede diferir en 2 unidades de la traza del arco de círculo calculado a partir del radio.

Puede ser necesario que una línea de cadena deba ser una línea de cadena simple, es decir, prácticamente que no se interseca a sí misma y, sobre todo, que el uso múltiple del mismo segmento de curva es imposible (palabra clave WITHOUT OVERLAPS). Si un arco del círculo y una línea recta (un segundo arco de círculo), como segmentos de curvas sucesivos de una cadena de líneas, no sólo tienen un vértice común, sino también un punto interior común (véase la definición de más arriba), entonces esto también está permitido en el caso de una cadena de línea simple, siempre que el segmento de círculo separado de la recta (el segmento de círculo doble separado del otro arco de círculo) tenga una altura de flecha menor o igual al decimal especificado después de WITHOUT OVERLAPS (véase la figura 15a).

Hay dos razones para esta regulación: Por un lado, razones numéricas y porque en algunos casos no se pueden evitar pequeñas superposiciones en arcos (arcos tangenciales). Por otra parte, al transferir datos que originalmente se han registrado en gráficos se han de tolerar incluso superposiciones (por ejemplo de varios centímetros), a menos que uno esté preparado para hacer frente a una enorme carga de trabajo para reparar estos solapamientos. Las tolerancias deberán ser listadas en las mismas unidades que las coordenadas del vértice. Por razones numéricas debe ser mayor que cero. Estos no pueden ser anulados y son obligatorios en el caso tanto de superficies como de teselas.

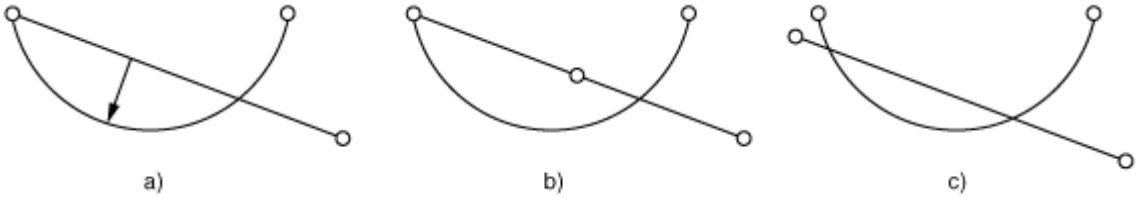

Figura 15: a) El parámetro de Altura (de la flecha) no podrá exceder de la tolerancia dada; b) superposición inadmisible de polilíneas, ya que otro vértice se sitúa entre vértice e intersección; c) superposición inadmisible de polilíneas, ya que no existe ningún vértice común.

Dentro del alcance de las definiciones de dominio el valor y las extensiones de atributos, las cadenas de líneas no dirigidas se pueden extender en cadenas de líneas dirigidas (véase el capítulo 2.8.13.4: de extensibilidad.)

Cuando las cadenas de línea son dirigidas, su dirección siempre debe ser conservada (incluso durante la transferencia de datos).

Para los vértices se define el dominio de valores de las coordenadas. Por medio de la restricción de existencia REQUIRED IN (ver los capítulos 2.12: Limitaciones y 2.13: Expresiones) es posible además exigir que las coordenadas no sean arbitrarias, sino que correspondan a los puntos de ciertas clases.

Si el tipo de coordenadas de los vértices es abstracto, entonces la cadena de línea también debe ser declarada abstracta.

Reglas de Sintaxis:

```
LineType = ( [ 'DIRECTED' ] 'POLYLINE' | 'SURFACE' | 'AREA' )
           [ LineForm ] [ ControlPoints ] [ IntersectionDef ]
           [ LineAttrDef ].

LineForm = 'WITH' '(' LineFormType { ',' LineFormType } ')'.

LineFormType = ( 'STRAIGHTS' | 'ARCS'
               | [ Model-Name '.' ] LineFormType-Name ).

ControlPoints = 'VERTEX' CoordType-DomainRef.

IntersectionDef = 'WITHOUT' 'OVERLAPS' '>' Dec.
```

Con el fin de poder asignar diferentes atributos a diferentes segmentos de los límites de las superficies (véase el capítulo 2.8.13: Superficies y mosaicos), es posible definir otros atributos para los objetos de la cadena de línea real (los llamados atributos de la línea, regla LineAttrDef solo con SURFACE y AREA). Sin embargo, esto no se traduce en el concepto de una división claramente definida en la frontera. Desde el punto de vista conceptual no tiene importancia si los segmentos de curva posteriores, con valores de atributo iguales, son considerados como objetos de una cadena de líneas individuales o como una cadena de línea completa (véase el capítulo 3.3.11.13: La codificación de las superficies y los mosaicos). La estructura empleada para la definición sólo puede presentar atributos locales y llamadas de función. Cuando se extiende un atributo de superficie o de tesela, para el cual se ha definido un atributo de línea mediante estructura, el atributo extendido no debe incluir un atributo de línea o su estructura debe ser una extensión abstracta de la estructura básica. Con las llamadas de función, el tipo resultante debe ser un atributo local.

Regla de sintaxis:

```
LineAttrDef = 'LINE' 'ATTRIBUTES' Structure-Name.
```

2.8.12.3 Otras formas de segmentos de curva

Además de segmentos de línea rectos y arcos de círculo, es posible definir otras formas de segmentos de curva. Además de sus nombres también tiene que especificarse según qué estructura se describe un segmento de curva. Estas definiciones de dichos segmentos de curva sólo son admisibles en el marco de un contrato, ya que no podemos asumir que un sistema es compatible con cualquier tipo de curvas.

Regla de sintaxis:

```
LineFormTypeDef = 'LINE' 'FORM'
                    { LineFormType-Name ':' LineStructure-Name ';' }.
```

Una estructura de línea siempre debe ser una extensión del segmento de línea definido por INTERLIS (ver el capítulo 2.8.12.2: Cadenas de líneas con segmentos de líneas rectas y arcos de círculo como segmentos de curva predefinidos).

2.8.13 Las superficies y teselas

2.8.13.1 Geometría de superficies

En la mayoría de los casos las superficies planas son suficientes para el modelado de datos geográficos. Además, INTERLIS también soporta superficies planas generales. En la práctica, una superficie general plana, está limitada por uno exterior y posiblemente por uno o varios límites interiores (véase la figura 20). Las propias líneas de límite deben consistir en cadenas de líneas simples que, desde un punto de vista geométrico, pueden combinarse en cadenas de línea simples cerradas. Además, deben situarse de tal manera que desde cualquier punto en el interior de la superficie a cualquier otro punto en el interior de la superficie exista una forma que ni se cruce con una línea límite ni contiene vértices de una línea límite (véase la figura 19). Mientras esta restricción no sea violada, los límites pueden reunirse en vértices. En tales situaciones, hay varias posibilidades concebibles que permitirían dividir el límite de la superficie como un todo en cadenas de líneas individuales (véase la figura 22). INTERLIS no insiste en una posibilidad concreta. Si dicha superficie se transfiere varias veces, se puede producir una posibilidad diferente en cada transferencia diferente.

> Definiciones exactas (términos matemáticos que no se explican aquí, pero cuya definición se pueden encontrar en los libros de texto, están escritos "en cursiva y entre comillas"):
>
> Elemento de superficie significa un subconjunto del espacio, que es el "conjunto de imágenes" de un "mapa" liso e "inyectivo" de un "polígono regular" planar (véanse las figuras 16 y 17).
>
> Superficie significa la unión F de un número finito de elementos de superficie, que están "conectados" y cumplen con la siguiente condición: Para cada punto P de la superficie existe una "zona", que es una deformación (es decir, un "mapeo homeomorfo") de un polígono regular plano. Si el punto P es una deformación de un punto del polígono de límite, se denomina punto de frontera F, de lo contrario punto interior de F. Se sostiene: el "límite" (es decir, el conjunto de todos los puntos del límite) de una superficie es la unión de un número finito de segmentos de curva, que se reúnen sólo en los puntos limítrofes (inicio o final). Una superficie plana es una superficie de un subconjunto de un plano. Representa: El límite de una superficie plana "simple continua" (gráficamente hablando: una superficie sin agujeros) es una línea poligonal cerrada simple, el llamado límite exterior. El límite de una superficie plana "n-veces continua" (gráficamente hablando: una superficie con n-1 orificios) consiste en el límite exterior correspondiente y n-1 otras cadenas simples cerradas de línea (los llamados límites internos). El límite exterior y todas las fronteras interiores no tienen puntos comunes. Una parte de la superficie cortada por un límite interior se llama un enclave (ver figuras 18, 19 y 20).

Una superficie general es una superficie con un número finito de puntos singulares pero con interior "conectado" (conjunto de puntos interiores). Un punto se llama punto singular si existe una deformación del punto junto con un "vecindario" en un conjunto plano de la hélice, el punto mismo en el centro. Conjunto de hélice significa la unión de un número finito de superficies triangulares que se encuentran, exactamente, en un punto llamado centro. Superficie general plana significa una superficie general que es subconjunto de un plano (véase la figura 21). Representa: Existen diferentes posibilidades para componer el límite de una superficie general plana por un número finito de cadenas de línea planas, cerradas, que tienen un máximo de un número finito de puntos en común y cada uno un número máximo finito de puntos dobles (véase la figura 22).

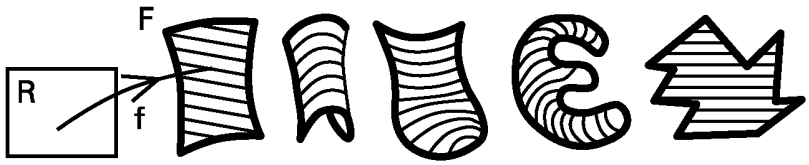

Figura 16: Ejemplos de elementos de superficie.

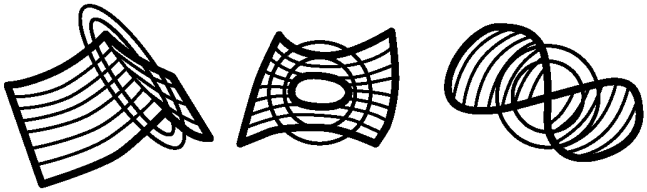

Figura 17: Ejemplos de conjuntos de puntos en el espacio, que no son elementos de superficie (en este caso un doble círculo significa "no es suave").

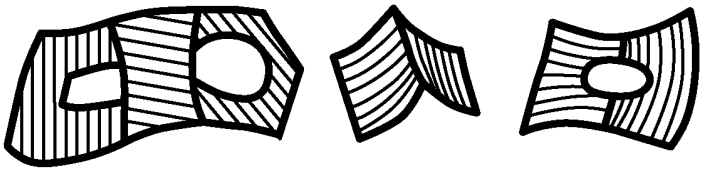

Figura 18 : Ejemplos de superficies en el espacio.

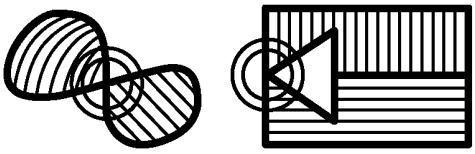

Figura 19: Ejemplos de conjuntos de puntos planos que no son superficies (un círculo doble marca un "punto singular").

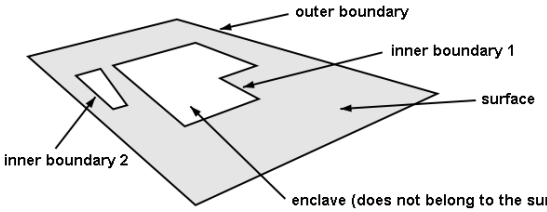

Figura 20: superficie plana con los límites y enclaves.

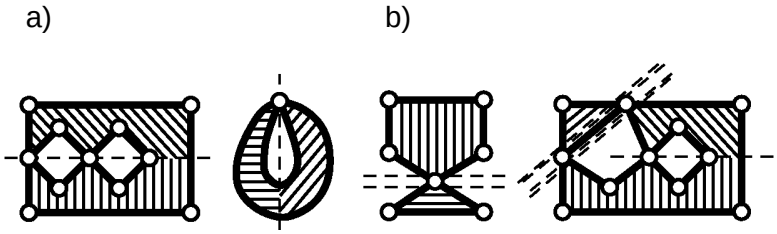

Figura 21: a) Ejemplos de superficies planas generales; b) Ejemplos de conjuntos de planos que no son superficies generales, porque su interior no está conectado. Pero estos conjuntos de planos se pueden subdividir en superficies generales ("---" muestra la subdivisión en elementos de superficie y "===" la subdivisión en superficies generales).

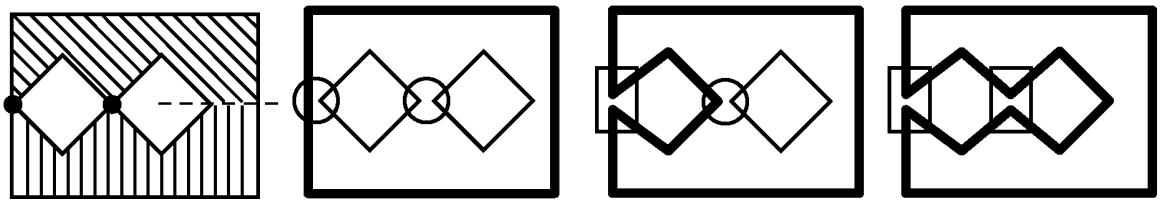

Figura 22: Diferentes posibles subdivisiones de los límites de una superficie general.

Junto con la definición de las superficies (generales), o superficies (generales) de una tesela determinadas, a partir de la cual los segmentos de curva de tolerancia de la frontera no pueden superponerse (para todas las definiciones concretas de las superficies y los mosaicos, WITHOUT OVERLAPS deberá ser directamente especificado heredado). Con superficies con prohibición de superposiciones, la intersección sólo se aplica a los segmentos de curva de una cadena de línea individual, sino a todos los segmentos de la curva de todas las cadenas de líneas del límite de superficie. En el caso de superficies de un mosaico, se aplia incluso a todas las cuerdas de línea conectadas con el mosaico. Además, y de acuerdo con la definición de la superficie (general), se excluyen cadenas de línea que no son parte del límite de una superficie (general).

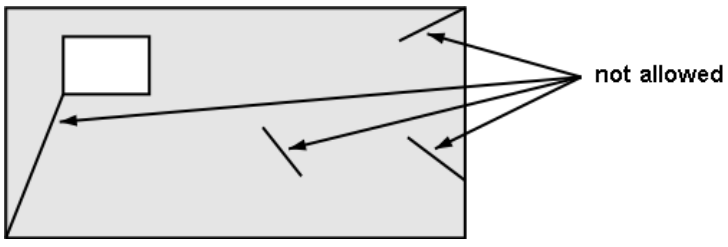

Figura 23: Configuraciones de límites prohibidas para las teselas.

2.8.13.2 Superficies.:

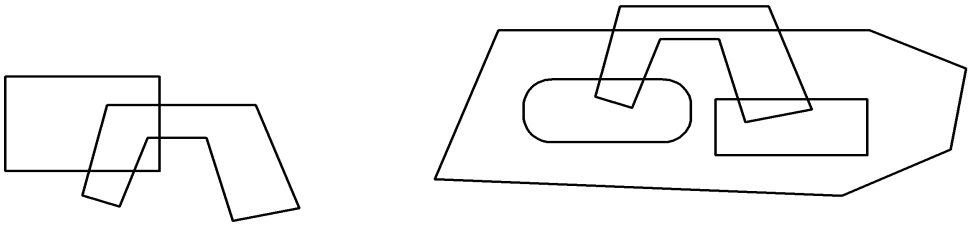

Figura 24: superficies individuales (SURFACE).

Para las superficies que se superponen parcialmente o en su totalidad, es decir, que no sólo tienen puntos de límite en común, el tipo de atributo geométrico SURFACE está disponible (ver figura 24). Este tipo se denomina superficie. Una superficie se compone de un límite exterior y posiblemente varios internos (alrededor de los enclaves). Cada límite consta de al menos una cadena de línea. Además de su geometría, cada cadena de línea cuenta con los atributos definidos (regla LineAttrDef).

2.8.13.3 Superficies de una tesela

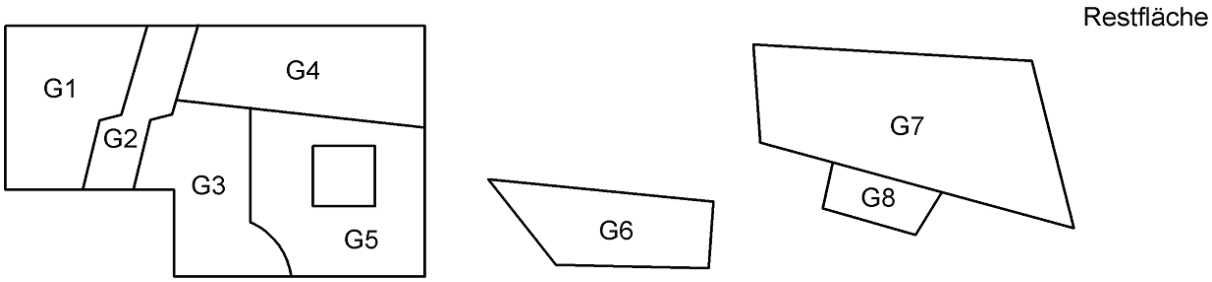

Figure 25: Mosaico (AREA).

Mosaico (plano) significa un conjunto finito de superficies (general) y entornos que cubren una capa sin superposiciones.

Para los mosaicos, el tipo de atributo geométrico AREA está a su disposición.

Un máximo de una superficie del mosaico (o exactamente una con la palabra clave adicional MANDATORY), pero nunca su entorno, se asigna al objeto de área. No es admisible que cada una de las dos superficies del mosaico con un límite común no corresponda a un objeto de área.

Así, cada objeto de área individual corresponde a una superficie. Como resultado, la misma estructura implícita se aplica a superficies y a objetos de área. No obstante, se aplican constricciones adicionales:

- Las líneas de cadena de un mosaico siempre deben ser verdaderos límites. Por lo tanto, no hay líneas de cadenas con la misma superficie a cada lado (véase la figura 23). Esto también se ha descartado por la definición de una superficie.
- Si hay objetos de área definidos a cada lado de una cadena de línea, entonces cada segmento de curva (unión entre dos vértices) de un objeto de área debe, en términos de geometría y de los atributos, corresponder exactamente al segmento de curva del otro objeto de área.
- Si se han definido atributos de línea para estas líneas, deben presentar los mismos valores para líneas coincidentes de las dos áreas vecinas.

Los mosaicos pueden no tener lugar en subestructuras.

Con el fin de poder hacer referencia a cadenas de línea de un mosaico como objetos individuales (y concretamente, como un objeto, incluso si la línea de la cadena es el límite de dos objetos de área), entonces AREA INSPECTION está a su disposición (véase el capítulo 2.15; Vistas).

2.8.13.4 Extensibilidad

Las superficies pueden extenderse en áreas. La extensión de una línea de cuerda en un área es inadmisible, ya que con una superficie se tienen que esperar varias cuerdas de línea, mientras que la definición de la cadena de línea sólo implica una cadena de línea.

Las superficies independientes y superficies de un mosaico se pueden extender en dos aspectos:

- Cuando es 'SURFACE' que se define principalmente y por lo tanto se permiten solapamientos, esto puede ser sustituido por 'AREA' en las extensiones, ya que esto no viola la definición básica.
- Se pueden adjuntar otros atributos de línea.

2.9 Unidades

Las unidades se describen siempre como un término y (en entre corchetes [] una contracción). Tanto el término como la contracción deben ser nombres. Cuando la contracción se omite, es la misma que el término mismo. Dependiendo del tipo de unidad se pueden seguir especificaciones. En el uso real de una unidad es siempre la contracción que ocurre. Por lo tanto, el término en sí solo es de carácter documental.

2.9.1 Unidades de base

unidades de base son metros, segundos, etc. No es necesario especificar más. Sin embargo, las unidades de base también se pueden definir como abstractas (palabra clave ABSTRACT), si la unidad misma es aún desconocida, pero la materia descrita es clara (por ejemplo, temperatura, dinero). Alas unidades abstractas no se les asigna ninguna contracción. Las unidades concretas no pueden extenderse.

Ejemplos:

```
UNIT
  Length (ABSTRACT);
  Time (ABSTRACT);
  Money (ABSTRACT);
  Temperature (ABSTRACT);
  Meter [m] EXTENDS Length;
  Second [s] EXTENDS Time;
  SwissFranc [CHF] EXTENDS Money;
  Celsius [C] EXTENDS Temperature;
```

INTERLIS define la unidad abstracta ANYUNIT. Todas las demás unidades heredan esta directa o indirectamente (véase el capítulo 2.10.3: Sistemas de referencia). Las siguientes unidades se han definido directa e internamente por INTERLIS

```
UNIT
  ANYUNIT             (ABSTRACT);
  DIMENSIONLESS       (ABSTRACT);
  LENGTH              (ABSTRACT);
  MASS                (ABSTRACT);
  TIME                (ABSTRACT);
  ELECTRIC_CURRENT    (ABSTRACT);
  TEMPERATURE         (ABSTRACT);
  AMOUNT_OF_MATTER    (ABSTRACT);
  ANGLE               (ABSTRACT);
  SOLID_ANGLE         (ABSTRACT);
  LUMINOUS_INTENSITY  (ABSTRACT);
  MONEY               (ABSTRACT);

  METER [m]           EXTENDS LENGTH;
  KILOGRAM [kg]       EXTENDS QUANTITY;
  SECOND [s]          EXTENDS TIME;
  AMPERE [A]          EXTENDS ELECTRIC_CURRENT;
  DEGREE_KELVIN [K]   EXTENDS TEMPERATURE;
  MOLE [mol]          EXTENDS AMOUNT_OF_MATTER;
  RADIAN [rad]        EXTENDS ANGLE;
  STERADIAN [sr]      EXTENDS SOLID_ANGLE;
  CANDELA [cd]        EXTENDS LUMINOUS_INTENSITY;
```

Observación: En el apéndice F: Definición de las unidades, las unidades más comunes han sido reunidas en un modelo de tipo extendido.

2.9.2 Unidades derivadas

Multiplicar o dividir unidades derivadas con constantes o funciones puede convertirlas en diferentes unidades. Ejemplo:

```
UNIT
  Kilometer [km] = 1000 [m];
  Centimeter [cm] = 1 / 100 [m];
  Inch [in] = 0.0254 [m];          !! 1 inch equals 2.54 cm
  Fahrenheit [oF] = FUNCTION // (oF + 459.67) / 1.8 // [K];
```

Los datos de kilómetros tienen que ser multiplicados por mil para convertirse en el equivalente en metros. Los datos en pulgadas tienen que ser multiplicados por 2,54 para convertirse en el equivalente en centímetros. Añadir 459,67 a los datos en grados Fahrenheit y luego dividir el resultado entre 1,8 para calcular la temperatura equivalente en grados Kelvin.

Una unidad derivada automáticamente se considera una extensión de la misma unidad abstracta en la que se puede convertir.

2.9.3 Unidades combinadas

Las unidades combinadas (por ejemplo, km por hora) son el resultado de una multiplicación o división de otras unidades (unidades básicas, unidades derivadas o combinados). Las unidades combinadas también se pueden definir como unidades abstractas. A continuación, deben referirse enteramente a otras unidades abstractas.

Por lo tanto, las unidades utilizadas en la extensión concreta deben ser una extensión concreta de las unidades que aparecen en la definición abstracta. Ejemplo:

```
UNIT
  Velocity (ABSTRACT) = ( Length / Time );
  Kilometer per hour [kmph] EXTENDS Velocity = ( km / h );
```

Reglas de Sintaxis:

```
UnitDef = 'UNIT'
            { Unit-Name
                [ '(' 'ABSTRACT' ')' | '[' UnitShort-Name ']' ]
                [ 'EXTENDS' Abstract-UnitRef ]
                [ '=' ( DerivedUnit | ComposedUnit ) ] ';' }.

DerivedUnit = [ DecConst { ( '*' | '/' ) DecConst }
              | 'FUNCTION' Explanation ] '[' UnitRef ']'.

ComposedUnit = '(' UnitRef { ( '*' | '/' ) UnitRef } ')'.

UnitRef = [ Model-Name '.' [ Topic-Name '.' ] ] UnitShort-Name.
```

2.10 Tratamiento de metaobjetos

2.10.1 Observaciones generales relativas a metaobjetos

En el sentido de INTERLIS 2, los metaobjetos son objetos que son de importancia en el ámbito de las descripciones de modelos de aplicación. Esto es útil en sistemas de referencia y símbolos gráficos.

La construcción de objetos del sistema de referencia o de objetos de símbolos, se debe definir con los medios habituales de clases y estructuras, ya sea en un modelo REFSYSTEM MODEL en un SYMBOLOGY MODEL. Las Clases de referencia del sistema, las clases de ejes, así como las clases de símbolos, deben ser extensiones de las clases COORDSYSTEM, REFSYSTEM AXIS o SIGN

suministradas por INTERLIS. Éstas, a su vez, son extensiones de la clase METAOBJECT. Esta clase posee un nombre de atributo que debe ser inequívoco, dentro del alcance de todos los metaobjetos de un contenedor.

Para la descripción de un modelo práctico, no se necesita el objeto meta como tal. Es suficiente con saber el nombre, como el representante del metaobjeto. Sin embargo, para un sistema en ejecución, por regla general, los metaobjetos deben ser conocidos por completo, es decir, con todos sus atributos. Por lo tanto, debe quedar claro para cualquier sistema, en tiempo de ejecución, que contenedor tiene un metaobjeto de nombre con cierto nombre. Antes de que los metaobjetos (o sus nombres) puedan ser utilizados en los modelos, tienen que ser declarados. Para lograr esto, se introduce un nombre de contenedor, se introduce el tema que se presume y luego (por clase) se enumeran los nombres de los objetos esperados (regla MetaDataBasketDef). El nombre del contenedor también se puede definir como la extensión de un nombre ya introducido (EXTENDS). Por consiguiente, el tema correspondiente debe ser el mismo que con el nombre de base o una extensión del mismo.

Si se hace referencia a un metaobjeto (regla MetaObjectRef) bajo el nombre del contenedor extendido, entonces el nombre del objeto meta debe figurar en él o en una definición heredada. Cualquier sistema de tiempo de ejecución primero tratará de encontrar el metaobjeto en su cesta correspondiente. Si no se encuentra tal metaobjeto, la búsqueda continúa de acuerdo con los nombres de contenedor que han sido heredados directa o indirectamente. Así, los metaobjetos se pueden preparar y refinar en varios niveles. Por ejemplo, para comenzar con símbolos gráficos comunes se pueden definir, y más tarde se refinan y complementan a nivel regional. Cómo hacer referencia al contenedor concreto dependiendo del nombre del contenedor se resuelve en el momento del tiempo de ejecución del sistema empleado.

Si un objeto se denomina meta (regla MetaObjectRef) bajo la canasta nombre extendida, entonces el nombre del objeto meta debe figurar en ella o en una definición heredada. Cualquier sistema de tiempo de ejecución en primer lugar tratar de encontrar el objeto meta en su contenedor correspondiente. Si no se encuentra dicho objeto meta la búsqueda continúa de acuerdo con las canastas nombres que han sido heredados directamente o indirectamente. Por lo tanto objetos meta se pueden preparar y refinar en varios niveles. Por ejemplo, para empezar símbolos gráficos comunes se puede definir, que luego son refinados y complementada a nivel regional. Cómo hacer referencia a la cesta de hormigón en función de la canasta-name es hasta el sistema de tiempo de ejecución empleado.

En un modelo de aplicación traducido (ver el capítulo 2.5.1: Modelos) un nombre de contenedor puede ser asignado a un contenedor concreto que sólo contiene traducciones (objetos METAOBJECTS_TRANSLATION; véase el apéndice A: El *modelo de datos interno INTERLIS*), traduciendo de ese modo esos metaobjetos que han sido introducidos por el contenedor correspondiente en el modelo, tomado como base. Si este modelo es una traducción, también es posible asignar a este contenedor de un contenedor concreto que sólo contiene traducciones. Si el modelo no es una traducción, se le puede asignar una cesta concreta que no contiene traducciones (es decir, sin objetos METAOBJECT_TRANSLATION)..

Reglas de Sintaxis:

```
MetaDataBasketDef = ( 'SIGN' | 'REFSYSTEM' ) 'CONTENEDOR' Contenedor-Name
                      Properties<FINAL>
                      [ 'EXTENDS' MetaDataBasketRef ]
                          '~' TopicRef
                      { 'OBJECTS' 'OF' Class-Name ':' MetaObject-Name
                          { ',' MetaObject-Name } } ';'.

MetaDataBasketRef = [ Model-Name '.' [ Topic-Name '.' ] ] Contenedor-Name.

MetaObjectRef = [ MetaDataBasketRef '.' ] Metaobject-Name.
```

Si en el contexto actual sólo es necesario un nombre de contenedor de metadatos, la referencia al contenedor de metadatos (MetaDataBasketRef) no debe estar indicado en la referencia al metaobjeto.

2.10.2 Parámetros.

Por medio de los parámetros, esas propiedades pueden ser designados en el metamodelo que no se refieren al mismo metaobjeto, sino a su uso dentro de la aplicación. Su definición se introduce por la palabra clave PARAMETER y se construye de una manera similar a la definición de atributos.

2.10.2.1 Parámetros para sistemas de referencia y coordenadas

Para sistemas de referencia y coordenadas, así como sus ejes, sólo es admisible el parámetro predefinido Unidad.

Si se hace referencia a un sistema de referencia o de coordenadas (regla RefSys) dentro de la definición de un tipo de datos numérico (véase el capítulo 2.8.5: Tipos de datos numéricos) o un tipo de coordenadas (véase el capítulo 2.8.8: Coordenadas) que sea compatible con la unidad del eje correspondiente del sistema de coordenadas, o con el único eje del sistema de referencia (véase el capítulo 2.10.3: Sistemas de referencia).

2.10.2.2 Parámetros de símbolos

Las definiciones de los parámetros de símbolos son arbitrarias. Estos parámetros corresponden a las especificaciones (por ejemplo, identificación de símbolo de punto, posición, rotación) que tienen que ser suministrados a un sistema con el fin de permitir la representación gráfica. Para empezar con el símbolo tiene que ser seleccionado. Esto se hace definiendo un parámetro para la referencia a la clase de símbolo dentro de la cual se define el parámetro (METAOBJECT). Este parámetro de un símbolo también puede ser una referencia a otro metaobjeto (METAOBJECT OF). En ambos casos se indicará un metaobjeto de referencia dentro de la aplicación (véase el capítulo 2.16: Descripciones gráficas), es decir, se indicarán tanto el nombre de referencia del contenedor como el del meta objeto. Así, la respectiva herramienta puede determinar la verdadera identidad del contenedor e identificar al metaobjeto.

Además de estos casos especiales de parámetros, los parámetros se pueden definir de la misma manera que los atributos.

Regla de sintaxis:

```
ParameterDef = Parameter-Name Properties<ABSTRACT,EXTENDED,FINAL>
                  ':' ( AttrTypeDef
                      | 'METAOBJECT' [ 'OF' MetaObject-ClassRef ] ) ';'.
```

2.10.3 Sistemas de referencia

Sin más especificaciones, los valores numéricos y coordenadas indican diferencias, no tienen una referencia absoluta. Para ello, debe ser definido un sistema de coordenadas, o un sistema escalar. La definición del modelo se ejecutará en un REFSYSTEM MODEL. En INTERLIS las siguientes clases están a su disposición:

```
CLASS METAOBJECT (ABSTRACT) =
  Name: MANDATORY NAME;
UNIQUE Name;
END METAOBJECT;

STRUCTURE AXIS =
  PARAMETER
    Unit: NUMERIC [ ANYUNIT ];
END AXIS;

CLASS REFSYSTEM (ABSTRACT) EXTENDS METAOBJECT =
 END REFSYSTEM;

CLASS COORDSYSTEM (ABSTRACT) EXTENDS REFSYSTEM =
```

```
  ATTRIBUTE
    Axis: LIST {1..3} OF AXIS;
END COORDSYSTEM;

CLASS SCALSYSTEM (ABSTRACT) EXTENDS REFSYSTEM =
  PARAMETER
    Unit: NUMERIC [ ANYUNIT ];
END SCALSYSTEM;
```

En las extensiones es posible añadir otros atributos típicos para este tipo de sistemas escalares y de coordenadas e, incluso, la unidad puede ser reemplaza por extensiones (parra la sintaxis de las definiciones de los parámetros, véanse los capítulos 2.10.2: Parámetros y 2.5.3: Clases y estructuras). Sin embargo, esta unidad tendrá que ser compatible con una unidad definida en el dominio del valor (ver el capítulo 2.9: Unidades).

2.11 Parámetros de tiempo de ejecución

Además de los datos reales y los metadatos, se pueden definir elementos de datos individuales que deberían ser suministrados por un sistema de procesamiento, evaluación o de representación gráfica. Se llaman parámetros de tiempo de ejecución.

Regla de sintaxis:

```
RunTimeParameterDef = 'PARAMETER'
                        { RunTimeParameter-Name ':' AttrTypeDef ';' }.
```

Los mosaicos (palabra clave AREA) no son admisibles para los parámetros de tiempo de ejecución. Dado que los parámetros de tiempo de ejecución definen condiciones para sistemas que van más allá del lenguaje INTERLIS 2, sólo pueden ser definidas dentro de los sistemas para los que se han firmado contratos.

Los parámetros de tiempo de ejecución típicos son, por ejemplo, la escala de representación o la fecha actual.

2.12 Constricciones

Las constricciones[1] sirven para definir las restricciones y obligaciones a las cuales los objetos están sometidos.

Las constricciones que se refieren a un solo objeto se describen mediante una expresión lógica relacionada con los atributos de los objetos (véase el capítulo siguiente). En ella es posible definir constricciones que son obligatorias y que se aplican a todos los objetos perentoriamente (palabra clave MANDATORY), y otros que sólo se aplican como regla general. En este último caso, se indica qué porcentaje de las instancias de una clase debe cumplir, normalmente, con la constricciones (regla PlausibilityConstraint).

Al indicar una constricción de existencia (regla ExistenceConstraint), es necesario que el valor de un atributo de cada objeto de la clase constreñida exista en un determinado atributo de una instancia de otra clase. Esto sólo es posible si el atributo de la constricción es compatible con el otro atributo y tiene los siguientes efectos:

- Si el dominio del atributo de la constricción es igual al dominio del otro atributo o a una de sus extensiones, el valor de la restricción debe existir en el atributo requerido de la otra instancia.
- Si el dominio de un atributo es una estructura, se comparan todos los atributos que contiene.
- Si el dominio del otro atributo es una coordenada o el dominio del atributo de constricción una polilínea, una superficie o un mosaico con el mismo dominio de coordenadas, o extendido, todas

[1] Constraint implica restricción, pero también obligación. En español, la restricción limita o reduce la posibilidad de acción. La constricción limita y reduce, pero también obliga a una acción concreta.

las coordenadas de los vértices de la polilínea, de la superficie o del mosaico, deben ocurrir dentro del atributo requerido de la otra instancia.

Las constricciones de singularidad se introducen con la palabra clave UNIQUE (regla UniquenessConstraint).

El significado conceptual de "globalmente" es que todos los objetos existentes en cualquier contenedor deben cumplir con estos requisitos.

Es posible exigir que dentro de una estructura, clase o asociación, una cierta combinación de atributos de subestructuras definidos por BAG OF o LIST OF o uno de los atributos de línea de dominios geométricos SURFACE o AREA sea localmente (LOCAL) inequívoca, es decir, dentro del alcance de todos los elementos estructurales asignados al objeto actual o al elemento estructural.

Ejemplo:

```
CLASS A =
  K: (A, B, C);
  ID: TEXT*10;
UNIQUE K, ID;
END A;
```

Se considera que una constricción está cumplida, si un atributo cuyo dominio es indefinido está conectado con una restricción de singularidad.

Con las constricciones que hacen referencia a la clase (SET CONSTRAINT), es posible definir las condiciones que se aplicarán a la cantidad total de objetos de la clase (la sección que cumple con la constricción WHERE). Las funciones que se utilizan dentro de esta restricción esperan un conjunto de objetos (OBJECTS OF) en un argumento (por lo general en la primera). Con el fin de entregar la cantidad total de objetos de esta clase (la sección que cumple con la restricción WHERE) a este argumento, se introduce ALL (sin indicar su propia clase). Puesto que tales constricciones de consistencia no se refieren al objeto individual, es imposible tocar los valores de los atributos. Por lo tanto, para todos los argumentos adicionales y para las comparaciones sólo se consideran adecuadas las constantes, parámetros de tiempo de ejecución, tipos de clase, así como tipos de atributo y otras llamadas de función que cumplan esta restricción.

Ejemplos:

```
CLASS B =
  Type: (a, b,  c);
  Geometry: SURFACE ...;
SET CONSTRAINT WHERE Type = #a :
  areAreas(ALL, UNDEFINED, >> Geometry);
END B;

STRUCTURE S =
  Geometry: SURFACE ...;
END S;

CLASS C =
  Surfaces: BAG OF S;
SET CONSTRAINT
  areAreas(ALL, >> Surfaces, >> S->Geometry);
END;
```

Los objetos de la clase B, cuyo tipo es ‘a’ formarán una superficie, debido a la utilización de la función estándar areAreas (ver el capítulo 2.14: Funciones). Todas las formas y objetos de la clase C forman un mosaico.

Se deben definir restricciones más extensas dentro de las vistas (por ejemplo, una vista que conecta una determinada clase con ella misma) y así permitir comparar cualquier combinación de atributos con todos

los demás objetos de la clase. Es perentorio que tales puntos de vista se definan dentro del modelo de datos.

Las constricciones que se extienden a una multitud de objetos (sobre todo, restricciones de singularidad) no son siempre totalmente controlables, ya que este control sólo se puede ejecutar con contenedores disponibles localmente. Sin embargo, conceptualmente, se aplican de forma global (excepto con los metamodelos, véase el apéndice E: Singularidad de las claves de usuario).

Reglas de Sintaxis:

```
ConstraintDef = ( MandatoryConstraint
                | PlausibilityConstraint
                | ExistenceConstraint
                | UniquenessConstraint
                | SetConstraint ).

MandatoryConstraint = 'MANDATORY' 'CONSTRAINT' Logical-Expression ';'.

PlausibilityConstraint = 'CONSTRAINT'
                           ( '<=' | '>=' ) Percentage-Dec '%'
                             Logical-Expression ';'.

ExistenceConstraint = 'EXISTENCE' 'CONSTRAINT'
                        AttributePath 'REQUIRED' 'IN'
                          ViewableRef ':' AttributePath
                            { 'OR' ViewableRef ':' AttributePath } ';'.

UniquenessConstraint = 'UNIQUE' [ 'WHERE' Logical-Expression ':' ]
                         ( GlobalUniqueness | LocalUniqueness ) ';'.

GlobalUniqueness = UniqueEl.

UniqueEl = ObjectOrAttributePath { ',' ObjectOrAttributePath }.

LocalUniqueness = '(' 'LOCAL' ')'
                    StructureAttribute-Name
                      { '->' StructureAttribute-Name } ':'
                               Attribute-Name { ',' Attribute-Name }.

SetConstraint = 'SET' 'CONSTRAINT' [ 'WHERE' Logical-Expression ':' ]
                  Logical-Expression ';'.
```

Sólo es posible definir posteriormente restricciones de coherencia para una cierta clase o asociación (típicamente siguiendo la definición de una asociación).

Regla de sintaxis:

```
ConstraintsDef = 'CONSTRAINTS' 'OF' ClassOrAssociationRef '='
                   { ConstraintDef }
                 'END' ';'.
```

2.13 Expresiones

En general, las expresiones (Expression) se utilizan, por ejemplo, como argumentos de funciones o en constricciones y selecciones. Con expresiones lógicas (Logical-Expression) el tipo de resultado debe ser booleano. Las expresiones se refieren a un objeto de contexto (es decir, un objeto para el que se formulan las constricciones). Proviniendo de este objeto es posible hacer referencia a un atributo, un elemento de estructura, una función, etc. Los parámetros se entrelazan como factores en los predicados. Un predicado

es una declaración que puede ser correcta o falsa. Por medio de los operadores booleanos los predicados pueden convertirse en una expresión lógica.

Reglas de Sintaxis:

```
Expression = Term.

Term = Term1 { 'OR' Term1 }.

Term1 = Term2 { 'AND' Term2 }.

Term2 = Predicate [ Relation Predicate ].

Predicate = ( Factor
            | [ 'NOT' ] '(' Logical-Expression ')'
            | 'DEFINED' '(' Factor ')' ).

Relation = ( '==' | '!=' | '<>' | '<=' | '>=' | '<' | '>' ).
```

Consideraciones respecto a la importancia de estas reglas de sintaxis:

- De acuerdo con las reglas de sintaxis para los términos, la obligación más forzada es la comparación (relación), seguido de AND y OR.
- NOT, seguido de la expresión lógica entre paréntesis, exige la negación de esta expresión.
- Comparación de factores. Dependiendo del tipo de factor, ciertas comparaciones son inadmisibles:
  - Con los tipos de cadena únicamente se admiten Igualdad (==) y Desigualdad (!=, <>). Las comparaciones adicionales deben realizarse por medio de funciones. Sobre todo, es de gran importancia definir con claridad cómo deben tratarse las particularidades regionales, como las diéresis o los acentos.
  - Con el tipo de datos numéricos y las comparaciones de dominios estructurados, se definen como de costumbre. Las comparaciones Más y Menos no son prácticas con el tipo de datos circular.
  - Las coordenadas en su conjunto sólo pueden ser examinadas en cuanto a su igualdad o desigualdad. Todas las demás comparaciones sólo están disponibles para sus componentes por separado (regla Factor).
  - Con enumeraciones, las comparaciones Más y Menos sólo son admisibles si la enumeración se ha definido como ordenada. Equivalencia significa equivalencia exacta. Si todos los subelementos de un nodo están incluidos, entonces se debe usar la función isEnumSubVal.
  - Las líneas sólo pueden ser probadas en cuanto a si son indefinidos (== UNDEFINED).
  - Un factor también puede designar un objeto. Entonces es posible examinarlo en términos de definición, igualdad y desigualdad.
- Si un factor no sólo consiste en el elemento en sí, sino también del camino que conduce a él, el elemento siempre se considera indefinido, siempre que cualquier atributo de la ruta sea indefinido. Así, la función incorporada DEFINED (a.b) es igual a (a.b != UNDEFINED).
- Cualquier expresión será examinada de izquierda a derecha, pero sólo hasta que su valor resultante aparezca definido. En otras palabras, los términos unidos con OR sólo serán evaluados si el valor hasta este punto es falso. Por otro lado, los términos unido con AND solamente serán evaluados si el valor hasta este punto es verdadero.

Los factores pueden ser formados de acuerdo con las siguientes reglas de sintaxis:

```
Factor = ( ObjectOrAttributePath
         | ( Inspection | 'INSPECTION' Inspection-ViewableRef )
             [ 'OF' ObjectOrAttributePath ]
         | FunctionCall
         | 'PARAMETER' [ Model-Name '.' ] RunTimeParameter-Name
```

```
          | Constant ).

  ObjectOrAttributePath = PathEl { '->' PathEl }.

  AttributePath = ObjectOrAttributePath.

  PathEl = ( 'THIS'
           | 'THISAREA' | 'THATAREA'
           | 'PARENT'
           | ReferenceAttribute-Name
           | AssociationPath
           | Role-Name [ '[' Association-Name ']' ]
           | Base-Name
           | AttributeRef ).

  AssociationPath = [ '\' ] AssociationAccess-Name.

  AttributeRef = ( Attribute-Name ( [ '[' ( 'FIRST'
                                          | 'LAST'
                                          | AxisListIndex-PosNumber ) ']' ] )
                 | 'AGGREGATES' ).

  FunctionCall = [ Model-Name '.' [ Topic-Name '.' ] ] Function-Name
                     '(' Argument { ',' Argument } ')'.

  Argument = ( Expression
             | 'ALL' [ '(' RestrictedClassOrAssRef | ViewableRef ')' ] ).
```

Los factores pueden referirse a los objetos y a sus atributos. Paso a paso, es posible configurar rutas completas de objetos dentro de este procedimiento. Cada construcción abre el camino desde el objeto actual al siguiente. El primer objeto actual resulta del contexto, por ejemplo, un objeto de la clase para la que se está definiendo una constricción.

- THIS: Designa el denominado objeto de contexto, es decir, el objeto actual de una clase, una vista o una definición gráfica, que requiere una ruta de objeto. THIS, por ejemplo, debe ser indicado cuando se llama a una función que cuenta con ANYCLASS o ANYSTRUCUTURE como parámetro.
- THISAREA and THATAREA: Designa un objeto de área en cuyo límite común se puede encontrar el objeto de cadena de línea actual. La aplicación de THISAREA y THATAREA sólo es posible dentro del ámbito de la inspección de un mosaico (véase el capítulo 2.15 Vistas).
- PARENT: Designa el elemento de superestructura o superobjeto del elemento u objeto de estructura actual. La vista debe ser una inspección ordinaria (sin inspección de área) (véase el capítulo 2.15: Vistas).
- Designa súper elementos-estructura o super-objeto del elemento de estructura u objeto actual. La vista debe ser una inspección ordinaria (sin inspección de la zona) (véase el capítulo 2.15 Vistas).
- Indicación de atributos de referencia: designa el objeto que es asignado al objeto actual, que a su vez es asignado desde el objeto actual (la estructura actual a través del atributo de referencia indicado).
- Indicación de rol (papel, o función desempeñada): La indicación de rol es válida siempre que exista una sola función correspondiente. La indicación de rol puede apuntar a un rol inicial (según el cual el objeto actual está relacionado con la referencia de asociación) o a un rol de destino (según el cual la referencia de asociación está relacionada con el objeto de referencia). Siempre que la indicación de la función se complementa con el nombre de referencia, sólo puede apuntar a los roles iniciales. Dependiendo de la posición del elemento de la trayectoria dentro de la trayectoria las funciones se buscan de una manera diferente. Si la indicación de rol es el primer elemento de

ruta dentro de la ruta, se busca la función en todos los accesos de relación dentro de la clase, siempre y cuando la ruta pueda ser utilizada en su contexto. Si la indicación de rol es un elemento siguiente de la ruta, se busca la función en todas las asociaciones disponibles dentro del tema donde se define la clase en cuyo contexto se puede utilizar la ruta. Sólo se tendrán en cuenta aquellas asociaciones que estén relacionadas a través de roles con la clase del objeto predecesor de la ruta.

- Indicación de vista básica: Mediante el nombre (local) de la vista básica designamos el objeto (virtual) correspondiente de la vista básica en la vista actual, en la relación derivada actual.

Cuando nos referimos a un atributo nos referimos al valor del atributo del objeto de contexto o del objeto designado por el camino. Además, todas las rutas que terminan con un atributo se denominan rutas de atributos y pueden utilizarse en diferentes reglas de sintaxis independientemente de los factores.

- En circunstancias normales, es suficiente con indicar el nombre del atributo.
- Cuando se trata de un atributo de coordenadas, se indica el número del eje para designar el componente correspondiente de la coordenada. El primer componente tiene índice 1.
- El atributo implícito AGGREGATES se define dentro de vistas de agregación (véase el capítulo 2.15: Vistas) y designa el conjunto (BAG OF) de los objetos base agregados.

En subestructuras ordenadas (LIST OF) se pueden abordar elementos individuales. Los índices admisibles son:

- FIRST: el primer elemento.
- LAST: el último elemento.
- Número de índice: El índice indicado debe ser menor o igual al número máximo determinado dentro de la cardinalidad. El primer elemento tiene índice 1. Si es menor o igual al número mínimo determinado dentro de la cardinalidad, siempre hay un elemento correspondiente en existencia; Si es mayor la existencia de tal elemento no puede ser garantizada. Posteriormente el factor puede ser indefinido.

Un factor también puede ser una inspección (véase el capítulo 2.15: Vistas). Si está precedida por una ruta de objeto, entonces la clase de objeto así dada debe coincidir con la clase de objeto de la inspección o ser una extensión de la misma. Para pertenecer al conjunto de elementos de estructura suministrados por la inspección, deben pertenecer al objeto definido por la trayectoria del objeto.

Los factores pueden también ser llamadas a funciones. Como sus argumentos podemos considerar:

- Expresiones: El tipo de resultado de una expresión debe ser compatible con el tipo de argumento.
- Si por medio de la expresión se hace una indicación de rol, entonces la expresión designa el conjunto de objetos de destino relacionados a través de este papel. Con un parámetro formal OBJECT OF o OBJECTS OF (sólo si se basa en la descripción del modelo es evidente que sólo es posible un objeto objetivo) debe solicitarse (cf. capítulo 2.14: Funciones).
- Todos los objetos (ALL) de la clase en cuyo contexto se está realizando la llamada a la función o se indican todos los objetos de la clase. Con un parámetro formal OBJETOS DE debe ser requerido (ver capítulo 2.14 Funciones). Esto siempre significa todos los objetos correspondientes a esta clase o sus extensiones.

Como valores de comparación, los siguientes elementos entran en cuestión: llamadas a funciones, parámetros de tiempo de ejecución (véase capítulo 2.16: Descripciones gráficas) y constantes.

2.14 Funciones

Por medio de su nombre, los parámetros formales, así como una descripción de la función corta, una función se define como una explicación. Los nombres de los parámetros son sólo de valor documental.

Esta definición sólo es admisible dentro de los contratos ya que, de lo contrario, no se garantizaría una evaluación automática de los modelos.

Como parámetros formales o resultados de la función podemos considerar:

- Todos esos tipos son admisibles para atributos, sobre todo estructuras. Los factores correspondientes (Regla Factor) pasan a funcionar como argumentos (es decir, parámetros actuales).
- Si se indica una estructura, son principalmente elementos estructurales que se consideran como argumentos. También es posible indicar rutas de objetos que conducen a objetos que son una extensión de la estructura. Sobre todo, con ANYSTRUCTURE es posible indicar cualquier tipo de trayectoria del objeto.
- Si se indica OBJECT OF, entonces los argumentos pueden ser todos estos objetos que son alcanzables a través de la ruta de objeto y que corresponden a la definición. Sobre todo con OBJECT OF ANYCLASS es posible indicar cualquier tipo de ruta de objeto. De manera similar a las referencias a otros objetos (véase el capítulo 2.6.3: Atributos de referencia) es posible definir la superclase admisible y restricciones eventuales y especializarse en las extensiones.
- Si se indica OBJECTS OF, los argumentos pueden ser conjuntos de objetos de una clase. Todos los objetos del conjunto deben cumplir con los requisitos establecidos con el argumento formal sobre la base de la descripción del modelo (es decir, el requisito establecido con el argumento formal no actúa como un filtro posterior).
- Si se indica ENUMVAL los argumentos pueden ser atributos o constantes que designan una hoja de cualquier tipo de enumeración (véase el capítulo 2.8.2: Enumeraciones).
- Si se indica ENUMTREEVA, los argumentos pueden ser atributos o constantes que designan un nodo o una hoja de cualquier tipo de enumeración.

Reglas de Sintaxis:

```
FunctionDef = 'FUNCTION' Function-Name
                '(' Argument-Name ':' ArgumentType
                { ';' Argument-Name ':' ArgumentType } ')'
                ':' ArgumentType [ Explanation ] ';'.

ArgumentType = ( AttrTypeDef
               | ( 'OBJECT' | 'OBJECTS' )
                     'OF' ( RestrictedClassOrAssRef | ViewRef )
               | 'ENUMVAL'
               | 'ENUMTREEVAL' ).
```

Se han definido las siguientes funciones estándar:

```
FUNCTION myClass (Object: ANYSTRUCTURE): STRUCTURE;
```

Suministra la clase del objeto.

```
FUNCTION isSubClass (potSubClass: STRUCTURE; potSuperClass: STRUCTURE): BOOLEAN;
```

Suministra verdadero si la clase del primer argumento corresponde a la clase o subclase del segundo argumento.

```
FUNCTION isOfClass (Object: ANYSTRUCTURE; Class: STRUCTURE): BOOLEAN;
```

Suministra verdadero si el objeto del primer argumento pertenece a la clase o a una subclase del segundo argumento.

```
FUNCTION elementCount (bag: BAG OF ANYSTRUCTURE): NUMERIC;
```

Suministra el número de elementos contenidos en la bolsa (o en la lista).

```
FUNCTION objectCount (Objects: OBJECTS OF ANYCLASS): NUMERIC;
```

Suministra el número de objetos pertenecientes al conjunto dado de objetos.

```
FUNCTION len (TextVal: TEXT): NUMERIC;
FUNCTION lenM (TextVal: MTEXT): NUMERIC;
```

Suministra la longitud del texto en términos de su número de símbolos.

```
FUNCTION trim (TextVal: TEXT): TEXT;
FUNCTION trimM (TextVal: MTEXT): MTEXT;
```

Suministra el texto sin espacios en blanco ni al principio ni al final.

```
FUNCTION isEnumSubVal (SubVal: ENUMTREEVAL; NodeVal: ENUMTREEVAL): BOOLEAN;
```

Suministra verdadero si SubVal es un subelemento, esto es, un subnodo o una hoja del nodo NodeVal.

```
FUNCTION inEnumRange (Enum: ENUMVAL;
                      MinVal: ENUMTREEVAL;
                      MaxVal: ENUMTREEVAL): BOOLEAN;
```

Suministra verdadero, si la enumeración pertenece a ENUM, se ordenan y se colocan en el rango de MinVal a MaxVal. Los subelementos de MinVal o MaxVal se considera que también están dentro.

```
FUNCTION convertUnit (from: NUMERIC): NUMERIC;
```

Convierte el valor numérico del parámetro "from" en el valor numérico de retorno y toma en consideración las unidades que están vinculados con el parámetro y con la aplicación del valor del resultado (normalmente con el atributo al cual se asigna el resultado). Esta función sólo se puede utilizar si los argumentos de "from" son compatibles con los del parámetro de retorno, es decir, si sus unidades derivan de una unidad común.

```
FUNCTION areAreas (Objects: OBJECTS OF ANYCLASS;
                   SurfaceBag: ATTRIBUTE OF @ Objects
                                     RESTRICTION (BAG OF ANYSTRUCTURE);
                   SurfaceAttr: ATTRIBUTE OF @ SurfaceBag
                                      RESTRICTION (SURFACE)): BOOLEAN;
```

Comprueba si las superficies forman un mosaico de acuerdo con el conjunto de objetos (primer parámetro) y el atributo (tercer parámetro). Si las superficies son una parte directa de la clase de objeto, entonces UNDEFINED debe indicarse para SurfaceBag, en todos los demás casos debe indicarse el camino que lleva al atributo *structure* con la estructura que contiene el atributo *surface*.

2.15 Vistas.

Las vistas son clases y estructuras cuyos objetos no son originales, sino virtuales, ya que se han derivado de los objetos de otras vistas o clases, o estructuras. Entre otras cosas, las vistas se utilizan para formular conceptos básicos para los gráficos y constricciones especiales. Otra aplicación consiste en transmitir datos en su forma derivada, sobre todo simplificada, a los sistemas de recepción.

Las vistas sólo se transfieren si han sido definidos en un VIEW TOPIC. En este caso, la transmisión tiene lugar de la misma manera que la transferencia completa ((palabra clave FULL) de las clases normales, posteriormente el receptor de datos (ver el capítulo 3: Transferencia secuencial) no necesita preocuparse de cómo se han creado los objetos (virtuales). Las vistas también pueden ser excluidas explícitamente de cualquier transferencia (TRANSIENT), si sólo son de importancia local, es decir, si sólo sirven como base

para otras vistas. Se excluye explícitamente toda la transferencia incremental de vistas, ya que no se puede asignar ninguna identificación de objeto a objetos de vista.

Las vistas pueden ser abstractas (ABSTRACT) o concretas. Las concretas también pueden fundarse en bases abstractas. Sólo es posible aproximarse a los atributos básicos que son concretos. Si este no es el caso, la propia vista debe ser declarada abstracta.

Las vistas también se pueden extender (EXTENDED o EXTENDS). Sin embargo, es imposible alterar la definición de la formación. Gracias a la extensión es posible concebir extensiones de vistas, clases y estructuras que sirvan de base a la vista de tal manera que se puedan formular más selecciones, atributos y limitaciones.

Reglas de Sintaxis:

```
ViewDef = 'VIEW' View-Name
            Properties<ABSTRACT,EXTENDED,FINAL,TRANSIENT>
            [ FormationDef | 'EXTENDS' ViewRef ]
                { BaseExtensionDef }
                { Selection }
            '='
            [ ViewAttributes ]
            { ConstraintDef }
          'END' View-Name ';'.

ViewRef = [ Model-Name '.' [ Topic-Name '.' ] ] View-Name.
```

Por medio de la definición de formación (FormationDef) de una vista, se define cómo y sobre qué base se pueden formar los objetos virtuales de una vista.

La proyección de la vista (palabra clave PROJECTION OF) es la forma más simple de una vista. Permite la visualización de la superclase (clase, estructura o vista) de forma alterada (por ejemplo, los atributos sólo en parte o en orden alterado).

Con una unión de tipo “*join*” (palabra clave JOIN OF) se genera el producto cartesiano (o producto cruzado) de las súperclases (clase o vista), es decir, hay tantos objetos de la clase resultante, como combinaciones de objetos de las diferentes superclases. También es posible definir las llamadas "uniones externas" (“*outer join*”), es decir, las combinaciones de objetos de la primera superclase con objetos vacíos (prácticamente inexistentes) de otras superclases (indicación "(OR NULL)"). Tales objetos vacíos se añaden si no se encuentra ningún objeto de la otra clase deseada que pueda corresponder a una cierta combinación de objetos precedentes. Todos los atributos del objeto vacío están sin definir. Por lo tanto, los atributos de la vista correspondiente pueden no ser obligatorios.

El uso de la unión (palabra clave UNION OF) permite la fusión de diferentes súperclases en una sola clase. Por lo general, todos los atributos de las diferentes súperclases se asignan a uno de los atributos de la clase fusionada. El tipo de atributo de la superclase debe ser compatible con el tipo de atributo de la vista de unión (del mismo tipo o del de una de sus extensiones).

Mediante una agregación (palabra clave AGGREGATION OF) es posible combinar en una instancia todas las instancias de un conjunto básico o aquellas cuya combinación requerida de atributos es idéntica. Dentro de la vista de agregación, el atributo implícito AGGREGATES (ver el capítulo 2.13: Expresiones) hace disponible el conjunto correspondiente de objetos originales en forma de BAG. Este atributo implícito no pertenece a los atributos reales de la vista y, por lo tanto, no será transferido, aunque se requiera una transferencia. Por ejemplo, se puede asignar al atributo correspondiente de la vista de agregación o en forma de argumento, que pueda transportarse a una función.

Por medio de la inspección (palabra clave INSPECTION OF) obtenemos el conjunto de todos los elementos de la estructura (con BAG OF o LIST OF o definido de acuerdo con polilínea, superficie o mosaico) que pertenece a un atributo subestructura (directo o indirecto) de una clase de objeto.

Cualquier inspección normal de un mosaico, respectivamente, la inspección de un atributo de la superficie suministrará los límites de todas las zonas o superficies de esta clase (estructura SurfaceBoundary). Basándose en los atributos de línea, las cadenas de línea de cada límite (estructura SurfaceEdge) se forman de tal manera que hacen que su número sea mínimo. Las cadenas de línea consecutivas, con los mismos atributos de línea, se combinan en una cadena de línea. Especialmente si no se han definido atributos de línea, resultará una cadena de una sola línea. Si intenta realizar una inspección adicional de las líneas de atributo, obtendría todas las cadenas de línea (estructura SurfaceEdge). Sin embargo de esta manera van a aparecer dos veces en un mosaico (una por cada área del objeto en cuestión).

Por medio de la inspección de un mosaico (palabra clave AREA INSPECTION OF) obtendrá exactamente una vez las cadenas de línea de los bordes de todas las áreas pertenecientes al mosaico (en forma de estructura SurfaceEdge). Las dos áreas que están bordeadas por una cadena de línea común se pueden referir a THISAREA o a THATAREA (véase el capítulo 2.13 Expresiones). Al igual que con una línea de inspección normal, las cadenas (estructura SurfaceEdge) se entregarán tan condensadas como sea posible.

```
STRUCTURE SurfaceEdge =
  Geometry: DIRECTED POLYLINE;
  LineAttrs: ANYSTRUCTURE;
END SurfaceEdge;

STRUCTURE SurfaceBoundary =
  Líneas LIST OF SurfaceEdge;
END SurfaceBoundary;
```

En la estructura como se indica debajo de la inspección POLILINEA) suministrará todos los segmentos (LineSegments) que forman los objetos de cadena de línea de esta clase:

```
STRUCTURE LineGeometry =
  Segments: LIST OF LineSegment;
MANDATORY CONSTRAINT isOfClass (Segments[FIRST],INTERLIS.StartSegment);
END LineGeometry;
```

En otras palabras, el primer segmento de curva representa un elemento denominado StartSegment con longitud 0, y todos los demás, llamados LineSegments, son líneas rectas, arcos o segmentos de curva de algún otro tipo, de acuerdo con la definición establecida en la estructura.

INTERLIS sólo se aspira a la descripción conceptual de una vista. Se omite expresamente para apoyar una realización eficaz de las vistas. Así, la generación de vista es parte de un grado especial de conformidad.

Reglas de Sintaxis:

```
FormationDef = ( Projection
               | Join
               | Union
               | Aggregation
               | Inspection ) ';'.

Projection = 'PROJECTION' 'OF' RenamedViewableRef.

Join = 'JOIN' 'OF' RenamedViewableRef
         (* ',' RenamedViewableRef
              [ '(' 'OR' 'NULL' ')' ] *).

Union = 'UNION' 'OF' RenamedViewableRef
```

```
            (* ',' RenamedViewableRef *).

  Aggregation = 'AGGREGATION' 'OF' RenamedViewableRef
                  ( 'ALL' | 'EQUAL' '(' UniqueEl ')' ).

  Inspection = [ 'AREA' ] 'INSPECTION' 'OF' RenamedViewableRef
                   '->' StructureOrLineAttribute-Name
                        { '->' StructureOrLineAttribute-Name }.
```

Todas las vistas básicas empleadas en una vista reciben un nombre dentro de la vista en uso, pueden ser referidos bajo este nombre. Este nombre también corresponde a la vista básica, siempre que no se cambie el nombre por medio de una base de definición de nombre (local) explícita. Por encima de todo el cambio de nombre se hace necesario cuando se definen las uniones que se refieren repetidamente a la misma superclase.

Reglas de Sintaxis:

```
  RenamedViewableRef = [ Base-Name '~' ] ViewableRef.

  ViewableRef = [ Model-Name '.' [ Topic-Name '.' ] ]
                ( Structure-Name
                | Class-Name
                | Association-Name
                | View-Name ).
```

Si dentro de una vista o la extensión de una vista se consideran extensiones de super clases, permitiendo así la formulación de otros atributos, selecciones o restricciones, debe aparecer una definición de extensión correspondiente (BaseExtensionDef). Procede de una vista básica ya definida y, a su vez, describe sus extensiones (que deben ser extensiones de anteriores vistas básicas) como vistas básicas. Si dicha extensión de vista se emplea en expresiones, se da el valor "UNDEFINED", si el objeto básico que pertenece al objeto virtual no coincide con esta extensión de vista.

Regla de sintaxis:

```
  BaseExtensionDef = 'BASE' Base-Name 'EXTENDED' 'BY'
                       RenamedViewableRef { ',' RenamedViewableRef }.
```

Mediante constricciones (palabra clave WHERE) es posible aplicar más restricciones al conjunto de objetos de vista definidos por la definición de la formación.

Regla de sintaxis:

```
  Selection = 'WHERE' Logical-Expression ';'.
```

En cuanto a los atributos (y por lo tanto las vistas del receptor) y las restricciones, las vistas principales se construyen de la misma manera que las clases y las estructuras. Con el fin de facilitar el proceso de escritura, se ofrece además la posibilidad de transferir todos los atributos de una vista-base en el mismo orden (ALL OF). Sin embargo, esto no tendría sentido en las unionesy en las inspecciones de AREA y, en consecuencia, es inadmisible.

Regla de sintaxis:

```
  ViewAttributes = [ 'ATTRIBUTE' ]
                       { 'ALL' 'OF' Base-Name ';'
                       | AttributeDef
                       | Attribute-Name
                           Properties <ABSTRACT,EXTENDED,FINAL,TRANSIENT>
                             ':=' Factor ';' }.
```

En aquellos casos simples en los que se transfiere un atributo desde la vista básica, basta indicar el nombre del atributo y la asignación al atributo básico. Dichas definiciones son siempre definitivas, lo que significa que no pueden extenderse más.

En las uniones es obligatorio indicar, para cada atributo, de qué atributos de la clase básica se ha derivado. Sin embargo, un atributo no debe referirse a todas las clases básicas siempre y cuando el tipo de atributo permita valores no definidos. Se considera como indefinido para todos los objetos básicos que faltan.

El siguiente ejemplo demuestra cómo se puede describir una vista que permita la definición de una relación apropiada mediante la palabra reservada FROM (véase el capítulo 2.7: Relaciones propiamente dichas).

```
DOMINIO
  CHSurface = ... ;

FUNCTION Intersect (Surface1: CHSurface;
                    Surface2: CHSurface): BOOLEAN;

CLASS A =
  A1: CHSurface;
END A;

CLASS B =
  b1: CHSurface;
END B;

VIEW ABIntersection
  JOIN OF A,B;
  WHERE Intersect (A.a1,B.b1);
  =
END ABIntersection;

ASSOCIATION IntersectedAB
  DERIVED FROM ABIntersection =
  ARole -- A := ABIntersection -> A;
  BRole -- B := ABIntersection -> B;
END IntersectedAB;
```

2.16 Descripciones gráficas

Una descripción gráfica consiste en definiciones de gráficos que se basan siempre en una vista o una clase (palabra clave BASED ON). Por medio de una definición gráfica se intenta conceptualmente asignar un símbolo gráfico (punto, línea, símbolo de área, etiqueta de texto) a través de una o varias reglas de dibujo (regla DrawingRule) a cada objeto de esta vista o clase - a menos que tal objeto haya sido descartado por una selección específica (palabra clave WHERE) - que se refiere a la vista o la clase. Así se crean uno o varios objetos gráficos, que a su vez producirán la representación respectiva (ver figura 5). Para ello, cada regla de dibujo debe seleccionar un símbolo gráfico (con nombre de metaobjeto) y determinar los argumentos para los parámetros correspondientes.

Entre paréntesis (regla Properties) se pueden definir las características de herencia. Cada vez que una descripción gráfica es abstracta, no puede producir objetos de símbolo. La extensión de un gráfico debe estar basada en la misma clase que el gráfico base (BASED ON) o en una de sus extensiones.

La regla de dibujo se identifica por un nombre que es inequívoco dentro de la descripción gráfica, con el fin de ser rastreado en las extensiones y posteriormente refinado. (Nota: en el sentido de la especialización se va refinando también valores de los parámetros adicionales). Cuando hay extensiones a una regla de dibujo en existencia (en la extensión de descripciones gráficas), estas no crean nuevos objetos gráficos, sino que solo influirán en los parámetros de símbolos del objeto gráfico determinados por la definición básica. Es admisible definir varias extensiones a una definición gráfica. Todos ellos son evaluados (en el

orden de su definición). Esto es especialmente útil cuando se planifican varias pilas de extensión para diversos aspectos (por ejemplo, varias reglas de dibujo). Posteriormente, se determinan los diferentes parámetros de símbolo. Esta definición puede ser producida en varias etapas. Es el valor definido último que se aplica a cada parámetro respectivo. En primer lugar, se evalúa la definición primaria, y sólo entonces las posibles extensiones. Además, es posible vincular asignaciones de parámetros a una constricción (regla CondSignParamAssignment), es decir, la asignación sólo está en vigor si se cumple la restricción. Si no se cumple la restricción de selección, las subextensiones eventuales ya no se tienen en cuenta. Dentro de las reglas de asignación (regla CondSignParamAssignment) el espacio de nombres de la clase básica o de la vista básica es válido para todos los nombres de atributos o roles, para los nombres de metaobjetos, de funciones y de parámetros de tiempo de ejecución, este es el espacio de nombres de la definición gráfica que es válido.

Tan pronto como las reglas de dibujo son concretas, debemos definir a qué clase pertenecen los símbolos gráficos que se van a asignar. En las extensiones de reglas de dibujo esta clase de símbolos gráficos debe ser reemplazada por una clase que es una extensión de la primera. Principalmente la clase "responsable" de símbolos gráficos es la clase a la que pertenece el objeto de símbolo gráfico asignado (un metaobjeto). Los valores concretos deben asignarse a los parámetros introducidos en la clase "responsable". Si los parámetros indicados corresponden a una clase extendida de símbolos gráficos, se convierte en la clase "responsable", siempre que esté de acuerdo con la clase del símbolo gráfico de la regla de dibujo o sea una de sus extensiones.

En las restricciones mencionadas anteriormente, los atributos de objeto (véase AttributePath en la regla SignParamAssignment) también se pueden comparar con parámetros de tiempo de ejecución (véase capítulo 2.11 Parámetros de tiempo de ejecución). Los parámetros de tiempo de ejecución que son importantes para los gráficos (por ejemplo, la escala del gráfico requerido) se definen típicamente en los modelos de simbología, ya que describen - de la misma manera que un parámetro de símbolos - las competencias gráficas que se esperan de un sistema. Para el parámetro de un símbolo gráfico que requiere un metaobjeto, debe indicarse una referencia de metaobjeto (ver capítulo 2.10.).

El valor del parámetro ordinario de un símbolo gráfico se indica en términos de una constante o referencia a un atributo de objeto (ver factor en la regla SignParamAssignment). Por lo tanto, siempre nos referimos al atributo de un objeto de la clase básica o vista básica que se ha especificado mediante BASED ON.

Como la representación a menudo depende de atributos que se han definido por medio de las enumeraciones, está disponible una construcción especial para este propósito: el dominio de enumeración. Un dominio de enumeración es un único nodo del árbol del tipo de enumeración o un intervalo entre nodos definidos por dos nodos del mismo nivel. Las definiciones de intervalos sólo son admisibles cuando se trata de tipos de enumeración ordenados. Si el valor del atributo se encuentra dentro del dominio de la enumeración indicada, se establece el valor del parámetro correspondiente. Los símbolos concretos son el resultado del modelo de simbología. Allí se definen todas las clases de símbolos más los parámetros de tiempo de ejecución necesarios (palabra clave PARAMETER) para su aplicación. Es admisible definir tipos de datos numéricos sólo de forma abstracta.

Reglas de Sintaxis:

```
GraphicDef = 'GRAPHIC' Graphic-Name Properties<ABSTRACT,FINAL>
               [ 'EXTENDS' GraphicRef ]
               [ 'BASED' 'ON' ViewableRef ] '='
               { Selection }
               { DrawingRule }
             'END' Graphic-Name ';'.

GraphicRef = [ Model-Name '.' [ Topic-Name '.' ] ] Graphic-Name.

DrawingRule = DrawingRule-Name Properties<ABSTRACT,EXTENDED,FINAL>
```

```
                [ 'OF' Sign-ClassRef ]
                    ':' CondSignParamAssignment
                        { ',' CondSignParamAssignment } ';'.

CondSignParamAssignment = [ 'WHERE' Logical-Expression ]
                          '(' SignParamAssignment
                              { ';' SignParamAssignment } ')'.

SignParamAssignment = SignParameter-Name
                        ':=' ( '{' MetaObjectRef '}'
                             | Factor
                             | 'ACCORDING' Enum-AttributePath
                                 '(' EnumAssignment
                                     { ',' EnumAssignment } ')' ).

EnumAssignment = ( '{' MetaObjectRef '}' | Constant )
                   'WHEN' 'IN' EnumRange.

EnumRange = EnumerationConst [ '..' EnumerationConst ].
```

Para la aplicación de los modelos de simbología de la SIGN clase ha sido predefinido por INTERLIS

```
CLASS SIGN (ABSTRACT) EXTENDS METAOBJECT =
  PARAMETER
    Sign: METAOBJECT;
END SIGN;
```

Para las clases de símbolos concretas, esta clase básica tiene que ser extendida, definiendo, por una parte, datos concretos sobre los otros parámetros.

El siguiente ejemplo describe cómo se definen los gráficos correspondientes (símbolos de puntos y etiquetas de texto) desde una clase de punto con coordenadas, cadena y una enumeración como atributo.

El modelo de simbología se define de la siguiente manera:

```
CONTRACTED SYMBOLOGY MODEL SimpleSignsSymbology (en) AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  DOMINIO
    S_COORD2 (ABSTRACT) = COORD NUMERIC, NUMERIC;


  TOPIC SignsTopic =

    CLASS Symbol EXTENDS INTERLIS.SIGN =
      PARAMETER
        POS MANDATORY S_COORD2;
    END Symbol;

    CLASS Textlabel EXTENDS INTERLIS.SIGN =
      PARAMETER
        Pos: MANDATORY S_COORD2;
        Text: MANDATORY TEXT;
    END Textlabel;

  END SignsTopic;

END SimpleSignsSymbology.
```

Además de este modelo de simbología, se supone que los modelos de (símbolo) objetos concretos han sido listados y archivados bajo el nombre de la biblioteca de símbolos (es decir, el nombre del contenedor) SimpleSignsBasket. Los objetos de símbolo listados (símbolo de clase) se denominan punto-símbolo,

cuadrado-símbolo, círculo-símbolo, y los tipos de fuente (etiqueta de texto de clase) etiquetado1 y etiquetado2.

```
MODEL DataModel (en) AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  DOMINIO
    LCoord = COORD
      0,0000.000 .. 200.000 [m],
      0,0000.000 .. 200.000 [m],
      ROTATION 2 -> 1;


  TOPIC DotTopic =

    DOMINIO
      DotType = (Stone
                  (large,
                   small),
                 Bolt,
                 Pipe,
                 Cross,
                 nonmaterialized) ORDERED;

    CLASS Dot =
      Posicion: LCoord;      !! LCoord ser un dominio de valores de coordenadas
      Type: DotType;
      DotName: TEXT*12;
    END Dot;

  END DotTopic;

END DataModel.


CONTRACTED MODEL SimpleGraphic (en) AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  IMPORTS DataModel;
  IMPORTS SimpleSignsSymbology;

  SIGN CONTENEDOR SimpleSignsBasket ~ SimpleSignsSymbology.SignsTopic;


  TOPIC DotGraphicsTopic =
    DEPENDS ON DataModel.DotTopic;

    GRAPHIC SimpleDotGraphic BASED ON DataModel.DotTopic.Dot =

      Symbol OF SimpleSignsSymbology.SignsTopic.Symbol: (
        Sign := {DotSymbol};
        Pos := Position
      );

    END SimpleDotGraphic;

  END DotGraphicsTopic;

END SimpleGraphic.
```

Por medio de este gráfico (basado en el modelo de simbología SimpleSignsSymbology y la representación gráfica SimpleGraphic) para todos los puntos de la clase, se dibujan símbolos de puntos simples.

También es deseable un gráfico mejorado. Esta mejora se puede hacer en diferentes aspectos, por ejemplo:

- Se desean símbolos adicionales (símbolos de puntos, símbolos cruzados, símbolos de triángulos). Esto requiere una biblioteca de símbolos suplementarios con nombre de SimpleSignsPlusBasket. Dado que se trata de una extensión de la biblioteca SimpleSignsBasket, los objetos de símbolo (o metaobjetos) se buscarán en ambas bibliotecas. Si la biblioteca SimpleSignBasket se extiende directamente (EXTENDED), entonces para todos los gráficos creados con GraphicPlus dentro del modelo - incluyendo aquellos que han sido heredados del modelo SimpleGraphic - los símbolos se buscarán primero en la biblioteca extendida y sólo después en la Biblioteca base SimpleSignsBasket.
- Los símbolos deben ser escalables, permitiendo así la creación de cuadrados pequeños y grandes con el mismo símbolo de punto. Esto requiere un modelo de simbología extendida que contenga un parámetro que defina la escala de los símbolos. Dado que las clases de símbolos no presentan ningún atributo adicional, no es obligatorio que existan bibliotecas correspondientes.
- Dependiendo del tipo de punto, deben dibujarse varios símbolos de puntos: piedras como cuadrados grandes o pequeños, pernos como círculos y cruces y tubos con el símbolo de la cruz. El símbolo de punto real puede derivarse directamente del tipo de punto. El factor de escala para cuadrados pequeños para la representación de piedras pequeñas se obtiene mediante una asignación adicional. Los puntos no materializados siguen siendo puntos simples; Por lo tanto, en este caso no se produce ninguna nueva asignación.

```
CONTRACTED SYMBOLOGY MODEL ScalableSignsSymbology (en) AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  IMPORTS SimpleSignsSymbology;


  TOPIC ScalableSignsTopic EXTENDS SimpleSignsSymbology.SignsTopic =

    CLASS Symbol (EXTENDED) =
      PARAMETER
        ScaleFactor: 0,1 10.0;  !! Default 1.0
    END Symbol;

  END ScalableSignsTopic;

END ScalableSignsSymbology.


CONTRACTED MODEL GraphicPlus (en) AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  IMPORTS SimpleGraphic;
  IMPORTS SimpleSignsSymbology;
  IMPORTS ScalableSignsSymbology;

  SIGN CONTENEDOR SimpleSignsPlusBasket EXTENDS
    SimpleGrafik.SimpleSignsBasket ~ ScalableSignsSymbology.ScalableSignsTopic;


  TOPIC DotGraphicsPlusTop EXTENDS SimpleGraphic.DotGraphicsTopic =

    GRAPHIC DotGraphicPlus EXTENDS SimpleDotGraphic =

      Symbol (EXTENDED) OF ScalableSignsSymbology.ScalableSignsTopic.Symbol: (
        Sign := ACCORDING Art (
          {SquareSymbol} WHEN IN #Stone,
          {CircleSymbol} WHEN IN #Bolt,
          {CrossSymbol} WHEN IN #Pipe .. #Cross
        )
      ),
      WHERE Type == #Stone.small (
          ScaleFactor := 0.5
```

```
        );

        Text OF SimpleSignsSymbology.Signs.Textlabel: (
          Sign := {Labeling1};
          Pos := Position;
          Texto: = DotName
        );

      END DotGraphicPlus;

    END DotGraphicsPlusTop;

  END GraphicPlus.
```

3 transferencia secuencial

3.1 Introducción

En este capítulo se describe el servicio de transferencia secuencial en INTERLIS. Permite el intercambio entre almacenes de datos de distintos sistemas y de forma neutra respecto a estos. Nuestro servicio de transferencia INTERLIS soporta el intercambio completo, así como el intercambio incremental (o diferencial) de almacenes de datos (replicación). Este servicio de transferencia es aplicable a todos los modelos INTERLIS. Así, es posible transferir datos (modelo de datos) y objetos de símbolo (modelos de simbología) por medio del mismo mecanismo.

En la actualidad, el servicio de transferencia de INTERLIS se define como un intercambio de archivos XML (www.w3.org/XML/). Para una utilización más extensa de estos archivos INTERLIS/XML es posible también, entre otros, crear documentos de esquema XML (www.w3.org/XML/Schema). No obstante, es concebible que en el futuro se definan servicios de transferencia para INTERLIS adicionales (por ejemplo, basados en servicios web o COBRA). Por esta razón, la descripción de nuestro servicio de transferencia INTERLIS ha sido subdividida en los párrafos “Reglas Generales” y “Codificación XML”. Las reglas generales se aplican a cada servicio de transferencia INTERLIS secuencial, independientemente de la codificación o transmisión concreta. Las reglas establecidas en Codificación XML se aplican especialmente a los archivos de transferencia con formato XML.

3.2 Reglas generales para la transferencia secuencial

3.2.1 Origen en el modelo de datos

Cada transferencia de INTERLIS puede tener origen en el modelo de datos correspondiente mediante la aplicación de reglas (transferencia de datos basada en modelos).

3.2.2 Lectura de modelos extendidos

Una transferencia de INTERLIS está siempre estructurada de tal manera que permita que un programa de lectura previsto y configurado para un modelo de datos específico, lea también datos de extensiones de este modelo de datos sin ningún conocimiento de las definiciones del modelo extendido.

3.2.3 Organización de una transferencia: Preliminares

Una transferencia de INTERLIS es un flujo secuencial de objetos. El flujo de objetos se subdivide en preliminares y dominio de datos.

En los preliminares de apertura al menos se incluye la siguiente información para la transferencia:

- Indicación del número versión actual de INTERLIS (véase el capítulo: 2.3 regla principal).
- Referencia al modelo de datos correspondiente (o modelos).
- Indicación del emisor (SENDER).

Dentro de los preliminares, hay una opción para dar información relativa al autor de la estructura de la identificación de objetos.

Los preliminares pueden contener comentarios (opcional).

La organización del dominio de datos se describirá con más precisión en los siguientes párrafos.

3.2.4 Objetos transferibles

Dentro de los objetos de dominio de datos (es decir, instancias de objeto) de clases concretas, se pueden transferir relaciones, vistas y definiciones gráficas. Dentro de la transferencia, los objetos de las vistas se tratan de la misma manera que los objetos de clases concretas. En la actualidad, la transferencia incremental de vistas aún no es posible. Los objetos de vistas sólo se transfieren siempre que las vistas

pertinentes hayan sido declaradas dentro de un VIEW TOPIC, de lo contrario no se transmitirán. Además, las vistas no se transferirán si se han marcado como TRANSIENT.

3.2.5 Orden de los objetos dentro del dominio de datos

Un dominio de datos consiste en una serie de contenedores (instancias de tema (topic)). Los contenedores sólo pueden ser transferidas como un todo. Con las transferencias incrementales sólo se producen alteraciones, esto es, los objetos suprimidos serán transferidos. Sin embargo, incluso con transferencias incrementales conceptualmente es el contenedor entero el que se transfiere junto con la historia previa. En principio es posible que una transferencia contenga un contenedor proporcionado por diferentes modelos. A su vez, cada contenedor contiene todos sus objetos. Dentro de la transferencia se permite cualquier orden de objetos, sobre todo los objetos no necesitan en ningún caso ser ordenados de acuerdo a la relación o agrupados en clases dentro de un contenedor (en contraposición a INTERLIS 1). Las cestas vacías no necesitan ser transferidas.

3.2.6 Codificación de los objetos

Dentro del flujo del objeto, cada contenedor y cada objeto recibe una identificación. La identificación del contenedor debe ser un identificador de objeto general y estable (OID). El identificador de contenedores y objetos debe ser inequívoco a lo largo de toda la transferencia. Además, con cada objeto se suministra una identificación de contendor dentro del cual se ha creado originalmente el objeto (cesta original). Con transferencias incrementales, así como con las transferencias iniciales, la identificación del contenedor y del objeto debe ser un identificador de objeto general y estable (véase el apéndice D Organización de los identificadores de objeto (OID)).

Todos los atributos del objeto (incluyendo COORD, SURFACE, AREA, POLYLINE, STRUCTURE, BAG OF, LIST OF, etc.) se memorizan directamente con el objeto. Los atributos del tipo AREA se codifican como atributos del tipo SURFACE. Los atributos del tipo BAG se codifican como atributos del tipo LIST. STRUCTURE se codifica como LIST {1}.

Para la transmisión de valores de atributo sólo están disponibles los símbolos imprimibles del conjunto de caracteres US-ASCII (32 a 126) y los símbolos de acuerdo con la tabla de símbolos del apéndice B.

3.2.7 Tipos de transferencia

Junto con cada contenedor se debe suministrar la siguiente información:

- Detalles relativos al tipo (KIND) de transferencia: FULL, INITIAL o UPDATE.
- Detalles relativos al estado inicial y el estado final (STARTSTATE y ENDSTATE) de la transferencia (sólo con los tipos INITIAL (ENDSTATE) o UPDATE (STARTSTATE y ENDSTATE)).
- Detalles relativos a la consistencia de su contenido: COMPLETE, INCOMPLETE, INCONSISTENT, ADAPTED.

Es admisible tener contenedores con diferentes tipos de transferencia (FULL, INITIAL, UPDATE) dentro de la misma transferencia. Los diversos tipos de transferencia tienen el siguiente significado:

- FULL – transferencia completa. Al recibir un contenedor completo, el receptor debe inicializar primero un nuevo contenedor e insertar todos los objetos en él mediante INSERT. FULL no es apropiado como base para transferencias ya que las identificaciones de objeto son válidas solamente para esta transferencia. Los archivos de transferencia de acuerdo con INTERLIS 1 corresponden a FULL. Dentro del tipo de transferencia FULL sólo puede ocurrir la operación INSERT.
- INITIAL - Transferencia primaria. Corresponde al tipo de transferencia FULL con la única diferencia de que tanto el contenedor como los objetos contenidos deben tener OID generales y estables. Al igual que dentro de la transferencia de tipo FULL, sólo puede darse la operación INSERT.

- UPDATE - Transferencia adicional. Un contenedor UPDATE contiene objetos con operaciones INSERT, UPDATE o DELETE. Todos los objetos y el contenedor tienen la característica de un OID general y estable. Los contenedores UPDATE sólo pueden ser procesadas por el sistema de destino, si el estado de inicio del contenedor ya se ha recibido con INITIAL o UPDATE.

Además, las siguientes reglas de transferencia son de aplicación al tipo de transferencia UPDATE:

- El sistema receptor puede continuar suponiendo que después de procesar completamente todos los datos de un contenedor UPDATE, se ha restablecido uno coherente, es decir, un contenedor UPDATE transfiere un contenedor desde un estado inicial consistente a un estado final consistente.
- Un contenedor UPDATE no es consistente en si mismo, ya que en la mayoría de los casos las referencias sólo pueden resolverse junto con transferencias anteriores.

Además, para cada objeto debe indicarse una operación de transferencia correspondiente (véase el capítulo 1.4.5 Contenedores, replicación y transferencia de datos). Las operaciones INSERT, UPDATE y DELETE tienen el siguiente significado:

- La operación INSERT significa "insertar un nuevo objeto" (Insertar objeto).
- La operación UPDATE significa "actualizar valores de atributo de objeto" (actualización del objeto). Todos los atributos (no sólo aquellos que han sido alterados) deben ser transferidos.
- La operación DELETE significa "eliminar objetos" (eliminar objetos). Todos los atributos (no sólo los de OID) deben ser transferidos.

En muchos casos no es un conjunto completo de datos el que tiene que ser transferido, sino sólo una parte de el. Dependiendo de la elección de la sección, algunas geometrías (polilíneas y superficies) pueden estar incompletas. Para permitir esto sin tener que crear un modelo adicional, debe ser posible que los objetos (y por lo tanto el contenedor correspondiente) puedan ser señalados INCOMPLETE. De manera similar con la integración de varios contenedores en uno, pueden surgir situaciones en las que la coherencia de los datos sea violada o sólo se garantice alterando los datos más allá de un grado tolerable. En ambos casos, los objetos en cuestión (y por lo tanto todo el contenedor) deben ser marcados como INCONSISTENT o ADAPTED.

3.3 Codificación XML

3.3.1 Introducción

A diferencia de las reglas del capítulo 3.2: Reglas generales para la transferencia secuencial, las reglas bajo codificación XML sólo se aplican a los archivos de transferencia formateados de acuerdo con el estándar XML-0.1 (ver www.w3.org./XML). Para formalizar las reglas de derivación de los formatos de transferencia se usa la notación EBNF ya introducida en el capítulo 2.1: Sintaxis aplicada. Por lo tanto, las siguientes reglas ya están predefinidas:

```
XML-Any = any XML-elements of your chosing (well-formed XML).
XML-base64Binary = any binary data coded in Base64.
XML-String = any text without tags (including carriage return (#xD),
             line feed (#xA) and tabulator (#x9)).
XML-NormalizedString = any text limited to one line.
XML-ValueDelimiter = '"' | #x27 (simple hyphen ').
XML-Value = XML-ValueDelimiter XML-NormalizedString XML-ValueDelimiter.
XML-NcName = ( Letter | '_' ) { Letter | Digit | '_' | '-' | '.' }.
XML-ID = XML-ValueDelimiter [ XML-NcName ':' ] ( Letter | Digit | '_' )
         { Letter | Digit | '_' | '-' | '.' } XML-ValueDelimiter.
```

Si el valor ID es un OID general y estable, entonces el nombre del dominio OID (véase el capítulo 3.3.4.1: Información sobre la estructura de las identificaciones de objetos) debe indicarse como un prefijo (delante del ':') , A menos que se trate de una identificación de transferencia pura e inestable.

Para mejorar la legibilidad de cada regla de derivación individual se sugiere un uso adicional de las macros TAG y ETAG.

```
TAG ( Tagwert )
```

Genera una regla parcial EBNF de la forma:

```
'<%Tagwert%>'
```

```
TAG ( Tagwert, Attribut1, Attribut2, ...)
```

Genera una regla parcial EBNF de la forma:

```
'<%Tagwert%' %Attribut1% %Attribut2% etc. '>'
```

```
ETAG ( Tagwert )
```

Genera una regla parcial EBNF de la forma:

```
'</%Tagwert%>'
```

En cada caso, la secuencia %argument% tiene que ser reemplazada por el contenido actual del argumento.

Ejemplos:

```
TAG (DATASECTION)                    produces '<DATASECTION>'
TAG (HEADERSECTION,'VERSION="2.3"') produces '<HEADERSECTION' 'VERSION="2.3"' '>'
ETAG (DATASECTION)                   produces '</DATASECTION>
```

3.3.2 Codificación de símbolos

Los únicos símbolos disponibles para la codificación de cadenas XML-String, así como XML-NormalizedString, son signos ASCII 32 a 126, y los símbolos indicados en el apéndice B: Tabla de símbolos. Estos símbolos están codificados de acuerdo a la regla de codificación UTF-8 o como carácter de referencia XML (Character Reference), así como Entidad de referencia XML (Entity reference). Además, los símbolos especiales de XML '&', '<' y '>' deben codificarse de la siguiente manera::

- '&' Debe ser reemplazada por la secuencia "&
- '<' Debe ser reemplazado por la secuencia '<'
- '>' debe ser reemplazado por la secuencia '>'

En la tabla de símbolos del Apéndice B se encuentra un resumen completo de esta codificación de símbolos con todas las formas posibles de codificación por símbolo. Si hay disponibles varios formularios de codificación por símbolo, corresponde a un programa de escritura INTERLIS 2 seleccionar un formulario pertinente. Un programa de lectura INTERLIS 2 debe ser capaz de reconocer todas las formas de codificación. Nota: Se admiten varios formularios de codificación por símbolo para lograr la máxima compatibilidad con las herramientas XML existentes.

3.3.3 Estructura general de un archivo de transferencia

Un archivo de transferencia de INTERLIS está estructurado de acuerdo con la siguiente regla principal EBNF:

```
Transfer = '<?xml version="1.0" encoding="UTF-8" ?>'
           TAG ( TRANSFER, 'xmlns="http://www.interlis.ch/INTERLIS2.3"' )
               HeaderSection
               DataSection
           ETAG ( TRANSFER ).
```

La regla HeaderSection genera la sección de encabezado del archivo de transferencia y la regla DataSection genera la sección de datos.

En cualquier momento, un archivo de transferencia INTERLIS generado por la regla de transferencia también es un archivo de transferencia XML 1.0 bien formado. Así, en un archivo de transferencia INTERLIS, cualquier número de líneas de comentario en forma de

```
<!-- Comment -->
```

puede darse en lugares asignados por XML 1.0. Sin embargo, el contenido de estas líneas de comentario no puede ser interpretado por el software de transferencia. La codificación UTF-8 se aplica para la codificación de todos los símbolos del archivo de transferencia. Sin embargo, como estándar, sólo se puede utilizar el conjunto de símbolos que se muestra en la tabla de símbolos del apéndice B.

Los datos se transfieren como objetos XML. Los nombres de variables de objetos XML derivan de sus respectivos nombres de objeto en el modelo de datos INTERLIS. Para los modelos de datos traducidos (TRANSLATION OF) esto implica que los nombres de las etiquetas existen en el idioma traducido dentro de la transferencia (sin embargo, las entradas adicionales deben realizarse en la tabla de alias).

3.3.4 Sección de cabecera

Una sección de cabecera está estructurada de la siguiente manera:

```
HeaderSection = TAG ( HEADERSECTION,
                      'VERSION="2.3"',
                      'SENDER=' XML-Value )
                    Models
                    [ Alias ]
                    [ OidSpaces ]
                    [ Comment ]
                  ETAG (HEADERSECTION).

Models = TAG ( MODELS )
             (* Model *)
         ETAG ( MODELS ).

Model = TAG ( MODEL,
              'NAME=' XML-Value,
              'VERSION=' XML-Value,
              'URI=' XML-Value )
        ETAG ( MODEL ).

Alias = TAG ( ALIAS )
            { Entries }
        ETAG ( ALIAS ).

Entries = TAG ( ENTRIES,
                'FOR=' XML-Value )
              { Tagentry | Valentry | Delentry }
          ETAG ( ENTRIES ).

Tagentry = TAG ( TAGENTRY,
                 'FROM=' XML-Value, 'TO=' XML-Value )
           ETAG ( TAGENTRY ).

Valentry = TAG ( VALENTRY,
                 'TAG=' XML-Value,
                 'ATTR=' XML-Value,
                 'FROM=' XML-Value, 'TO=' XML-Value )
           ETAG ( VALENTRY ).
```

```
Delentry = TAG ( DELENTRY,
                 'TAG=' XML-Value
                 [ , 'ATTR=' XML-Value ] )
           ETAG ( DELENTRY ).

OidSpaces = TAG ( OIDSPACES )
                (* OidSpace *)
            ETAG ( OIDSPACES ).

OidSpace = TAG ( OIDSPACE,
                 'NAME=' XML-Value,
                 'OIDDOMAIN=' XML-Value )
           ETAG ( OIDSPACE ).

Comment = TAG ( COMMENT )
              XML-String
          ETAG ( COMMENT ).
```

En el elemento HeaderSection deben introducirse los siguientes valores (atributos XML):

- VERSION. Versión de la codificación de INTERLIS (actualmente 2.3)
- SENDER. Quien envía el conjunto de datos.

En el elemento “Models” deben ser listados todos los modelos de datos, si los datos relacionados con sus temas se producen. En NAME se transfiere el nombre del modelo de datos, bajo VERSION su versión.

En el elemento OidSpaces se indica la información relativa a la estructura de las identificaciones de objetos (véase el capítulo 3.3.4.1: Información sobre la estructura de las identificaciones de objetos).

En el elemento “Alias” se realizan todas las entradas que permiten la lectura polimorfa de un conjunto de datos (véase el capítulo 3.3.4.2: Significado y contenido de la tabla Alias). El elemento “Alias” es opcional. Sin embargo, si falta, la lectura de polimorfos sólo es posible si el elemento Alias puede ser suministrado de una manera diferente gracias a su identificador de esquema.

En “Comment” se puede agregar un comentario que describa la transferencia (opcional).

3.3.4.1 Información relativa a la estructura de las identificaciones de objetos

Si una unidad de transferencia contiene identificaciones de objetos generales, estables, los dominios Oid aplicados deben ser indicados en los preliminares con sus nombres calificados y equipados con una abreviatura. Cuando no hay definición de un dominio Oid dentro de los preliminares, entonces el archivo de transferencia no contiene identificaciones de objetos generales y estables.

3.3.4.2 Significado y contenido de la tabla Alias

El elemento Alias en HeaderSection es una tabla especial que permite a un programa de lectura leer modelos extendidos sin ningún conocimiento de las extensiones (la llamada lectura polimorfa). Dado que XML no conoce la herencia y, en consecuencia, no hay polimorfismo, la tabla de alias se utiliza para transmitir información adicional necesaria a un programa de lectura.

EL Alias debe contener una tabla de representación (elementos de entrada) por modelo de datos X que ocurre dentro de la transferencia. A través de las entradas xxxENTRY (TAGENTRY, VALENTRY, DELENTRY) se indica en cada tabla de relaciones qué objetos transferibles (es decir inherentes, o sus extensiones) pueden ocurrir en los contenedores del modelo de datos X (no puede ocurrir en el caso de entradas DELENTRY). En caso de existir traducción (TRANSLATION OF) del modelo de datos X dentro de la transferencia, todas las etiquetas del modelo de datos traducido deben introducirse con TAGENTRY o VALENTRY en la tabla de representación del modelo de datos X. Las tablas de representación de cada uno de los modelos de datos se debe organizar de tal manera que las tablas de la representación de los

modelos bajos se introducen antes de las tablas de la representación de la extensión, así como los modelos traducidos. Las entradas individuales de la tabla Alias tienen la siguiente significación:

- TAGENTRY. La etiqueta se introduce de acuerdo con el modelo original en el atributo FROM (por ejemplo, 'Canton.TCanton.K1'), en el atributo TO, siendo la etiqueta según el modelo considerado actualmente (por ejemplo, 'Federation.TFederation.B1') . Además, todas las etiquetas deben introducirse según el modelo actual. En este caso se introduce el mismo valor en el FROM, considerando la etiqueta TO. El TAGENTRY debe ser indicado para temas (topics) concretos, clases, estructuras, relaciones y definiciones gráficas, así comovistas transferibles.
- VALENTRY. Sólo el nombre del atributo se indica dentro del atributo ATTR (por ejemplo, 'Color'). Tanto el modelo como la clase que presenta este atributo se enumeran como nombres calificados dentro del atributo TAG. El valor según el modelo original se introduce en el atributo FROM (por ejemplo, "red.crimson"), en el atributo TO es la etiqueta según el modelo actual (por ejemplo, "rojo"). Todos los valores deben introducirse según el modelo actual. En este caso se indica el mismo valor para el parámetro FROM, considerando la etiqueta TO. VALENTRY debe indicarse para todos los tipos de enumeración.
- DELENTRY. En el atributo TAG indicamos la etiqueta que no puede ocurrir desde el punto de vista del modelo actual, pero que podría existir en extensiones. El DELENTRY debe ser indicado para temas concretos, clases, estructuras, relaciones y definiciones gráficas, así como para atributos de clases, estructuras y vistas transferibles. Si una clase entera (o estructura, relación o vista transferible) no puede existir desde el punto de vista del modelo actual, es admisible designar solamente la clase inexistente con DELENTRY. En este caso, los atributos de la clase (o estructura, relación o vista transferible) no se deben transferir con DELENTRY.
- Nota sobre las relaciones sin identidad propia: Las reglas anteriores también se aplican de manera análoga a las relaciones sin identidad propia. Por ejemplo, los valores de un atributo de enumeración se deben enumerar en la misma clase en la que se enumera la función (es decir, en class. role. attribute).
- En el siguiente ejemplo se ilustran una vez más las características de la tabla Alias.

La siguiente descripción de los datos:

```
MODEL Federation AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  CLASS B (ABSTRACT) =
  END B;


  TOPIC TFederation =

    CLASS B1 EXTENDS B =
      Color : (red, green, blue);
    END B1;

  END TFederation;

END Federation.


MODEL TFederation AT "http://www.interlis.ch/"
  VERSION "2005-06-16" TRANSLATION OF Federation ["2005-06-16"] =

  CLASS TB (ABSTRACT) =
  END TB;


  TOPIC TTFederation =

    CLASS TB1 EXTENDS TB =
```

```
        TColor : (tred, tgreen, tblue);
      END TB1;

    END TTFederation;

  END TFederation.


  MODEL Canton AT "http://www.interlis.ch/"
    VERSION "2005-06-16" =

    IMPORTS Federation;


    TOPIC TCanton EXTENDS Federation.TFederation =

      CLASS K (ABSTRACT) EXTENDS Federation.B =
      END K;

      CLASS K1 EXTENDS Federation.TFederation.B1 =
        Color (EXTENDED): (red (dark, crimson, light));
        TXT: TEXT*40;
      END K1;

      CLASS K2 =
        TXT: TEXT*40;
      END K2;

    END TCanton;

  END Canton.


  MODEL County AT "http://www.interlis.ch/"
    VERSION "2005-06-16" =

    IMPORTS Canton;


    TOPIC TCounty EXTENDS Canton.TCanton =
    END TCounty;

  END County.
```

conduce a la siguiente tabla Alias:

```
  <ALIAS>

    <ENTRIES FOR="Federation">

      <!—Entries according to personal model -->
      <TAGENTRY FROM="Federation.TFederation" TO="Federation.TFederation"></TAGENTRY>
      <TAGENTRY FROM="Federation.TFederation.B1"
                TO="Federation.TFederation.B1"></TAGENTRY>
      <VALENTRY ATTR="Federation.TFederation.B1.Color"
                FROM="red" TO="red"></VALENTRY>
      <VALENTRY ATTR="Federation.TFederation.B1.Color"
                FROM="green" TO="green"></VALENTRY>
      <VALENTRY ATTR="Federation.TFederation.B1.Color"
                FROM="blue" TO="blue"></VALENTRY>

      <!-- Entries for TFederation (TRANSLATION OF) -->
      <TAGENTRY FROM="TFederation.TTFederation"
                TO="Federation.TFederation"></TAGENTRY>
      <TAGENTRY FROM="TFederation.TTFederation.TB1"
                TO="Federation.TFederation.B1"></TAGENTRY>
      <VALENTRY ATTR="TFederation.TTFederation.TB1.TColor"
                FROM="tred" TO="red"></VALENTRY>
```

```
    <VALENTRY ATTR="TFederation.TTFederation.TB1.TColor"
              FROM="tgreen" TO="green"></VALENTRY>
    <VALENTRY ATTR="TFederation.TTFederation.TB1.TColor"
              FROM="tblue" TO="blue"></VALENTRY>

    <!—Entries according to model Canton -->
    <TAGENTRY FROM="Canton.TCanton" TO="Federation.TFederation"> </TAGENTRY>
    <TAGENTRY FROM="Canton.TCanton.K1" TO="Federation.TFederation.B1"> </TAGENTRY>
    <DELENTRY TAG="Canton.TCanton.K1.Txt"> </DELENTRY>
    <DELENTRY TAG="Canton.TCanton.K2"> </DELENTRY>
    <VALENTRY ATTR="Canton.TCanton.K1.Color" FROM="red.dark" TO="red"> </VALENTRY>
    <VALENTRY ATTR="Canton.TCanton.K1.Color" FROM="red.crimson" TO="red"> </VALENTRY>
    <VALENTRY ATTR="Canton.TCanton.K1.Color" FROM="red.light" TO="red"> </VALENTRY>
    <VALENTRY ATTR="Canton.TCanton.K1.Color" FROM="green" TO="green"> </VALENTRY>
    <VALENTRY ATTR="Canton.TCanton.K1.Color" FROM="blue" TO="blue"> </VALENTRY>

    <!-- Entries according to model County -->
    <TAGENTRY FROM="County.TCounty" TO="Federation.TFederation"> </TAGENTRY>

</ENTRIES>

  <ENTRIES FOR="Canton">

    <!—Entries according to personal model -->
    <TAGENTRY FROM="Canton.TCanton" TO="Canton.TCanton"> </TAGENTRY>
    <TAGENTRY FROM="Canton.TCanton.K1" TO="Canton.TCanton.K1"> </TAGENTRY>
    <TAGENTRY FROM="Canton.TCanton.K2" TO="Canton.TCanton.K2"> </TAGENTRY>
    <VALENTRY ATTR="Canton.TCanton.K1.Color"
              FROM="red.dark" TO="red.dark"> </VALENTRY>
    <VALENTRY ATTR="Canton.TCanton.K1.Color"
              FROM="red.crimson" TO="red.crimson"> </VALENTRY>
    <VALENTRY ATTR="Canton.TCanton.K1.Color"
              FROM="red.light" TO="red.light"> </VALENTRY>
    <VALENTRY ATTR="Canton.TCanton.K1.Color"
              FROM="green" TO="green"> </VALENTRY>
    <VALENTRY ATTR="Canton.TCanton.K1.Color"
              FROM="blue" TO="blue"> </VALENTRY>

    <!-- Entries according to model County -->
    <TAGENTRY FROM="County.TCounty" TO="Canton.TCanton"> </TAGENTRY>

  </ENTRIES>

  <ENTRIES FOR="County">

    <!-- Entries according to personal model -->
    <TAGENTRY FROM="County.TCounty" TO="County.TCounty"> </TAGENTRY>

  </ENTRIES>

</ALIAS>
```

Por lo tanto, un programa de lectura, escrito y configurado para el modelo Federado, puede leer los datos proporcionados por Federación, TFederation, Cantón o Condado como sigue:

- Todas las etiquetas proporcionadas por Federation se asignan a sí mismas (por ejemplo, Federation.TFederation a Federation.TFederation).
- Todas las etiquetas de TFederation (TRANSLATION OF) se asignan a su contraparte en Federation (por ejemplo, T.Federation.TTFederation a Federation.TFederation).
- A través de su Tagentry, un objeto XML <Canton.TCanton.K1> está asignado a <Federation.TFederation.B1>. El objeto <Federation.TFederation.B1> es conocido por el programa de lectura para el modelo Federate, por lo que puede interpretar el objeto en consecuencia.

- Mediante un Tagentry se transfiere un objeto XML <County.TCounty.K1> y se representa en <Federation.TFederation.B1>. El objeto <Federation.TFederation.B1> es conocido en el programa de lectura para el modelo federal, por lo que el objeto se puede interpretar de la manera correspondiente.
- El valor "red.crimson" del atributo de enumeración Canton.TCanton.K1.Color o County.TCounty.Color se asigna a través de su Valentry a "rojo". El valor "rojo" es un valor válido según el modelo Federado.
- La clase abstracta Canton.TCanton.K, resp. County.TCounty.K no necesita ser insertada como tagentry, ya que en su conjunto de datos no hay instancias de Canton.TCanton.K, County.TCounty.K puede ocurrir.
- El atributo <Canton.TCanton.K1.Txt>, o <County.TCounty.K1.Txt> debe ignorarse, ya que este atributo no existe en el modelo federado.
- La clase <Canton.TCanton.K2>,o <County.TCounty.K2>, debe ignorarse, ya que desde el punto de vista de las instancias del modelo Federal de <Canton.TCanton.K2>, o <County.TCounty.K2> no existen. Nota: El atributo <Canton.TCanton.K2.Txt>, o <County.TCounty.K2.Txt> no es necesario introducirse específicamente con Delentry, ya que desde el punto de vista del modelo Federate esta clase entera no existe.

NOTAS:

- La tabla de alias puede ser generada con el compilador de INTERLIS 2.
- Para cada modelo de datos contenido en el conjunto de datos (incluyendo todos los modelos base) debe introducirse un elemento de entrada. El nombre del modelo debe insertarse en el atributo XML FOR.
- Los nombres de las etiquetas, que después de su asignación a través de la tabla Alias no conducen a ningún nombre de etiqueta conocido en lo que respecta al modelo de lectura, deben ser declarados como errores por el programa de lectura.
- Los valores de los atributos, que después de su asignación a través de la tabla Alias no conducen a ningún valor conocido en lo que respecta al modelo de lectura, deben ser declarados como errores por el programa de lectura.

3.3.5 Sección de datos

La sección de datos se estructura de la siguiente manera:

```
DataSection = TAG ( DATASECTION )
                  { Basket }
              ETAG ( DATASECTION ).
```

3.3.6 Codificación de los temas

Los contenedores son instancias de un TOPIC concreto, también VIEW TOPIC. Los contenedores se codifican de la siguiente manera:

```
Basket = TAG ( %Model.Topic%,
               'BID=' XML-ID
               [ , 'TOPICS=' XML-Value ]
               [ , 'KIND=' XML-Value ]
               [ , 'STARTSTATE=' XML-Value ]
               [ , 'ENDSTATE=' XML-Value ]
               [ , 'CONSISTENCY=' XML-Value ] )
             { Object | Link | DeleteObject | SetOrderPos }
         ETAG ( %Model.Topic% ).
```

El valor% Model.Topic% tiene que ser sustituido de forma correspondiente para cada tema concreto (por ejemplo, datos básicos de los puntos de control de Cadastral Surveying.Control). Los atributos XML del contenedor tienen el siguiente significado:

- BID. En BID se debe introducir la identificación del contenedor. Con la actualización incremental, la identificación del contenedor debe ser un OID.
- TOPICS. En TOPICS se indican todos los temas, excepto los temas básicos, existentes en el contenedor, separados por comas (por ejemplo, "Canton1.Topic1, Canton2.Topic2"). Los temas indicados deben ser extensiones del tema básico común% Model.Topic% (posiblemente heredado a través de varios niveles). El modelo que contiene la definición del tema básico, así como todos los modelos que contienen las definiciones de los temas extendidos, tienen que ser indicados en la lista de modelos dentro de los preliminares de la transferencia de datos.
- KIND. Tipo de transferencia (valores posibles: FULL, UPDATE, INITIAL.) Cuando se omite el atributo, se supone FULL.
- STARTSTATE. Estado inicial del contenedor antes de la transferencia (sólo en relación con la actualización incremental).
- ENDSTATE. Estado final del contenedor después de la transferencia (sólo en relación con la actualización incremental).
- CONSISTENCY. La consistencia del contenedor (los posibles valores son: COMPLETE, INCOMPLETE, INCONSISTENT, ADAPTED). En caso de que falte el atributo, se asumirá COMPLETE.

3.3.7 Codificación de las clases

Instancias de objetos de una clase concreta se codifican como sigue:

```
Object = TAG ( %Model.Topic.Class%,
                 'TID=' XML-ID
                 [ , 'BID=' XML-ID ]
                 [ , 'OPERATION=' XML-Value ]
                 [ , 'CONSISTENCY=' XML-Value ] )
               { Attribute | EmbeddedLink }
          ETAG ( %Model.Topic.Class% ).

Link = TAG ( %Model.Topic.Association%
               [ , 'TID=' XML-ID ]
               [ , 'BID=' XML-ID ]
               [ , 'OPERATION=' XML-Value ]
               [ , 'CONSISTENCY=' XML-Value ] )
             { Attribute | Role | EmbeddedLink }
          ETAG ( %Model.Topic.Association% ).

DeleteObject = TAG ( DELETE [ , 'TID=' XML-ID ] )
                     { TAG ( %RoleName%,
                             'REF=' XML-ID [ , 'BID=' XML-ID ] )
                       ETAG ( %RoleName% ) }
                 ETAG ( DELETE ).
```

El valor% Model.Topic.Class% debe sustituirse de forma correspondiente para cada clase concreta (por ejemplo, BaseDataSet.ControlPoints.LFP). Además de los atributos definidos dentro del modelo, cada clase - y por lo tanto cada instancia de objeto - se asigna implícitamente a una identificación de transferencia (atributo TID de XML). Las instancias de relación (enlace) sólo tendrán una identificación de transferencia, si se ha requerido explícitamente introduciendo la propiedad OID dentro del ámbito de la definición (véase el capítulo 2.7.1: Descripción de las relaciones). En relación con la transferencia de tipo 'FULL' todos los TID, incluyendo todos los BIDs, deben ser inequívocos a lo largo de toda la transferencia. En relación con el tipo de transferencia INITIAL o UPDATE todos los TID y BID deben ser OID. En BID, la

identificación del contenedor se refiere al contenedor en la que el objeto originalmente se había creado (contenedor original). Si el objeto debe ubicarse dentro de su contenedor original, se pueden omitir los BID. Además, en relación con los tipos de transferencia INITIAL y UPDATE, a cada objeto se le asigna un atributo para la operación de actualización (atributo OPERATION de XML). El atributo de XML OPERATION puede adoptar los valores INSERT, UPDATE o DELETE. Siempre que no se indique, a OPERATION se le asignará el valor INSERT. El atributo de XML CONSISTENCIA puede tomar cualquiera de los valores COMPLETE, INCOMPLETE, INCONSISTENT o ADAPTED. Si el atributo falta, se supondrá COMPLETE.

Con las actualizaciones incrementales es posible exigir la supresión de un determinado objeto del contenedor a través de su OID mediante DeleteObject. En el caso de relaciones sin OID, la instancia (enlace) se identifica mediante la combinación de todos los OID de los objetos a los que se hace referencia. A diferencia de OPERATION = "DELETE", con DeleteObject no se deben proporcionar otros atributos del objeto.

Lo siguiente se aplica al orden de los atributos, roles, EmbeddedLink, atributos de referencia dentro de la clase: Primero son codificados todos los roles de la clase básica, seguidos por los atributos o atributos de referencia de la clase básica, después todos los EmbeddedLink de la clase básica, todos atributos y atributos de referencia de la extensión, todo EmbeddedLink de la extensión, etc. Dentro del mismo nivel de extensión, los atributos y atributos de referencia y roles se codifican según el orden de definición dentro del archivo de modelo. Dentro del mismo nivel de extensión EmbeddedLink se ordenan en orden alfabético.

Los parámetros no se transfieren con la única excepción que se describe en el capítulo 3.3.10 Codificación de definiciones gráficas.

3.3.8 Codificación de vista

Para la codificación de las vistas ver el Capítulo 3.2.4 Objetos transferibles. Son transmitidos los atributos TID y BID de XML, sin embargo, no OPERATION. En términos de atributos de objetos de vista sólo se transmiten aquellos atributos que han sido indicados explícitamente bajo ATTRIBUTES, Implícitamente con ALL OF.

3.3.9 Codificación de relaciones

Hay dos formas diferentes para codificar las relaciones: integrado o enlace. Una relación integrada se codifica en términos de un subelemento de una clase involucrada en la asociación. La instancia de un enlace se codifica de la misma manera que la instancia de una clase.

Las relaciones se integran siempre, a menos que

- tienen más de dos roles
- la cardinalidad máxima es mayor que 1 para ambos roles (básicos) o
- se requiere un OID para la relación o
- en el caso de ciertas relaciones que abarquen temas (véase más adelante).

Si la cardinalidad máxima es mayor que 1 en uno de los dos roles (básicos), la incorporación tiene lugar con la clase objetivo de esta función. Si esta clase de destino se ha definido dentro de un tema diferente que la asociación (básica), no puede tener lugar la incrustación.

Si con ambas funciones (básicas) la cardinalidad máxima es menor o igual a 1, la incorporación tiene lugar con la clase de destino del segundo rol. Si esta clase de destino se ha definido en un tema diferente al de la asociación (básica) y la clase de destino del primer rol se ha definido en el mismo tema que la asociación (básica), la incorporación tendrá lugar con la clase de destino de la primera. En otras palabras: si las clases objetivo de ambos roles se han definido en un tema diferente a la asociación (básica), no puede tener lugar la incorporación.

3.3.9.1 Relaciones integradas

Las relaciones incrustadas se transfieren de la misma manera que los atributos de estructura de la clase donde la relación está incrustada.

La subestructura se construye como sigue:

```
EmbeddedLink = TAG ( %RoleName%,
                     'REF=' XML-ID [ , 'BID=' XML-ID ]
                     [ , 'ORDER_POS=' PosNumber ] )
                   [ EmbeddedLinkStruct ]
                ETAG ( %RoleName% ).

EmbeddedLinkStruct = TAG ( %Model.Topic.Association% )
                         (* Attribute *)
                     ETAG ( %Model.Topic.Association% ).
```

Para %RoleName% tiene que indicar el nombre del rol que se refiere al objeto opuesto (el otro rol no se codificará). En EmbeddedLink se codifican posibles atributos de la relación. Los atributos de XML REF, ORDER_POS y BID tienen la misma significación que con las relaciones no integradas.

3.3.9.2 Las relaciones no integradas

Las relaciones no integradas se transfieren de la misma manera que las instancias de objetos de las clases.

Nota: Para las relaciones sin nombres explícitos, el nombre (clase) es el resultado de combinar los nombres de rol individuales (por ejemplo,% RoleName1RoleName2%).

```
Role = TAG ( %RoleName%,
             'REF=' XML-ID [ , 'BID=' XML-ID ]
             [ , 'ORDER_POS=' PosNumber ] )
       ETAG ( %RoleName% ).

SetOrderPos = TAG ( SETORDERPOS [ , 'TID=' XML-ID ] )
                  { TAG ( %RoleName%,
                          'REF=' XML-ID [ , 'BID=' XML-ID ]
                          [ , 'ORDER_POS=' PosNumber ] )
                    ETAG ( %RoleName% ) }
                ETAG ( SETORDERPOS ).
```

Si la referencia apunta a un objeto dentro del mismo contenedor, la referencia se codifica con REF. Es la identificación de transferencia del objeto al que se ha hecho referencia la que se indica en REF.

Si la referencia apunta a un objeto en un contenedor diferente (dentro de la misma transferencia o incluso en otra parte), la referencia se codificará adicionalmente con BID, introduciendo así la identificación del contenedor del objeto al que se ha hecho referencia en BID.

En las relaciones ordenadas, el atributo ORDER_POS (value> 0!) Define la posición absoluta de esta referencia en la lista ordenada de referencias que forman parte de este contenedor de transferencia.

Con transferencias incrementales es posible transferir la posición absoluta de un objeto de referencia no modificado, mediante SetOrderPos. Incluso con la transferencia incremental ORDER_POS define el orden sólo dentro de la transferencia, por lo tanto, no es estable.

3.3.10 Codificación de las definiciones gráficas

En la transferencia se transmiten las clases de símbolos referidas por la definición gráfica (Sign-ClassRef) para cada definición gráfica. Las instancias de objeto de las clases de símbolos se crean ejecutando definiciones gráficas en un conjunto de datos de entrada concreto. Los parámetros se codifican de la misma manera que los atributos.

3.3.11 Codificación de atributos

3.3.11.1 Reglas generales para la codificación de los atributos

Cada atributo de una instancia de objeto (incluyendo atributos complejos como POINT, POLYLINE, SURFACE, AREA, STRUCTURE, LIST OF; BAG OF, etc.) se codifica de la siguiente manera:

```
Attribute = [ TAG ( %AttributeName% )
                AttributeValue
              ETAG ( %AttributeName% )
            | OIDAttributeValue ].

AttributeValue = ( MTextValue | TextValue | EnumValue
                 | NumericValue | FormattedValue
                 | BlackboxValue
                 | ClassTypeValue | AttributePathTypeValue | StructureValue
                 | BagValue | ListValue
                 | CoordValue | PolylineValue | SurfaceValue).
```

Con valores de atributo indefinidos, el atributo no se transfiere. La unidad de medida del valor del atributo no está codificada. Ejemplo de un atributo simple:

```
<Number>12345</Number>
```

3.3.11.2 Codificación de cadenas

Los atributos del tipo básico TEXT, así como MTEXT (y por consiguiente también NAME y URI) se codifican de la siguiente manera:

```
MTextValue = XML-String.
TextValue = XML-NormalizedString.
```

3.3.11.3 Codificación de enumeraciones

Las enumeraciones se codifican de la siguiente manera:

```
EnumValue = ( EnumElement-Name { '.' EnumElement-Name } ) | 'OTHERS'.
```

Para la codificación de enumeraciones (sin tener en cuenta si el dominio comprende sólo sus hojas o los nodos también) se aplica la sintaxis de las constantes de enumeración (regla EnumValue). Se omite el signo #. Los tipos de orientación de texto predefinidos HALIGNMENT y VALIGNMENT se codifican de la misma forma que las enumeraciones. Igualmente el tipo BOOLEAN se transmite como una enumeración..

3.3.11.4 Codificación de tipos de datos numéricos

Los valores numéricos se codifican como sigue:

```
NumericValue = NumericConst.
```

Nota: Con números enteros, no se admiten ceros colocados delante (007 se transfiere como 7). Con los números reales se permite un máximo de uno (por ejemplo, no 00,07 sino 0,07). Los números de flotación se pueden transferir en diferentes representaciones (con o sin mantisa). También se pueden transferir con un grado de precisión superior al requerido por el dominio. Sin embargo, es esencial que el valor del número flotante no viole el dominio requerido. Así, es posible que, por ejemplo, el número 100 (para un dominio supuesto de 0..999) sea transferido como 100, 100.0000001, 10.0e1 o 1.0e2.

3.3.11.5 Codificación de dominios con formato

Los dominios están codificados con formato de acuerdo con la definición del formato:

```
FormattedValue = XML-NormalizedString.
```

3.3.11.6 Codificación de cajas negras

Los valores de atributo del tipo BLACKBOX se codifican como sigue:

```
BlackboxValue = TAG ( XMLBLBOX )
                    XML-Any
                ETAG ( XMLBLBOX )
              | TAG ( BINBLBOX )
                    XML-base64Binary
                ETAG ( BINBLBOX ).
```

La variedad XML del tipo BLACK BOX está codificada como Any en XML, la variedad binaria como base64Binary del XML.

3.3.11.7 Codificación de los tipos de clase

Los valores de atributo de la clase CLASS o STRUCTURE están codificados de la siguiente manera:

```
ClassTypeValue = XML-NormalizedString.
```

El valor XML normalizedString contiene la clase, estructura o relación de nombre totalmente calificado (por ejemplo DM01AVCH24D.ControlPointsCategory1.LFP1).

3.3.11.8 Codificación de tipos de atributos de ruta

Los valores de atributo de ATTRIBUTE están codificados de la siguiente manera:

```
AttributePathTypeValue = XML-NormalizedString.
```

El valor normalizedString de XML contiene el nombre completo de la clase, seguido por el nombre del atributo, separados por un punto (por ejemplo BaseDataSet.ControlPoints.LFP.Number).

3.3.11.9 Codificación de atributos de la estructura

Los elementos de la estructura de tipo STRUCTURE se codifican como sigue:

```
StructureValue = TAG ( %StructureName% )
                     { Attribute | ReferenceAttribute }
                 ETAG ( %StructureName% ).
```

StructureName se forma ya sea como Model.StructureName en el nivel de modelo, o como Model.Topic.StructureName en el nivel de Topic..

3.3.11.10 Codificación de las subestructuras ordenadas y no ordenadas

Los valores de los atributos del tipo BAG OF y LIST OF se codifican de la siguiente manera:

```
BagValue = (* StructureValue *).
ListValue = (* StructureValue *).
```

La secuencia de los elementos ListValue no se pueden aalterar durante la transferencia .

3.3.11.11 Codificación de coordenadas

Los valores del tipo COORD están codificados de la siguiente forma:

```
CoordValue = TAG ( COORD )
                 TAG ( C1 )
                     NumericConst
                 ETAG ( C1 )
                 [ TAG ( C2 )
                       NumericConst
                   ETAG ( C2 )
                   [ TAG ( C3 )
                         NumericConst
                     ETAG ( C3 ) ] ]
             ETAG ( COORD ).
```

Los subobjetos individuales XML deben rellenarse como sigue:

- C1. Primer componente de la coordenada (codificado como valor numérico).
- C2. Segundo componente de la coordenada (sólo coordenadas 2D y 3D, codificado como valor numérico).
- C3. Tercer componente de la coordenada (sólo con coordenadas 3D, codificados como valor numérico).

3.3.11.12 Codificación de cadenas de líneas

Los valores de atributo del tipo POLYLINE se codifica como sigue:

```
PolylineValue = TAG ( POLYLINE )
                    [ LineAttr ]
                    SegmentSequence | (* ClippedSegment *)
                ETAG ( POLYLINE ).

StartSegment = CoordValue.

StraightSegment = CoordValue.

ArcSegment = TAG ( ARC )
                 TAG ( C1 )
                     NumericConst
                 ETAG ( C1 )
                 TAG ( C2 )
                     NumericConst
                 ETAG ( C2 )
                 [ TAG ( C3 )
                       NumericConst
                   ETAG ( C3 ) ]
                 TAG ( A1 )
                     NumericConst
                 ETAG ( A1 )
                 TAG ( A2 )
                     NumericConst
                 ETAG ( A2 )
                 [TAG (R)
                       NumericConst
                   ETAG ( R ) ]
             ETAG ( ARC ).

LineFormSegment = StructureValue.

SegmentSequence = StartSegment (* StraightSegment
                                | ArcSegment
                                | LineFormSegment *).

ClippedSegment = TAG ( CLIPPED )
                     SegmentSequence
                 ETAG ( CLIPPED ).

LineAttr = TAG ( LINEATTR )
               StructureValue
           ETAG ( LINEATTR ).
```

Los segmentos rectos de una cadena de línea se codifican de acuerdo con la regla StraightSegment, para segmentos de arco se aplica la regla ArcSegment. Los segmentos de línea definidos con LINE FORM se codifican como estructuras (LineStructure).

Nota: Para los segmentos de arco (regla ArcSegment), el radio (atributo opcional de XML, R) se transmite de forma redundante a la coordenada del punto intermedio (A1/A2). El punto intermedio de un arco sólo tiene importancia para su posición. Su altura debe ser interpolado linealmente entre el punto inicial y el final. Si el arco se ha definido en sentido horario (desde el inicio hasta el punto final), el radio tendrá un signo positivo, de lo contrario negativo. Si se producen diferencias entre los valores de radio y coordenadas, es el radio el que prevalece (ver capítulo 2.8.12.2: Cadenas de línea con segmentos de línea recta y arcos de círculo como segmentos de curva predefinidos). La altura del vértice (C3) sólo tiene que ser transferida con cadenas de líneas 3D.

En el caso de un conjunto completo de datos, SegmentSequence es obligatorio y ClippedSegments no puede darse. Si fuera de un conjunto completo de datos sólo se transfiere una sección, puede darse cualquier número de ClippedSegments en lugar de SegmentSequence.

3.3.11.13 Codificación de superficies y mosaicos

SURFACE y AREA se codifican como sigue:

```
SurfaceValue = TAG ( SURFACE )
                    Boundaries | (* ClippedBoundaries *)
                ETAG ( SURFACE ).

Boundaries = OuterBoundary { InnerBoundary }.

OuterBoundary = Boundary.

InnerBoundary = Boundary.

ClippedBoundaries = TAG ( CLIPPED )
                        Boundaries
                    ETAG ( CLIPPED ).

Boundary = TAG ( BOUNDARY )
                (* PolylineValue *)
            ETAG ( BOUNDARY ).
```

Las superficies se transmiten como una secuencia de límites. Una frontera es una secuencia de líneas límite, la siguiente línea límite que comienza con el punto final de la línea límite anterior. El punto final de la última línea de límite es idéntico al punto de inicio de la primera línea de límite. Así, las líneas fronterizas forman una cadena de líneas cerradas (polígono). Un límite puede dividirse en líneas de límite en cualquier vértice de su elección. Los segmentos pueden diferir con cada transferencia, sobre todo con actualizaciones incrementales.

El primer límite de una superficie (OuterBoundary) es el límite exterior de una superficie, posiblemente seguido por los límites interiores (InnerBoundary) de la superficie que limitan los recintos de la superficie. Los límites interiores deben encontrarse completamente dentro de los límites externos. Los límites individuales de una superficie no pueden superponerse.

Si SURFACE o AREA han sido definidos con atributos de línea, el elemento de estructura del atributo “line” debe ser transmitido con cada PolylineValue (regla LineAttr en PolylineValue).

Con el teselado (AREA) todas las líneas límite de la superficie deben coincidir con las líneas límite de la(s) superficie(s) vecina(s), a menos que formen parte del perímetro de la red de área. Se consideran idénticas dos líneas fronterizas si en cada segmento de la línea límite todos los vértices son idénticos al segmento correspondiente de la superficie vecina. Con los vértices de arco sólo puede diferir el signo del radio del arco. Si se han definido atributos de línea para la red de área, los valores de los atributos de línea para las líneas de límite deben ser idénticos en parejas.

En el caso de un conjunto de datos completo, los límites son obligatorios con la regla SurfaceValue y no se pueden producir ClippedBoundaries. Si de un conjunto completo de datos sólo se transfiere una sección, se puede producir cualquier número de ClippedBoundaries en lugar de Boundaries. Cada elemento ClippedBoundaries debe cumplir las reglas del tipo SURFACE.

3.3.11.14 Codificación de referencias

Los atributos del tipo REFERENCE TO se codifican como sigue:

```
ReferenceAttribute = [ TAG ( %AttributeName%,
                             'REF=' XML-ID [ , 'BID=' XML-ID ] )
                       ETAG ( %AttributeName% ) ].
```

Los atributos de XML REF y BID tienen el mismo significado que en las relaciones propiamente dichas.

3.3.11.15 Codificación de metaobjetos

Los atributos del tipo METAOBJECT (Véase la clase correspondiente en el apéndice A: El modelo interno de datos INTERLIS) se codifican de acuerdo con el capítulo 3.3.11.10 Codificación de subestructuras ordenadas y no ordenadas. Sin embargo, los parámetros del tipo METAOBJECT (regla ParameterDef) no se transmiten. Los parámetros del tipo METAOBJECT OF se transmiten como atributos del tipo NAME.

3.3.11.16 Codificación del OIDType

Los valores de atributo del tipo OIDType se codifican de la misma manera que un ID XML incluido el dominio Oid. Si OIDType es un NumericType, las reglas que gobiernan la codificación de los tipos numéricos también deben aplicarse para el valor (sin el dominio Oid).

```
OIDAttributeValue = [ TAG ( %AttributeName%,
                            'OID=' XML-ID )
                      ETAG ( %AttributeName% ) ].
```

3.4 Aplicación de herramientas XML

Dado que la transferencia de INTERLIS 2 se basa totalmente en XML 1.0, es posible utilizar INTERLIS o herramientas XML para el procesamiento (así como análisis) de los objetos INTERLIS. Sin embargo, hay que tener en cuenta las siguientes diferencias:

- Además del conjunto de datos XML, las herramientas INTERLIS 2 también reconocerán los correspondientes modelos de datos INTERLIS 2. Por ejemplo, una herramienta de control de INTERLIS en general será capaz de ejecutar pruebas más rigurosas de lo que podría ser una simple herramienta XML.
- Las herramientas INTERLIS 2 conocen todos los tipos de datos INTERLIS específicos como COORD, POLYLINE, SURFACE, etc. Por lo tanto, un navegador INTERLIS 2 podrá representar gráficamente los conjuntos de datos, mientras que un simple explorador XML sólo visualizará la estructura del documento.
- Las herramientas INTERLIS 2 soportan la lectura polimórfica de datos mediante la tabla Alias. Por lo tanto, un programa de importación de INTERLIS será capaz de leer un modelo ampliado de la misma manera que los datos de un modelo base. Con una simple herramienta XML la lectura de polimorfos no es automáticamente posible.
- Además, las herramientas de INTERLIS 2 pueden soportar la traducción de nombres de esquema a través de la tabla Alias. De nuevo esto no sería posible con simples herramientas XML.

A pesar de todas estas diferencias, las herramientas XML generales pueden emplearse directamente para varios propósitos. Entre otros, cabe citar el filtrado de conjuntos de datos, la edición de conjuntos de datos con editores XML, el examen de conjuntos de datos de transferencia, la traducción a otros formatos, etc.

Apéndice A (normativo) El modelo de datos interno de INTERLIS

A continuación, se ha resumido una vez más todo el modelo interno de datos. Este modelo sólo sirve para ilustrar las explicaciones y no puede ser compilado (por ejemplo, utiliza nombres que son palabras clave de acuerdo a la definición del lenguaje (ver capítulo 2.2.7 Símbolos especiales y palabras reservadas). Cualquier software INTERLIS puesto a su disposición por KOGIS, como el compilador INTERLIS, debe ser capaz de reconocer todos los elementos de este modelo.

```
INTERLIS 2.3;

CONTRACTED TYPE MODEL INTERLIS (en) AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  LINE FORM
    STRAIGHTS;
    ARCS;

  UNIT
    ANYUNIT (ABSTRACT);
    DIMENSIONLESS (ABSTRACT);
    LENGTH (ABSTRACT);
    MASS (ABSTRACT);
    TIME (ABSTRACT);
    ELECTRIC_CURRENT (ABSTRACT);
    TEMPERATURE (ABSTRACT);
    AMOUNT_OF_MATTER (ABSTRACT);
    ANGLE (ABSTRACT);
    SOLID_ANGLE (ABSTRACT);
    LUMINOUS_INTENSITY (ABSTRACT);
    MONEY (ABSTRACT);

    METER [m] EXTENDS LENGTH;
    KILOGRAM [kg] EXTENDS MASS;
    SECOND [s] EXTENDS TIME;
    AMPERE [A] EXTENDS ELECTRIC_CURRENT;
    DEGREE_KELVIN [K] EXTENDS TEMPERATURE;
    MOLE [mol] EXTENDS AMOUNT_OF_MATTER;
    RADIAN [rad] EXTENDS ANGLE;
    STERADIAN [sr] EXTENDS SOLID_ANGLE;
    CANDELA [cd] EXTENDS LUMINOUS_INTENSITY;

  DOMAIN
    URI (FINAL) = TEXT*1023;
    NAME (FINAL) = TEXT*255;
    INTERLIS_1_DATE (FINAL) = TEXT*8;
    BOOLEAN (FINAL) = (
      falso,
      true) ORDERED;
    HALIGNMENT (FINAL) = (
      Left,
      Center,
      Right) ORDERED;
    VALIGNMENT (FINAL) = (
      Top,
      Cap,
      Half,
      Base,
      Bottom) ORDERED;
    ANYOID = OID ANY;
    I32OID = OID 0 .. 2147483647;
    STANDARDOID = OID TEXT*16;
    UUIDOID = OID TEXT*36;
```

```
  LineCoord (ABSTRACT) = COORD NUMERIC, NUMERIC;

FUNCTION myClass (Object: ANYSTRUCTURE): STRUCTURE;
FUNCTION isSubClass (potSubClass: STRUCTURE; potSuperClass: STRUCTURE):
  BOOLEAN;
FUNCTION isOfClass (Object: ANYSTRUCTURE; Class: STRUCTURE): BOOLEAN;
FUNCTION elementCount (bag: BAG OF ANYSTRUCTURE): NUMERIC;
FUNCTION objectCount (Objects: OBJECTS OF ANYCLASS): NUMERIC;
FUNCTION len (TextVal: TEXT): NUMERIC;
FUNCTION lenM (TextVal: MTEXT): NUMERIC;
FUNCTION trim (TextVal: TEXT): TEXT;
FUNCTION trimM (TextVal: MTEXT): MTEXT;
FUNCTION isEnumSubVal (SubVal: ENUMTREEVAL; NodeVal: ENUMTREEVAL): BOOLEAN;
FUNCTION inEnumRange (Enum: ENUMVAL;
                      MinVal: ENUMTREEVAL;
                      MaxVal: ENUMTREEVAL): BOOLEAN;
FUNCTION convertUnit (from: NUMERIC): NUMERIC;
FUNCTION areAreas (Objects: OBJECTS OF ANYCLASS;
                   SurfaceBag: ATTRIBUTE OF @ Objects
                                    RESTRICTION (BAG OF ANYSTRUCTURE);
                   SurfaceAttr: ATTRIBUTE OF @ SurfaceBag
                                     RESTRICTION (SURFACE)): BOOLEAN;

CLASS METAOBJECT (ABSTRACT) =
  Name: MANDATORY NAME;
UNIQUE Name;
END METAOBJECT;

CLASS METAOBJECT_TRANSLATION =
  Name: MANDATORY NAME;
  NameInBaseLanguage: MANDATORY NAME;
UNIQUE Name;
UNIQUE NameInBaseLanguage;
END METAOBJECT_TRANSLATION;

STRUCTURE AXIS =
  Parámetro
    Unit: NUMERIC [ANYUNIT];
END AXIS;

CLASS REFSYSTEM (ABSTRACT) EXTENDS METAOBJECT =
END REFSYSTEM;

CLASS COORDSYSTEM (ABSTRACT) EXTENDS REFSYSTEM =
  ATTRIBUTE
    Axis: LIST {1..3} OF AXIS;
END COORDSYSTEM;

CLASS SCALSYSTEM (ABSTRACT) EXTENDS REFSYSTEM =
  Parámetro
    Unit: NUMERIC [ANYUNIT];
END SCALSYSTEM;

CLASS SIGN (ABSTRACT) EXTENDS METAOBJECT =
  PARAMETER
    Sign: METAOBJECT;
END SIGN;


TOPIC TIMESYSTEMS =

  CLASS CALENDAR EXTENDS INTERLIS.SCALSYSTEM =
    Parámetro
      Unit(EXTENDED): NUMERIC [TIME];
  END CALENDAR;

  CLASS TIMEOFDAYSYS EXTENDS INTERLIS.SCALSYSTEM =
    Parámetro
```

```
      Unit(EXTENDED): NUMERIC [TIME];
  END TIMEOFDAYSYS;

END TIMESYSTEMS;

UNIT
  Minute [min] = 60 [INTERLIS.s];
  Hour   [h]   = 60 [min];
  Day    [d]   = 24 [h];
  Month [M] EXTENDS INTERLIS.TIME;
  Year [Y] EXTENDS INTERLIS.TIME;

REFSYSTEM BASKET BaseTimeSystems ~ TIMESYSTEMS
  OBJECTS OF CALENDAR: GregorianCalendar
  OBJECTS OF TIMEOFDAYSYS: UTC;

STRUCTURE TimeOfDay (ABSTRACT) =
  Hours: 0 .. 23 CIRCULAR [h];
  CONTINUOUS SUBDIVISION Minutes: 0 .. 59 CIRCULAR [min];
  CONTINUOUS SUBDIVISION Seconds: 0,0000.000 .. 59.999 CIRCULAR [INTERLIS.s];
END TimeOfDay;

STRUCTURE UTC EXTENDS TimeOfDay =
  Hours(EXTENDED): 0 .. 23 {UTC};
END UTC;

DOMINIO
  GregorianYear = 1582 .. 2999 [Y] {GregorianCalendar};

STRUCTURE GregorianDate =
  Año: GregorianYear;
  SUBDIVISION Month: 1 .. 12 [M];
  SUBDIVISION Day: 1 .. 31 [d];
END GregorianDate;

STRUCTURE GregorianDateTime EXTENDS GregorianDate =
  SUBDIVISION Hours: 0 .. 23 CIRCULAR [h] {UTC};
  CONTINUOUS SUBDIVISION Minutes: 0 .. 59 CIRCULAR [min];
  CONTINUOUS SUBDIVISION Seconds: 0,0000.000 .. 59.999 CIRCULAR [INTERLIS.s];
END GregorianDateTime;

DOMAIN XMLTime = FORMAT BASED ON UTC ( Hours/2 ":" Minutes ":" Seconds );
DOMAIN XMLDate = FORMAT BASED ON GregorianDate ( Year "-" Month "-" Day );
DOMAIN XMLDateTime EXTENDS XMLDate = FORMAT BASED ON GregorianDateTime
                                        ( INHERITANCE "T" Hours/2 ":" Minutes
                                                      ":" Seconds );

STRUCTURE LineSegment (ABSTRACT) =
  SegmentEndPoint: MANDATORY LineCoord;
END LineSegment;

STRUCTURE StartSegment (FINAL) EXTENDS LineSegment =
END StartSegment;

STRUCTURE StraightSegment (FINAL) EXTENDS LineSegment =
END StraightSegment;

STRUCTURE ArcSegment (FINAL) EXTENDS LineSegment =
  ArcPoint: MANDATORY LineCoord;
  Radius: NUMERIC [LENGTH];
END ArcSegment;

STRUCTURE SurfaceEdge =
  Geometry: DIRECTED POLYLINE;
  LineAttrs: ANYSTRUCTURE;
END SurfaceEdge;

STRUCTURE SurfaceBoundary =
```

```
      Líneas LIST OF SurfaceEdge;
    END SurfaceBoundary;

    STRUCTURE LineGeometry =
      Segments: LIST OF LineSegment;
    MANDATORY CONSTRAINT isOfClass (Segments[FIRST],StartSegment);
    END LineGeometry;

  END INTERLIS.
```

Apéndice B (normativo para CH) Tabla de símbolos

En la tabla de abajo se pueden encontrar todos los símbolos estándar disponibles de INTERLIS 2, así como símbolos especiales, diéresis y marcas diacríticas, así como su codificación en una transferencia INTERLIS. Para algunos de estos símbolos, están a su disposición varias formas de codificación y en estos casos se indican todas las formas posibles de codificación del símbolo. Siempre que existan varias posibilidades de codificación, un programa de escritura de INTERLIS 2 seleccionará una de estas posibilidades. Un programa de lectura de INTERLIS 2 debe ser capaz de reconocer todas estas formas de codificación para un símbolo.

La tabla sólo se aplica al contenido XML (es decir, XML String, XML NormalizedString o XML Value). Las etiquetas XML se transmiten exclusivamente como símbolos codificados en ASCII de acuerdo con la sintaxis del capítulo 3.

Además de los símbolos estándar que figuran en la tabla, pueden utilizarse otros símbolos en las solicitudes INTERLIS 2, siempre que se haya acordado un contrato entre las partes interesadas.

Tabulador (TAB, #x9), Retorno de carro (CR, #xD) y Nueva línea (LF, #xA) son los únicos códigos de control admisibles. Sin embargo, sólo pueden ocurrir dentro del ámbito de la codificación de MTEXT (véase capítulo 2.8.1 Cadenas y capítulo 3.3.11.2 Codificación de cadenas).

UCS Hex	UCS Dec	Codificación UTF-8 Octet Hex	Codificación XML Referencia de carácter Dec	Codificación XML Referencia de carácter Hex	Codificación XML entidad de Referencia	Representado como
0020	32	20				
0021	33	21				!
0022	34		&#34;	&#x22;	&quot;	"
0023	35	23				#
0024	36	24				$
0025	37	25				%
0026	38		&#38;	&#x26;	&amp;	&
0027	39		&#39;	&#x27;	&apos;	'
0028	40	28				(
0029	41	29				)
002A	42	2A				*
002B	43	2B				+
002C	44	2C				,
002D	45	2D				-
002E	46	2E				.
002F	47	2F				/
0030	48	30				0
0031	49	31				1
0032	50	32				2
0033	51	33				3
0034	52	34				4
0035	53	35				5
0036	54	36				6
0037	55	37				7
0038	56	38				8
0039	57	39				9
003A	58	3A				:
003B	59	3B				;

UCS Hex	UCS Dec	Codificación UTF-8 Octet Hex	Codificación XML Referencia de carácter Dec	Codificación XML Referencia de carácter Hex	Codificación XML entidad de Referencia	Representado como
003C	60		&#60;	&#x3c;	&lt;	<,
003D	61	3D				=
003E	62		&#62;	&#x3e;	&gt;	>
003F	63	3F				?
0040	64	40				@
0041	65	41				A
0042	66	42				B
0043	67	43				C
0044	68	44				D
0045	69	45				E
0046	70	46				F
0047	71	47				G
0048	72	48				H
0049	73	49				I
004A	74	4A				J
004B	75	4B				K
004C	76	4C				L
004D	77	4D				M
004E	78	4E				N
004F	79	4F				O
0050	80	50				P
0051	81	51				Q
0052	82	52				R
0053	83	53				S
0054	84	54				T
0055	85	55				U
0056	86	56				V
0057	87	57				W
0058	88	58				X
0059	89	59				Y
005A	90	5A				Z
005B	91	5B				[
005C	92	5C				\\
005D	93	5D				]
005E	94	5E				^
005F	95	5F				_
0060	96	60				`
0061	97	61				a
0062	98	62				b
0063	99	63				C
0064	100	64				d
0065	101	65				e
0066	102	66				f
0067	103	67				g
0068	104	68				h
0069	105	69				i
006A	106	6A				j
006B	107	6B				k
006C	108	6C				l
006D	109	6D				m
006E	110	6E				n
006F	111	6F				o

UCS Hex	UCS Dec	Codificación UTF-8 Octet Hex	Codificación XML Referencia de carácter Dec	Codificación XML Referencia de carácter Hex	Codificación XML entidad de Referencia	Representado como
0070	112	70				p
0071	113	71				q
0072	114	72				r
0073	115	73				s
0074	116	74				t
0075	117	75				u
0076	118	76				v
0077	119	77				w
0078	120	78				x
0079	121	79				y
007A	122	7A				z
007B	123	7B				{
007C	124	7C				|
007D	125	7D				}
007E	126	7E				~
00A7	167	C2 A7)	&#167	&#xa7		§
00AB	171	C2 AB	&#171	&#xab		«
00BB	187	C2 BB	&#187	&#xbb		»
00C4	196	C3 84	&#196;	&#xc4;		UN
00C6	198	C3 86	&#198;	&#xc6;		Æ
00C7	199	C3 87	&#199;	&#xc7;		Ç
00C8	200	C3 88	&#200;	&#xc8;		È
00C9	201	C3 89	&#201;	&#xc9;		É
00D1	209	C3 91	&#209;	&#xd1;		Ñ
00D6	214	C3 96	&#214;	&#xd6;		Ö
00DC	220	C3 9C	&#220;	&#xdc;		Ü
00E0	224	C3 A0)	&#224;	&#xe0;		à
00E1	225	C3 A1)	&#225;	&#xe1;		á
00E2	226	C3 A2)	&#226;	&#xe2;		â
00E4	228	C3 A4)	&#228;	&#xe4;		ä
00E6	230	C3 A6)	&#230;	&#xe6;		æ
00E7	231	C3 A7)	&#231;	&#xe7;		ç
00E8	232	C3 A8)	&#232;	&#xe8;		è
00E9	233	C3 A9)	&#233;	&#xe9;		é
00EA	234	C3 AA	&#234;	&#xea;		ê
00EB	235	C3 AB	&#235;	&#xeb;		ë
00EC	236	C3 AC	&#236;	&#xec;		ì
00ED	237	C3 AD	&#237;	&#xed;		í
00EE	238	C3 AE	&#238;	&#xee;		î
00EF	239	C3 AF	&#239;	&#xef;		ï
00F1	241	C3 B1)	&#241;	norte		ñ
00F2	242	C3 B2)	&#242;	&#xf2;		ò
00F3	243	C3 B3)	&#243;	&#xf3;		ó
00F4	244	C3 B4)	&#244;	&#xf4;		ô
00F5	245	C3 B5)	&#245;	&#xf5;		õ
00F6	246	C3 B6)	&#246;	&#xf6;		ö
00F9	249	C3 B9)	&#249;	&#xf9;		ù
00FA	250	C3 BA	&#250;	&#xfa;		ú
00FB	251	C3 BB	&#251;	&#xfb;		û
00FC	252	C3 BC	&#252;	&#xfc;		Ü
20AC	8364	E2 82 AC	&#8364	%#x20ac		€

Tabla 2: símbolos Unicode permitidos en INTERLIS y su codificación.

NOTAS:

- En las columnas UCS/Hex y UCS/Dec, se ha indicado el código de símbolo UCS (Unicode) (hexadecimal y decimal).
- En la columna UTF-8, la codificación del símbolo se ha indicado de acuerdo a UTF-8 como 8bit bytes (octeto) en notación hexadecimal. Los símbolos ≥ Hex 80 se codifican como secuencias de bytes múltiples. Nota: La notación hexadecimal sólo se utiliza para ilustrar la codificación. Sólo los octetos codificados en binario se transmiten en el archivo de transferencia.
- En las columnas de codificación XML de referencia de caracteres (Decimal y Hexadecimal), la codificación del símbolo se indica como referencia de carácter XML (decimal y hexadecimal). Este valor es una secuencia de símbolos ASCII y debe utilizarse estrictamente preciso en la transferencia. Siempre que sea posible, un programa de escritura debe seleccionar la codificación de referencia de carácter para el símbolo. La ventaja de la codificación de referencia de caracteres radica en su capacidad para que pueda ser indicada, así como editado con editores ASCII simples en cualquier plataforma (Unix, PC, etc.). Nota: Con la variante de codificación hexadecimal las letras a-f pueden ser listadas en mayúsculas o minúsculas (sin embargo, la x en la variante hexadecimal debe ser siempre una letra minúscula).
- La codificación XML (Entity Reference) sólo es admisible para algunos pocos símbolos especiales. Su valor es una secuencia ASCII que debe utilizarse con precisión en la transferencia. Nota: entidades XML como ü (para el símbolo ü) no son admisibles en un archivo de transferencia INTERLIS 2, ya que no se hará referencia a ningún DTD mediante un archivo de transferencia INTERLIS 2 (entidades permitidas como & han sido predefinidas en la especificación XML 1.0).
- En la representación de la columna encontrará la representación del símbolo en un editor compatible con UCS o unicode.

Apéndice C (informativo) un pequeño ejemplo de carreteras

Introducción

Con el fin de facilitar el entendimiento de INTERLIS 2, ofrecemos un ejemplo pequeño pero no obstante completo. Describe un conjunto de datos diseñado para una transferencia completa de datos. Para ejemplos de aplicación con actualización incremental (incluyendo OID), se suministran los conjuntos de datos de prueba y manuales de usuario necesarios..

El ejemplo consiste en las siguientes partes:

- Modelo de datos RoadsExdm2ben y RoadsExdm2ien.
- Conjunto de datos XML RoadsExdm2ien (archivo RoadsExdm2ien.xml) que contiene objetos de acuerdo con el modelo de datos RoadsExdm2ien.
- Modelo gráfico RoadsExgm2ien. Una representación posible para el modelo de datos RoadsExdm2ien se define en el modelo gráfico (Nota: cualquier número de representaciones es posible para un solo modelo de datos).
- Colección de objetos de símbolo (biblioteca de símbolos) en RoadsExgm2ien_Symbols (archivo RoadsExgm2ien_Symbols.xml). La biblioteca de símbolos es un conjunto de datos XML de acuerdo con el modelo de simbología StandardSymbology (ver apéndice J, Modelos de simbología). La biblioteca de símbolos se utiliza en el modelo gráfico RoadsExgm2ien para la representación de datos ejemplares del conjunto de datos RoadsExdm2ien.xml.

El nombre "RoadsExdm2ben" es una abreviatura de "RoadsExample, modelo de datos, INTERLIS 2, modelo básico, inglés". A continuación se describirán más detalladamente las partes individuales.

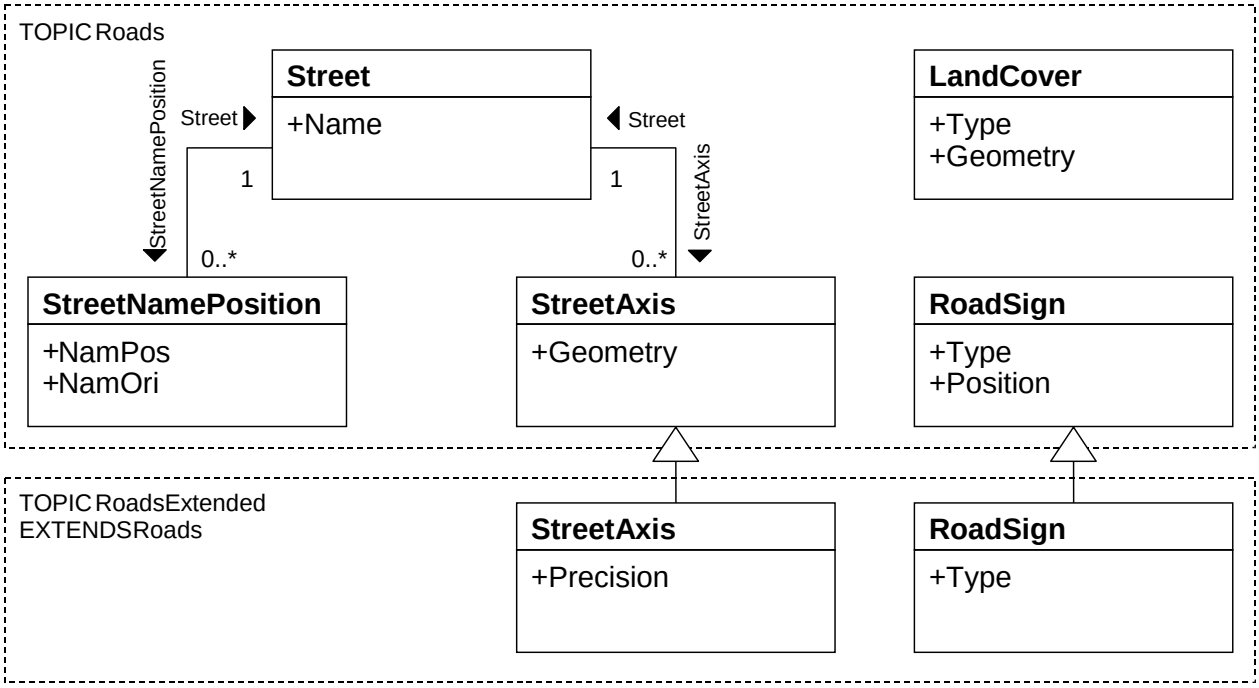

Figura 26: diagrama de clase UML de los modelos de datos.

modelo de datos RoadsExdm2ben y RoadsExdm2ien

El modelo de datos RoadsExdm2ben contiene los objetos LandCover, StreetAxis, StreetName y PointObject. El modelo de datos RoadsExdm2ien es una extensión de RoadsExdm2ben. El diagrama de clases UML anterior (véase la figura 26) proporciona una visión completa de los modelos de datos.

Nota: el ejemplo escogido es muy simple a propósito, no pretendiéndose ser exhaustivos. Los modelos correspondientes definidos en INTERLIS 2 son los siguientes:

```
!! File RoadsExdm2ben.ili Release 2005-06-16

INTERLIS 2.3;

MODEL RoadsExdm2ben (en) AT "http://www.interlis.ch/models"
  VERSION "2005-06-16" =

  UNIT
    Angle_Degree = 180 / PI [INTERLIS.rad];

  DOMAIN
    Point2D = COORD
      0.000 .. 200.000 [INTERLIS.m], !! Min_East  Max_East
      0.000 .. 200.000 [INTERLIS.m], !! Min_North Max_North
      ROTATION 2 -> 1;
    Orientation = 0.0 .. 359.9 CIRCULAR [Angle_Degree];


  TOPIC Roads =

    STRUCTURE LAttrs =
      LArt: (
        welldefined,
        fuzzy);
    END LAttrs;

    CLASS LandCover =
      Type: MANDATORY (
        building,
        street,
        water,
        other);
      Geometry: MANDATORY SURFACE WITH (STRAIGHTS)
        VERTEX Point2D WITHOUT OVERLAPS > 0.100
        LINE ATTRIBUTES LAttrs;
    END LandCover;

    CLASS Street =
      Name: MANDATORY TEXT*32;
    END Street;

    CLASS StreetAxis =
      Geometry: MANDATORY POLYLINE WITH (STRAIGHTS)
        VERTEX Point2D;
    END StreetAxis;

    ASSOCIATION StreetAxisAssoc =
      Street -- {1} Street;
      StreetAxis -- StreetAxis;
    END StreetAxisAssoc;

    CLASS StreetNamePosition =
      NamPos: MANDATORY Point2D;
      NamOri: MANDATORY Orientation;
    END StreetNamePosition;

    ASSOCIATION StreetNamePositionAssoc =
      Street -- {0..1} Street;
      StreetNamePosition -- StreetNamePosition;
    END StreetNamePositionAssoc;
```

```
    CLASS RoadSign =
      Type: MANDATORY (
        prohibition,
        indication,
        danger,
        velocity);
      Position: MANDATORY Point2D;
    END RoadSign;

  END Roads; !! of TOPIC

END RoadsExdm2ben. !! of MODEL



!! File RoadsExdm2ien.ili Release 2005-06-16

INTERLIS 2.3;

MODEL RoadsExdm2ien (en) AT "http://www.interlis.ch/models"
  VERSION "2005-06-16" =

  IMPORTS RoadsExdm2ben;


  TOPIC RoadsExtended EXTENDS RoadsExdm2ben.Roads =

    CLASS StreetAxis (EXTENDED) =
      Precision: MANDATORY (
        precise,
        unprecise);
    END StreetAxis;

    CLASS RoadSign (EXTENDED) =
      Type (EXTENDED): (
        prohibition (
          noentry,
          noparking,
          other));
    END RoadSign;

  END RoadsExtended; !! of TOPIC

END RoadsExdm2ien. !! of MODEL
```

Conjunto de datos RoadsExdm2ien de acuerdo con el modelo de datos RoadsExdm2ien

A continuación, se encuentra un ejemplo de conjunto de datos para el modelo de datos RoadsExdm2ien. El formato XML se ha derivado del modelo de datos RoadsExdm2ien mediante las reglas establecidas en el capítulo 3 Transferencia secuencial.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- File RoadsExdm2ien.xml 2005-06-16 (http://www.interlis.ch/models) -->

<TRANSFER xmlns="http://www.interlis.ch/INTERLIS2.3"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.interlis.ch/INTERLIS2.3
                              RoadsExdm2ien.xsd">
  <HEADERSECTION VERSION="2.3" SENDER="KOGIS">
    <MODELS>
      <MODEL NAME="RoadsExdm2ben" URI="http://www.interlis.ch/models"
             VERSION="2005-06-16"/>
      <MODEL NAME="RoadsExdm2ien" URI="http://www.interlis.ch/models"
             VERSION="2005-06-16"/>
    </MODELS>
```

```
<ALIAS>
  <ENTRIES FOR="RoadsExdm2ben">
    <TAGENTRY FROM="RoadsExdm2ben.Roads"
              TO="RoadsExdm2ben.Roads"/>
    <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended"
              TO="RoadsExdm2ben.Roads"/>
    <TAGENTRY FROM="RoadsExdm2ben.Roads.LAttrs"
              TO="RoadsExdm2ben.Roads.LAttrs"/>
    <TAGENTRY FROM="RoadsExdm2ben.Roads.LandCover"
              TO="RoadsExdm2ben.Roads.LandCover"/>
    <TAGENTRY FROM="RoadsExdm2ben.Roads.Street"
              TO="RoadsExdm2ben.Roads.Street"/>
    <TAGENTRY FROM="RoadsExdm2ben.Roads.StreetAxis"
              TO="RoadsExdm2ben.Roads.StreetAxis"/>
    <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended.StreetAxis"
              TO="RoadsExdm2ben.Roads.StreetAxis"/>
    <DELENTRY TAG="RoadsExdm2ien.RoadsExtended.StreetAxis"
              ATTR="Precision"/>
    <TAGENTRY FROM="RoadsExdm2ben.Roads.StreetAxisAssoc"
              TO="RoadsExdm2ben.Roads.StreetAxisAssoc"/>
    <TAGENTRY FROM="RoadsExdm2ben.Roads.StreetNamePosition"
              TO="RoadsExdm2ben.Roads.StreetNamePosition"/>
    <TAGENTRY FROM="RoadsExdm2ben.Roads.StreetNamePositionAssoc"
              TO="RoadsExdm2ben.Roads.StreetNamePositionAssoc"/>
    <TAGENTRY FROM="RoadsExdm2ben.Roads.RoadSign"
              TO="RoadsExdm2ben.Roads.RoadSign"/>
    <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended.RoadSign"
              TO="RoadsExdm2ben.Roads.RoadSign"/>
    <VALENTRY TAG="RoadsExdm2ben.Roads.LAttrs" ATTR="LArt"
              FROM="welldefined" TO="welldefined"/>
    <VALENTRY TAG="RoadsExdm2ben.Roads.LAttrs" ATTR="LArt"
              FROM="fuzzy" TO="fuzzy"/>
    <VALENTRY TAG="RoadsExdm2ben.Roads.LandCover" ATTR="Type"
              FROM="building" TO="building"/>
    <VALENTRY TAG="RoadsExdm2ben.Roads.LandCover" ATTR="Type"
              FROM="street" TO="street"/>
    <VALENTRY TAG="RoadsExdm2ben.Roads.LandCover" ATTR="Type"
              FROM="water" TO="water"/>
    <VALENTRY TAG="RoadsExdm2ben.Roads.LandCover" ATTR="Type"
              FROM="other" TO="other"/>
    <VALENTRY TAG="RoadsExdm2ben.Roads.RoadSign" ATTR="Type"
              FROM="prohibition" TO="prohibition"/>
    <VALENTRY TAG="RoadsExdm2ben.Roads.RoadSign" ATTR="Type"
              FROM="indication" TO="indication"/>
    <VALENTRY TAG="RoadsExdm2ben.Roads.RoadSign" ATTR="Type"
              FROM="danger" TO="danger"/>
    <VALENTRY TAG="RoadsExdm2ben.Roads.RoadSign" ATTR="Type"
              FROM="velocity" TO="velocity"/>
    <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
              FROM="prohibition.noentry" TO="prohibition"/>
    <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
              FROM="prohibition.noparking" TO="prohibition"/>
    <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
              FROM="prohibition.other" TO="prohibition"/>
  </ENTRIES>

  <ENTRIES FOR="RoadsExdm2ien">
    <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended"
              TO="RoadsExdm2ien.RoadsExtended"/>
    <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended.StreetAxis"
              TO="RoadsExdm2ien.RoadsExtended.StreetAxis"/>
    <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended.RoadSign"
              TO="RoadsExdm2ien.RoadsExtended.RoadSign"/>
    <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.StreetAxis" ATTR="Precision"
              FROM="precise" TO="precise"/>
    <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.StreetAxis" ATTR="Precision"
              FROM="unprecise" TO="unprecise"/>
```

```
        <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
                  FROM="prohibition.noentry" TO="prohibition.noentry"/>
        <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
                  FROM="prohibition.noparking" TO="prohibition.noparking"/>
        <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
                  FROM="prohibition.other" TO="prohibition.other"/>
      </ENTRIES>
    </ALIAS>

    <COMMENT>
      example dataset ili2 refmanual appendix C
    </COMMENT>
  </HEADERSECTION>

  <DATASECTION>
    <RoadsExdm2ien.RoadsExtended BID="REFHANDB00000001">

      <!-- === LandCover === -->
      <RoadsExdm2ben.Roads.LandCover TID="16">
        <Type>water</Type>
        <Geometry>
          <SURFACE>
            <BOUNDARY>
              <POLYLINE>
                <LINEATTR>
                  <RoadsExdm2ben.Roads.LAttrs>
                    <LArt>welldefined</LArt>
                  </RoadsExdm2ben.Roads.LAttrs>
                </LINEATTR>
                <COORD><C1>39.038</C1><C2>60.315</C2></COORD>
                <COORD><C1>41.200</C1><C2>59.302</C2></COORD>
                <COORD><C1>43.362</C1><C2>60.315</C2></COORD>
                <COORD><C1>44.713</C1><C2>66.268</C2></COORD>
                <COORD><C1>45.794</C1><C2>67.662</C2></COORD>
                <COORD><C1>48.766</C1><C2>67.408</C2></COORD>
                <COORD><C1>53.360</C1><C2>64.115</C2></COORD>
                <COORD><C1>56.197</C1><C2>62.595</C2></COORD>
                <COORD><C1>57.818</C1><C2>63.862</C2></COORD>
                <COORD><C1>58.899</C1><C2>68.928</C2></COORD>
                <COORD><C1>55.927</C1><C2>72.348</C2></COORD>
                <COORD><C1>47.955</C1><C2>75.515</C2></COORD>
                <COORD><C1>42.281</C1><C2>75.388</C2></COORD>
                <COORD><C1>39.308</C1><C2>73.235</C2></COORD>
                <COORD><C1>36.741</C1><C2>69.688</C2></COORD>
                <COORD><C1>35.525</C1><C2>66.268</C2></COORD>
                <COORD><C1>35.661</C1><C2>63.735</C2></COORD>
                <COORD><C1>37.957</C1><C2>61.455</C2></COORD>
                <COORD><C1>39.038</C1><C2>60.315</C2></COORD>
              </POLYLINE>
            </BOUNDARY>
          </SURFACE>
        </Geometry>
      </RoadsExdm2ben.Roads.LandCover>

      <RoadsExdm2ben.Roads.LandCover TID="18">
        <Type>building</Type>
        <Geometry>
          <SURFACE>
            <BOUNDARY>
              <POLYLINE>
                <LINEATTR>
                  <RoadsExdm2ben.Roads.LAttrs>
                    <LArt>welldefined</LArt>
                  </RoadsExdm2ben.Roads.LAttrs>
                </LINEATTR>
                <COORD><C1>101.459</C1><C2>65.485</C2></COORD>
                <COORD><C1>108.186</C1><C2>69.369</C2></COORD>
                <COORD><C1>102.086</C1><C2>79.936</C2></COORD>
```

```
          <COORD><C1>95.359</C1><C2>76.053</C2></COORD>
          <COORD><C1>101.459</C1><C2>65.485</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
    </SURFACE>
  </Geometry>
</RoadsExdm2ben.Roads.LandCover>

<RoadsExdm2ben.Roads.LandCover TID="20">
  <Type>building</Type>
  <Geometry>
    <SURFACE>
      <BOUNDARY>
        <POLYLINE>
          <LINEATTR>
            <RoadsExdm2ben.Roads.LAttrs>
              <LArt>welldefined</LArt>
            </RoadsExdm2ben.Roads.LAttrs>
          </LINEATTR>
          <COORD><C1>60.489</C1><C2>49.608</C2></COORD>
          <COORD><C1>79.900</C1><C2>55.839</C2></COORD>
          <COORD><C1>75.351</C1><C2>70.932</C2></COORD>
          <COORD><C1>67.678</C1><C2>68.781</C2></COORD>
          <COORD><C1>69.938</C1><C2>61.721</C2></COORD>
          <COORD><C1>57.582</C1><C2>58.029</C2></COORD>
          <COORD><C1>60.489</C1><C2>49.608</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
    </SURFACE>
  </Geometry>
</RoadsExdm2ben.Roads.LandCover>

<RoadsExdm2ben.Roads.LandCover TID="22">
  <Type>street</Type>
  <Geometry>
    <SURFACE>
      <BOUNDARY>
        <POLYLINE>
          <LINEATTR>
            <RoadsExdm2ben.Roads.LAttrs>
              <LArt>welldefined</LArt>
            </RoadsExdm2ben.Roads.LAttrs>
          </LINEATTR>
          <COORD><C1>45.067</C1><C2>58.655</C2></COORD>
          <COORD><C1>50.669</C1><C2>42.579</C2></COORD>
          <COORD><C1>57.060</C1><C2>44.638</C2></COORD>
          <COORD><C1>51.432</C1><C2>60.469</C2></COORD>
          <COORD><C1>45.067</C1><C2>58.655</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
    </SURFACE>
  </Geometry>
</RoadsExdm2ben.Roads.LandCover>

<RoadsExdm2ben.Roads.LandCover TID="24">
  <Type>other</Type>
  <Geometry>
    <SURFACE>
      <BOUNDARY>
        <POLYLINE>
          <LINEATTR>
            <RoadsExdm2ben.Roads.LAttrs>
              <LArt>welldefined</LArt>
            </RoadsExdm2ben.Roads.LAttrs>
          </LINEATTR>
          <COORD><C1>114.027</C1><C2>99.314</C2></COORD>
          <COORD><C1>31.351</C1><C2>99.314</C2></COORD>
          <COORD><C1>31.140</C1><C2>92.530</C2></COORD>
```

```
          <COORD><C1>70.419</C1><C2>86.177</C2></COORD>
          <COORD><C1>86.481</C1><C2>79.710</C2></COORD>
          <COORD><C1>114.027</C1><C2>99.314</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
    </SURFACE>
  </Geometry>
</RoadsExdm2ben.Roads.LandCover>

<RoadsExdm2ben.Roads.LandCover TID="26">
  <Type>other</Type>
  <Geometry>
    <SURFACE>
      <BOUNDARY>
        <POLYLINE>
          <LINEATTR>
            <RoadsExdm2ben.Roads.LAttrs>
              <LArt>welldefined</LArt>
            </RoadsExdm2ben.Roads.LAttrs>
          </LINEATTR>
          <COORD><C1>113.559</C1><C2>62.880</C2></COORD>
          <COORD><C1>114.027</C1><C2>99.314</C2></COORD>
          <COORD><C1>86.481</C1><C2>79.710</C2></COORD>
          <COORD><C1>94.381</C1><C2>66.289</C2></COORD>
          <COORD><C1>96.779</C1><C2>57.177</C2></COORD>
          <COORD><C1>113.559</C1><C2>62.880</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
      <BOUNDARY>
        <POLYLINE>
          <LINEATTR>
            <RoadsExdm2ben.Roads.LAttrs>
              <LArt>welldefined</LArt>
            </RoadsExdm2ben.Roads.LAttrs>
          </LINEATTR>
          <COORD><C1>108.186</C1><C2>69.369</C2></COORD>
          <COORD><C1>101.459</C1><C2>65.485</C2></COORD>
          <COORD><C1>95.359</C1><C2>76.053</C2></COORD>
          <COORD><C1>102.086</C1><C2>79.936</C2></COORD>
          <COORD><C1>108.186</C1><C2>69.369</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
    </SURFACE>
  </Geometry>
</RoadsExdm2ben.Roads.LandCover>

<RoadsExdm2ben.Roads.LandCover TID="29">
  <Type>street</Type>
  <Geometry>
    <SURFACE>
      <BOUNDARY>
        <POLYLINE>
          <LINEATTR>
            <RoadsExdm2ben.Roads.LAttrs>
              <LArt>welldefined</LArt>
            </RoadsExdm2ben.Roads.LAttrs>
          </LINEATTR>
          <COORD><C1>100.621</C1><C2>24.239</C2></COORD>
          <COORD><C1>109.729</C1><C2>24.239</C2></COORD>
          <COORD><C1>105.640</C1><C2>48.068</C2></COORD>
          <COORD><C1>96.779</C1><C2>45.088</C2></COORD>
          <COORD><C1>100.621</C1><C2>24.239</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
    </SURFACE>
  </Geometry>
</RoadsExdm2ben.Roads.LandCover>
```

```
<RoadsExdm2ben.Roads.LandCover TID="31">
  <Type>other</Type>
  <Geometry>
    <SURFACE>
      <BOUNDARY>
        <POLYLINE>
          <LINEATTR>
            <RoadsExdm2ben.Roads.LAttrs>
              <LArt>welldefined</LArt>
            </RoadsExdm2ben.Roads.LAttrs>
          </LINEATTR>
          <COORD><C1>30.900</C1><C2>24.478</C2></COORD>
          <COORD><C1>100.621</C1><C2>24.239</C2></COORD>
          <COORD><C1>96.779</C1><C2>45.088</C2></COORD>
          <COORD><C1>30.900</C1><C2>24.478</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
    </SURFACE>
  </Geometry>
</RoadsExdm2ben.Roads.LandCover>

<RoadsExdm2ben.Roads.LandCover TID="33">
  <Type>other</Type>
  <Geometry>
    <SURFACE>
      <BOUNDARY>
        <POLYLINE>
          <LINEATTR>
            <RoadsExdm2ben.Roads.LAttrs>
              <LArt>welldefined</LArt>
            </RoadsExdm2ben.Roads.LAttrs>
          </LINEATTR>
          <COORD><C1>31.110</C1><C2>83.750</C2></COORD>
          <COORD><C1>31.140</C1><C2>36.458</C2></COORD>
          <COORD><C1>50.669</C1><C2>42.579</C2></COORD>
          <COORD><C1>45.067</C1><C2>58.655</C2></COORD>
          <COORD><C1>51.432</C1><C2>60.469</C2></COORD>
          <COORD><C1>57.060</C1><C2>44.638</C2></COORD>
          <COORD><C1>87.839</C1><C2>54.138</C2></COORD>
          <COORD><C1>85.282</C1><C2>63.410</C2></COORD>
          <COORD><C1>78.847</C1><C2>73.433</C2></COORD>
          <COORD><C1>67.549</C1><C2>77.788</C2></COORD>
          <COORD><C1>31.110</C1><C2>83.750</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
      <BOUNDARY>
        <POLYLINE>
          <LINEATTR>
            <RoadsExdm2ben.Roads.LAttrs>
              <LArt>welldefined</LArt>
            </RoadsExdm2ben.Roads.LAttrs>
          </LINEATTR>
          <COORD><C1>41.200</C1><C2>59.302</C2></COORD>
          <COORD><C1>39.038</C1><C2>60.315</C2></COORD>
          <COORD><C1>37.957</C1><C2>61.455</C2></COORD>
          <COORD><C1>35.661</C1><C2>63.735</C2></COORD>
          <COORD><C1>35.525</C1><C2>66.268</C2></COORD>
          <COORD><C1>36.741</C1><C2>69.688</C2></COORD>
          <COORD><C1>39.308</C1><C2>73.235</C2></COORD>
          <COORD><C1>42.281</C1><C2>75.388</C2></COORD>
          <COORD><C1>47.955</C1><C2>75.515</C2></COORD>
          <COORD><C1>55.927</C1><C2>72.348</C2></COORD>
          <COORD><C1>58.899</C1><C2>68.928</C2></COORD>
          <COORD><C1>57.818</C1><C2>63.862</C2></COORD>
          <COORD><C1>56.197</C1><C2>62.595</C2></COORD>
          <COORD><C1>53.360</C1><C2>64.115</C2></COORD>
          <COORD><C1>48.766</C1><C2>67.408</C2></COORD>
          <COORD><C1>45.794</C1><C2>67.662</C2></COORD>
```

```
            <COORD><C1>44.713</C1><C2>66.268</C2></COORD>
            <COORD><C1>43.362</C1><C2>60.315</C2></COORD>
            <COORD><C1>41.200</C1><C2>59.302</C2></COORD>
          </POLYLINE>
        </BOUNDARY>
        <BOUNDARY>
          <POLYLINE>
            <LINEATTR>
              <RoadsExdm2ben.Roads.LAttrs>
                <LArt>welldefined</LArt>
              </RoadsExdm2ben.Roads.LAttrs>
            </LINEATTR>
            <COORD><C1>79.900</C1><C2>55.839</C2></COORD>
            <COORD><C1>60.489</C1><C2>49.608</C2></COORD>
            <COORD><C1>57.582</C1><C2>58.029</C2></COORD>
            <COORD><C1>69.938</C1><C2>61.721</C2></COORD>
            <COORD><C1>67.678</C1><C2>68.781</C2></COORD>
            <COORD><C1>75.351</C1><C2>70.932</C2></COORD>
            <COORD><C1>79.900</C1><C2>55.839</C2></COORD>
          </POLYLINE>
        </BOUNDARY>
      </SURFACE>
    </Geometry>
  </RoadsExdm2ben.Roads.LandCover>

  <RoadsExdm2ben.Roads.LandCover TID="37">
    <Type>street</Type>
    <Geometry>
      <SURFACE>
        <BOUNDARY>
          <POLYLINE>
            <LINEATTR>
              <RoadsExdm2ben.Roads.LAttrs>
                <LArt>welldefined</LArt>
              </RoadsExdm2ben.Roads.LAttrs>
            </LINEATTR>
            <COORD><C1>31.140</C1><C2>92.530</C2></COORD>
            <COORD><C1>31.110</C1><C2>83.750</C2></COORD>
            <COORD><C1>67.549</C1><C2>77.788</C2></COORD>
            <COORD><C1>78.847</C1><C2>73.433</C2></COORD>
            <COORD><C1>85.282</C1><C2>63.410</C2></COORD>
            <COORD><C1>87.839</C1><C2>54.138</C2></COORD>
            <COORD><C1>96.779</C1><C2>57.177</C2></COORD>
            <COORD><C1>94.381</C1><C2>66.289</C2></COORD>
            <COORD><C1>86.481</C1><C2>79.710</C2></COORD>
            <COORD><C1>70.419</C1><C2>86.177</C2></COORD>
            <COORD><C1>31.140</C1><C2>92.530</C2></COORD>
          </POLYLINE>
        </BOUNDARY>
      </SURFACE>
    </Geometry>
  </RoadsExdm2ben.Roads.LandCover>

  <RoadsExdm2ben.Roads.LandCover TID="39">
    <Type>other</Type>
    <Geometry>
      <SURFACE>
        <BOUNDARY>
          <POLYLINE>
            <LINEATTR>
              <RoadsExdm2ben.Roads.LAttrs>
                <LArt>welldefined</LArt>
              </RoadsExdm2ben.Roads.LAttrs>
            </LINEATTR>
            <COORD><C1>113.811</C1><C2>51.168</C2></COORD>
            <COORD><C1>105.640</C1><C2>48.068</C2></COORD>
            <COORD><C1>109.729</C1><C2>24.239</C2></COORD>
            <COORD><C1>114.269</C1><C2>24.017</C2></COORD>
```

```
          <COORD><C1>113.811</C1><C2>51.168</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
    </SURFACE>
  </Geometry>
</RoadsExdm2ben.Roads.LandCover>

<RoadsExdm2ben.Roads.LandCover TID="41">
  <Type>street</Type>
  <Geometry>
    <SURFACE>
      <BOUNDARY>
        <POLYLINE>
          <LINEATTR>
            <RoadsExdm2ben.Roads.LAttrs>
              <LArt>welldefined</LArt>
            </RoadsExdm2ben.Roads.LAttrs>
          </LINEATTR>
          <COORD><C1>105.640</C1><C2>48.068</C2></COORD>
          <COORD><C1>113.811</C1><C2>51.168</C2></COORD>
          <COORD><C1>113.559</C1><C2>62.880</C2></COORD>
          <COORD><C1>96.779</C1><C2>57.177</C2></COORD>
          <COORD><C1>87.839</C1><C2>54.138</C2></COORD>
          <COORD><C1>57.060</C1><C2>44.638</C2></COORD>
          <COORD><C1>50.669</C1><C2>42.579</C2></COORD>
          <COORD><C1>31.140</C1><C2>36.458</C2></COORD>
          <COORD><C1>30.900</C1><C2>24.478</C2></COORD>
          <COORD><C1>96.779</C1><C2>45.088</C2></COORD>
          <COORD><C1>105.640</C1><C2>48.068</C2></COORD>
        </POLYLINE>
      </BOUNDARY>
    </SURFACE>
  </Geometry>
</RoadsExdm2ben.Roads.LandCover>

<!-- === Street === -->
<RoadsExdm2ben.Roads.Street TID="1">
  <Name>Austrasse</Name>
</RoadsExdm2ben.Roads.Street>

<RoadsExdm2ben.Roads.Street TID="2">
  <Name>Eymattstrasse</Name>
</RoadsExdm2ben.Roads.Street>

<RoadsExdm2ben.Roads.Street TID="3">
  <Name>Feldweg</Name>
</RoadsExdm2ben.Roads.Street>

<RoadsExdm2ben.Roads.Street TID="4">
  <Name>Seeweg</Name>
</RoadsExdm2ben.Roads.Street>

<!-- === StreetAxis / StreetAxisAssoc === -->
<RoadsExdm2ien.RoadsExtended.StreetAxis TID="8">
  <Geometry>
    <POLYLINE>
      <COORD><C1>55.600</C1><C2>37.649</C2></COORD>
      <COORD><C1>15.573</C1><C2>25.785</C2></COORD>
    </POLYLINE>
  </Geometry>
  <Street REF="1"></Street>
  <Precision>precise</Precision>
</RoadsExdm2ien.RoadsExtended.StreetAxis>

<RoadsExdm2ien.RoadsExtended.StreetAxis TID="9">
  <Geometry>
    <POLYLINE>
      <COORD><C1>55.600</C1><C2>37.649</C2></COORD>
```

```
      <COORD><C1>94.990</C1><C2>50.109</C2></COORD>
    </POLYLINE>
  </Geometry>
  <Street REF="1"></Street>
  <Precision>precise</Precision>
</RoadsExdm2ien.RoadsExtended.StreetAxis>

<RoadsExdm2ien.RoadsExtended.StreetAxis TID="10">
  <Geometry>
    <POLYLINE>
      <COORD><C1>94.990</C1><C2>50.109</C2></COORD>
      <COORD><C1>101.099</C1><C2>52.279</C2></COORD>
    </POLYLINE>
  </Geometry>
  <Street REF="1"></Street>
  <Precision>precise</Precision>
</RoadsExdm2ien.RoadsExtended.StreetAxis>

<RoadsExdm2ien.RoadsExtended.StreetAxis TID="11">
  <Geometry>
    <POLYLINE>
      <COORD><C1>101.099</C1><C2>52.279</C2></COORD>
      <COORD><C1>126.100</C1><C2>62.279</C2></COORD>
    </POLYLINE>
  </Geometry>
  <Street REF="1"></Street>
  <Precision>precise</Precision>
</RoadsExdm2ien.RoadsExtended.StreetAxis>

<RoadsExdm2ien.RoadsExtended.StreetAxis TID="12">
  <Geometry>
    <POLYLINE>
      <COORD><C1>94.990</C1><C2>50.109</C2></COORD>
      <COORD><C1>89.504</C1><C2>65.795</C2></COORD>
      <COORD><C1>83.594</C1><C2>75.598</C2></COORD>
      <COORD><C1>71.774</C1><C2>80.712</C2></COORD>
      <COORD><C1>11.423</C1><C2>91.154</C2></COORD>
    </POLYLINE>
  </Geometry>
  <Street REF="2"></Street>
  <Precision>precise</Precision>
</RoadsExdm2ien.RoadsExtended.StreetAxis>

<RoadsExdm2ien.RoadsExtended.StreetAxis TID="13">
  <Geometry>
    <POLYLINE>
      <COORD><C1>101.099</C1><C2>52.279</C2></COORD>
      <COORD><C1>107.400</C1><C2>14.603</C2></COORD>
    </POLYLINE>
  </Geometry>
  <Street REF="3"></Street>
  <Precision>unprecise</Precision>
</RoadsExdm2ien.RoadsExtended.StreetAxis>

<RoadsExdm2ien.RoadsExtended.StreetAxis TID="15">
  <Geometry>
    <POLYLINE>
      <COORD><C1>55.600</C1><C2>37.649</C2></COORD>
      <COORD><C1>49.359</C1><C2>56.752</C2></COORD>
    </POLYLINE>
  </Geometry>
  <Street REF="4"></Street>
  <Precision>unprecise</Precision>
</RoadsExdm2ien.RoadsExtended.StreetAxis>

<!-- === StreetNamePosition / StreetNamePositionAssoc === -->
<RoadsExdm2ben.Roads.StreetNamePosition TID="5">
  <NamPos>
```

```
            <COORD><C1>71.660</C1><C2>45.231</C2></COORD>
          </NamPos>
          <NamOri>15.0</NamOri>
          <Street REF="1"></Street>
        </RoadsExdm2ben.Roads.StreetNamePosition>

        <RoadsExdm2ben.Roads.StreetNamePosition TID="6">
          <NamPos>
            <COORD><C1>58.249</C1><C2>85.081</C2></COORD>
          </NamPos>
          <NamOri>351.0</NamOri>
          <Street REF="2"></Street>
        </RoadsExdm2ben.Roads.StreetNamePosition>

        <RoadsExdm2ben.Roads.StreetNamePosition TID="7">
          <NamPos>
            <COORD><C1>106.095</C1><C2>33.554</C2></COORD>
          </NamPos>
          <NamOri>280.0</NamOri>
          <Street REF="3"></Street>
        </RoadsExdm2ben.Roads.StreetNamePosition>

        <RoadsExdm2ben.Roads.StreetNamePosition TID="14">
          <NamPos>
            <COORD><C1>53.031</C1><C2>51.367</C2></COORD>
          </NamPos>
          <NamOri>291.3</NamOri>
          <Street REF="4"></Street>
        </RoadsExdm2ben.Roads.StreetNamePosition>

        <!-- === RoadSign === -->
        <RoadsExdm2ien.RoadsExtended.RoadSign TID="501">
          <Type>prohibition.noparking</Type>
          <Position>
            <COORD><C1>69.389</C1><C2>92.056</C2></COORD>
          </Position>
        </RoadsExdm2ien.RoadsExtended.RoadSign>

        <RoadsExdm2ien.RoadsExtended.RoadSign TID="502">
          <Type>prohibition.noparking</Type>
          <Position>
            <COORD><C1>80.608</C1><C2>88.623</C2></COORD>
          </Position>
        </RoadsExdm2ien.RoadsExtended.RoadSign>

        <RoadsExdm2ien.RoadsExtended.RoadSign TID="503">
          <Type>prohibition.noparking</Type>
          <Position>
            <COORD><C1>58.059</C1><C2>93.667</C2></COORD>
          </Position>
        </RoadsExdm2ien.RoadsExtended.RoadSign>

        <RoadsExdm2ien.RoadsExtended.RoadSign TID="504">
          <Type>danger</Type>
          <Position>
            <COORD><C1>92.741</C1><C2>38.295</C2></COORD>
          </Position>
        </RoadsExdm2ien.RoadsExtended.RoadSign>
      </RoadsExdm2ien.RoadsExtended>
      <!-- end of basket REFHANDB00000001 -->
    </DATASECTION>
  </TRANSFER>
```

Descripción gráfica RoadsExgm2ien

Con respecto al modelo de datos se define una representación por medio de la descripción gráfica RoadsExgm2ien. El modelo gráfico se lee como sigue:

```
!! File RoadsExgm2ien.ili Release 2005-06-16

INTERLIS 2.3;

CONTRACTED MODEL RoadsExgm2ien (en) AT "http://www.interlis.ch/models"
  VERSION "2005-06-16" = !! Roads graphics

  IMPORTS RoadsExdm2ben;
  IMPORTS RoadsExdm2ien;
  IMPORTS StandardSymbology;

  SIGN BASKET StandardSymbology ~ StandardSymbology.StandardSigns
    OBJECTS OF SurfaceSign: Building, Street, Water, Other
    OBJECTS OF PolylineSign: continuous, dotted
    OBJECTS OF TextSign: Linefont_18
    OBJECTS OF SymbolSign: NoParking, GP;


  TOPIC Graphics =
    DEPENDS ON RoadsExdm2ben.Roads, RoadsExdm2ien.RoadsExtended;

    GRAPHIC Surface_Graphics
      BASED ON RoadsExdm2ien.RoadsExtended.LandCover =

      Building OF StandardSymbology.StandardSigns.SurfaceSign:
        WHERE Type == #building (
          Sign := {Building};
          Geometry := Geometry;
          Priority := 100);

      Street OF StandardSymbology.StandardSigns.SurfaceSign:
        WHERE Type == #street (
          Sign := {Street};
          Geometry := Geometry;
          Priority := 100);

      Water OF StandardSymbology.StandardSigns.SurfaceSign:
        WHERE Type == #water (
          Sign := {Water};
          Geometry := Geometry;
          Priority := 100);

      Other OF StandardSymbology.StandardSigns.SurfaceSign:
        WHERE Type == #other (
          Sign := {Other};
          Geometry := Geometry;
          Priority := 100);

    END Surface_Graphics;

    VIEW Surface_Boundary
      INSPECTION OF RoadsExdm2ien.RoadsExtended.LandCover -> Geometry;
      =
    ATTRIBUTE
      ALL OF LandCover;
    END Surface_Boundary;

    VIEW Surface_Boundary2
      INSPECTION OF Base ~ Surface_Boundary -> Lines;
      =
    ATTRIBUTE
      Geometry := Base -> Geometry;
      LineAttr := Base -> LineAttrs;
    END Surface_Boundary2;

    GRAPHIC SurfaceBoundary_Graphics
      BASED ON Surface_Boundary2 =
```

```
        Boundary OF StandardSymbology.StandardSigns.PolylineSign: (
          Sign := {continuous};
          Geometry := Geometry;
          Priority := 101);

      END SurfaceBoundary_Graphics;

      GRAPHIC Polyline_Graphics
        BASED ON RoadsExdm2ien.RoadsExtended.StreetAxis =

        Street_precise OF StandardSymbology.StandardSigns.PolylineSign:
          WHERE Precision == #precise (
            Sign := {continuous};
            Geometry := Geometry;
            Priority := 110);

        Street_unprecise OF StandardSymbology.StandardSigns.PolylineSign:
          WHERE Precision == #unprecise (
            Sign := {dotted};
            Geometry := Geometry;
            Priority := 110);

      END Polyline_Graphics;

      GRAPHIC Text_Graphics
        BASED ON RoadsExdm2ien.RoadsExtended.StreetNamePosition =

        StreetName OF StandardSymbology.StandardSigns.TextSign: (
            Sign := {Linefont_18};
            Txt := Street -> Name;
            Geometry := NamPos;
            Rotation := NamOri;
            Priority := 120);

      END Text_Graphics;

      GRAPHIC Point_Graphics
        BASED ON RoadsExdm2ien.RoadsExtended.RoadSign =

        Tree OF StandardSymbology.StandardSigns.SymbolSign:
          WHERE Type == #prohibition.noparking (
            Sign := {NoParking};
            Geometry := Position;
            Priority := 130);

        GP OF StandardSymbology.StandardSigns.SymbolSign:
          WHERE Type == #danger (
            Sign := {GP};
            Geometry := Position;
            Priority := 130);

      END Point_Graphics;

    END Graphics;

  END RoadsExgm2ien.
```

El modelo gráfico RoadsExgm2ien tiene recurre a los símbolos en la biblioteca de símbolos Roads-Exgm2ien_Symbols (archivo RoadsExgm2ien_Symbols.xml). Una descripción de la biblioteca de símbolos se encuentra en el siguiente párrafo.

Biblioteca de símbolos RoadsExgm2ien_Symbols.xml

A continuación, se representa la biblioteca de símbolos RoadsExgm2ien_Symbols en forma de conjunto de datos XML (archivo RoadsExgm2ien_Symbols.xml). La biblioteca de símbolos contiene definiciones de

símbolos para puntos de control y árboles, así como símbolos de línea, texto y superficie. El modelo de simbología correspondiente (StandardSymbology) se encuentra en el apéndice J Modelos de simbología.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- File RoadsExgm2ien_Symbols.xml 2005-06-16 (http://www.interlis.ch/models) -->

<TRANSFER xmlns="http://www.interlis.ch/INTERLIS2.3"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.interlis.ch/INTERLIS2.3
                              RoadsExgm2ien_Symbols.xsd">
  <HEADERSECTION VERSION="2.3" SENDER="KOGIS">
    <MODELS>
      <MODEL NAME="RoadsExdm2ben" URI="http://www.interlis.ch/models"
             VERSION="2005-06-16"/>
      <MODEL NAME="RoadsExdm2ien" URI="http://www.interlis.ch/models"
             VERSION="2005-06-16"/>
      <MODEL NAME="AbstractSymbology" URI="http://www.interlis.ch/models"
             VERSION="2005-06-16"/>
      <MODEL NAME="StandardSymbology" URI="http://www.interlis.ch/models"
             VERSION="2005-06-16"/>
      <MODEL NAME="RoadsExgm2ien" URI="http://www.interlis.ch/models"
             VERSION="2005-06-16"/>
    </MODELS>

    <ALIAS>
      <ENTRIES FOR="RoadsExdm2ben">
        <TAGENTRY FROM="RoadsExdm2ben.Roads"
                  TO="RoadsExdm2ben.Roads"/>
        <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended"
                  TO="RoadsExdm2ben.Roads"/>
        <TAGENTRY FROM="RoadsExdm2ben.Roads.LAttrs"
                  TO="RoadsExdm2ben.Roads.LAttrs"/>
        <TAGENTRY FROM="RoadsExdm2ben.Roads.LandCover"
                  TO="RoadsExdm2ben.Roads.LandCover"/>
        <TAGENTRY FROM="RoadsExdm2ben.Roads.Street"
                  TO="RoadsExdm2ben.Roads.Street"/>
        <TAGENTRY FROM="RoadsExdm2ben.Roads.StreetAxis"
                  TO="RoadsExdm2ben.Roads.StreetAxis"/>
        <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended.StreetAxis"
                  TO="RoadsExdm2ben.Roads.StreetAxis"/>
        <DELENTRY TAG="RoadsExdm2ien.RoadsExtended.StreetAxis"
                  ATTR="Precision"/>
        <TAGENTRY FROM="RoadsExdm2ben.Roads.StreetAxisAssoc"
                  TO="RoadsExdm2ben.Roads.StreetAxisAssoc"/>
        <TAGENTRY FROM="RoadsExdm2ben.Roads.StreetNamePosition"
                  TO="RoadsExdm2ben.Roads.StreetNamePosition"/>
        <TAGENTRY FROM="RoadsExdm2ben.Roads.StreetNamePositionAssoc"
                  TO="RoadsExdm2ben.Roads.StreetNamePositionAssoc"/>
        <TAGENTRY FROM="RoadsExdm2ben.Roads.RoadSign"
                  TO="RoadsExdm2ben.Roads.RoadSign"/>
        <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended.RoadSign"
                  TO="RoadsExdm2ben.Roads.RoadSign"/>
        <VALENTRY TAG="RoadsExdm2ben.Roads.LAttrs" ATTR="LArt"
                  FROM="welldefined" TO="welldefined"/>
        <VALENTRY TAG="RoadsExdm2ben.Roads.LAttrs" ATTR="LArt"
                  FROM="fuzzy" TO="fuzzy"/>
        <VALENTRY TAG="RoadsExdm2ben.Roads.LandCover" ATTR="Type"
                  FROM="building" TO="building"/>
        <VALENTRY TAG="RoadsExdm2ben.Roads.LandCover" ATTR="Type"
                  FROM="street" TO="street"/>
        <VALENTRY TAG="RoadsExdm2ben.Roads.LandCover" ATTR="Type"
                  FROM="water" TO="water"/>
        <VALENTRY TAG="RoadsExdm2ben.Roads.LandCover" ATTR="Type"
                  FROM="other" TO="other"/>
        <VALENTRY TAG="RoadsExdm2ben.Roads.RoadSign" ATTR="Type"
                  FROM="prohibition" TO="prohibition"/>
        <VALENTRY TAG="RoadsExdm2ben.Roads.RoadSign" ATTR="Type"
```

```
            FROM="indication" TO="indication"/>
  <VALENTRY TAG="RoadsExdm2ben.Roads.RoadSign" ATTR="Type"
            FROM="danger" TO="danger"/>
  <VALENTRY TAG="RoadsExdm2ben.Roads.RoadSign" ATTR="Type"
            FROM="velocity" TO="velocity"/>
  <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
            FROM="prohibition.noentry" TO="prohibition"/>
  <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
            FROM="prohibition.noparking" TO="prohibition"/>
  <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
            FROM="prohibition.other" TO="prohibition"/>
</ENTRIES>

<ENTRIES FOR="RoadsExdm2ien">
  <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended"
            TO="RoadsExdm2ien.RoadsExtended"/>
  <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended.StreetAxis"
            TO="RoadsExdm2ien.RoadsExtended.StreetAxis"/>
  <TAGENTRY FROM="RoadsExdm2ien.RoadsExtended.RoadSign"
            TO="RoadsExdm2ien.RoadsExtended.RoadSign"/>
  <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
            FROM="prohibition.noentry" TO="prohibition.noentry"/>
  <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
            FROM="prohibition.noparking" TO="prohibition.noparking"/>
  <VALENTRY TAG="RoadsExdm2ien.RoadsExtended.RoadSign" ATTR="Type"
            FROM="prohibition.other" TO="prohibition.other"/>
</ENTRIES>

<ENTRIES FOR="AbstractSymbology">
  <TAGENTRY FROM="AbstractSymbology.Signs"
            TO="AbstractSymbology.Signs"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns"
            TO="AbstractSymbology.Signs"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.Color"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.PolylineAttrs"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.FontSymbol_Polyline"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.FontSymbol_Surface"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.FontSymbol"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.Font"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.FontAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.LineStyle_Solid"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.LineStyle_SolidColorAssoc"/>
  <DELENTRY TAG=
    "StandardSymbology.StandardSigns.LineStyle_SolidPolylineAttrsAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.DashRec"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.LineStyle_Dashed"/>
  <DELENTRY TAG=
    "StandardSymbology.StandardSigns.LineStyle_DashedColorAssoc"/>
  <DELENTRY TAG=
    "StandardSymbology.StandardSigns.LineStyle_DashedLineAttrsAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.Pattern_Symbol"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.LineStyle_Pattern"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.TextSign"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.TextSignFontAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.TextSignColorAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.TextSignClipFontAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.SymbolSign"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.SymbolSignSymbolAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.SymbolSignClipSymbolAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.SymbolSignColorAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.PolylineSign"/>
  <DELENTRY TAG=
    "StandardSymbology.StandardSigns.PolylineSignLineStyleAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.PolylineSignColorAssoc"/>
  <DELENTRY TAG=
    "StandardSymbology.StandardSigns.PolylineSignClipStyleAssoc"/>
  <DELENTRY TAG=
    "StandardSymbology.StandardSigns.PolylineSignStartSymbolAssoc"/>
```

```
  <DELENTRY TAG=
    "StandardSymbology.StandardSigns.PolylineSignEndSymbolAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.SurfaceSign"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.SurfaceSignColorAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.SurfaceSignBorderAssoc"/>
  <DELENTRY TAG="StandardSymbology.StandardSigns.SurfaceSignHatchSymbAssoc"/>
</ENTRIES>

<ENTRIES FOR="StandardSymbology">
  <TAGENTRY FROM="StandardSymbology.StandardSigns"
            TO="StandardSymbology.StandardSigns"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.Color"
            TO="StandardSymbology.StandardSigns.Color"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.PolylineAttrs"
            TO="StandardSymbology.StandardSigns.PolylineAttrs"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.FontSymbol_Polyline"
            TO="StandardSymbology.StandardSigns.FontSymbol_Polyline"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.FontSymbol_Surface"
            TO="StandardSymbology.StandardSigns.FontSymbol_Surface"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.FontSymbol"
            TO="StandardSymbology.StandardSigns.FontSymbol"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.Font"
            TO="StandardSymbology.StandardSigns.Font"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.FontAssoc"
            TO="StandardSymbology.StandardSigns.FontAssoc"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.LineStyle_Solid"
            TO="StandardSymbology.StandardSigns.LineStyle_Solid"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.LineStyle_SolidColorAssoc"
            TO="StandardSymbology.StandardSigns.LineStyle_SolidColorAssoc"/>
  <TAGENTRY FROM=
    "StandardSymbology.StandardSigns.LineStyle_SolidPolylineAttrsAssoc"
            TO=
    "StandardSymbology.StandardSigns.LineStyle_SolidPolylineAttrsAssoc"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.DashRec"
            TO="StandardSymbology.StandardSigns.DashRec"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.LineStyle_Dashed"
            TO="StandardSymbology.StandardSigns.LineStyle_Dashed"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.LineStyle_DashedColorAssoc"
            TO="StandardSymbology.StandardSigns.LineStyle_DashedColorAssoc"/>
  <TAGENTRY FROM=
    "StandardSymbology.StandardSigns.LineStyle_DashedLineAttrsAssoc"
            TO=
    "StandardSymbology.StandardSigns.LineStyle_DashedLineAttrsAssoc"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.Pattern_Symbol"
            TO="StandardSymbology.StandardSigns.Pattern_Symbol"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.LineStyle_Pattern"
            TO="StandardSymbology.StandardSigns.LineStyle_Pattern"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.TextSign"
            TO="StandardSymbology.StandardSigns.TextSign"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.TextSignFontAssoc"
            TO="StandardSymbology.StandardSigns.TextSignFontAssoc"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.TextSignColorAssoc"
            TO="StandardSymbology.StandardSigns.TextSignColorAssoc"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.TextSignClipFontAssoc"
            TO="StandardSymbology.StandardSigns.TextSignClipFontAssoc"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.SymbolSign"
            TO="StandardSymbology.StandardSigns.SymbolSign"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.SymbolSignSymbolAssoc"
            TO="StandardSymbology.StandardSigns.SymbolSignSymbolAssoc"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.SymbolSignClipSymbolAssoc"
            TO="StandardSymbology.StandardSigns.SymbolSignClipSymbolAssoc"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.SymbolSignColorAssoc"
            TO="StandardSymbology.StandardSigns.SymbolSignColorAssoc"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.PolylineSign"
            TO="StandardSymbology.StandardSigns.PolylineSign"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.PolylineSignLineStyleAssoc"
            TO="StandardSymbology.StandardSigns.PolylineSignLineStyleAssoc"/>
  <TAGENTRY FROM="StandardSymbology.StandardSigns.PolylineSignColorAssoc"
```

```
                TO="StandardSymbology.StandardSigns.PolylineSignColorAssoc"/>
      <TAGENTRY FROM="StandardSymbology.StandardSigns.PolylineSignClipStyleAssoc"
                TO="StandardSymbology.StandardSigns.PolylineSignClipStyleAssoc"/>
      <TAGENTRY FROM=
        "StandardSymbology.StandardSigns.PolylineSignStartSymbolAssoc"
                TO=
        "StandardSymbology.StandardSigns.PolylineSignStartSymbolAssoc"/>
      <TAGENTRY FROM="StandardSymbology.StandardSigns.PolylineSignEndSymbolAssoc"
                TO="StandardSymbology.StandardSigns.PolylineSignEndSymbolAssoc"/>
      <TAGENTRY FROM="StandardSymbology.StandardSigns.SurfaceSign"
                TO="StandardSymbology.StandardSigns.SurfaceSign"/>
      <TAGENTRY FROM="StandardSymbology.StandardSigns.SurfaceSignColorAssoc"
                TO="StandardSymbology.StandardSigns.SurfaceSignColorAssoc"/>
      <TAGENTRY FROM="StandardSymbology.StandardSigns.SurfaceSignBorderAssoc"
                TO="StandardSymbology.StandardSigns.SurfaceSignBorderAssoc"/>
      <TAGENTRY FROM="StandardSymbology.StandardSigns.SurfaceSignHatchSymbAssoc"
                TO="StandardSymbology.StandardSigns.SurfaceSignHatchSymbAssoc"/>
      <VALENTRY TAG="StandardSymbology.StandardSigns.PolylineAttrs" ATTR="Join"
                FROM="bevel" TO="bevel"/>
      <VALENTRY TAG="StandardSymbology.StandardSigns.PolylineAttrs" ATTR="Join"
                FROM="round" TO="round"/>
      <VALENTRY TAG="StandardSymbology.StandardSigns.PolylineAttrs" ATTR="Join"
                FROM="miter" TO="miter"/>
      <VALENTRY TAG="StandardSymbology.StandardSigns.PolylineAttrs" ATTR="Caps"
                FROM="round" TO="round"/>
      <VALENTRY TAG="StandardSymbology.StandardSigns.PolylineAttrs" ATTR="Caps"
                FROM="butt" TO="butt"/>
      <VALENTRY TAG="StandardSymbology.StandardSigns.Font" ATTR="Type"
                FROM="symbol" TO="symbol"/>
      <VALENTRY TAG="StandardSymbology.StandardSigns.Font" ATTR="Type"
                FROM="text" TO="text"/>
      <VALENTRY TAG="StandardSymbology.StandardSigns.SurfaceSign" ATTR="Clip"
                FROM="inside" TO="inside"/>
      <VALENTRY TAG="StandardSymbology.StandardSigns.SurfaceSign" ATTR="Clip"
                FROM="outside" TO="outside"/>
    </ENTRIES>

    <ENTRIES FOR="RoadsExgm2ien">
      <TAGENTRY FROM="RoadsExgm2ien.Graphics"
                TO="RoadsExgm2ien.Graphics"/>
    </ENTRIES>
  </ALIAS>

  <COMMENT>
    example symbology dataset ili2 refmanual appendix C
  </COMMENT>
</HEADERSECTION>

<DATASECTION>
  <StandardSymbology.StandardSigns BID="REFHANDB00000002">

    <!-- Color Library -->
    <StandardSymbology.StandardSigns.Color TID="1">
      <Name>red</Name>
      <L>40.0</L>
      <C>70.0</C>
      <H>0.0</H>
      <T>1.0</T>
    </StandardSymbology.StandardSigns.Color>

    <StandardSymbology.StandardSigns.Color TID="2">
      <Name>green</Name>
      <L>49.4</L>
      <C>48.5</C>
      <H>153.36</H>
      <T>1.0</T>
    </StandardSymbology.StandardSigns.Color>
```

```
<StandardSymbology.StandardSigns.Color TID="3">
  <Name>light_gray</Name>
  <L>75.0</L>
  <C>0.0</C>
  <H>0.0</H>
  <T>1.0</T>
</StandardSymbology.StandardSigns.Color>

<StandardSymbology.StandardSigns.Color TID="4">
  <Name>dark_grey</Name>
  <L>25.0</L>
  <C>0.0</C>
  <H>0.0</H>
  <T>1.0</T>
</StandardSymbology.StandardSigns.Color>

<StandardSymbology.StandardSigns.Color TID="5">
  <Name>dark_blue</Name>
  <L>50.3</L>
  <C>43.5</C>
  <H>261.1</H>
  <T>1.0</T>
</StandardSymbology.StandardSigns.Color>

<StandardSymbology.StandardSigns.Color TID="6">
  <Name>black</Name>
  <L>0.0</L>
  <C>0.0</C>
  <H>0.0</H>
  <T>1.0</T>
</StandardSymbology.StandardSigns.Color>

<StandardSymbology.StandardSigns.Color TID="7">
  <Name>white</Name>
  <L>100.0</L>
  <C>0.0</C>
  <H>0.0</H>
  <T>1.0</T>
</StandardSymbology.StandardSigns.Color>

<StandardSymbology.StandardSigns.PolylineAttrs TID="4001">
  <Width>0.01</Width>
  <Join>round</Join>
  <Caps>butt</Caps>
</StandardSymbology.StandardSigns.PolylineAttrs>

<StandardSymbology.StandardSigns.PolylineAttrs TID="4002">
  <Width>0.01</Width>
  <Join>miter</Join>
  <MiterLimit>2.0</MiterLimit>
  <Caps>butt</Caps>
</StandardSymbology.StandardSigns.PolylineAttrs>

<!-- Font/Symbol Library -->
<StandardSymbology.StandardSigns.FontSymbol TID="101">
  <Name>Triangle</Name>
  <Geometry>
    <StandardSymbology.StandardSigns.FontSymbol_Surface>
      <Geometry>
        <SURFACE>
          <BOUNDARY>
            <POLYLINE>
              <COORD><C1>-0.5</C1><C2>-0.5</C2></COORD>
              <COORD><C1>0.0</C1><C2>0.5</C2></COORD>
              <COORD><C1>0.5</C1><C2>-0.5</C2></COORD>
            </POLYLINE>
          </BOUNDARY>
        </SURFACE>
```

```
        </Geometry>
      </StandardSymbology.StandardSigns.FontSymbol_Surface>
      <StandardSymbology.StandardSigns.FontSymbol_Polyline>
        <Color REF="6"></Color>
        <LineAttrs REF="4001"></LineAttrs>
        <Geometry>
          <POLYLINE>
            <COORD><C1>-0.5</C1><C2>0.0</C2></COORD>
            <ARC><C1>0.5</C1><C2>0.0</C2>
              <A1>0.0</A1><A2>0.5</A2><R>0.5</R>
            </ARC>
            <ARC><C1>-0.5</C1><C2>0.0</C2>
              <A1>0.0</A1><A2>-0.5</A2><R>0.5</R>
            </ARC>
          </POLYLINE>
        </Geometry>
      </StandardSymbology.StandardSigns.FontSymbol_Polyline>
    </Geometry>
    <Font REF="10"></Font>
  </StandardSymbology.StandardSigns.FontSymbol>

  <StandardSymbology.StandardSigns.FontSymbol TID="102">
    <Name>NoParking</Name>
    <Geometry>
      <StandardSymbology.StandardSigns.FontSymbol_Polyline>
        <Color REF="6"></Color>
        <LineAttrs REF="4001"></LineAttrs>
        <Geometry>
          <POLYLINE>
            <COORD><C1>-0.5</C1><C2>0.0</C2></COORD>
            <ARC><C1>0.5</C1><C2>0.0</C2>
              <A1>0.0</A1><A2>0.5</A2><R>0.5</R>
            </ARC>
            <ARC><C1>-0.5</C1><C2>0.0</C2>
              <A1>0.0</A1><A2>-0.5</A2><R>0.5</R>
            </ARC>
          </POLYLINE>
        </Geometry>
      </StandardSymbology.StandardSigns.FontSymbol_Polyline>
      <StandardSymbology.StandardSigns.FontSymbol_Surface>
        <FillColor REF="1"></FillColor>
        <Geometry>
          <SURFACE>
            <BOUNDARY>
              <POLYLINE>
                <COORD><C1>-0.233</C1><C2>0.325</C2></COORD>
                <ARC><C1>0.325</C1><C2>-0.233</C2>
                  <A1>0.283</A1><A2>0.283</A2><R>0.4</R>
                </ARC>
                <COORD><C1>-0.233</C1><C2>0.325</C2></COORD>
              </POLYLINE>
            </BOUNDARY>
          </SURFACE>
        </Geometry>
      </StandardSymbology.StandardSigns.FontSymbol_Surface>
      <StandardSymbology.StandardSigns.FontSymbol_Surface>
        <FillColor REF="1"></FillColor>
        <Geometry>
          <SURFACE>
            <BOUNDARY>
              <POLYLINE>
                <COORD><C1>0.228</C1><C2>-0.324</C2></COORD>
                <ARC><C1>-0.327</C1><C2>0.238</C2>
                  <A1>-0.283</A1><A2>-0.283</A2><R>0.4</R>
                </ARC>
                <COORD><C1>0.228</C1><C2>-0.324</C2></COORD>
              </POLYLINE>
            </BOUNDARY>
```

```
          </SURFACE>
        </Geometry>
      </StandardSymbology.StandardSigns.FontSymbol_Surface>
      <StandardSymbology.StandardSigns.FontSymbol_Surface>
        <FillColor REF="5"></FillColor>
        <Geometry>
          <SURFACE>
            <BOUNDARY>
              <POLYLINE>
                <COORD><C1>-0.5</C1><C2>0.0</C2></COORD>
                <ARC><C1>0.5</C1><C2>0.0</C2>
                  <A1>0.0</A1><A2>0.5</A2><R>0.5</R>
                </ARC>
                <ARC><C1>-0.5</C1><C2>0.0</C2>
                  <A1>0.0</A1><A2>-0.5</A2><R>0.5</R>
                </ARC>
              </POLYLINE>
            </BOUNDARY>
            <BOUNDARY>
              <POLYLINE>
                <COORD><C1>-0.233</C1><C2>0.325</C2></COORD>
                <ARC><C1>0.325</C1><C2>-0.233</C2>
                  <A1>0.283</A1><A2>0.283</A2><R>0.4</R>
                </ARC>
                <COORD><C1>-0.233</C1><C2>0.325</C2></COORD>
              </POLYLINE>
            </BOUNDARY>
            <BOUNDARY>
              <POLYLINE>
                <COORD><C1>0.228</C1><C2>-0.324</C2></COORD>
                <ARC><C1>-0.327</C1><C2>0.238</C2>
                  <A1>-0.283</A1><A2>-0.283</A2><R>0.4</R>
                </ARC>
                <COORD><C1>0.228</C1><C2>-0.324</C2></COORD>
              </POLYLINE>
            </BOUNDARY>
          </SURFACE>
        </Geometry>
      </StandardSymbology.StandardSigns.FontSymbol_Surface>
      <StandardSymbology.StandardSigns.FontSymbol_Polyline>
        <Color REF="7"></Color>
        <LineAttrs REF="4001"></LineAttrs>
        <Geometry>
          <POLYLINE>
            <COORD><C1>-0.233</C1><C2>0.325</C2></COORD>
            <ARC><C1>0.325</C1><C2>-0.233</C2>
              <A1>0.283</A1><A2>0.283</A2><R>0.4</R>
            </ARC>
            <COORD><C1>-0.233</C1><C2>0.325</C2></COORD>
          </POLYLINE>
        </Geometry>
      </StandardSymbology.StandardSigns.FontSymbol_Polyline>
      <StandardSymbology.StandardSigns.FontSymbol_Polyline>
        <Color REF="7"></Color>
        <LineAttrs REF="4001"></LineAttrs>
        <Geometry>
          <POLYLINE>
            <COORD><C1>0.228</C1><C2>-0.324</C2></COORD>
            <ARC><C1>-0.327</C1><C2>0.238</C2>
              <A1>-0.283</A1><A2>-0.283</A2><R>0.4</R>
            </ARC>
            <COORD><C1>0.228</C1><C2>-0.324</C2></COORD>
          </POLYLINE>
        </Geometry>
      </StandardSymbology.StandardSigns.FontSymbol_Polyline>
    </Geometry>
    <Font REF="10"></Font>
  </StandardSymbology.StandardSigns.FontSymbol>
```

```
<!-- Internal Symbol Font "Symbols" -->
<StandardSymbology.StandardSigns.Font TID="10">
  <Name>Symbols</Name>
  <Internal>true</Internal>
  <Type>symbol</Type>
</StandardSymbology.StandardSigns.Font>

<!-- External Text Font "Leroy" -->
<StandardSymbology.StandardSigns.Font TID="11">
  <Name>Leroy</Name>
  <Internal>false</Internal>
  <Type>text</Type>
  <BottomBase>0.3</BottomBase>
</StandardSymbology.StandardSigns.Font>

<!-- LineStyles -->
<StandardSymbology.StandardSigns.LineStyle_Solid TID="21">
  <Name>LineSolid_01</Name>
  <Color REF="6"></Color>
  <LineAttrs REF="4001"></LineAttrs>
</StandardSymbology.StandardSigns.LineStyle_Solid>

<StandardSymbology.StandardSigns.LineStyle_Dashed TID="22">
  <Name>LineDashed_01</Name>
  <Dashes>
    <StandardSymbology.StandardSigns.DashRec>
      <DLength>0.1</DLength>
    </StandardSymbology.StandardSigns.DashRec>
    <StandardSymbology.StandardSigns.DashRec>
      <DLength>0.1</DLength>
    </StandardSymbology.StandardSigns.DashRec>
  </Dashes>
  <Color REF="6"></Color>
  <LineAttrs REF="4002"></LineAttrs>
</StandardSymbology.StandardSigns.LineStyle_Dashed>

<!-- Text Signs -->
<StandardSymbology.StandardSigns.TextSign TID="1001">
  <Name>Linefont_18</Name>
  <Height>1.8</Height>
  <Font REF="11"></Font>
</StandardSymbology.StandardSigns.TextSign>

<!-- Symbol Signs -->
<StandardSymbology.StandardSigns.SymbolSign TID="2001">
  <Name>GP</Name>
  <Scale>1.0</Scale>
  <Color REF="2"></Color>
  <Symbol REF="101"></Symbol>
</StandardSymbology.StandardSigns.SymbolSign>

<StandardSymbology.StandardSigns.SymbolSign TID="2002">
  <Name>NoParking</Name>
  <Scale>1.0</Scale>
  <Symbol REF="102"></Symbol>
</StandardSymbology.StandardSigns.SymbolSign>

<!-- Polyline Signs -->
<StandardSymbology.StandardSigns.PolylineSign TID="3001">
  <Name>continuous</Name>
  <Style REF="21">
    <StandardSymbology.StandardSigns.PolylineSignLineStyleAssoc>
      <Offset>0.0</Offset>
    </StandardSymbology.StandardSigns.PolylineSignLineStyleAssoc>
  </Style>
</StandardSymbology.StandardSigns.PolylineSign>
```

```
        <StandardSymbology.StandardSigns.PolylineSign TID="3002">
          <Name>dotted</Name>
          <Style REF="22">
            <StandardSymbology.StandardSigns.PolylineSignLineStyleAssoc>
              <Offset>0.0</Offset>
            </StandardSymbology.StandardSigns.PolylineSignLineStyleAssoc>
          </Style>
        </StandardSymbology.StandardSigns.PolylineSign>

        <!-- Surface Signs -->
        <StandardSymbology.StandardSigns.SurfaceSign TID="5001">
          <Name>Building</Name>
          <FillColor REF="4"></FillColor>
        </StandardSymbology.StandardSigns.SurfaceSign>

        <StandardSymbology.StandardSigns.SurfaceSign TID="5002">
          <Name>Street</Name>
          <FillColor REF="3"></FillColor>
        </StandardSymbology.StandardSigns.SurfaceSign>

        <StandardSymbology.StandardSigns.SurfaceSign TID="5003">
          <Name>Water</Name>
          <FillColor REF="5"></FillColor>
        </StandardSymbology.StandardSigns.SurfaceSign>

        <StandardSymbology.StandardSigns.SurfaceSign TID="5005">
          <Name>Other</Name>
          <FillColor REF="2"></FillColor>
        </StandardSymbology.StandardSigns.SurfaceSign>
      </StandardSymbology.StandardSigns>
      <!-- end of basket REFHANDB00000002 -->
    </DATASECTION>
  </TRANSFER>
```

Representación gráfica de nuestro ejemplo

Combinando la información proporcionada por el conjunto de datos RoadsExdm2ien (archivo RoadsExdm2ien.xml), las descripciones en el modelo gráfico RoadsExgm2ien (archivo RoadsExgm2ien.ili) y la biblioteca de símbolos RoadsExgm2ien_Symbols (archivo RoadsExgm2ien_Symbols.xml), un procesador gráfico de INTERLIS 2 generará el siguiente gráfico:

Figura 27: Gráfico generado a partir de descripciones gráficas y datos.

Apéndice D (sugerencia de extensión estándar)
Organización de los identificadores de objeto (OID)

Nota preliminar

La siguiente especificación no es un componente normativo de INTERLIS. Esta es una sugerencia de extensión estándar basada en el Manual de Referencia de INTERLIS Versión 2 en el sentido de una recomendación. Sin embargo, se tiene la intención de ponerlo en discusión y posiblemente convertirlo en una regulación más definida. Consulte los correspondientes manuales de INTERLIS 2 para obtener ejemplos de aplicación.

Introducción

La disponibilidad cada vez mayor de geodatos, exige su actualización e integración en diversas bases de datos. Estas son algunas de las razones por las que existe una demanda de regulaciones uniformes relativas a los identificadores de objetos (OID): Un OID identificará una instancia de objeto desde su inicio hasta su fin, incluso si los valores de atributo deben alterarse. A diferencia de las claves de usuario (véase el apéndice E, Unicidad de claves de usuario en el Manual de referencia de INTERLIS 2), el usuario debe considerar un OID como un atributo silencioso ("opaco") que normalmente será administrado por las funciones del sistema.

Al menos dentro de una comunidad de transferencia, un OID debe ser único, inequívoco e inmutable.

Se realizan las siguientes demandas, entre otras, sobre la generación y utilización de OID:

- El OID es un identificador general y estable, incluso con grandes cantidades de datos. Como identificador es un atributo cuyo valor designa inequívocamente un objeto en su clase. Al ser un identificador general, su valor no sólo designa claramente un objeto dentro de su clase sino dentro de todas las clases de una comunidad de transferencia. Además, siendo un identificador estable, es independiente del tiempo, es decir, durante el ciclo de vida de un objeto no se puede modificar y el OID de cualquier objeto eliminado ya no se puede utilizar.
- Es independiente de los productores de hardware y software.
- Es independiente de plataformas.
- Es utilizable tanto para usuarios múltiples como individuales, o en sistemas autónomos (por ejemplo, en el trabajo de campo).
- Se requiere poco espacio, pero si se necesita, se puede usar más para ser optimizado.
- Fácil de implementar.

Otras demandas posibles no son necesariamente de naturaleza técnica, por ejemplo, un mínimo de gastos en organización, bajo control nacional, también utilizable con sistemas antiguos y aprobación de proveedores de sistemas. Estas son altas exigencias que apuntan parcialmente a direcciones opuestas. Un requisito especial establece que un OID puede ser colocado, al menos, 10 millones de veces por un sistema en producción; también, que el OID tiene una longitud determinada para facilitar su manipulación (excluyendo así otros procedimientos bien conocidos, tales como el denominado URI como un prefijo). No hay llamada para números de control, se supone que los niveles de comunicación más bajos proporcionan las herramientas necesarias.

En principio, la singularidad de un OID siempre se logrará a través de un mecanismo central. Los dos extremos, es decir, la colocación de cada OID a través de una autoridad central, por un lado, y la generación completamente descentralizada y autónoma de OID por el otro, conducen a resultados insatisfactorios. Obtener el OID a partir de una dirección MAC de un adaptador de red y de un sello de

tiempo, por ejemplo, no se considera práctico ni muy prometedor, ya que significaría que cada equipo esté equipado con un MAC y no es posible prever si esta tecnología no estará anticuada en unos años.

Hay una larga historia en el desarrollo de esta especificación. Con los años se han realizado estudios, conferencias y revisiones. Algunos de los documentos establecidos en su curso están a disposición de los interesados (haga su pedido en www.interlis.ch, info@interlis.ch).

Estructura de un identificador de objeto (OID)

Un identificador de objeto (OID) consiste en un prefijo y un sufijo y tiene una longitud en conjunto de 16 caracteres alfanuméricos. Un OID siempre se trata como una unidad, sobre todo en la interfaz de datos y en el usuario. El dominio OID STANDARDOID del modelo INTERLIS corresponde precisamente a esta definición. Sin embargo, sólo define toda su longitud y no su estructura detallada.

```
OIDDef = Prefix Postfix.
```

Prefijo

Una autoridad central genera el prefijo. Por lo tanto, la unicidad está garantizada dentro de una comunidad de transferencia. Normalmente, cada contenedor (es decir, un proceso de base de datos, que administra datos de un tema concreto) exige un nuevo prefijo. Es el país del prefijo el que crea el proceso considerado como destino del prefijo. Este proceso no se encuentra automáticamente en el mismo lugar que el sistema productor que crea el OID completo.

Un prefijo consta de 8 caracteres, siendo admisibles los siguientes símbolos:

```
Prefix = Letter { Letter | Digit }.  !! sequence of 8 characters
Letter = ( 'A' | .. | 'Z' | 'a' | .. | 'z' ).
Digit = ( '0' | '1' | .. | '9' ).
```

Un prefijo se define como una secuencia de letras y dígitos, el primer símbolo tiene que ser una letra (véase también la estructura de los nombres de las etiquetas XML o los nombres de los capítulos en el Manual de Referencia de la Versión 2 de INTERLIS).

Además, los dos primeros prefijos tienen que ser determinados de acuerdo a los códigos de país de la Norma ISO 3166. Así se han seleccionado las letras "ch" para todos los prefijos creados en Suiza, "de" para Alemania, "at" para Austria etc. Para la creación de un prefijo, 62 variedades diferentes están disponibles por carácter (0..9: ASCII 48 a 57; A..Z: ASCII 65 a 90; a..z: ASCII 97 a 122). La combinación de 62 símbolos con el número de 10 caracteres resulta en un número que excede las exigencias probables de la mayoría de las aplicaciones en la actualidad concebibles.

Sufijo

Un postfijo es creado por el productor de datos, así como por el propio sistema productor. Se compone de 8 caracteres, compatible con ASCII; El enfoque orientado a columnas exige que los posibles caracteres "vacíos" a la izquierda se llenen con ceros ("0") (ver ejemplos 1 y 3 de un OID a continuación). Así, el valor ordinal más pequeño posible de la parte de sufijo se representa como "00000000"..

```
Postfix = { Letter | Digit }.  !! sequence of 8 characters
```

Si es necesario, se pueden definir más restricciones en la parte del prefijo o del sufijo en especificaciones adicionales.

Resumen y ejemplos de aplicaciones

OID	*Length*	*Significancia*	*Notas*
Prefijo	2 + 6 Char.	Especificación de un país +, una parte de identificación única y global, asignada por una autoridad central.	especificación país inequívoca mundial por ejemplo, DE (Alemania), AT (Austria), CH (Suiza) de acuerdo con la norma ISO-3166 norma. Otras restricciones requieren especificaciones adicionales.
Sufijo	8 Char.	Secuencia (numérica o alfanumérica) del sistema de producción como parte de identificación "local"	Otras restricciones, como por ejemplo sello de fecha con número de secuencia requieren especificaciones adicionales.

Ejemplos

```
1234567812345678    Comment
---------------- -------

A000000000000000        En teoría, el OID más pequeño posible.

zwzzzzzzzzzzzzz         El OID máximo posible con zw para Zimbabue.

deg5mQXX2000004a        OID de origen germano (de) generado aleatoriamente.

chgAAAAAAAAA0azD        OID de origen suizo (ch) generado aleatoriamente.
```

Organización

Alguna autoridad (posiblemente federal), universalmente reconocida por la comunidad de transferencia, mantiene un servicio central encargado de la generación de OID. A través de canales de comunicación adecuados, los productores de datos pueden obtener uno o varios prefijos. Esto podría ser, por ejemplo, una página de Internet conectada con un servicio de correo electrónico. Tal servicio puede hacerse relativamente seguro y protegido contra abusos.

Corresponde a la implementación del sistema fuente y objetivo utilizar las características explícitamente indicadas en esta especificación y utilizar un OID apropiadamente, por ejemplo, para la clasificación u optimización interna. La administración de la parte del prefijo dentro del sistema en una localización central puede conseguir dicha optimización. Además, los diferentes objetos contendrían solamente la parte del sufijo y además una relación a una parte del prefijo común. Puede economizarse de otra manera si la parte del sufijo se memoriza internamente en el sistema como número binario.

Para practicar ejercicios use:

- The OID-prefix "chB00000" with OID's for baskets.
- El -prefijo-OID "ch100000" con el resto de OID.

Apéndice E (sugerencia de extensión estándar)
Unicidad de las claves de usuario

Nota

La siguiente especificación no es un componente normativo de INTERLIS. Esta es una sugerencia de extensión estándar basada en el Manual de Referencia de INTERLIS Versión 2 en el sentido de una recomendación. Sin embargo, se tiene la intención de ponerlo bajo discusión y posiblemente convertirlo en una regulación más definida. Consulte los correspondientes manuales de INTERLIS 2 para obtener ejemplos de aplicación.

Alternativas de modelado

Si la unicidad es un requisito en las claves de usuario, surge la pregunta de cuáles son los límites dentro de los que se aplica esta singularidad. Desde un punto de vista puramente técnico, a menudo es obvio que la unicidad sólo puede garantizarse dentro de un contenedor específico, ya que todos los demás no son accesibles. Sin embargo, desde el punto de vista del modelado, un contenedor carece de sentido mientras no se pueda hacer ninguna declaración sobre su alcance.

Es dudoso, incluso, si la unicidad es necesaria en un modelo base. Tal como lo ve una autoridad superior (por ejemplo, la Federación), es perfectamente concebible que la unicidad no sea una regla para todos, sino sólo para el modelo de datos interno (federado).

A continuación, se presentan dos formas posibles de tratar el problema de las claves de usuario únicas:

- Regulación central de la variación.
- Regulación descentralizada de la variación (principio de delegación).

Variante de Regulación centralizada

Sin más reflexión, una regulación centralizada estaría probablemente en primer plano. Una autoridad central determina, para todos los objetos de una clase, que una clave de usuario determinada tiene que ser única en el área entera. Esto puede lograrse adoptando ciertas medidas organizativas, o todas las partes interesadas pueden tener acceso a una base de datos central.

```
TOPIC Property =

  CLASS Allotment =
    Number: 1 .. 99999
    Geometry: AREA WITH (STRAIGHTS, ARCS) VERTEX CHCoord
              WITHOUT OVERLAPS > 0.005;
  UNIQUE
    Number;
  END Allotment;

END Property.
```

A menudo, la autoridad central determinará una tesela y, por tanto, números de área únicos dentro de toda la zona. Si las asignaciones, que a su vez están situadas dentro de estas áreas, deben ser únicas en todos los aspectos, entonces la clave de usuario debe, necesariamente, consistir en una combinación del número de área como del número de asignación:

```
  CLASS Allotment =
    Area_Number: 1 .. 9999.
    Number: 1 .. 99999
    Geometry: AREA WITH (STRAIGHTS, ARCS) VERTEX CHCoord
              WITHOUT OVERLAPS > 0.005;
```

```
    UNIQUE
      Area_Number, Number; !! User key
    END Allotment;
```

Variante de Regulación Descentralizada (Principio de Delegación)

Si se han establecido las estructuras de datos necesarias en un modelo de datos, es posible incluir objetos principalmente en contenedores más pequeñas (por ejemplo, un contenedor por condado), que pueden ser incluidas, sin problemas, en contenedores más grandes (por ejemplo, uno para todo un cantón). Suponiendo, además, que la autoridad federal exige números de adjudicación de cinco dígitos, sin determinar los límites cuando se requiere la unicidad, y presumiendo al mismo tiempo que un cantón requiere singularidad dentro de los límites de un condado, entonces se hace posible el siguiente modelo:

```
MODEL Federation (en) AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  DOMAIN
    CHCoord = COORD
      0,0000.000 .. 200.000 [INTERLIS.m], !! Min_East  Max_East
      0,0000.000 .. 200.000 [INTERLIS.m], !! Min_North Max_North
    ROTATION 2 -> 1;


  TOPIC Property =

    CLASS Allotment =
      Number: 1..99999;
    Geometry: AREA WITH (STRAIGHTS, ARCS) VERTEX CHCoord
                WITHOUT OVERLAPS > 0.005;
    END Allotment;

  END Property;

END Federation.



MODEL CantonA (en) AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  IMPORTS Federation;


  TOPIC OrgStructure =

    CLASS County =
      Name: TEXT*30;
    UNIQUE
      Name;
    END County;

  END OrgStructure;


  TOPIC Property EXTENDS Federation.Property =
    DEPENDS ON OrgStruktur;

    ASSOCIATION CountyAllotment =
      County (EXTERNAL) -- {1} CantonA.OrgStruktur.County;
      Allotment -- Allotment;
    END CountyAllotment;

    CONSTRAINTS OF Allotment =
      UNIQUE}
        Number, County;
    END;
```

```
    END Property;

  END CantonA.
```

Según la definición, los nombres de los condados deben ser únicos dentro del alcance de todos los objetos de una clase. Es irrelevante si la observancia de este requisito puede comprobarse en vista de su distribución en contenedores concretos. Sin embargo, predomina este requisito.

Para determinar la unicidad del número de asignación dentro de un condado, se establece una relación entre asignación y condado y se requiere que la combinación de condado y número sea única. Una vez más, es irrelevante si un contenedor comprende parte de un condado, un condado en su totalidad o varios condados. Desde el punto de vista del modelado predomina el requisito.

Partiendo de la suposición de que un sistema contiene las parcelas o lotes de un determinado condado, es muy posible que se omita la relación, de forma interna al sistema, entre el condado y las parcelas o lotes, sólo para ser incluido cuando se transfieren datos a otros sistemas.

Apéndice F (sugerencia de extensión estándar)
Definición de unidades

Nota

La siguiente especificación no es un componente normativo de INTERLIS. Esta es una sugerencia de extensión estándar basada en el Manual de Referencia de INTERLIS Versión 2 en el sentido de una recomendación. Sin embargo, se tiene la intención de ponerlo en discusión y posiblemente convertirlo en una regulación más definida. Consulte los correspondientes manuales de INTERLIS 2 para obtener ejemplos de aplicación.

El modelo tipo

El siguiente modelo tipo comprende las unidades más comunes. Extiende las unidades que han sido definidas directamente por INTERLIS (véase el Apéndice A: El modelo interno de datos de INTERLIS).

```
!! File Units.ili Release 2005-06-16

INTERLIS 2.3;

CONTRACTED TYPE MODEL Units (en) AT "http://www.interlis.ch/models"
  VERSION "2005-06-06" =

  UNIT
    !! abstract Units
    Area (ABSTRACT) = (INTERLIS.LENGTH*INTERLIS.LENGTH);
    Volume (ABSTRACT) = (INTERLIS.LENGTH*INTERLIS.LENGTH*INTERLIS.LENGTH);
    Velocity (ABSTRACT) = (INTERLIS.LENGTH/INTERLIS.TIME);
    Acceleration (ABSTRACT) = (Velocity/INTERLIS.TIME);
    Force (ABSTRACT) = (INTERLIS.MASS*INTERLIS.LENGTH/INTERLIS.TIME/INTERLIS.TIME);
    Pressure (ABSTRACT) = (Force/Area);
    Energy (ABSTRACT) = (Force*INTERLIS.LENGTH);
    Power (ABSTRACT) = (Energy/INTERLIS.TIME);
    Electric_Potential (ABSTRACT) = (Power/INTERLIS.ELECTRIC_CURRENT);
    Frequency (ABSTRACT) = (INTERLIS.DIMENSIONLESS/INTERLIS.TIME);

    Millimeter [mm] = 0.001 [INTERLIS.m];
    Centimeter [cm] = 0.01 [INTERLIS.m];
    Decimeter [dm] = 0.1 [INTERLIS.m];
    Kilometer [km] = 1000 [INTERLIS.m];

    Square_Meter [m2] EXTENDS Area   = (INTERLIS.m*INTERLIS.m);
    Cubic_Meter  [m3] EXTENDS Volume = (INTERLIS.m*INTERLIS.m*INTERLIS.m);

    Minute [min] = 60 [INTERLIS.s];
    Hour   [h]   = 60 [min];
    Day    [d]   = 24 [h];

    Kilometer_per_Hour [kmh] EXTENDS Velocity = (km/h);
    Meter_per_Second [ms] = 3.6 [kmh];
    Newton [N]  EXTENDS Force = (INTERLIS.kg*INTERLIS.m/INTERLIS.s/INTERLIS.s);
    Pascal [Pa] EXTENDS Pressure = (N/m2);
    Joule  [J]  EXTENDS Energy = (N*INTERLIS.m);
    Watt   [W]  EXTENDS Power = (J/INTERLIS.s);
    Volt   [V]  EXTENDS Electric_Potential = (W/INTERLIS.A);

    Inch [in] = 2.54 [cm];
    Foot [ft] = 0.3048 [INTERLIS.m];
    Mile [mi] = 1.609344 [km];

    Are [a] = 100 [m2];
    Hectare [ha] = 100 [a];
```

```
      Square_Kilometer [km2] = 100 [ha];
      Acre [acre] = 4046.873 [m2];

      Liter [L] = 1 / 1000 [m3];
      US_Gallon [USgal] = 3.785412 [L];

      Angle_Degree = 180 / PI [INTERLIS.rad];
      Angle_Minute = 1 / 60 [Angle_Degree];
      Angle_Second = 1 / 60 [Angle_Minute];

      Gon = 200 / PI [INTERLIS.rad];

      Gram [g] = 1 / 1000 [INTERLIS.kg];
      Ton [t] = 1000 [INTERLIS.kg];
      Pound [lb] = 0.4535924 [INTERLIS.kg];

      Calorie [cal] = 4.1868 [J];
      Kilowatt_Hour [kWh] = 0.36E7 [J];

      Horsepower = 746 [W];

      Techn_Atmosphere [at] = 98066.5 [Pa];
      Atmosphere [atm] = 101325 [Pa];
      Bar [bar] = 10000 [Pa];
      Millimeter_Mercury [mmHg] = 133.3224 [Pa];
      Torr = 133.3224 [Pa]; !! Torr = [mmHg]

      Decibel [dB] = FUNCTION // 10**(dB/20) * 0.00002 // [Pa];

      Degree_Celsius [oC] = FUNCTION // oC+273.15 // [INTERLIS.K];
      Degree_Fahrenheit [oF] = FUNCTION // (oF+459.67)/1.8 // [INTERLIS.K];

      CountedObjects EXTENDS INTERLIS.DIMENSIONLESS;

      Hertz [Hz] EXTENDS Frequency = (CountedObjects/INTERLIS.s);
      KiloHertz [KHz] = 1000 [Hz];
      MegaHertz [MHz] = 1000 [KHz];

      Percent = 0.01 [CountedObjects];
      Permille = 0.001 [CountedObjects];

      !! ISO 4217 Currency Abbreviation
      USDollar    [USD] EXTENDS INTERLIS.MONEY;
      Euro        [EUR] EXTENDS INTERLIS.MONEY;
      SwissFrancs [CHF] EXTENDS INTERLIS.MONEY;

  END Units.
```

Ejemplos

Véase el capítulo *Unidades Base* en el manual de referencia de INTERLIS versión 2.

Apéndice G (sugerencia extensión estándar)

Las definiciones de tiempo

Nota

La siguiente especificación no es un componente normativo de INTERLIS. Esta es una sugerencia de extensión estándar basada en el Manual de Referencia de INTERLIS Versión 2 en el sentido de una recomendación. Sin embargo, se tiene la intención de ponerlo en discusión y posiblemente convertirlo en una regulación más definida. Consulte los correspondientes manuales de INTERLIS 2 para obtener ejemplos de aplicación.

El modelo de tiempo

```
!! File Time.ili Release 2005-06-16

INTERLIS 2.3;

CONTRACTED REFSYSTEM MODEL Time (en) AT "http://www.interlis.ch/models"
  VERSION "2005-06-16" =

  IMPORTS Units;

  STRUCTURE DayOfYear =
    Month: 1 .. 12 [INTERLIS.M];
    SUBDIVISION Day: 1..31 [INTERLIS.d];
  END DayOfYear;

  STRUCTURE HMDiffWithinDay =
    Hours: -23 .. 23 CIRCULAR [INTERLIS.h];
    CONTINUOUS SUBDIVISION Minutes: 0 .. 59 [INTERLIS.min];
  END HMDiffWithinDay;

  DOMAIN
    WeekDay = (WorkingDay (Monday, Tuesday, Wednesday,
                           Thursday, Friday, Saturday),
               Sunday) CIRCULAR;

    HMDiffWDay = FORMAT BASED ON HMDiffWithinDay (Hours ":" Minutes);
    DifferenceToUTC EXTENDS HMDiffWDay = MANDATORY "-13:00" .. "13:00";
                                         !! UTC := LocTime + Diff

  FUNCTION AppropriateDate (dayOfYear: MANDATORY DayOfYear;
                            weekDay: WeekDay): DayOfYear
    // returns first parameter if second is undefined,
       returns first day from (incl) first parameter being the
       requested weekday //;

  FUNCTION DSTOrdered (day1: DayOfYear; day2: DayOfYear) : BOOLEAN
    // returns TRUE if the second parameter comes after the
       first parameter or if both parameters are equal //;

  STRUCTURE DSTransition =
    TransitionDSTime: MANDATORY HMDiffWDay;
    FirstDate: MANDATORY DayOfYear;
    DayOfWeek: WeekDay;
    TransitionDate: DayOfYear := AppropriateDate (FirstDate, DayOfWeek);
  END DSTransition;

  STRUCTURE DaylightSavingPeriod =
    DSToUTC: DifferenceToUTC;
    From: MANDATORY INTERLIS.GregorianYear;
    To:   MANDATORY INTERLIS.GregorianYear;
```

```
    DSStart: MANDATORY DSTransition;
    DSEnd: MANDATORY DSTransition;
  MANDATORY CONSTRAINT
    DSTOrdered (DSStart, DSEnd);
  MANDATORY CONSTRAINT
    To >= From;
  END DaylightSavingPeriod;

  FUNCTION DSPOverlaps (periods: BAG {1..*} OF DaylightSavingPeriod) : BOOLEAN
    // returns TRUE if any one of the periods overlap //;


  TOPIC TimeZone =

    CLASS TimeZone (ABSTRACT) EXTENDS INTERLIS.SCALSYSTEM =
    PARAMETER
      Unit (EXTENDED): NUMERIC [INTERLIS.TIME];
    END TimeZone;

    CLASS BaseTimeZone EXTENDS INTERLIS.TIMESYSTEMS.TIMEOFDAYSYS =
                              !! TimeZone without daylight saving
      DiffToUTC: DifferenceToUTC;
    END BaseTimeZone;

    CLASS DaylightSavingTZ EXTENDS INTERLIS.TIMESYSTEMS.TIMEOFDAYSYS =
      Periods: BAG {1..*} OF DaylightSavingPeriod;
    MANDATORY CONSTRAINT
      NOT ( DSPOverlaps (Periods) );
    END DaylightSavingTZ;

    ASSOCIATION DaylightSavingTZOf =
      BaseTZ -<> BaseTimeZone;
      DSTZ -- DaylightSavingTZ;
    END DaylightSavingTZOf;

  END TimeZone;

END Time.
```

datos de ejemplo para el modelo de tiempo

El siguiente ejemplo corresponde al modelo de tiempo anterior.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- File SwissTimeData.xml 2005-06-16 (http://www.interlis.ch/models) -->

<TRANSFER xmlns="http://www.interlis.ch/INTERLIS2.3"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.interlis.ch/INTERLIS2.3
                              SwissTimeData.xsd">
  <HEADERSECTION VERSION="2.3" SENDER="KOGIS">
    <MODELS>
      <MODEL NAME="Units" URI="http://www.interlis.ch/models"
             VERSION="2005-06-16"/>
      <MODEL NAME="Time" URI="http://www.interlis.ch/models"
             VERSION="2005-06-16"/>
    </MODELS>

    <ALIAS>
      <ENTRIES FOR="Time">
        <TAGENTRY FROM="Time.DayOfYear" TO="Time.DayOfYear"/>
        <TAGENTRY FROM="Time.HMDiffWithinDay" TO="Time.HMDiffWithinDay"/>
        <TAGENTRY FROM="Time.DSTransition" TO="Time.DSTransition"/>
        <TAGENTRY FROM="Time.DaylightSavingPeriod" TO="Time.DaylightSavingPeriod"/>
        <TAGENTRY FROM="Time.TimeZone" TO="Time.TimeZone"/>
        <TAGENTRY FROM="Time.TimeZone.BaseTimeZone"
                  TO="Time.TimeZone.BaseTimeZone"/>
```

```
      <TAGENTRY FROM="Time.TimeZone.DaylightSavingTZ"
                TO="Time.TimeZone.DaylightSavingTZ"/>
      <TAGENTRY FROM="Time.TimeZone.DaylightSavingTZOf"
                TO="Time.TimeZone.DaylightSavingTZOf"/>
    </ENTRIES>
  </ALIAS>

  <COMMENT>
    example dataset ili2 refmanual appendix G
  </COMMENT>
</HEADERSECTION>

<DATASECTION>
  <Time.TimeZone BID="BTimeZones">
    <Time.TimeZone.BaseTimeZone TID="BTimeZones.MEZ">
      <Name>MEZ</Name>
      <DiffToUTC>-1:00</DiffToUTC>
    </Time.TimeZone.BaseTimeZone>

    <Time.TimeZone.DaylightSavingTZ TID="BTimeZones.MESZ">
      <Name>MESZ</Name>
      <Periods>
        <Time.DaylightSavingPeriod>
          <DSToUTC>-2:00</DSToUTC>
          <From>1983</From>
          <To>1995</To>
          <DSStart>
            <Time.DSTransition>
              <TransitionDSTime>3:00</TransitionDSTime>
              <FirstDate>
                <Time.DayOfYear>
                  <Month>3</Month>
                  <Day>25</Day>
                </Time.DayOfYear>
              </FirstDate>
              <DayOfWeek>Sunday</DayOfWeek>
            </Time.DSTransition>
          </DSStart>
          <DSEnd>
            <Time.DSTransition>
              <TransitionDSTime>3:00</TransitionDSTime>
              <FirstDate>
                <Time.DayOfYear>
                  <Month>9</Month>
                  <Day>24</Day>
                </Time.DayOfYear>
              </FirstDate>
              <DayOfWeek>Sunday</DayOfWeek>
            </Time.DSTransition>
          </DSEnd>
        </Time.DaylightSavingPeriod>
        <Time.DaylightSavingPeriod>
          <DSToUTC>-2:00</DSToUTC>
          <From>1996</From>
          <To>2999</To>
          <DSStart>
            <Time.DSTransition>
              <TransitionDSTime>3:00</TransitionDSTime>
              <FirstDate>
                <Time.DayOfYear>
                  <Month>3</Month>
                  <Day>25</Day>
                </Time.DayOfYear>
              </FirstDate>
              <DayOfWeek>Sunday</DayOfWeek>
            </Time.DSTransition>
          </DSStart>
          <DSEnd>
```

```
                <Time.DSTransition>
                  <TransitionDSTime>3:00</TransitionDSTime>
                  <FirstDate>
                    <Time.DayOfYear>
                      <Month>10</Month>
                      <Day>25</Day>
                    </Time.DayOfYear>
                  </FirstDate>
                  <DayOfWeek>Sunday</DayOfWeek>
                </Time.DSTransition>
              </DSEnd>
            </Time.DaylightSavingPeriod>
          </Periods>
        </Time.TimeZone.DaylightSavingTZ>

        <Time.TimeZone.DaylightSavingTZOf TID="DaylightSavingTZOf">
          <BaseTZ REF="BTimeZones.MEZ"></BaseTZ>
          <DSTZ REF="BTimeZones.MESZ"></DSTZ>
        </Time.TimeZone.DaylightSavingTZOf>
      </Time.TimeZone>
    </DATASECTION>
  </TRANSFER>
```

Apéndice H (sugerencia de extensión estándar) Definiciones de Color

Nota

La siguiente especificación no es un componente normativo de INTERLIS. Esta es una sugerencia de extensión estándar basada en el Manual de Referencia de INTERLIS Versión 2 en el sentido de una recomendación. Sin embargo, se tiene la intención de ponerlo en discusión y posiblemente convertirlo en una regulación más definida. Consulte los correspondientes manuales de INTERLIS 2 para obtener ejemplos de aplicación.

Introducción

Esta especificación establece en detalle por qué un cierto espacio de color llamado $L^*C^*_{ab}h^*_{ab}$ es el más adecuado para las definiciones de color. Proporciona una descripción exhaustiva de este espacio de color, cita fórmulas de conversión relacionadas con otros espacios de color y da instrucciones sobre cómo puede implementarse una transformación de coordenadas $L^*C^*_{ab}h^*_{ab}$ en el sistema de coordenadas de color de una pantalla o impresora concretas. Además, establece las bases para los dominios y las precisiones seleccionadas a continuación e indica coordenadas de ejemplos especialmente elegidos.

Dado que, entre otras facultades, INTERLIS 2 permite la descripción de gráficos, debe ser posible especificar colores. Sin embargo, una definición neutra de "color" del sistema y del equipo es sorprendentemente compleja y exige la comprensión de conceptos que no son conocimiento general.

El color es un producto de la luz (= estímulo del color), del ojo (= valencia del color) y de la función del cerebro (= sensación). Es prácticamente imposible describir los colores a través de los números de tal manera que dos personas los perciban de manera idéntica. Sin embargo, los valores de color se pueden medir de una manera universalmente reconocida, permitiendo así una comprensión precisa entre los expertos.

Un método para especificar colores como cadenas debe cumplir varios requisitos:

- Independencia del equipoi - Debe definirse claramente qué color corresponde realmente a una determinada indicación. Este es el único medio para asegurar que el resultado cumpla con todas las expectativas, sea cual sea el equipo utilizado.
- Expresividad - Debería ser posible especificar todos los colores que el equipo "normal" (especialmente las impresoras y trazadores de buena calidad) pueda representar. El espectro de colores a especificar debe ser lo más amplio posible. Idealmente comprenderá todos los colores que un ser humano pueda percibir.
- Intuición - Al leer una descripción de color, un ser humano debe intuitivamente tener una noción del color que se describe. Un modelo de INTERLIS siempre tiene un cierto carácter documental y debe ser comprensible para los interesados sin exigir mayores esfuerzos.
- Neutralidad del sistema - Las formas y métodos de indicar el color no deben dar prioridad a cierto sistema (SIG, sistema operativo, hardware) ni obligar a la adquisición de dispositivos especiales.

Espacio de color

La siguiente tabla muestra la idoneidad de diferentes espacios de color en lo que respecta a la aplicación en las descripciones gráficas de INTERLIS:

Espacio de color	Independencia de Equipo	Expresividad	Inteligibilidad intuitiva	sistema de Neutralidad
RGB	–	–	–	–
HLS	–	–	+ +	–
HSV	–	–	+ +	–
CMY(K)	–	–	–	–
XYZ	+	+	– –	+
SRGB	+	–	–	–
L a* b**	+	+	+	+
$L^* C^*_{ab} h^*_{ab}$	+	+	+ +	+

Figura H.1: Adecuación de los diferentes espacios de color para los propósitos de INTERLIS.

El último de los espacios de color mencionados en la figura H.1 $L^* C^*_{ab} h^*_{ab}$ (d.h. L * a * b * con coordenadas polares) cumple con los requisitos establecidos anteriormente de la forma más satisfactoria.

L*a*b*

El espacio de color L * a * b * (a veces también llamado CIELAB) ampliamente utilizado en la industria gráfica, se puede derivar a través de la transformación de XYZ como se describe en la figura H.2.

$$L^* = 116 \cdot f\left(\frac{Y}{Y_n}\right) - 16$$

$$a^* = 500 \cdot \left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right]$$

, donde $f(x) = \begin{cases} \sqrt[3]{x} & \text{if } x > 0.008856; \\ 7.787\,x + \frac{16}{116} & \text{else} \end{cases}$

$$b^* = 200 \cdot \left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right]$$

Figura H.2: La conversión de XYZ a L * a * b *.

En el cálculo de la figura H.2 se introduce un "blanco de referencia" mediante <Xn, Yn, Zn> para compensar un eventual tinte de luz. Muy a menudo se emplean los valores de las fuentes de luz estándar CIE (principalmente D50, ocasionalmente D65). Las coordenadas XYZ de estas fuentes de luz se pueden encontrar, por ejemplo, en [Sangwine / Horne, 1989], Tabla 3.1.

Esta gama posee una serie de propiedades útiles:

- Independencia del equipo - L * a * b * se deriva de XYZ y por lo tanto independiente de un determinado equipo. Se define inequívocamente qué color pertenece a un L * a * b * Triple.
- Expresividad - En L * a * b * se asigna un punto a cada color que puede ser emitido por una superficie reflectante.
- La inteligibilidad intuitiva - L* significa luminancia, por lo que una superficie completamente negra (que no refleja luz alguna) posee un L* de 0 y un reflector perfecto (que refleja toda la luz), un L* de 100. Un observador humano juzgará un color con L* = 50 como brillo medio. a* es el eje rojo-verde: los colores con un a* = 0 serán percibidos como ni rojo ni verde, los colores con un valor negativo de a* son rojos, los colores con valor positivo de a* son verdes. Análogamente b* es el

eje azul-amarillo. Dentro de un plano que se extiende con a* y b*, hay una distancia desde el punto cero a un valor de color específico, cuanto mayor es la distancia más saturado se vuelve un color.

- Neutralidad del sistema - *L*a*b** es absolutamente neutral respecto al sistema; siendo un estándar internacional, el espacio de color es independiente de una empresa específica.
- Utilización creciente – El uso de *L*a*b** en la impresión profesional está ampliamente difundido. Programas como Adobe Photoshop o Acrobat (PDF) admiten *L*a*b**.
- Fácil transformabilidad a RGB – la tripleta *L*a*b** se puede transformar en valores RGB de cualquier pantalla a través de la multiplicación de una matriz 3x3, seguido de un aumento a mayor potencia (corrección gamma), que puede llevarse a cabo eficientemente mediante una tabla (véase [Adobe, 1992], capítulo 23). Así, los desarrolladores de sistemas sólo tendrán que hacer frente a esfuerzos mínimos.
- Buena capacidad de compresión - Sólo hay una diferencia marginal entre *L*a*b** y RGB en procesos que es probable que impliquen pérdidas al comprimir imágenes. Sin embargo, en relación con INTERLIS esto es irrelevante.

$$C^*_{ab} = \sqrt{\left(a^*\right)^2 + \left(b^*\right)^2} \qquad h^*_{ab} = \tan^{-1}\left(\frac{b^*}{a^*}\right)$$

Figura H.3: Conversión del espacio cartesiano *L*a*b** a la forma polar L*C*abh*ab (de acuerdo con [Sangwine / Horne, 1998]).

L*C*abh*ab

Como se ha descrito anteriormente, en el espacio *L*a*b** cada eje único L* (luz oscura), a* (verde - rojo) y b* (azul - amarillo) corresponde a una propiedad de color que puede percibirse de forma inmediatamente.

Sin embargo, la inteligibilidad intuitiva puede aumentarse aún más indicando coordenadas de color en un sistema polar en lugar de un sistema cartesiano (véase la figura H.4).

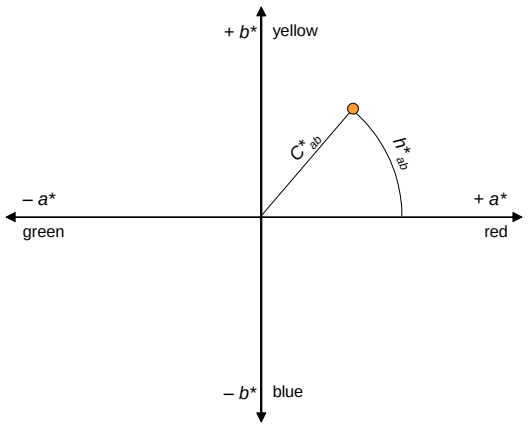

Figura H.4: las Funciones del espacio de color L*C*abh*ab con coordenadas polares en L* a* b*.

La fórmula de la figura H.3 para h*ab sólo es aplicable para a* y b* positivos; Una versión correcta proporcionaría diferenciación de casos para cada cuadrante único. Este sistema polar combina la inteligibilidad intuitiva de HLS y HSV con las numerosas ventajas de L* a* b* descritas anteriormente, ya que significa que los ejes L* (luminancia), C* (croma) y h* (tonalidad) están disponibles de forma separada.

En los modelos de INTERLIS, siempre que se deseen indicaciones de color precisas, deben basarse en este sistema de coordenadas de color.

Precisión requerida

Es parte de un modelo INTERLIS indicar el grado de precisión que se debe aplicar al registrar valores numéricos. El espacio L* a* b* se define de tal manera que la diferencia entre dos colores es sólo perceptible si el valor calculado, como se muestra en la figura 1.5, es igual a 1.

Nota: En [Has/Newman, o.D.] se afirma que la perceptibilidad de las diferencias de color también depende de la cantidad de tiempo permitido para la comparación. El artículo se refiere a un experimento, en el que el tiempo necesario para observar las diferencias se midió en el caso de un observador inexperto. Las cifras mencionadas son 5 segundos para ΔEab = 15, 10 segundos para ΔEab = 10 y 15 segundos para ΔEab = 5.

$$\Delta E_{ab} = \sqrt{\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2}}$$

Figura H.5: Cálculo de diferencias de color en el espacio cartesiano L* a* b*.

Precisión del eje L*: Para la luminancia es suficiente una precisión de un decimal..

Precisión de los ejes C^*_{ab} y h^*_{ab}: Teóricamente, a* y b* pueden ser ilimitados, pero en realidad los límites de ± 128, redondeados a números enteros, se considera que son suficientemente amplios (véase Adobe, 1992). Así, ¿qué grado de precisión es necesario para C^*_{ab} y h^*_{ab}, para asegurar que la inexactitud en la superficie a*/b* no exceda de 1?

La inexactitud introducida a través de la indicación de ángulo aumenta al aumentar la distancia desde el punto cero. Así, la precisión puede considerarse suficiente siempre que se puedan distinguir <127, 128> y <128, 128> dentro de la superficie a*/b*. Como puede verse en la figura H.6, un decimal puede ser suficiente en este caso extremo. Se trata de dos tonalidades de naranja apenas distinguibles, saturadas hasta tal punto que parece improbable que cualquier aparato sea capaz de reproducirlas..

a^*	b^*	C^*_{ab}	h^*_{ab}
127	128	180.3	45.2
128	128	181.0	45.0

Figura H.6: coordenadas cartesianas y polares de un color muy lejos del punto cero (véase el gráfico de conversión H.3).

Combinación con nombres

nombres de los colores son más fáciles de manejar que los códigos de color (es decir, números), sin embargo, esto resulta ser una desventaja, ya que de este modo se dispone de sólo un número limitado de colores. En INTERLIS nombres se pueden combinar con una especificación numérica, lo que permite a los usuarios definir sus propios nombres de los colores y para intercambiarlos entre sí por los medios comunes de INTERLIS.

Gracias a esta definición también es posible emplear INTERLIS - si es necesario - en la documentación y utilización de sistemas de nombres de colores existentes o catálogos de muestras de colores, como el Sistema Pantone o HKS.

Esto requiere la definición de una metaclase (véase el capítulo Metamodelos y metaobjetos en el Manual de Referencia de INTERLIS Versión 2). Sus instancias, los llamados metaobjetos, se conservan en un archivo de transferencia especial y son leídos por el compilador INTERLIS 2. Están disponibles para los

modelos de datos INTERLIS y, por tanto, pueden utilizarse en las definiciones gráficas para determinar el color de un cierto símbolo, etc.

Ejemplos de aplicación en los modelos de INTERLIS

```
!! Component of the symbology model

CONTRACTED SYMBOLOGY MODEL SymbologyExample AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =


  TOPIC Signs =

    CLASS LChColor EXTENDS INTERLIS.METAOBJECT =
      !! Attribute "Name" inherited from INTERLIS.METAOBJECT
      Luminance = MANDATORY 0.0 .. 100.0;
      Chroma = MANDATORY 0.0 .. 181.1;
      Hue = MANDATORY 0.0 .. 359.9 CIRCULAR [DEGREE] COUNTERCLOCKWISE;
    END LChColor;

    ...

    !! Component of the symbol class definition within the symbology model
    CLASS ColoredSymbology EXTENDS SIGN =
      ...
    PARAMETER
      Color: METAOBJECT OF SymbologyExample.LChColor;
    END ColoredSymbology;

    ...

  END Signs;
  ...
```

En un comando de visualización definido por el usuario (denominado SimplePointGr), el color de un símbolo de color definido por el usuario podría aparecer de la siguiente manera (véase el capítulo Descripción gráfica en el Manual de referencia INTERLIS versión 2):

```
...

CONTRACTED MODEL SimpleGraphic AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  IMPORTS SymbologyExample, Data;

  SIGN BASKET SimpleSigns ~ SymbologyExample.Signs
    OBJECTS OF Color: Brown
    OBJECTS OF ColoredSymbology: Dot;


  TOPIC ColoredDotGraphic =
    DEPENDS ON Data.Dots;

    GRAPHIC SimpleColoredDotGr BASED ON Data.Dots.Dot =
      Symbol OF SymbologyExample.Signs.ColoredSymbology: (
        Sign := {Dot};
        Pos := Position;
        Color := Brown;
      );
    END SimpleColoredDotGr;

  END ColoredDotGraphic;

END SimpleGraphic.
...
```

No se da un ejemplo completo ni se representa la tabla meta necesaria, sino que se hace referencia al ejemplo indicado en el Apéndice C Un pequeño ejemplo de Carreteras en el Manual de Referencia de INTERLIS Versión 2.

Valores de ejemplo

La figura H.7 indica algunos colores, así como sus coordenadas. Debido a no saber si este documento ha sido concebido en un sistema (e impreso) capaz de representar los colores correctamente, en esta etapa se prescinde de una representación con colores.

Nombre	*L**	*a**	*b**	C^*_{ab}	h^*_{ab}
Negro	0.0	0	0	0.0	0.0
Gris oscuro	25.0	0	0	0.0	0.0
Gris Medio	50.0	0	0	0.0	0.0
Gris claro	75.0	0	0	0.0	0.0
Blanco	100.0	0	0	0.0	0.0
Fucsia	40.0	70	0	70.0	0.0
Azul claro	80.0	0	-30	30	270.0
Amarillo intenso	90.0	0	100	100.0	90.0
Marrón	50.0	30	50	58.3	59.0
Lila	50.0	50	-50	70.7	315.0

Figura H.7 coordenadas cartesianas y polares de algunos colores.

Un ejemplo concreto de aplicación con el color-definiciones se encuentra en el apéndice CA pequeño ejemplo de rutas.

Notas para desarrolladores de sistemas

Los desarrolladores de sistemas compatibles con INTERLIS deben tratar la cuestión de cómo transformar valores de color desde el sistema independiente *L** C^*_{ab} h^*_{ab} en un sistema de coordenadas de color de una pantalla o impresora específica.

Un formato de archivo estandarizado le permitirá registrar las distorsiones de color de un componente de imagen determinado en los denominados perfiles de concordancia de componentes o colores (el denominado formato de perfil ICC). Entre otros, estos archivos contienen parámetros necesarios en la conversión de un espacio de color independiente a un sistema de coordenadas de color específico del componente. Los primeros son XYZ o L*a*b*, este último comúnmente RGB o CMYK. El formato y las funciones de conversión necesarias se definen en [ICC, 1996].

Así, en su producto, un desarrollador de sistemas podrá hacer uso directamente de perfiles ICC. El formato de archivo es de una estructura relativamente simple, y las funciones de conversión serán fácilmente implementadas. Para algunas plataformas, están disponibles las bibliotecas de programas ya hechas (como Apple ColorSync o Kodak KCMS).

En este contexto, queremos llamar su atención sobre el hecho de que PDF soporta directamente el espacio de color L*a*b*. PostScript incluso le permite definir sus propios rangos de colores en términos de cualquier transformación dada desde XYZ. La función de inversión de la fórmula indicada en la figura H.2 se encuentra en el ejemplo 4.11 en [Adobe, 1990]. Es relativamente sencillo programar una función análoga en PostScript que acepte directamente $L^* C^*_{ab} h^*_{ab}$.

Bibliografía

[Adobe, 1990] Adobe Systems: Manual de Referencia del lenguaje PostScript. 2ª Ed., 1990. ISBN 0-201-18127-4. 764 páginas.

*El manual de referencia para PostScript, también proporciona recomendaciones para el tratamiento de los colores y diferentes métodos de conversión disponibles en PostScript. El ejemplo 4.11 en la página 191 define el sistema L * a * b * para dar color en el rango de PostScript.*

[Adobe, 1992] Adobe Developers Association: TIFF Revision 6.0.
http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf

*El capítulo 23 define una variedad del formato TIFF para fotografías en el sistema L * a * b * gama de color y los nombres de una serie de ventajas en comparación con RGB. Además, encontrará las líneas generales de un método rápido para la conversión de L * a * b * a RGB.*

[Apple, 1998] Apple Computer, Inc .: Introducción a los Sistemas de Gestión de color Color y. In: *Inside Macintosh — Managing Color with ColorSync*.
https://developer.apple.com/documentation/mac/ACI/ACI-46.html

Introducción, fácil de entender, a diferentes espacios de color, con gráfico ilustrado.

[Apple, o.D.] Apple Computer, Inc.: Una breve descripción de color.
http://devworld.apple.com/documentation/GraphicsImaging/Conceptual/csintro/

Concisa, fácil de entender y la introducción en bruto en diferentes conceptos en relación con los colores. Destinado a los no especialistas.

[Has/Newman, o.D.] Michael Has, Todd Newman: Color Management: Current Practice and The Adoption of a New Standard. www.color.org/wpaper1.html

Nombra los datos para el punto de referencia rojo, verde y azul de dos monitores de computadora típicos y muestra que difieren ampliamente de los valores xy de los colores estándar de fósforo NTSC citados con frecuencia. Indica una transformación de XYZ a RGB de una determinada pantalla.

[ICC, 1996] Consorcio Internacional del Color: ICC Profile Format Specification.
www.color.org/profile.html

Define un formato de archivo que permite la caracterización de cualquier componente dado con respecto a su representación de color. Apéndice A comentarios sobre varios espacios de colores.

[Poynton, 1997] Charles A. Poynton: Frequently Asked Questions about Color.
www.poynton.com/ColorFAQ.html

Explica, en el párrafo 36, por qué HLS y HSV no son adecuados para la especificación de colores.

[Sangwine/Horne, 1998] Sangwine, Stephen J. und Horne, Robin E. N. [Hrsg.]: Manual de Procesamiento de Imagen Color. Chapman & Hall: London […], 1998. ISBN 0-412-80620-7. 440 páginas.

Introducción bien fundada en los fundamentos científicos de la percepción del color y su aplicación en el procesamiento de imágenes.

[Stokes et al., 1996] Michael Stokes, Matthew Anderson, Srinivasan Chandrasekar und Ricardo Motta: Un espacio de color predeterminado estándar para el Internet - sRGB. November 1996.
www.color.org/sRGB.html

Especificación de sRGB.

Apéndice I (sugerencia de extensión estándar) Sistemas de coordenadas y sistemas de referencia de coordenadas

Nota

La siguiente especificación no es un componente normativo de INTERLIS. Esta es una sugerencia de extensión estándar basada en el Manual de Referencia de INTERLIS Versión 2 en el sentido de una recomendación. Sin embargo, se tiene la intención de ponerlo en discusión y posiblemente convertirlo en una regulación más definida. Consulte los correspondientes manuales de INTERLIS 2 para obtener ejemplos de aplicación.

Introducción

Las coordenadas describen la posición de un punto en el espacio, siempre que se haya configurado un sistema de coordenadas correspondiente. Si un sistema de coordenadas está colocado de forma fija en relación con la tierra - en otros términos: referenciado - entonces se denomina sistema de referencia de coordenadas. No obstante, las coordenadas no sólo determinan posiciones, sino también cantidades métricas que pueden derivarse de coordenadas, tales como distancias, superficies, volúmenes, ángulos y direcciones, así como otras propiedades, por ejemplo, grados y curvas.

Hay una multitud de clases (tipos) de sistemas de coordenadas y un mayor número de objetos, es decir, realizaciones (instancias) de sistemas de coordenadas (véase también, por ejemplo, [Voser1999]). Las coordenadas federales suizas, por ejemplo, se basan en una instancia especial (objeto) de un sistema de referencia de coordenadas [Gubler et al. 1996], que puede derivarse de un sistema de referencia geodésico mediante proyección de mapas [Snyder 1987, Bugayevskiy 1995]. Estos sistemas de referencia geodésica forman una categoría propia de sistemas de referencia de coordenadas que describen la geometría del modelo terrestre. Por ejemplo, para describir una posición bidimensional, se utiliza una esfera o un elipsoide sobre cuya superficie pueden definirse coordenadas geográficas. Es un poco más complicado en lo que se refiere a la altitud: Para servir como modelo geométrico físico de la tierra empleamos un geoide [Marti 1997] o un modelo de gravedad [Torge 1975] que define las altitudes ortométricas, es decir, normales. Sin embargo a menudo, en la práctica, es frecuente sólo se aplica a las alturas en uso.

Dado que los geodatos de las aplicaciones geomáticas siempre están relacionados con el espacio, cada conjunto de geodatos debe basarse en un sistema de coordenadas. Teniendo en cuenta que los sistemas de coordenadas individuales difieren ampliamente, es necesario proporcionar los datos de referencia correspondientes junto con los geodatos. Es por eso que INTERLIS permite describir los datos pertenecientes a un sistema de coordenadas.

Sólo a través del conocimiento del sistema de coordenadas subyacente es posible transferir los geodatos a otro sistema de coordenadas. Esto también es necesario si los geodatos proporcionados por diferentes sistemas de coordenadas son de uso común [Voser 1996].

Primero se consideran los sistemas de coordenadas de un tipo general, luego las relaciones (representaciones) entre sistemas de coordenadas (generales), después introducimos sistemas de coordenadas de referencia y tratamos con ellos.

Sistemas de coordenadas

Un *sistema de coordenadas* permite la "medición" del espacio métrico. Un sistema de coordenadas posee un origen, eje de coordenadas (su número correspondiente a la dimensión del espacio abarcado), así como unidades de medida asignadas al eje. Dependiendo de si el espacio en cuestión es mono, bi o tri

dimensional, el sistema de coordenadas asigna un dígito, un dígito doble o un dígito triple a cada punto en el espacio como su coordenada.

El espacio euclídeo de una, dos o tres dimensiones se define por 1, 2 o 3 ejes rectos. Los espacios curvos requieren parámetros adicionales para definir la incorporación de su eje curvado en un espacio euclídeo.

Para fines geodésicos, se necesitan espacios elípticos bidimensionales, así como varios sistemas de altura, ambos tratados como casos especiales de espacios euclídeos unidimensionales, además de los espacios euclídeos de diferentes dimensiones. Esto es cuando se requiere un modelo de gravedad o geoide.

Ligeramente diferente al uso empleado hasta ahora en geodesia, se emplea el término de datum geodésico como sinónimo de sistema de referencia geodésico, designando así nada más que un sistema de coordenadas especial, es decir un sistema de coordenadas cartesiano 3D que se ha colocado en relación con la tierra. Esto puede lograrse de dos maneras diferentes:

(a) El centro de gravedad medio de la Tierra se define como el punto cero del sistema de coordenadas, el primer eje a través del eje de rotación medio de la Tierra, el segundo eje perpendicular al anterior a través del meridiano medio de Greenwich y el tercer eje también perpendicular a los dos primeros, creando así un sistema de rotación en el sentido de las agujas del reloj. Por ejemplo, el sistema de coordenadas WGS84 se define de esta manera.

(b) La superficie de la tierra de una determinada zona (sobre todo un país) se aproxima de manera óptima mediante un globo o una elipse de rotación cuyo eje de rotación se establece paralelo al eje de rotación medio de la Tierra. Este elipsoide de rotación define un sistema de coordenadas cartesianas a través de su medio eje más pequeño que es paralelo al eje de tierra, a través de uno de su medio eje más largo y a través de un tercer eje perpendicular a los dos primeros, creando de nuevo un sistema de rotación en sentido horario.

Un sistema de coordenadas cartesiano 3D situado sobre la tierra de acuerdo a lo indicado en (a) o en (b) se denomina datum geodésico o sistema de referencia geodésico.

Diferentes orígenes de sistemas de coordenadas en geomática

Los diferentes antecedentes conducen a varias definiciones de sistemas de coordenadas en:

Técnicas de sensor: Los métodos de captura de datos en geodesia clásica (por ejemplo, con teodolitos), así como fotogrametría y teledetección utilizan un sistema de coordenadas (local) de acuerdo con cada respectivo método en sus sensores de captura de datos.

Geoposicionamiento: La descripción de la posición en la tierra por medio de un modelo terrestre (geodésico). Hay tres tipos diferentes de modelos geodésicos de la tierra [Voser 1999]::

- *físico:* El modelo de la tierra se describe mediante un campo gravitacional o un geoide.
- *matemático:* El modelo de la tierra es un cuerpo simétrico (por ejemplo, una esfera o un elipsoide).
- *topográfico:* El modelo de la tierra también tiene en cuenta las montañas y los valles (modelo de la superficie terrestre).

Los modelos de la tierra mencionados se corresponden a diferentes sistemas de coordenadas.

Posicionamiento del mapa: Dado que las superficies de los modelos de la tierra antes mencionados son de forma curva o incluso más compleja, el cálculo de distancias, ángulos, etc. es muy complejo. Por lo tanto, se emplean proyecciones de mapa que representan la superficie bidimensional en un plano. Una proyección de mapa es una forma geométricamente definida de representar la superficie de un modelo matemático de la tierra en un plano. Este proceso implica distorsiones, que se pueden determinar y controlar por adelantado.

Asignaciones entre sistemas de coordenadas

Dado que los geodatos normalmente se registran en diferentes sistemas de coordenadas o son administrados por diferentes instituciones en varios sistemas, es necesario conocer los métodos que permiten la conversión de datos suministrados por un sistema de coordenadas fuente A en un sistema de coordenadas Z objetivo. Del sistema de coordenadas A y del espacio definido por A en el sistema de coordenadas Z y el espacio definido por Z. Las asignaciones entre dos sistemas de coordenadas y entre los espacios que definen están determinadas por las clases de los dos sistemas de coordenadas correspondientes.

Hay que distinguir entre dos asignaciones fundamentalmente diferentes de sistemas de coordenadas en lo que se refiere al origen de las fórmulas y sus parámetros, que son la conversión y la transformación.

El término *conversión* significa asignación entre dos sistemas de coordenadas estrictamente definidos por fórmulas y sus parámetros. Estas fórmulas y especialmente los valores de los parámetros necesarios se determinan de antemano [Voser 1999]. En la categoría de conversiones se incluyen, entre otras, proyecciones de mapa, es decir, asignaciones de superficies elipsoidales en un plano, además de la conversión de coordenadas elípticas en coordenadas cartesianas correspondientes con su origen en el centro de la elipse o viceversa.

Una *transformación* es una correlación entre dos sistemas de coordenadas donde las reglas (fórmulas) se determinan sobre la base de hipótesis, y los parámetros se establecen mediante un análisis estadístico de las mediciones en ambos sistemas de coordenadas [Voser 1999]. Las transformaciones típicas se efectúan cuando se reemplaza un modelo geodésico de la tierra con otro (transformación de datum geodésico) o cuando se ajustan coordenadas locales en un sistema superior, por ejemplo, con digitalización: la transformación de coordenadas de mapa o tabla de coordenadas en coordenadas proyectadas.

Sistemas de Coordenadas de Referencia

El término Sistema de Coordenadas de Referencia describe un sistema de coordenadas que puede derivarse de un sistema de referencia geodésico (es decir, un datum geodésico) mediante la conversión a través de una secuencia de sistemas de coordenadas intermedios.

Los sistemas de referencia geodésicos (o datum geodésicos) como tales son los sistemas de referencia de coordenadas más importantes. Se refieren a un modelo geodésico de la tierra (véase arriba).

Estudio de los más importantes sistemas de coordenadas de referencia

En la parte inferior de la figura I.1 se representan algunas de las expresiones geodésicas y cartográficas más importantes de los sistemas de coordenadas de referencia. Es la tierra misma la que está en el origen de cualquier secuencia de sistemas o asignaciones de coordenadas. Tratamos de asignarle un modelo geométrico de la tierra que permita la descripción de las posiciones sobre ella. Para empezar, podemos asignar a la Tierra en su conjunto un sistema de coordenadas cartesiano 3D con su punto cero en el centro de gravedad de la Tierra (ver método “a” en el capítulo Sistemas de coordenadas arriba). Posteriormente, sin embargo, se trata la posición y la altura de un punto independientemente. En primer lugar, se considera sólo lo que hay que hacer para determinar su posición. Las mediciones geodésicas proporcionan la información necesaria para determinar el tamaño y la forma de un elipsoide de rotación que se aproxima a la superficie terrestre localmente de una manera óptima. Según el método “b” en el capítulo Sistemas de coordenadas, este elipsoide de rotación puede tener un datum geodésico. Muchos de los modelos de tierra seleccionados para el levantamiento nacional son "localmente" referenciados, es decir, el centro del elipsoide no coincide con el centro de gravedad de la tierra. Sin embargo, como se indicó anteriormente (“a” en el capítulo Sistemas de coordenadas), existen sistemas de referencia geodésicos que se referencian al centro de gravedad. Así, hoy en día es relativamente fácil determinar los parámetros de una transformación de datum a un sistema superior. Una vez que se ha decidido tal elipsoide de rotación local,

por otra parte, es posible representar su superficie en un plano mediante una proyección de mapa apropiada y de acuerdo con todos los requisitos.

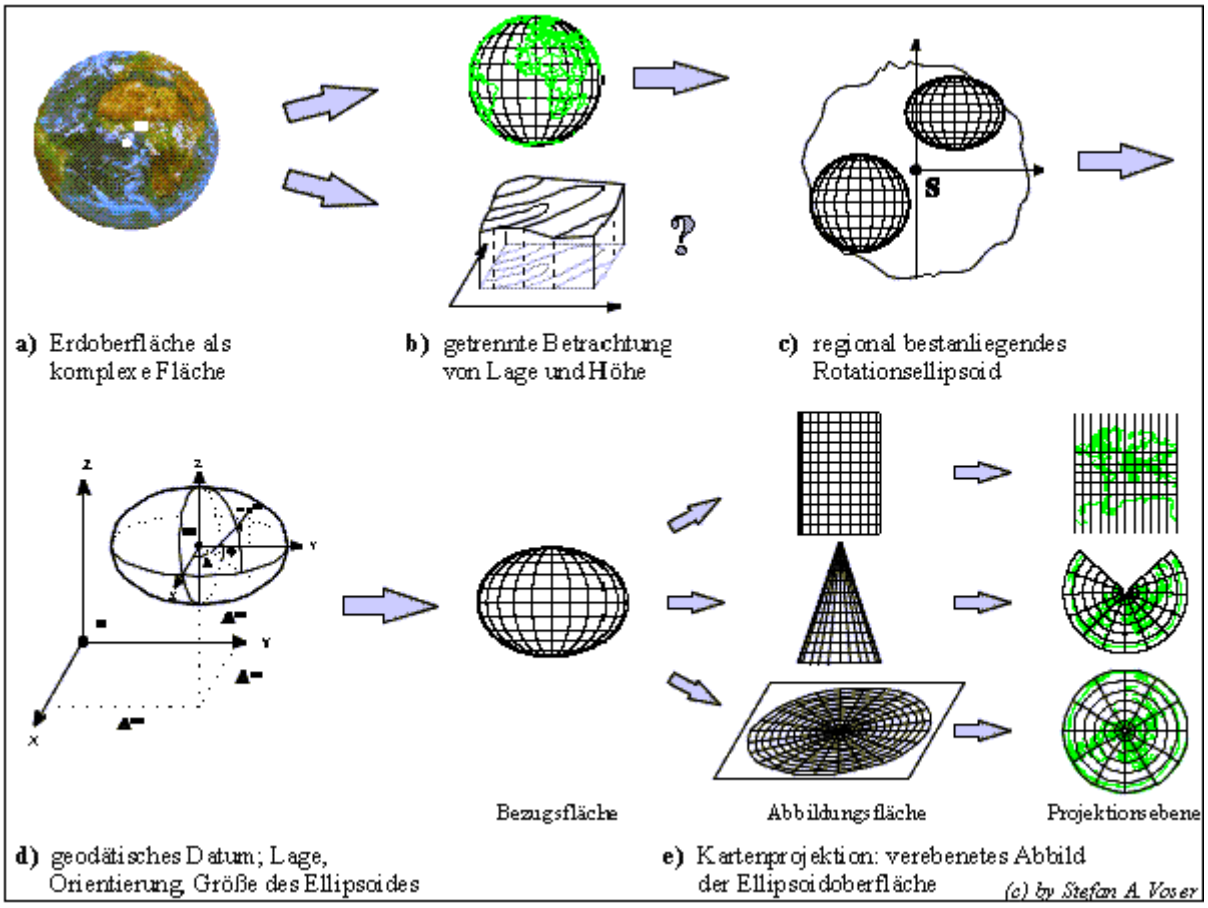

Figura I.1: Cómo transformar la superficie de la tierra en coordenadas horizontales en 2D.

Estructura de datos para los sistemas de referencia y asignaciones entre ellos

La estructura propuesta para los datos necesarios para la descripción de los sistemas de coordenadas y la correspondencia entre ellos, no se limita a los sistemas de coordenadas de referencia, sino que se ha concebido a propósito para los sistemas de coordenadas en general. Nuestra intención es también permitir la descripción de los sistemas de coordenadas digitalizadoras y de pantalla o sistemas de coordenadas de símbolos, sin ninguna referencia explícita a la superficie de la Tierra.

Los sistemas de coordenadas y la correspondencia entre ellos, son los dos conceptos clave para la caracterización exacta de la referencia espacial de los geodatos. En consecuencia, el modelo conceptual (o esquema conceptual) de la estructura de datos presenta dos grupos principales de clases, que son "sistemas de coordenadas para fines geodésicos" y "asignaciones entre sistemas de coordenadas". La tercera dimensión, altitud, se trata de la siguiente manera: En un sistema de coordenadas cartesianas 3D, la altitud se ha integrado implícitamente como la tercera coordenada. Sin embargo, en el uso cotidiano, los sistemas de coordenadas suelen ser una combinación de sistemas de coordenadas horizontales 2D y un sistema de altura 1D adicional. Los datos (meta) de los sistemas de coordenadas de este tipo son descritos por dos conjuntos de datos independientes, primero por los datos de un sistema de coordenadas 2D (2D cartesiano o elíptico) y secundariamente por los datos de un sistema de altitud del tipo apropiado (normal, ortométrico o elíptico).

¿Cómo ayudan las estructuras de datos propuestas a realizar la correspondencia entre sistemas de coordenadas? De la siguiente manera: Los sistemas de coordenadas forman nodos y las correspondencias

entre ellos constituyen bordes en una estructura gráfica. En la sección DOMAIN de un modelo de aplicación (esquema de aplicación) se encuentra el nombre del sistema de coordenadas en uso. Si las coordenadas dadas han de ser asignadas a otro sistema de coordenadas o, por ejemplo, si se van a calcular los parámetros GeoTIFF que corresponden a la asignación, entonces un programa apropiado, dentro de la estructura gráfica de sistemas de coordenadas y asignaciones, tiene que encontrar el camino más corto posible desde el nodo del sistema de coordenadas dado (según DOMAIN) hasta el nodo del sistema de destino. A partir de ahí, deben calcularse las asignaciones necesarias desde el sistema fuente a través de posibles sistemas de coordenadas intermedias al sistema objetivo.

Para la descripción de los sistemas de coordenadas están disponibles en INTERLIS dos clases internas y palabras clave: AXIS y COORDSYSTEM. Estos se emplean dentro del modelo de datos conceptuales (el modelo del sistema de coordenadas o el esquema del sistema de coordenadas) "CoordSys" (ver más abajo). Encontrará más detalles en el capítulo Sistemas de referencia en el Manual de referencia INTERLIS versión 2.

Bibliografía

[Bugayevskiy 1995] Bugayevskiy Lev M., Snyder John P.: Map Projections, A Reference Manual, Taylor&Francis, London, Bristol 1995.

[Gubler et al. 1996] Gubler E., Gutknecht D., Marti U., Schneider D. Signer Th., Voge B., Wiget A.: Die neue Landesvermessung der Schweiz LV95; VPK 2/96.

[Marti 1997] Marti, Urs: Geoid der Schweiz 1997. Geodätisch-geophysikalische Arbeiten in der Schweiz, Schweizerische Geodätische Kommission, Volume 56, Zürich, 1997.

[Torge 1975] Torge, Wolfgang: Geodäsie. Sammlung Göschen 2163, de Gruyter, Berlin – New York, 1975.

[Snyder 1987] Snyder, John P.: Map Projections - A Working Manual, U.S. Geological Survey Professional Paper 1395, Washington, 1987.

[Voser 1996] Voser, Stefan A; Anforderungen an die Geometrie zur gemeinsamen Nutzung unterschiedlicher Datenquellen; 4. deutsche Arc/Info-Anwender-Konferenz, Proceedings, März 1996, Freising.

[Voser 1999] Voser, Stefan. A.: MapRef - The Internet Collection of Map Projections and Reference Systems for Europe; 14. ESRI European User Conference, Presentation and Proceedings; 15.-17. Nov. 1999; Munich: www.mapref.org/.

El modelo del sistema de referencia

Estructura de datos para sistemas de coordenadas y sistemas de referencia de coordenadas, así correspondencia entre ellos. Modelo conceptual de datos (esquema conceptual) con INTERLIS.

```
!! File CoordSys.ili Release 2005-06-16

INTERLIS 2.3;

REFSYSTEM MODEL CoordSys (en) AT "http://www.interlis.ch/models"
  VERSION "2005-06-16" =

  UNIT
    Angle_Degree = 180 / PI [INTERLIS.rad];
    Angle_Minute = 1 / 60 [Angle_Degree];
    Angle_Second = 1 / 60 [Angle_Minute];

  STRUCTURE Angle_DMS_S =
    Degrees: -180 .. 180 CIRCULAR [Angle_Degree];
    CONTINUOUS SUBDIVISION Minutes: 0 .. 59 CIRCULAR [Angle_Minute];
    CONTINUOUS SUBDIVISION Seconds: 0.000 .. 59.999 CIRCULAR [Angle_Second];
  END Angle_DMS_S;

  DOMAIN
    Angle_DMS = FORMAT BASED ON Angle_DMS_S (Degrees ":" Minutes ":" Seconds);
    Angle_DMS_90 EXTENDS Angle_DMS = "-90:00:00.000" .. "90:00:00.000";


  TOPIC CoordsysTopic =

    !! Special space aspects to be referenced
    !! ***************************************

    CLASS Ellipsoid EXTENDS INTERLIS.REFSYSTEM =
      EllipsoidAlias: TEXT*70;
      SemiMajorAxis: MANDATORY 6360000.0000 .. 6390000.0000 [INTERLIS.m];
      InverseFlattening: MANDATORY 0.00000000 .. 350.00000000;
      !! The inverse flattening 0 characterizes the 2-dim sphere
      Remarks: TEXT*70;
    END Ellipsoid;

    CLASS GravityModel EXTENDS INTERLIS.REFSYSTEM =
      GravityModAlias: TEXT*70;
      Definition: TEXT*70;
    END GravityModel;

    CLASS GeoidModel EXTENDS INTERLIS.REFSYSTEM =
      GeoidModAlias: TEXT*70;
      Definition: TEXT*70;
    END GeoidModel;


    !! Coordinate systems for geodetic purposes
    !! ***************************************

    STRUCTURE LengthAXIS EXTENDS INTERLIS.AXIS =
      ShortName: TEXT*12;
      Description: TEXT*255;
    PARAMETER
      Unit (EXTENDED): NUMERIC [INTERLIS.LENGTH];
    END LengthAXIS;

    STRUCTURE AngleAXIS EXTENDS INTERLIS.AXIS =
      ShortName: TEXT*12;
      Description: TEXT*255;
    PARAMETER
      Unit (EXTENDED): NUMERIC [INTERLIS.ANGLE];
    END AngleAXIS;
```

```
CLASS GeoCartesian1D EXTENDS INTERLIS.COORDSYSTEM =
  Axis (EXTENDED): LIST {1} OF LengthAXIS;
END GeoCartesian1D;

CLASS GeoHeight EXTENDS GeoCartesian1D =
  System: MANDATORY (
    normal,
    orthometric,
    ellipsoidal,
    other);
  ReferenceHeight: MANDATORY -10000.000 .. +10000.000 [INTERLIS.m];
  ReferenceHeightDescr: TEXT*70;
END GeoHeight;

ASSOCIATION HeightEllips =
  GeoHeightRef -- {*} GeoHeight;
  EllipsoidRef -- {1} Ellipsoid;
END HeightEllips;

ASSOCIATION HeightGravit =
  GeoHeightRef -- {*} GeoHeight;
  GravityRef -- {1} GravityModel;
END HeightGravit;

ASSOCIATION HeightGeoid =
  GeoHeightRef -- {*} GeoHeight;
  GeoidRef -- {1} GeoidModel;
END HeightGeoid;

CLASS GeoCartesian2D EXTENDS INTERLIS.COORDSYSTEM =
  Definition: TEXT*70;
  Axis (EXTENDED): LIST {2} OF LengthAXIS;
END GeoCartesian2D;

CLASS GeoCartesian3D EXTENDS INTERLIS.COORDSYSTEM =
  Definition: TEXT*70;
  Axis (EXTENDED): LIST {3} OF LengthAXIS;
END GeoCartesian3D;

CLASS GeoEllipsoidal EXTENDS INTERLIS.COORDSYSTEM =
  Definition: TEXT*70;
  Axis (EXTENDED): LIST {2} OF AngleAXIS;
END GeoEllipsoidal;

ASSOCIATION EllCSEllips =
  GeoEllipsoidalRef -- {*} GeoEllipsoidal;
  EllipsoidRef -- {1} Ellipsoid;
END EllCSEllips;


!! Mappings between coordinate systems
!! **********************************

ASSOCIATION ToGeoEllipsoidal =
  From -- {1..*} GeoCartesian3D;
  To -- {1..*} GeoEllipsoidal;
  ToHeight -- {1..*} GeoHeight;
MANDATORY CONSTRAINT
  ToHeight -> System == #ellipsoidal;
MANDATORY CONSTRAINT
  To -> EllipsoidRef -> Name == ToHeight -> EllipsoidRef -> Name;
END ToGeoEllipsoidal;

ASSOCIATION ToGeoCartesian3D =
  From2 -- {1..*} GeoEllipsoidal;
  FromHeight-- {1..*} GeoHeight;
  To3 -- {1..*} GeoCartesian3D;
```

```
MANDATORY CONSTRAINT
  FromHeight -> System == #ellipsoidal;
MANDATORY CONSTRAINT
  From2 -> EllipsoidRef -> Name == FromHeight -> EllipsoidRef -> Name;
END ToGeoCartesian3D;

ASSOCIATION BidirectGeoCartesian2D =
  From -- {1..*} GeoCartesian2D;
  To -- {1..*} GeoCartesian2D;
END BidirectGeoCartesian2D;

ASSOCIATION BidirectGeoCartesian3D =
  From -- {1..*} GeoCartesian3D;
  To2 -- {1..*} GeoCartesian3D;
  Precision: MANDATORY (
    exact,
    measure_based);
  ShiftAxis1: MANDATORY -10000.000 .. 10000.000 [INTERLIS.m];
  ShiftAxis2: MANDATORY -10000.000 .. 10000.000 [INTERLIS.m];
  ShiftAxis3: MANDATORY -10000.000 .. 10000.000 [INTERLIS.m];
  RotationAxis1: Angle_DMS_90;
  RotationAxis2: Angle_DMS_90;
  RotationAxis3: Angle_DMS_90;
  NewScale: 0.000001 .. 1000000.000000;
END BidirectGeoCartesian3D;

ASSOCIATION BidirectGeoEllipsoidal =
  From4 -- {1..*} GeoEllipsoidal;
  To4 -- {1..*} GeoEllipsoidal;
END BidirectGeoEllipsoidal;

ASSOCIATION MapProjection (ABSTRACT) =
  From5 -- {1..*} GeoEllipsoidal;
  To5 -- {1..*} GeoCartesian2D;
  FromCo1_FundPt: MANDATORY Angle_DMS_90;
  FromCo2_FundPt: MANDATORY Angle_DMS_90;
  ToCoord1_FundPt: MANDATORY -10000000 .. +10000000 [INTERLIS.m];
  ToCoord2_FundPt: MANDATORY -10000000 .. +10000000 [INTERLIS.m];
END MapProjection;

ASSOCIATION TransverseMercator EXTENDS MapProjection =
END TransverseMercator;

ASSOCIATION SwissProjection EXTENDS MapProjection =
  IntermFundP1: MANDATORY Angle_DMS_90;
  IntermFundP2: MANDATORY Angle_DMS_90;
END SwissProjection;

ASSOCIATION Mercator EXTENDS MapProjection =
END Mercator;

ASSOCIATION ObliqueMercator EXTENDS MapProjection =
END ObliqueMercator;

ASSOCIATION Lambert EXTENDS MapProjection =
END Lambert;

ASSOCIATION Polyconic EXTENDS MapProjection =
END Polyconic;

ASSOCIATION Albus EXTENDS MapProjection =
END Albus;

ASSOCIATION Azimutal EXTENDS MapProjection =
END Azimutal;

ASSOCIATION Stereographic EXTENDS MapProjection =
END Stereographic;
```

```
    ASSOCIATION HeightConversion =
      FromHeight -- {1..*} GeoHeight;
      ToHeight -- {1..*} GeoHeight;
      Definition: TEXT*70;
    END HeightConversion;

  END CoordsysTopic;

END CoordSys.
```

El archivo MiniCoordSysData, cuyos nombres pueden aparecer en MetadataBasketDef, contiene los siguientes datos en el formato de transferencia de INTERLIS 2.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- File MiniCoordSysData.xml 2005-06-16 (http://www.interlis.ch/models) -->

<TRANSFER xmlns="http://www.interlis.ch/INTERLIS2.3"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.interlis.ch/INTERLIS2.3
                              MiniCoordSysData.xsd">
  <HEADERSECTION VERSION="2.3" SENDER="KOGIS">
    <MODELS>
      <MODEL NAME="CoordSys" URI="http://www.interlis.ch/models"
             VERSION="2005-06-16"/>
    </MODELS>

    <ALIAS>
      <ENTRIES FOR="CoordSys">
        <TAGENTRY FROM="CoordSys.Angle_DMS_S" TO="CoordSys.Angle_DMS_S"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic"
                  TO="CoordSys.CoordsysTopic"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.Ellipsoid"
                  TO="CoordSys.CoordsysTopic.Ellipsoid"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.GravityModel"
                  TO="CoordSys.CoordsysTopic.GravityModel"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.GeoidModel"
                  TO="CoordSys.CoordsysTopic.GeoidModel"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.LengthAXIS"
                  TO="CoordSys.CoordsysTopic.LengthAXIS"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.AngleAXIS"
                  TO="CoordSys.CoordsysTopic.AngleAXIS"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.GeoCartesian1D"
                  TO="CoordSys.CoordsysTopic.GeoCartesian1D"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.GeoHeight"
                  TO="CoordSys.CoordsysTopic.GeoCartesian1D"/>
        <DELENTRY TAG="CoordSys.CoordsysTopic.GeoHeight"
                  ATTR="System"/>
        <DELENTRY TAG="CoordSys.CoordsysTopic.GeoHeight"
                  ATTR="ReferenceHeight"/>
        <DELENTRY TAG="CoordSys.CoordsysTopic.GeoHeight"
                  ATTR="ReferenceHeightDescr"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.GeoHeight"
                  TO="CoordSys.CoordsysTopic.GeoHeight"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.HeightEllips"
                  TO="CoordSys.CoordsysTopic.HeightEllips"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.HeightGravit"
                  TO="CoordSys.CoordsysTopic.HeightGravit"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.HeightGeoid"
                  TO="CoordSys.CoordsysTopic.HeightGeoid"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.GeoCartesian2D"
                  TO="CoordSys.CoordsysTopic.GeoCartesian2D"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.GeoCartesian3D"
                  TO="CoordSys.CoordsysTopic.GeoCartesian3D"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.GeoEllipsoidal"
                  TO="CoordSys.CoordsysTopic.GeoEllipsoidal"/>
        <TAGENTRY FROM="CoordSys.CoordsysTopic.EllCSEllips"
```

```
                TO="CoordSys.CoordsysTopic.EllCSEllips"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.ToGeoEllipsoidal"
                TO="CoordSys.CoordsysTopic.ToGeoEllipsoidal"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.ToGeoCartesian3D"
                TO="CoordSys.CoordsysTopic.ToGeoCartesian3D"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.BidirectGeoCartesian2D"
                TO="CoordSys.CoordsysTopic.BidirectGeoCartesian2D"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.BidirectGeoCartesian3D"
                TO="CoordSys.CoordsysTopic.BidirectGeoCartesian3D"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.BidirectGeoEllipsoidal"
                TO="CoordSys.CoordsysTopic.BidirectGeoEllipsoidal"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.TransverseMercator"
                TO="CoordSys.CoordsysTopic.TransverseMercator"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.SwissProjection"
                TO="CoordSys.CoordsysTopic.SwissProjection"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.Mercator"
                TO="CoordSys.CoordsysTopic.Mercator"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.ObliqueMercator"
                TO="CoordSys.CoordsysTopic.ObliqueMercator"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.Lambert"
                TO="CoordSys.CoordsysTopic.Lambert"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.Polyconic"
                TO="CoordSys.CoordsysTopic.Polyconic"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.Albus"
                TO="CoordSys.CoordsysTopic.Albus"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.Azimutal"
                TO="CoordSys.CoordsysTopic.Azimutal"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.Stereographic"
                TO="CoordSys.CoordsysTopic.Stereographic"/>
      <TAGENTRY FROM="CoordSys.CoordsysTopic.HeightConversion"
                TO="CoordSys.CoordsysTopic.HeightConversion"/>
    </ENTRIES>
  </ALIAS>

  <COMMENT>
    example dataset ili2 refmanual appendix I
  </COMMENT>
</HEADERSECTION>

<DATASECTION>
  <CoordSys.CoordsysTopic BID="BCoordSys">
    <CoordSys.CoordsysTopic.Ellipsoid TID="BCoordSys.Bessel">
      <Name>Bessel</Name>
      <EllipsoidAlias>Bessel (1841)</EllipsoidAlias>
      <SemiMajorAxis>6377397.1550</SemiMajorAxis>
      <InverseFlattening>299.1528128</InverseFlattening>
      <Remarks>Documentation swisstopo 19031266</Remarks>
    </CoordSys.CoordsysTopic.Ellipsoid >

    <CoordSys.CoordsysTopic.Ellipsoid TID="BCoordSys.WGS72">
      <Name>WGS72</Name>
      <EllipsoidAlias>World Geodetic System 1972</EllipsoidAlias>
      <SemiMajorAxis>6378135.000</SemiMajorAxis>
      <InverseFlattening>298.2600000</InverseFlattening>
    </CoordSys.CoordsysTopic.Ellipsoid>

    <CoordSys.CoordsysTopic.GravityModel
      TID="BCoordSys.CHDeviationOfTheVertical">
      <Name>CHDeviationOfTheVertical</Name>
      <Definition>See software LAG swisstopo</Definition>
    </CoordSys.CoordsysTopic.GravityModel>

    <CoordSys.CoordsysTopic.GeoidModel TID="BCoordSys.CHGeoid">
      <Name>CHGeoid</Name>
      <Definition>See new Swiss Geoid swisstopo</Definition>
    </CoordSys.CoordsysTopic.GeoidModel>

    <CoordSys.CoordsysTopic.GeoHeight TID="BCoordSys.SwissOrthometricAlt">
```

```
    <Name>SwissOrthometricAlt</Name>
    <Axis>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>h</ShortName>
        <Description>Swiss Orthometric Altitude</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
    </Axis>
    <System>orthometric</System>
    <ReferenceHeight>373.600</ReferenceHeight>
    <ReferenceHeightDescr>Pierre du Niton</ReferenceHeightDescr>
    <EllipsoidRef REF="BCoordSys.Bessel"/>
    <GeoidRef REF="BCoordSys.CHGeoid"/>
    <GravityRef REF="BCoordSys.CHDeviationOfTheVertical"/>
  </CoordSys.CoordsysTopic.GeoHeight>

  <CoordSys.CoordsysTopic.GeoHeight TID="BCoordSys.SwissEllipsoidalAlt">
    <Name>SwissEllipsoidalAlt</Name>
    <Axis>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>H</ShortName>
        <Description>Swiss Ellipsoidal Altitude</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
    </Axis>
    <System>ellipsoidal</System>
    <ReferenceHeight>0.000</ReferenceHeight>
    <ReferenceHeightDescr>Sea level</ReferenceHeightDescr>
    <EllipsoidRef REF="BCoordSys.Bessel"/>
    <GeoidRef REF="BCoordSys.CHGeoid"/>
    <GravityRef REF="BCoordSys.CHDeviationOfTheVertical"/>
  </CoordSys.CoordsysTopic.GeoHeight>

  <CoordSys.CoordsysTopic.GeoCartesian2D TID="BCoordSys.COORD2">
    <Name>COORD2</Name>
    <Axis>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>X</ShortName>
        <Description>X-axis</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>Y</ShortName>
        <Description>Y-axis</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
    </Axis>
    <Definition>Mathematical Cartesian 2D Refsystem</Definition>
  </CoordSys.CoordsysTopic.GeoCartesian2D>

  <CoordSys.CoordsysTopic.GeoCartesian2D TID="BCoordSys.CHLV03">
    <Name>CHLV03</Name>
    <Axis>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>Y</ShortName>
        <Description>East-value</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>X</ShortName>
        <Description>North-value</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
    </Axis>
    <Definition>Geodetic Cartesian 2D Refsystem</Definition>
  </CoordSys.CoordsysTopic.GeoCartesian2D>

  <CoordSys.CoordsysTopic.GeoCartesian3D TID="BCoordSys.COORD3">
    <Name>COORD3</Name>
    <Axis>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>X</ShortName>
        <Description>X-axis</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
```

```
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>Y</ShortName>
        <Description>Y-axis</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>Z</ShortName>
        <Description>Z-axis</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
    </Axis>
    <Definition>Mathematical Cartesian 3D Refsystem</Definition>
  </CoordSys.CoordsysTopic.GeoCartesian3D>

  <CoordSys.CoordsysTopic.GeoCartesian3D TID="BCoordSys.CH1903">
    <Name>CH1903</Name>
    <Axis>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>XC</ShortName>
        <Description>Equator Greenwich</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>YC</ShortName>
        <Description>Equator East</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>ZC</ShortName>
        <Description>North</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
    </Axis>
    <Definition>Swiss Geodetic Cartesian 3D Refsystem</Definition>
  </CoordSys.CoordsysTopic.GeoCartesian3D>

  <CoordSys.CoordsysTopic.GeoCartesian3D TID="BCoordSys.WGS84">
    <Name>WGS84</Name>
    <Axis>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>XW</ShortName>
        <Description>Equator Greenwich</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>YW</ShortName>
        <Description>Equator East</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
      <CoordSys.CoordsysTopic.LengthAXIS>
        <ShortName>ZW</ShortName>
        <Description>North</Description>
      </CoordSys.CoordsysTopic.LengthAXIS>
    </Axis>
    <Definition>World Geodetic System 1984</Definition>
  </CoordSys.CoordsysTopic.GeoCartesian3D>

  <CoordSys.CoordsysTopic.GeoEllipsoidal TID="BCoordSys.Switzerland">
    <Name>Switzerland</Name>
    <Axis>
      <CoordSys.CoordsysTopic.AngleAXIS>
        <ShortName>Lat</ShortName>
        <Description>Latitude</Description>
      </CoordSys.CoordsysTopic.AngleAXIS>
      <CoordSys.CoordsysTopic.AngleAXIS>
        <ShortName>Long</ShortName>
        <Description>Longitude</Description>
      </CoordSys.CoordsysTopic.AngleAXIS>
    </Axis>
    <Definition>Coordinates on the Swiss Ellipsoid 1903</Definition>
    <EllipsoidRef REF="BCoordSys.Bessel"/>
  </CoordSys.CoordsysTopic.GeoEllipsoidal>

  <CoordSys.CoordsysTopic.ToGeoEllipsoidal
    TID="BCoordSys.FromCH1903toSwitzerland">
```

```
        <From REF="BCoordSys.CH1903"></From>
        <To REF="BCoordSys.Switzerland"></To>
        <ToHeight REF="BCoordSys.SwissEllipsoidalAlt"></ToHeight>
      </CoordSys.CoordsysTopic.ToGeoEllipsoidal>

      <CoordSys.CoordsysTopic.ToGeoCartesian3D
        TID="BCoordSys.FromSwitzerlandToCH1903">
        <From2 REF="BCoordSys.Switzerland"></From2>
        <FromHeight REF="BCoordSys.SwissEllipsoidalAlt"></FromHeight>
        <To3 REF="BCoordSys.CH1903"></To3>
      </CoordSys.CoordsysTopic.ToGeoCartesian3D>

      <CoordSys.CoordsysTopic.BidirectGeoCartesian3D
        TID="BCoordSys.WGS84toCH1903">
        <From REF="BCoordSys.WGS84"></From>
        <To2 REF="BCoordSys.CH1903"></To2>
        <Precision>measure_based</Precision>
        <ShiftAxis1>-660.077</ShiftAxis1>
        <ShiftAxis2>-13.551</ShiftAxis2>
        <ShiftAxis3>-369.344</ShiftAxis3>
        <RotationAxis1>-0:0:2.484</RotationAxis1>
        <RotationAxis2>-0:0:1.783</RotationAxis2>
        <RotationAxis3>-0:0:2.939</RotationAxis3>
        <NewScale>0.99444</NewScale>
      </CoordSys.CoordsysTopic.BidirectGeoCartesian3D>

      <CoordSys.CoordsysTopic.BidirectGeoCartesian3D
        TID="BCoordSys.CH1903toWGS84">
        <From REF="BCoordSys.CH1903"></From>
        <To2 REF="BCoordSys.WGS84"></To2>
        <Precision>measure_based</Precision>
        <ShiftAxis1>660.077</ShiftAxis1>
        <ShiftAxis2>13.551</ShiftAxis2>
        <ShiftAxis3>369.344</ShiftAxis3>
        <RotationAxis1>0:0:2.484</RotationAxis1>
        <RotationAxis2>0:0:1.783</RotationAxis2>
        <RotationAxis3>0:0:2.939</RotationAxis3>
        <NewScale>1.00566</NewScale>
      </CoordSys.CoordsysTopic.BidirectGeoCartesian3D>

      <CoordSys.CoordsysTopic.TransverseMercator
        TID="BCoordSys.FromCH1903ToSwitzerland">
        <From5 REF="BCoordSys.Switzerland"></From5>
        <To5 REF="BCoordSys.CHLV03"></To5>
        <FromCo1_FundPt>46:57:08.66</FromCo1_FundPt>
        <FromCo2_FundPt>7:26:22.50</FromCo2_FundPt>
        <ToCoord1_FundPt>600000</ToCoord1_FundPt>
        <ToCoord2_FundPt>200000</ToCoord2_FundPt>
      </CoordSys.CoordsysTopic.TransverseMercator>

      <CoordSys.CoordsysTopic.HeightConversion TID="BCoordSys.ElliphToOrth">
        <FromHeight REF="BCoordSys.SwissEllipsoidalAlt"></FromHeight>
        <ToHeight REF="BCoordSys.SwissOrthometricAlt"></ToHeight>
      </CoordSys.CoordsysTopic.HeightConversion>

      <CoordSys.CoordsysTopic.HeightConversion TID="BCoordSys.OrthToElliph">
        <FromHeight REF="BCoordSys.SwissOrthometricAlt"></FromHeight>
        <ToHeight REF="BCoordSys.SwissEllipsoidalAlt"></ToHeight>
      </CoordSys.CoordsysTopic.HeightConversion>
    </CoordSys.CoordsysTopic>
  </DATASECTION>
</TRANSFER>
```

Ejemplo

¿Qué información dentro de un modelo de aplicación (o esquema de aplicación) es necesaria para identificar inequívocamente el sistema de coordenadas empleado, el sistema de referencia de coordenadas?

```
MODEL Example (en) AT "http://www.interlis.ch/"
  VERSION "2005-06-16" =

  IMPORTS CoordSys;

  REFSYSTEM BASKET BCoordSys ~ CoordSys.CoordsysTopic
    OBJECTS OF GeoCartesian2D: CHLV03
    OBJECTS OF GeoHeight: SwissOrthometricAlt;

  DOMAIN
    LCoord = COORD
      480000.000 .. 850000.000 [INTERLIS.m] {CHLV03[1]},
       60000.000 .. 320000.000 [INTERLIS.m] {CHLV03[2]},
      ROTATION 2 -> 1;
    Height = COORD
      -200.000 .. 5000.000 [INTERLIS.m] {SwissOrthometricAlt[1]};
    HCoord = COORD
      480000.000 .. 850000.000 [INTERLIS.m] {CHLV03[1]},
       60000.000 .. 320000.000 [INTERLIS.m] {CHLV03[2]},
        -200.000 ..   5000.000 [INTERLIS.m] {SwissOrthometricAlt[1]},
      ROTATION 2 -> 1;


  TOPIC T =

    CLASS ControlPoint =
      Name: TEXT*20;
      Position: LCoord;
    END ControlPoint;

  END T;

END Example.
```

Apéndice J (sugerencia de extensión estándar) modelos de simbología

Nota

La siguiente especificación no es un componente normativo de INTERLIS. Esta es una sugerencia de extensión estándar basada en el Manual de Referencia de INTERLIS Versión 2 en el sentido de una recomendación. Sin embargo, se tiene la intención de ponerlo en discusión y posiblemente convertirlo en una regulación más definida. Consulte los correspondientes manuales de INTERLIS 2 para obtener ejemplos de aplicación.

Modelo simbología abstracta

Descripción del modelo de simbología abstracta.

```
!! File AbstractSymbology.ili Release 2005-06-16

INTERLIS 2.3;

CONTRACTED SYMBOLOGY MODEL AbstractSymbology (en)
  AT "http://www.interlis.ch/models"
  VERSION "2005-06-16" =

  UNIT
    Millimeter [mm] = 0.001 [INTERLIS.m];
    Angle_Degree = 180 / PI [INTERLIS.rad];

  DOMAIN
    Style_COORD2 (ABSTRACT)   = COORD NUMERIC, NUMERIC;
    Style_COORD3 (ABSTRACT)   = COORD NUMERIC, NUMERIC, NUMERIC;
    Style_POLYLINE (ABSTRACT) = POLYLINE WITH (STRAIGHTS, ARCS)
                                VERTEX Style_COORD2;  !! {Planar}?
    Style_SURFACE (ABSTRACT)  = SURFACE WITH (STRAIGHTS, ARCS)
                                VERTEX Style_COORD2;
    Style_INT (ABSTRACT)      = NUMERIC; !! [Units?]
    Style_FLOAT (ABSTRACT)    = NUMERIC; !! [Units?]
    Style_ANGLE (ABSTRACT)    = 0.000 .. 359.999 CIRCULAR [Angle_Degree]
                                COUNTERCLOCKWISE;  !! RefSystem?


  TOPIC Signs =

    !! Graphic interface

    CLASS TextSign (ABSTRACT) EXTENDS INTERLIS.SIGN =
    PARAMETER
      Txt      : MANDATORY TEXT;
      Geometry : MANDATORY Style_COORD2;
      Rotation : Style_ANGLE; !! Default 0.0
      HAli     : HALIGNMENT;  !! Default Center
      VAli     : VALIGNMENT;  !! Default Half
    END TextSign;

    CLASS SymbolSign (ABSTRACT) EXTENDS INTERLIS.SIGN =
    PARAMETER
      Geometry : MANDATORY Style_COORD2;
      Scale    : Style_FLOAT;
      Rotation : Style_ANGLE; !! Default 0.0
    END SymbolSign;

    CLASS PolylineSign (ABSTRACT) EXTENDS INTERLIS.SIGN =
    PARAMETER
```

```
      Geometry : MANDATORY Style_POLYLINE;
    END PolylineSign;

    CLASS SurfaceSign (ABSTRACT) EXTENDS INTERLIS.SIGN =
    PARAMETER
      Geometry : MANDATORY Style_SURFACE;
    END SurfaceSign;

  END Signs;

END AbstractSymbology.
```

Modelo de simbología estándar

Descripción del modelo extendido de simbología estándar construido sobre su versión abstracta.

```
!! File StandardSymbology.ili Release 2005-06-16

INTERLIS 2.3;

CONTRACTED SYMBOLOGY MODEL StandardSymbology (en)
  AT "http://www.interlis.ch/models"
  VERSION "2005-06-16" =

  !! Extended symbology model with symbol libraries and priorities.

  IMPORTS AbstractSymbology;

  UNIT
    Angle_Degree = 180 / PI [INTERLIS.rad];

  DOMAIN
    SS_Priority = 0 .. 9999;
    SS_Float    = -2000000000.000 .. 2000000000.000;
    SS_Angle    = 0.000 .. 359.999
                  CIRCULAR [Angle_Degree] COUNTERCLOCKWISE;
    SS_Coord2   = COORD -2000000000.000 .. 2000000000.000 [INTERLIS.m],
                        -2000000000.000 .. 2000000000.000 [INTERLIS.m],
                  ROTATION 2 -> 1;
    SS_Polyline = POLYLINE WITH (STRAIGHTS, ARCS)
                  VERTEX SS_Coord2;
    SS_Surface  = SURFACE WITH (STRAIGHTS, ARCS)
                  VERTEX SS_Coord2;


  TOPIC StandardSigns EXTENDS AbstractSymbology.Signs =

    !! StandardSigns contains symbol libraries and symbol interfaces.
    !! The libraries (colors, fonts/symbols and line patterns) form the
    !! base for the construction of concrete symbols. The symbol interfaces
    !! extend the symbol interfaces of AbstractSymbology by priorites.

    !! Library section
    !! +++++++++++++++

    !! Color library
    !! =============
    !! Colors are defined by LCh values with transparency.

    CLASS Color =
      Name: TEXT*40; !! name of color, i.e. "light green"
      L: MANDATORY 0.0 .. 100.0; !! Luminance
      C: MANDATORY 0.0 .. 181.1; !! Chroma
      H: MANDATORY 0.0 .. 359.9 CIRCULAR [Angle_Degree] COUNTERCLOCKWISE; !! Hue
      T: MANDATORY 0.000 .. 1.000; !! Transparency: 0=totally transparent, 1=opaque
    END Color;

    !! Polyline attributes
```

```
!! ++++++++++++++++++
!! Presentation parameters for simple continuous lines. Polyline attributes
!! are used by all other polyline definitions (see below).

CLASS PolylineAttrs =
  Width       : SS_Float;
  Join        : ( !! connection form for line segments
    bevel,
    round,
    miter
  );
  MiterLimit  : 1.0 .. 1000.0; !! only for Join = miter
  Caps        : (  !! termination form at end of line
    round,
    butt
  );
END PolylineAttrs;

!! Font- and symbol library
!! ========================
!! Symbols are a collection of lines and surfaces. Symbols are
!! organized in fonts. A font can be either a text font or a symbol
!! font. If the font is a text font (Type = #text), every symbol
!! (Character) has an UCS4 code (Unicode) and a spacing parameter assigned.

STRUCTURE FontSymbol_Geometry (ABSTRACT) =
  !! Basic structure for uniform treatment of all symbol geometries.
END FontSymbol_Geometry;

STRUCTURE FontSymbol_Polyline EXTENDS FontSymbol_Geometry =
  Color       : REFERENCE TO Color; !! only for symbols
  LineAttrs   : REFERENCE TO PolylineAttrs;
  Geometry    : MANDATORY SS_Polyline;
END FontSymbol_Polyline;

STRUCTURE FontSymbol_Surface EXTENDS FontSymbol_Geometry =
  FillColor   : REFERENCE TO Color; !! only for symbols
  Geometry    : MANDATORY SS_Surface;
  !! Remark: Has no line symbology, because the boundary is *not* part
  !! of the surface. With FillColor you define only the color of the
  !! surface filling.
END FontSymbol_Surface;

CLASS FontSymbol =
  !! All font symbols are defined for size 1.0 and scale 1.0.
  !! The value is measured in user units (i.e. normally [m]).
  Name        : TEXT*40; !! Symbol name, if known
  UCS4        : 0 .. 4000000000; !! only for text symbols (characters)
  Spacing     : SS_Float; !! only for text symbols (characters)
  Geometry    : LIST OF FontSymbol_Geometry
                RESTRICTION (FontSymbol_Polyline; FontSymbol_Surface);
END FontSymbol;

CLASS Font =
  Name        : MANDATORY TEXT*40; !! Font name or name of external font
  Internal    : MANDATORY BOOLEAN; !! Internal or external font
                                   !! Only for internal fonts the geometric
                                   !! definitions of the symbols is contained
                                   !! in FontSymbol.
  Type        : MANDATORY (
    symbol,
    text
  );
  BottomBase  : SS_Float; !! Only for text fonts, measured relative to text
                          !! height 1.0
END Font;

ASSOCIATION FontAssoc =
```

```
  Font -<#> {1} Font;
  Symbol -- {0..*} FontSymbol;
END FontAssoc;

!! Line symbology library
!! ======================
!! With the line sybology library the user can define continuous, dashed or
!! patterned lines. It is also possible to define multi line symbologies.
!! Each line in a multi line symbology can be continuous, dashed or patterned
!! for itself. The offset indicates the distance from the middle axis. All
!! are stored in the library relative to the width 1.0. The width can be over
!! written by the symbology parameter Width in the symbology interface. For
!! continuous lines the Width parameter defines the total width of the line,
!! for multi lines the parameter Width scales the attribute value offset.

CLASS LineStyle (ABSTRACT) =
  Name        : MANDATORY TEXT*40;
END LineStyle;

CLASS LineStyle_Solid EXTENDS LineStyle =
END LineStyle_Solid;

ASSOCIATION LineStyle_SolidColorAssoc =
  Color -- {0..1} Color;
  LineStyle -- {1} LineStyle_Solid;
END LineStyle_SolidColorAssoc;

ASSOCIATION LineStyle_SolidPolylineAttrsAssoc =
  LineAttrs -- {0..1} PolylineAttrs;
  LineStyle -- {1} LineStyle_Solid;
END LineStyle_SolidPolylineAttrsAssoc;

STRUCTURE DashRec =
  DLength     : SS_Float; !! Length of dash
END DashRec;

CLASS LineStyle_Dashed EXTENDS LineStyle =
  Dashes      : LIST OF DashRec; !! 1. dash is continuous
                                 !! 2. dash is not visible
                                 !! 3. dash is continuous
                                 !! etc.
END LineStyle_Dashed;

ASSOCIATION LineStyle_DashedColorAssoc =
  Color -- {0..1} Color;
  LineStyle_Dashed -- {1} LineStyle_Dashed;
END LineStyle_DashedColorAssoc;

ASSOCIATION LineStyle_DashedLineAttrsAssoc =
  LineAttrs -- {0..1} PolylineAttrs;
  LineStyle_Dashed -- {1} LineStyle_Dashed;
END LineStyle_DashedLineAttrsAssoc;

STRUCTURE Pattern_Symbol =
  FontSymbRef : MANDATORY REFERENCE TO FontSymbol;
  ColorRef    : REFERENCE TO Color;
  Weight      : SS_Float; !! Width for symbol lines
  Scale       : SS_Float; !! Default: 1.0
  Dist        : MANDATORY SS_Float; !! Distance along polyline
  Offset      : MANDATORY SS_Float; !! Vertical distance to polyline axis
END Pattern_Symbol;

CLASS LineStyle_Pattern EXTENDS LineStyle =
  PLength     : MANDATORY SS_Float;
  Symbols     : LIST OF Pattern_Symbol;
  !! after PLength the pattern is repeated
END LineStyle_Pattern;
```

```
!! Symbology interface
!! ++++++++++++++++++

!! Text interface
!! ==============

CLASS TextSign (EXTENDED) =
  Height      : MANDATORY SS_Float;
  Weight      : SS_Float; !! line width for line fonts
  Slanted     : BOOLEAN;
  Underlined  : BOOLEAN;
  Striked     : BOOLEAN;
  ClipBox     : SS_Float; !! Defines a rectangular surface around the text
                          !! with distance ClipBox from text.
PARAMETER
  Priority    : MANDATORY SS_Priority;
END TextSign;

ASSOCIATION TextSignFontAssoc =
  Font -- {1} Font;
  TextSign -- {0..*} TextSign;
MANDATORY CONSTRAINT
  Font -> Type == #text;
END TextSignFontAssoc;

ASSOCIATION TextSignColorAssoc =
  Color -- {0..1} Color;
  TextSign -- {0..*} TextSign;
END TextSignColorAssoc;

ASSOCIATION TextSignClipFontAssoc =
  ClipFont -- {0..1} Font;
  TextSign2 -- {0..*} TextSign;
END TextSignClipFontAssoc;

!! Symbol interface
!! ================

CLASS SymbolSign (EXTENDED) =
  Scale       : SS_Float;
  Rotation    : SS_Angle;
PARAMETER
  Priority    : MANDATORY SS_Priority;
END SymbolSign;

ASSOCIATION SymbolSignSymbolAssoc =
  Symbol -- {1} FontSymbol;
  SymbolSign -- {0..*} SymbolSign;
END SymbolSignSymbolAssoc;

ASSOCIATION SymbolSignClipSymbolAssoc =
  ClipSymbol -- {0..1} FontSymbol;
  SymbolSign2 -- {0..*} SymbolSign;
END SymbolSignClipSymbolAssoc;

ASSOCIATION SymbolSignColorAssoc =
  Color -- {0..1} Color;
  SymbolSign -- {0..*} SymbolSign;
END SymbolSignColorAssoc;

!! Polyline interface
!! ==================

CLASS PolylineSign (EXTENDED) =
  !! The parameter Width of the interface influences the width *and*
  !! the scale of start- and endsymbols.
PARAMETER
  Priority    : MANDATORY SS_Priority;
```

```
      Width       : SS_Float; !! Width of line symbology, default = 1.0
    END PolylineSign;

    ASSOCIATION PolylineSignLineStyleAssoc =
      Style -- {1} LineStyle;
      PolylineSign -- {0..*} PolylineSign;
    ATTRIBUTE
      Offset      : SS_Float; !! Default 0.0
    END PolylineSignLineStyleAssoc;

    ASSOCIATION PolylineSignColorAssoc =
      Color -- {0..1} Color;
      PolylineSign -- {0..*} PolylineSign;
    END PolylineSignColorAssoc;

    ASSOCIATION PolylineSignClipStyleAssoc =
      ClipStyle -- {0..1} LineStyle; !! Used as a mask for clipping
      PolylineSign2 -- {0..*} PolylineSign;
    END PolylineSignClipStyleAssoc;

    ASSOCIATION PolylineSignStartSymbolAssoc =
      StartSymbol -- {0..1} SymbolSign; !! Symbol at start of line in opposite
                                        !! direction of line
      PolylineSign -- {0..*} PolylineSign;
    END PolylineSignStartSymbolAssoc;

    ASSOCIATION PolylineSignEndSymbolAssoc =
      EndSymbol -- {0..1} SymbolSign; !! Symbol at end of line in same
                                      !! direction as line
      PolylineSign3 -- {0..*} PolylineSign;
    END PolylineSignEndSymbolAssoc;

    !! Surface interface
    !! ==================

    CLASS SurfaceSign (EXTENDED) =
      Clip        : (
        inside,
        outside
      );
      HatchOffset : SS_Float;
    PARAMETER
      Priority    : MANDATORY SS_Priority;
      HatchAng    : SS_Angle;  !! Default 0.0
      HatchOrg    : SS_Coord2; !! Default 0.0/0.0, Anchor point for hatching
                               !! or filling
    END SurfaceSign;

    ASSOCIATION SurfaceSignColorAssoc =
      FillColor -- {0..1} Color; !! Fill color
      SurfaceSign -- {0..*} SurfaceSign;
    END SurfaceSignColorAssoc;

    ASSOCIATION SurfaceSignBorderAssoc =
      Border -- {0..1} PolylineSign; !! Border symbology
      SurfaceSign -- {0..*} SurfaceSign;
    END SurfaceSignBorderAssoc;

    ASSOCIATION SurfaceSignHatchSymbAssoc =
      HatchSymb -- {0..1} PolylineSign; !! Hatch symbology
      SurfaceSign2 -- {0..*} SurfaceSign;
    END SurfaceSignHatchSymbAssoc;

  END StandardSigns;

END StandardSymbology.
```

Ejemplo

Ver el apéndice C: Un pequeño ejemplo de carreteras.

Apendice K (Informativo) Glosario

Abreviaturas comunes, abreviatura de terminología técnica ver definiciones

Abrev.	Abreviatura.
Art.	artículo (en los textos legales).
Párr.	párrafo (en los textos legales).
Def.	Definición.
de	Deutsch (alemán).
en	English (inglés).
fr	Français (francés).
Sin.	Sinónimo.
INTERLIS	La nota INTERLIS, por ejemplo INTERLIS 2.6.4, significa que en el párrafo 2.6.4 de este Manual de Referencia de INTERLIS Versión 2 (SN 612031) se puede encontrar más información sobre este término.
→A	"A" es un término que se ha definido en este glosario.

Definiciones

Abstract class (Clase abstracta)

→Clase que no puede contener → objetos.

Nota: Una clase abstracta es siempre incompleta y forma la base para las → subclases (es decir, → especializaciones) cuyo objeto entonces no necesita ser nulo.

Aggregation (Agregación)

→ Relación real dirigida entre una → superclase y una clase inferior.→ Varias partes (subobjetos) de una → clase derivada están asignados a una entidad (metaobjeto de la → superclase). También es posible asignar varias entidades a una parte. Al copiar una entidad todas las partes asignadas se copian también. Al borrar una entidad partes asignadas pueden seguir existiendo.

Nota 1: Por medio de una agregación →se describe la relación entre una entidad y sus partes (por ejemplo, el coche / motor). El →rol de la →subclase puede ser denominado con "es-parte-de".

Nota 2: En→ INTERLIS 2 una agregación se indica análoga a la notación →diagrama de clases UML con un rombo (void) (- <>).

Nota 3: Ver también →composición.

Amendment (Enmienda)

→ Operación de ahorro de consistencia en una → base de datos.

Area (Área)

Superficie general → plana de una → división de area.

Sin. →Objeto área.

Area division (División de área)

Conjunto de →superficies generales planas que no tienen ningún →punto o sólo puntos límite en común.

Area object (Objeto Área)

Sin. para la →zona.

Argument (Argumento)

→Valor de un →parámetro.

Association (Asociación)

Una relación real, que no limita la independencia de las →clases en cuestión. Los →objetos asignados pueden ser copiados o borrados independientemente unos de otros.

Nota 1: En →INTERLIS 2 la →clase de asociación está disponible para describir una asociación.

Nota 2: Véase también la → atributo de referencia, →agregación →y composición.

Association class (Clase de Asociación)

→Elemento de clase para describir una → asociación, →agregación o →composición.

Attribute (Atributo)

De datos (elementos) que corresponden a una característica específica de →objetos de una →clase y de elementos de la →estructura de una →estructura (cf. INTERLIS 2.6.4). Cada atributo tiene un nombre de atributo y un →dominio.

Sin. property (en).

Nota: Cada →objeto de una→ clase contiene asimismo→ un elemento de datos de un atributo con un valor →individual. Gráficamente un atributo corresponde a la columna de una→ tabla.

Attribute specialization (Especialización de Atributo)

La restricción del → dominio de un →atributo.

Nota: Especialización de atributo también se emplea en la definición de →relaciones de herencia.

Basic class (Clase base)

Sin. ambiguo para la →superclase y la →clase de vista base.

Basic data type (Tipo básico de datos)

→Dominio de valor predefinido, como TEXT o BOOLEAN (ver INTERLIS 2.8).

Basic view (Vista básica)

→Vista cuyos →objetos contribuyen a crear un nuevo punto de →vista.

Basket (Contenedor)

Colección de → objetos que pertenecen a un → topic o a sus → extensiones.

Bi-directional association (Asociación bidireccional)

Def. ver →asociación.

Boundary of a surface (Límite de una superficie)

Conjunto de todos los puntos de los límites de una →superficie.

Cardinality (Cardinalidad)

Número de →objetos de la →clase B (o A) que puede ser asignado a un →objeto de la → clase A (o B) a través de una → relación entre las →clases A y B.

Sin. Multiplicity (Multiplicidad).

Nota: En →UML también se emplea el término →multiplicidad; en cuyo caso "cardinalidad" significa el número concreto de →relaciones de objeto entre →instancias de objetos.

Cartesian coordinate system (Sistema de coordenadas Cartesianas)

→sistema del espacio euclidiano cuyos → ejes son rectas perpendiculares entre sí y en pares de coordenadas.

Cartographic sign system (Sistema de signos cartográficos)

Conjunto de posibilidades de representación gráfica de los → símbolos gráficos.

Nota 1: Un →elemento gráfico concreto mostrado en pantalla o impreso en papel es el resultado de un proceso de múltiples niveles →mediante el cual se seleccionan los objetos (selección), a continuación, representada en →símbolos gráficos (cartografía) y montado, de forma gráfica mostrada (representación) y representado (pantalla).

Nota 2: En→ INTERLIS 2 los dos primeros niveles se determinan por medio de una →descripción de representación, todos los demás niveles están sujetos a la aplicación de cada sistema resp. "Conductor", en algunos casos existen ciertas normas gráficas tales como PostScript, HPGL, OpenGL, Java2D, SVG.

Change database (Base de datos de cambios)

→ base de datos temporal en la que se pueden ejecutar →cambios de →objetos. Una base de datos de cambio recibe sus →objetos de una→ base de datos principal y los devuelve después de su procesamiento →(actualización).

Nota: Una base de datos de cambios puede operar en el mismo →sistema que la base de datos primaria (base de datos de cambio interno) o en otro →sistema (base de datos de cambio externo).

Class (Clase)

Conjunto de →objetos con las mismas propiedades y →operaciones. Cada propiedad es descrita por un →atributo, cada →operación mediante una →firma.

Sin. clase de objeto, un conjunto de entidades, el tipo de objeto, tipo de característica, función.

Nota 1: Una clase descrita con →INTERLIS 2 corresponde a una clase UML con nada más que "público", es decir, los atributos→ visibles.

Nota 2: Cf. →super clase, →subclase, →tabla→, así como elemento de clase.

Nota 3: Las clases no necesitan contener necesariamente→ objetos. Si ellas contienen →objetos hablamos de→ clases concretas, si no de →clases abstractas.

Class diagram (Diagrama de clase)

Representación gráfica de las→ clases y sus →relaciones.

Class element (Elemento de clase)

→elemento de modelado "de la clase nivel de modelado". Para ser exactos: elementos de la clase se denominan →clase,→ estructura, →clase de asociación,→ vista, →vista de proyección y→definición gráfica.

Class interface (Clase de interface)

llamada a la función de una parte o de la totalidad de las →operaciones de una →clase.

Sin. interfaz de programa, interfaz de software, interfaz.

Nota 1: Una →clase puede tener varias interfaces de clase. Para cada una de ellas se puede definir una clase de →interfaz separada. El →esquema conceptual de una →clase de interfaz se compone sólo de →firmas.

Nota 2: Véase también la → interfaz de usuario y la →interfaz de datos.

Class specialization (Especialización de clase)

Restricciones de una →clase a través de →atributos adicionales, →relaciones,→ restricciones de consistencia o especialización de →atributo.

Nota: Las especializaciones de clase se utilizan para definir relaciones de →herencia.

Complete data transfer (Transferencia de datos completa)

→Transferencia de datos del →estado completo de una base de datos desde el →emisor (fuente) hacia el →receptor (destino).

Comportment rule (Regla de comportamiento)

Las condiciones bajo las cuales, por una parte, se recibirán →mensajes de un sistema remitente y, por otra parte será transferido un →mensaje, conteniendo los argumentos de salida de la →interfaz de clase, al sistema remitente del →mensaje, incluyendo una llamada de una interfaz de clase.

Composition (Composición)

→Relación real dirigida entre una superclase y una →clase →subordinada. Varias partes (subobjetos de la →clase subordinada) se asignan a un conjunto (superobjeto de la →superclase), mientras que hay un máximo de un conjunto que se puede asignar a una parte. Al copiar una totalidad, todas las partes asignadas se copian al mismo tiempo. Del mismo modo, cuando se elimina un conjunto también se eliminan todas las partes asignadas.

Nota 1: Todas las partes son dependientes; que inalterablemente forman parte de la totalidad. De esta manera todas las →clases involucradas no llevan →relaciones iguales, sino que forman una jerarquía compuesta.

Nota 2: En→ INTERLIS 2 una composición se define como una clase de →asociación.

Nota 3: Cf. →atributo de estructura.

Conceptual schema (Esquema conceptual)

Def. cf. → data schema (note 2).

Sin. esquema conceptual de →datos.

Concrete class (Clase concreta)

→Clase que puede contener →objetos.

Nota: Cf. →clase abstracta.

Consistency constraints (Constricciones de consistencia)

Restricciones que todos los →objetos deben cumplir.

Sin. Condición, condición limitante, garantía, →constricción (es).

Nota: Algunas condiciones de constricción están predefinidas en → INTERLIS 2. Otras pueden ser definidas formalmente por medio de→ funciones, →expresiones lógicas o reglas y están sujetas a un →contrato.

Constraint attribute (Atributo de constricción)

→Atributo, para el que se ha definido una constricción de →consistencia.

Constraint class (Clase de restricción)

→Clase, para la que se ha definido una constricción de →consistencia.

Constraint (Constricción)

Sin. para la constricción de → consistencia.

Nota de la versión en español:

Constricción: acción y efecto de constreñir: Obligar, precisar, compeler por fueza a hacer algo. Limitar o reducir.

Restrincción: acción y efecto de restringir: Ceñir, circunscribir o reducir a menores posibilidades. 2ª acepción: apretar, constreñir, restriñir (es decir, como sinónimo de apretar como oprimiendo).

Podría decirse que restringir es poner límites, por lo que una restricción podría considerarse como el establecimiento de prohibiciones o de imposibilidades. Por el contrario, constreñir se refiere a limitar, pero también a obligar a que se haga algo, por lo que es un concepto más próximo a “Constraint”, por tanto, aunque una lectura técnica aceptará “Restricción”, en este manual se considerar mejor el uso de “Constricción”

Contract (Contrato)

Acuerdo con los proveedores de herramientas de software .

Nota: Se usan por ejemplo en los modelos de INTERLIS que usan →funciones no predefinidas, modelos de →simbología o formatos de tipos de →líneas no predefinidas.

Conversion (Conversión)

→Reasignación de un sistema de →coordenadas (de su espacio) a otro →sistema de coordenadas, definido estrictamente por una serie de fórmulas y sus →parámetros.

Nota: El término conversión en ocasiones se utiliza para el cambio de formato de →archivos de transferencia.

Coordinate reference system (Sistema de coordenadas de referencia)

Sin. →para el sistema de referencia.

Coordinate system (Sistema de coordenadas)

Base de un espacio vectorial euclidiano, esto es, base original del espacio vectorial euclidiano asignado cuando se trata de un homomorfismo de mapa de una diversidad (para detalles ver análisis vectorial).

Nota: Desde el punto de vista de los datos, un sistema de coordenadas se define por sus ejes que son rectas (en →INTERLIS 2 denominado LongitudAXIS) o arcos elípticos (denominados AngleAXIS), dependiendo del tipo de →espacio que permitan medir..

Corner (Esquina)

Parte ni lisa de una → cadena de líneas.

CSL

Abr. para el esquema conceptual de idiomas.

Curve segment (Segmento Curva)

Subconjunto del →espacio, es el conjunto de imágenes de una asignación lisa e→ inyectiva de un intervalo de la recta numérica.

Sin. line segment.

Data abstraction (Abstraccion de datos)

Hacer abstracción (omitiendo, entre otros) de detalles sin importancia en los datos.

Nota 1: Separando el qué (→Tipo de interfaz de →clase) del cómo (implementación concreta de →clase), son posibles principios de la abstracción →generalización y la →especialización.

Nota 2: La realización efectiva de las →operaciones y la estructura interna del →objeto o elemento de →estructura están ocultos, es decir, las características son consideradas de una manera abstracta y no se presta atención a la implementación real.

Data catalogue (Catálogo de datos)

Sin. para el →catálogo de objetos.

Data description (Descripcion de datos)

Sin. para →el esquema de datos y →modelo de datos.

Data description language (Lenguaje de Descripción de Datos (DDL))

El lenguaje formal para la descripción exacta de las estructuras de datos.

Sin. Lenguaje de esquema conceptual (CSL).

Data element (Elementos de datos:)

Def. cf. Informática, cf. →dominio.

Data interface (Interfaz de datos)

Programa para el cambio de formato de →archivos de transferencia o →protocolo para la →transferencia datos.

Sin. interface.

Nota: Véase también→ la interfaz de la clase y →la interfaz de usuario.

Data model (Modelo de Datos)

Descripción exacta de una estructura de datos (el llamado →esquema conceptual de datos), que es una unidad completa y auto contenida. Desde el punto de vista de una jerarquía, es el elemento más alto →de modelado.

Sin. model, data description.

Nota 1: ¡Advertencia! En la teoría de bases de datos, el modelo de datos es un sinónimo común para el formalismo conceptual (es decir, un modelo de datos se considera como un →método para la creación de un esquema conceptual).

Nota 2: Un modelo de datos tiene al menos →un tema.

Nota 3: En →INTERLIS 2 se describe mediante la palabra clave MODEL. El paquete, →que corresponde a este modelo de datos, está por encima de los otros →paquetes, que corresponden a los temas→ de un modelo de datos.

Data schema (Esquema de datos)

Descripción del contenido y organización de los datos que caracterizan una faceta de la realidad del usuario, así como las normas que las rigen y de las operaciones que se pueden ejecutar con dichos datos.

Sin. Descripción de datos, esquema, esquema conceptual, ontología.

Nota 1: esquemas de datos: plural.

Nota 2: Dependiendo del nivel de abstracción en el que se describen los datos, se distingue entre el esquema →conceptual, el esquema lógico y el esquema físico. Al formular un esquema de datos disponemos de lenguajes de →descripción de datos apropiados.

Nota 3: Cuando se trata de bases de datos, el esquema lógico formulado de acuerdo con el esquema conceptual y las posibilidades específicas de organización del sistema, también se denomina esquema interno. Tanto esquemas lógicos como físicos de los instrumentos periféricos o archivos de intercambio se llaman a menudo esquemas externos o esquemas de formato.

Data transfer (Transferencia de datos)

La transferencia de datos de una base de→ datos “A” a otra →base de datos “Z”. A se conoce como sistema primario, fuente, →remitente o sistema emisor. Z como →sistema de destino, →receptor. El envío de datos a ser transferido por el →sistema A, también se llama exportar; su aceptación por el →sistema Z se llama importar.

Sin. transferencia, transmisión de datos.

Data transfer mechanism (Mecanismo de transferencia de datos)

→Lenguaje de Descripción de datos (Conceptual) y → formato de transferencia (física), así como las reglas que rigen la derivación de un formato de dicha→ transferencia de una estructura de datos que se describe por medio de un →lenguaje de descripción de datos.

Data type (Tipo de datos)

Sin. para el →dominio.

Database (Base de datos)

La unidad de administración lógica para el tratamiento a largo plazo y la memorización de los →objetos.

Abr. DB.

Nota: Es posible ejecutar varias bases de datos en un →sistema. También es concebible que una base de datos se ha dividido en→ varios sistemas.

Database state (Estado de la base de datos)

Totalidad de todos los datos y →las relaciones de una →base de datos en un momento dado. Cada estado de base de datos tiene su nombre.

Nota: Por medio de una o varias →enmiendas una base →de datos se transfiere de un estado de la base de datos al siguiente (→actualización).

Date (Fecha)

syn ambigua. →Fecha geodésica y la indicación de tiempo (por ejemplo, 2002-06-25).

Date transformation (Fecha de transformación)

→Transformación de una fecha →geodésica (Del →espacio así definido) en otra fecha→ geodésica (Su→espacio).

Derived attribute (Atributo derivado)

→Un atributo, cuyo →dominio se calcula por medio de la regulación de una función→ (expresión lógica, cálculo).

Nota 1: Los atributos derivados no pueden ser alterados.

Nota 2: En → INTERLIS 2 la regulación de una función se define a través de una →función.

Directed relationship (Relación dirigida)

La →agregación o →composición o →referencia de atributo o →relación de herencia.

Domain (Dominio)

Conjunto de elementos de datos →homogéneos. Un →elemento de datos de un dominio se llama→ valor.

Sin. data type.

Nota 1: Cf. →tipo de datos básico.

Nota 2: Un valor también puede consistir en→ elementos de la estructura de una →subestructura.

Domain of a name (Dominio de un nombre)

→Espacio de nombres de la →categoría del nombre al que pertenece uno dado y que se corresponde con el elemento → modelado, en el que se define este nombre.

Nota 1: Dentro del dominio de un nombre, cada nombre sólo puede tener una definición o significado. Sin embargo, el mismo nombre puede definirse una vez dentro del →espacio de nombres de cada →categoría de nombres del mismo →elemento de modelado.

Nota 2: El dominio de un nombre es parte del dominio de →visibilidad de un nombre.

Drawing rule (Reglas de diseño)

Elemento de lenguaje de una →definición gráfica. Una regla de dibujo asigna un →símbolo gráfico para los →objetos de una→ clase y determina los correspondientes argumentos de símbolos de gráficos de acuerdo con los valores de atributo (es decir, datos) de los →objetos.

Sin. symbol attribute.

Element (Elemento)

idea fundamental de la teoría de conjuntos. Un juego se compone de elementos.

Sin. instance.

Nota: Ver también →elemento modelar o →elemento gráfico.

Ellipsoid coordinate system (Sistema de coordenadas Elipsoidales)

→Sistema de coordenadas en la superficie límite de 2 dimensiones de un elipsoide de 3 dimensiones (rotacional) .

Ellipsoid height (Altura del elipsoide)

Distancia euclidiana de un punto medido desde el elipsoide a lo largo de la línea normal hasta la superficie a través de este punto.

End point of a curve segment (Punto final de un segmento de curva)

Imagen del otro punto final de intervalo con la →asignación que define el →segmento de curva.

Entity (Entidad)

Sin. de → objecto.

Expression (Expresion)

Sin. para →expresión lógica.

Extension (Extension)

Sin. →para especialización.

Feature (Caracteristica)

Sin. para →objeto, a menudo también para →clase.

Feature type (Tipo de entidad)

Sin. para → clase.

File (Archivo)

Def. cf. informática.

Force (Fuerza)

Enlace entre las partes (subobjetos de la →clase subordinada) y la totalidad (superobjeto de la→ superclase) en una →relación real.

Function (Función)

→Equivalencia de →dominios de valores de parámetros de entrada con un dominio de valores de un parámetro de salida mediante una regla de cálculo (parámetro)..

Nota: En → INTERLIS 2 ciertas funciones están predefinidas, otros están sujetos a un→ contrato.

General identification (Identificación general)

→La identificación de todos los objetos (modelado) →de una comunidad →de transferencia.

Nota: Ver también la →identificación de objetos.

General surface (Superficie general)

→Superficie con un número finito adicional de puntos →singulares pero con un interior →continuo de superficie.

Generalization (Generalización)

→Papel de la →superclase en una →relación de herencia.

Nota 1: La generalización se utiliza ocasionalmente como sinónimo de →herencia (a pesar de que en realidad significa la dirección opuesta).

Nota 2: En la cartografía, generalización describe a todas las actividades debidas al escalado a y la reducción en la representación de los objetos del →mundo real.

Geodetical date (datum geodésico)

Sistema de coordenadas cartesianas tridimensionales, cuyos ejes tienen una posición y orientación fija en cuanto al centro de gravedad y al eje de rotación de la tierra.

Sin. sistema de referencia geodésico.

Geodetical reference system (Sistema de referencia geodésico)

Sin. →datum geodésico.

Geoid (Geoide)

superficie equipotencial del campo de gravedad.

Nota: Un geoide suministra un modelo físico de la tierra que se ajusta al campo de gravedad de la tierra. Es de forma irregular, ya que toma en consideración la distribución de masa irregular de la tierra. Tiene que ser imaginado como si la media de la superficie del océano fuera a continuar debajo de los continentes.

GIS

Abr. para sistemas de geoinformación o sistema de información geográfica.

Graphic definition (Definición gráfica)

→elemento de clase de un →tema gráfico, es decir, cada →tema gráfico→ de una descripción gráfica es una colección de definiciones gráficas (no de →clases!). Cada definición gráfica pertenece a→ una clase (BASADO EN) del-tema de datos correspondiente, asigna por medio del→ dibujo regla→s uno o varios símbolos→ gráficos para objetos de esta →clase y determina los →argumentos del →símbolo gráfico de acuerdo con los datos de los →objetos.

Nota: Los datos de lo→s símbolos gráficos, es decir, sus nombres y visualización gráfica están comprendidas en una biblioteca de →símbolos se describe en el modelo de simbología→ correspondiente.

Graphic description (Descripción gráfica)

Sin. →Descripción de la representación.

Graphic element (Elemento gráfico)

Descripción gráfica de un objeto→ teniendo en cuenta su geometría de 2 dimensiones y otros →atributos de este objeto, posiblemente después de l→a transformación necesaria listo para la salida de un dispositivo periférico adecuado.

Sin. graphic object.

Graphic model (Modelo gráfico)

Sin. para la →descripción gráfica.

Graphic object (Objeto gráfico)

Sin. →para el elemento gráfico.

Graphic parameter (Parámetro gráfico)

Sin. para →el parámetro de →un símbolo gráfico.

Graphic symbol (Símbolo gráfico)

Los datos para la representación →gráfica de un objeto todavía independiente de la geometría de 2 dimensiones y otros valores de los atributos de este→ objeto. Un →parámetro gráfico se denomina parámetro del →símbolo gráfico.

Sin. symbol, style.

Nota 1: Hay cuatro tipos de símbolos gráficos: (1) texto, resp. símbolo de texto (a veces llamada etiqueta de texto o simplemente etiqueta), (2) símbolo del punto (a veces también llamado signo de punto o →símbolo o pictograma simplemente), (3) y el símbolo de línea (4) (individual) símbolo superficie.

Nota 2: En →INTERLIS 2 la estructura de datos y →parámetros posibles de un símbolo gráfico se especifican dentro →de un modelo simbología y los datos correspondientes se almacenan en una→ biblioteca de símbolos. Un símbolo gráfico está referenciado a través de su nombre de símbolo -gráfico dentro de una →definición gráfica. De esta manera los →argumentos correspondientes para los →parámetros de eventuales tienen que ser definidos.

Graphic topic (Tema gráfico)

Def. cf. →Descripción representación.

Gravity model (Modelo de gravedad)

Descripción del campo de gravedad de la tierra.

Height (Altura)

Ya sea →altura elipsoidal o →altura normal o →altura orthometrical.

Help line (Linea de ayuda)

elemento →gráfico lineal que une dos elementos →gráficos o un elemento →gráfico y una etiqueta.

Nota: Un caso típico de una línea de ayuda es la representación de una combinación de una línea o superficie símbolo a una etiqueta o su correspondiente línea de medición.

IDDL

Abr. →INTERLIS (lenguaje de descripción de datos IDDL).

Identification (Identificación)

→Atributo o combinación de atributos cuyo →valor determina inequívocamente un o→bjeto dentro de su →clase.

Abr. ID.

Sin. identificador, identidad.

Nota: Dentro de un archivo INTERLIS de transferencia →a cada objeto se le asigna una identificación, además de los valores de los atributos descritos en el →esquema de datos, por lo tanto ser identificados de forma inequívoca dentro del →archivo de transferencia. Esta es una llamada de →identificación de transferencia→ (TID). Si un →TID es una →identificación general →y estable, entonces se llama una →identificación de objeto (→OID).

Identifier (Identificador)

Sin. for → identification.

Identity (Identidad)

Sin. for → identification.

ILI

Abrev. for → INTERLIS.

Nota: También comunes como extensión de nombre de los →archivos de datos que contienen un →esquema de datos concebido en→ INTERLIS (versión 1 y 2).

Implemented class (Clase implementada)

módulo de software ejecutable con→ operaciones realizadas →como métodos.

Incremental data transfer (Transferencia de datos incrementales)

→La transferencia de datos de la diferencia entre dos →estados de la base de datos desde un →emisor a un →sistema de destino.

Incremental update (Actualización incremental

→la transferencia de datos→ completa o incremental→ de un estad→o de base de→ datos de la base de datos→ principal a una base de datos secundaria.

Nota 1: Una actualización incremental procede siempre de forma secuencial→, es decir, una base de datos secundaria no tendrá que recibir varias actualizaciones incrementales al mismo tiempo.

Nota 2: Cf.→ sincronización.

Information layer (Capa de Informacion)

Conjunto de temas- →No nulos

Inheritance (Herencia)

→Método para la definición de →las relaciones de herencia entre→ súper clases y →subclases. Estos→ métodos son→ la especialización y la especialización de clase de →atributo.→

Nota 1:→ LAS Subclases corresponden a la misma idea; que tienen las mismas propiedades que sus →súper clases en las que se especializan.

Nota 2: Distinguimos entre la →herencia simple y →herencia múltiple. En el caso de una sola →herencia (de: Einfachvererbung; fr: Heritage singulaire) una →subclase hereda →sólo de una →superclase directa. En el caso de un →múltiplo herencia una clase→ hereda de varias →superclases.

Nota 3: → INTERLIS 2 sólo admite la herencia sencilla (como Java).

Inheritance relationship (Relación de herencia)

Relación dirigida entre una clase superior, llamada superclase, y una clase subordinada, llamada subclase, definida por herencia. El papel de la superclase se llama generalización; el papel de la subclase se llama especialización.

Nota 1: Los→ objetos de la →superclase son →generalizaciones de →los objetos de la →subclase. Los →objetos de la →subclase son restricciones →(especializaciones,→ extensiones) de los →objetos de la →superclase.

Nota 2: La relación de herencia es una relación de subconjunto, los →objetos de la →subclase forman un subconjunto de los →objetos de la →superclase. Así, en el caso →de los objetos de la→ subclase no nos ocupamos de nuevos →objetos, pero con una parte o sección de los →objetos de la→ superclase. Ambos →objetos de un par objeto de la relación de herencia poseen el mismo →OID.

Inner boundary (Límite interior)

Subconjunto →del borde →de una superficie plana, se trata de una línea→ poligonal cerrada simple interior.

Instance (Instancia)

Sin. →para el elemento (espécimen concreto) de un conjunto (abstracción).

Nota: Ejemplos de una instancia: Un→ valor es una instancia de un →tipo de datos. Un →objeto es una instancia de una →clase. Una →contenedor es un ejemplo de un →tema. Un par objeto es una instancia de una →clase de asociación.

Interface (Interface)

Sin. ambiguo para →interfaz de la clase, →interfaz de usuario e → interfaz de datos.

Interface class (Clase de interfaz)

Def. cf.→ interfaz de clase.

Sin. clase de interfaz de clase.

Interior of a surface (Interior de una superficie)

Conjunto de todos los puntos interiores de una →superficie.

INTERLIS 2

→Mecanismo de transferencia de datos para geodatos que consiste en INTERLIS Data Description Language (IDDL) y el formato de transferencia INTERLIS-XML (IXML), así como reglas para la derivación de IXML para una estructura de datos descrita con IDDL. IDDL, IXML y reglas de conversión se definen en la norma suiza SN 612031.

Abr. para los "sistemas de información territorial INTER" (es decir, entre →SIG).

INTERLIS-Compilador

Programa que se deriva de la descripción del formato de transferencia →INTERLIS correspondiente de un→esquema de datos→ en IDDL. Al mismo tiempo, se examina la corrección sintáctica del esquema de datos (la llamada de análisis sintáctico), cf. INTERLIS apéndice A.

INTERLIS de datos (lenguaje de descripción IDDL)

(Conceptual) →Descripción de los datos de idioma→ del mecanismo de transferencia de datos INTERLIS.

Nota: Un →esquema de datos se describe en →IDDL se puede memorizar como un archivo de texto. Para este tipo de archivos de esquema que es común añadir la abreviatura "ILI" al nombre del archivo. Ejemplo: El archivo de esquema de base de datos del conjunto de Levantamiento Catastral se llama así DM01AV.ILI.

Layer (Capa)

Dentro del ámbito de CAD es un término común para la recogida de datos gráficos de un tipo →determinado. A veces se usa en SIG para→ tema.

Layout of the plan (Diseño del plan)

Descripción del plan por medio del título del →metadatos,→ leyenda, descripción productor, fecha de emisión, la definición del tipo de signo y representación gráfica de los→ elementos adicionales tales como intersecciones de la cuadrícula y la dirección norte.

Sin. diseño del mapa.

Legend (Leyenda)

El lectura y la explicación de un mapa, el símbolo →gráfico que trabaja en ella.

Nota: Cf. →descripción gráfica, así como la →biblioteca de símbolos.

Line form type (Tipo de forma de Línea)

Forma de curvas que componen una cadena de líneas (rectas, arcos y otras geometrías de conexión). Para la definición de las geometrías de objetos apoyados por → INTERLIS 2, véase INTERLIS 2.8.12 y 2.8.13 .

Line segment (Linea de segment)

Sin. for → segmento de curva.

Line string (Linea simple)

Subconjunto del espacio, conjunto de imágenes de un mapeado continuo y parcialmente liso (pero no necesariamente inyectivo) y que sólo cuenta con un número finito de partes no lisas (denominadas esquinas).

Logical expression (Expresión lógica)

Predicados unidos mediante operadores booleanos.

Sin. expression.

Map frame (Marco del mapa)

Confines de los límites dentro de los cuales se representan los contenidos de un mapa.

Nota: Hacia el borde exterior, es posible definir las regiones que incluyan graduadas.

Map projection (Proyección de mapa)

→La conversión de un espacio elíptico o esférico →en un plano euclidiano.

Map symbol (Simbolo Mapa)

Sin. de → símbolo gráfico.

Mapping (Asignación, mapeo)

Equivalencia del espacio A, definido por un sistema de →coordenadas, en otro espacio Z que se define por un segundo →sistema de coordenadas. Reglas que asigna exactamente un punto z de Z a cada punto a de A.

Nota: Casos especiales de asignaciones son la →transformación y la →conversión.

Message (Mensaje)

Los datos que contienen las→ llamadas para las interfaces de clase, como→ argumentos para la entrada de resp. los datos que contienen los →argumentos para la salida de las→ interfaces de clase.

Metadata (Metadatos)

Los datos acerca de los datos.

Nota: Especialmente en geoinformática, metadatos son datos que indican, entre otras descripciones, en lenguaje coloquial los →objetos, la organización, la referencia espacial, calidad, disponibilidad, origen, etc.

Metamodel (Metamodelo)

→modelo de datos de →metadatos.

Metaobject (Metaobjeto)

→Objeto, cuyo objeto del mundo real es un conjunto de→ objetos.

Nota 1: Así, un metaobjeto consta de →metadatos. Existen Metaobjects para objetos→ individuales y / o para todos →los objetos de un →elemento de modelado.

Note 2:→ Metadatos→ para los valores de los →atributos →individuales de los →objetos son atributos adicionales de la →clase de estos →objetos.

Metaobject-names (Metaobjeto-nombres)

→Nombre de la categoría que únicamente se compone de nombres de →metaobjects.

Method (Método)

Ejecución de una operación por una →serie de instrucciones (es decir, un programa).

Nota: término ambiguo, que se utiliza a menudo como sinónimo de →operación.

Model (Modelo)

Sin. para el →modelo de datos.

Nota: El modelado orientado a objetos distingue entre modelos de objetos (como sinónimo de la parte de un esquema de datos que describe el contenido y la organización de los datos) y los modelos de comportamiento (como sinónimo→ de la parte de un esquema de datos→ que describe las operaciones que se pueden →ejecutar con el datos).

Model driven approach (Enfoque basado en modelos)

Enfoque que lleva desde de la realidad con el detalle observado por el usuario, a través de un esquema conceptual, a los datos y programas para su procesamiento.

Sin. Model Driven Architecture: Arquitectura Dirigida por el modelo (es).

Abrev. MDA (es).

Nota 1: Hay cuatro fases en el enfoque basado en modelos que conducen a los siguientes resultados: (1) Descripción del mundo real en lenguaje coloquial, (2) esquema de datos conceptual, (3) esquema de datos lógico, (4) esquema de datos físicos. Ambas fases (1) y (2) y sus resultados son independientes del sistema.

Nota 2: Para el establecimiento de un esquema conceptual se utilizarán herramientas como UML e INTERLIS 2. INTERLIS 2 también proporciona reglas de codificación que permiten derivar el esquema de datos físicos desde un archivo de transferencia (formato de transferencia) de un esquema conceptual (en INTERLIS 2 CSL).

Nota 3: Una de las principales ventajas de un enfoque basado en el modelo consiste en la redacción precisa, sobre todo del esquema conceptual, que permite entonces la comunicación y la comprensión de la estructura de datos entre expertos.

Model driven method (Método dirigido por el modelo)

Sin. para el →enfoque basado en modelos.

Model driven protocol (Protocolo dirigido por el Modelo)

→Protocolo cuyas→ interfaces de clase y los→ mensajes se describen por medio de un esquema conceptual →(independiente del sistema).

Modeling element (Elemento de modelado)

elemento →especial de esquema. Hay tres elementos de modelado, a saber, el →modelo de datos, el→ tema y el →elemento de clase.

Nota: El elemento de modelado y la →categoría nombre definen el →espacio de nombres.

Multiple inheritance (Herencia múltiple)

→relación de herencia que asigna más de una →Superclase para una →subclase.

Nota: La herencia múltiple no se proporciona en→ INTERLIS 2.

Multiplicity (Multiplicidad)

Sin. de→ cardinalidad.

Name category (Nombre de la categoría)

Subconjunto de los nombres de un esquema conceptual de →datos. Hay tres categorías: tipo nombre, →nombres, nombres →y nombres →metaobjeto.

Nota: Nombre y categoría →de elemento de modelado definen el →namespace.

Namespace

Conjunto de nombres (inequívocos) de una →categoría de nombre en un →elemento modelado.

Nota: El espacio de nombres es de importancia para determinar el →dominio y el dominio de la→ visibilidad de un nombre.

Normal height (Altura normal (de un punto))

Distancia entre el punto y el cuasi-geoide.

Nota: La altura normal es de una altura riguroso con respecto a la teoría del potencial. La gravedad normal media se tiene en cuenta.

Object (Objeto)

De datos de un objeto del mundo real junto con las→ operaciones que se pueden ejecutar con estos datos y con una identificación de →objetos.

Sin. entidad, tupla, instancia de objeto, característica, caso de fenómeno.

Nota 1: Cf.→ ejemplo,→ clase.

Nota 2: A diferencia de un →valor, un objeto posee una i→dentidad, existe en el tiempo y el →espacio, puede ser alterado, manteniendo su→ identidad y por medio de una referencia que puede ser de uso común. Un objeto es concreto. Está estrechamente relacionado con la existencia de las cosas reales.

Nota 3: En la literatura orientada a objetos, encontramos la siguiente definición del término objeto: Una unidad existente de hormigón con su propia identidad (inmutable) y los límites→ definidos (en el sentido figurado de la palabra) que encapsulan estado y características. Su estado está representado por →atributos y →relaciones, por sus características de →operaciones. Cada objeto pertenece exactamente a una →clase. La estructura definida de sus →atributos, así como sus características, se aplica igualmente a todos los objetos de una →clase. Sin embargo, los →valores de los→ atributos son específicos para cada objeto.

Object catalogue (Catálogo de objetos)

enumeración informal de →clases con las definiciones coloquiales (nombre y la descripción de la →clase) de todos los objetos de datos relevantes para una utilización.

Abr. OC.

Sin. data catalogue.

Nota 1: Un cataloque objeto comprende indicaciones relativas grado de requisitos detallados Descripción de calidad (sobre todo de calidad geométrica), así como normas para la grabación.

Nota 2: Un catálogo de objetos es un preliminar y un complemento conceptual del →modelo de datos.

Object class (Clase de Objetos)

Sin. de → clase.

Object identification (Identificación de objetos)

→identificación general →y estable.

Sin. identificador de objeto, la identidad del objeto.

Nota 1: Por lo general, la identificación de objetos solamente se ve alterada por un sistema y no→ por un usuario. Una identificación de objetos es una característica que distingue un →objeto de todos los demás, a pesar de que pueden poseer los mismos valores de los atributos.

Nota 2: Cf.→ transferencia de identificación.

Nota 3: En el apéndice D del Manual de Referencia Versión INTERLIS 2 encontrará una sugerencia para una identificación de objetos.

Object identifier (Identificador de objeto)

Sin. de → identificación de objeto.

Object identity (Identidad de objeto)

Sin. de → identificación de objeto.

Object instance (Instancia de objeto)

Sin. de → objecto.

Object relationship (Relación de objeto)

Dos →objetos asignados entre si por una→ relación entre las →clases a las que pertenecen.

Sin. enlace.

Object type (Tipo de Objeto)

Sin. De → clase.

Official height (Altura oficial)

Suma de todas las mediciones de nivelación (diferencias de altura) a lo largo de una carrera de nivelación de un punto de altura 0 a un punto con G deseado.

OID

Abr. para → identificador de objetos.

Onesided relationship (Relación unilateral)

Sin. para →atributo de referencia.

Ontology (Ontologia)

Sin. para → esquema de datos.

Nota 1: La ontología es una "especificación formal explícita de una conceptualización compartida", es decir, en términos gráficos, un repositorio de conceptos.

Nota 2: Las ontologías usan UML / OCL o sus propios lenguajes como DAM / OIL (DARPA Agent Markup Language + Ontology Interchange Language). Normalmente las ontologías consisten en un esquema conceptual de datos, una jerarquía taxonómica de clases (vocabulario, tesauro) y axiomas, lo que restringe las posibles interpretaciones de los términos definidos (en la mayoría de los casos con un lenguaje lógico). (En el futuro) las ontologías deben ser utilizadas como abstracciones más altas de esquemas de datos para la especificación de software y para la comunicación entre individuos.

Operation (Operación)

→Mapeo de los dominios de atributo de una clase y / o de los dominios →de entrada-parámetros en el d→ominio de un parámetro →de salida.

Nota 1: La ejecución de una operación por medio de una serie de instrucciones (es decir, un programa) se llama al →método.

Nota 2: La descripción de una operación se denomina →signature (firma) y se compone de nombres de funcionamiento y descripción de los →parámetros.

Optional (Opcional)

No es obligatorio, no tiene por qué existir o ser aplicable. Contrario: obligatorio.

Nota 1:→ Los atributos son opcionales, a menos que se indique que son de carácter obligatorio. Para los atributos obligatorios d→isponemos de la palabra clave obligatoria en →IDDL.

Nota 2: En →IDDL el término "no es obligatorio" se refiere a la→ transferencia de archivos.

Orthometric height (Altura ortométrica)

Longitud de la curva de la perpendicular (curvada) entre el →geoide y el punto.

Outer boundary (Límite exterior)

Subconjunto del → límite de una superficie plana, la cadena de →línea cerrada simple más exterior.

Package (Paquete)

Elemento del lenguaje UML para la descripción de →modelos,→ temas y partes de los temas.

Nota 1: Un paquete define un espacio de nombres, →es decir, dentro de un paquete, los nombres de los elementos contenidos deben ser inequívocos. Cada elemento del esquema →designado, puede ser referenciado en un paquete difereten, pero pertenece exactamente a un paquete (el de origen).

Nota 2: En el caso de los paquetes →UML pueden contener otros paquetes. El paquete superior contiene todo el sistema de acuerdo con el →modelo de datos de →INTERLIS 2.

Parameter (Parámetros)

Datos (elementos), cuyos →valores se transmiten a una →función, una →operación o un →metaobjeto y/o han sido devueltos por →funciones u →operaciones. Cada parámetro se suministra con un nombre, un →dominio y, en función de las →funciones o →operaciones, una dirección de transferencia (in, out, inout). El valor →concreto de un parámetro se denomina argumento.

Nota 1: Cf. → parámetro de tiempo de ejecución.

Nota 2: Mediante parámetros se describen aquellas propiedades de objetos meta que no conciernen al objeto meta, sino su uso dentro de la aplicación.

Part names (nombre de las partes)

→Categoría de nombres que consiste en nombres de parámetros de →tiempo de ejecución, →atributos, →reglas de dibujo, →parámetros, →roles, acceso a →relaciones y →vistas básicas.

Path (Ruta)

Series de nombres de →atributos y/o →clases y/o →roles de las clases de asociación, que definen un →objeto o el →valor de un →atributo que se van→ a procesar por una →expresión lógica.

Planar curve segment (Segmento de curva plana)

→segmento de curva que es un subconjunto de un →plano.

Planar general surface (Superficie general plana)

→superficie general que es un subconjunto de →un plano.

Planar surface (Superficie plana)

→Superficie que es un subconjunto de un →plano.

Plane (Plano)

Parte del espacio bidimensional.

Point (Punto)

elemento (Set) del espacio→ (considerado como un conjunto).

Polymorphism of objects (Polimorfismo de objetos)

Siempre que se esperan objetos de una superclase, también es posible tener objetos de una extensión.

Sin. polymorphy, polymorphism.

Nota 1: Ver también el →polimorfismo de las operaciones.

Nota 2: En →INTERLIS 2 nos referimos principalmente a polimorfismo de objetos.

Polymorphism of operations (Polimorfismo de operaciones)

Basándose en la firma, es concebible que objetos de clases diferentes respondan a nombres de operación idénticos (mensajes), es decir, se procesan mediante operaciones con nombres idénticos.

Sin. polymorphy, polymorphism.

Nota 1: Ver también →polimorfismo de objetos.

Nota 2: En INTERLIS 2 principalmente →se aplica polimorfismo de objetos.

Primary database (Base de datos primaria)

→Base de datos que se ocupa de la administración a largo plazo de→ los objetos de ciertos temas de →un determinado campo.

Program (Programa)

Sin. de →método.

Program interface (Interfaz del programa)

Sin. de → interfaz de clase.

Ejemplos: la API de Java o el driver Open Database Connectivity (ODBC).

Program system (Sistema de programa)

Totalidad de todos los métodos de clases necesarios para el procesamiento de una solicitud por medio de procesamiento electrónico de datos.

Propeller set (Conjunto en hélice)

Unión de un número finito de superficies triangulares que tienen exactamente un →punto en común, el centro.

Proper relationship (Relación propiamente dicha o real)

Def. cf.→ relación.

Nota de la traducción en español: Se ha traducido por Relación Real y, alguna vez, por relación propiamente dicha, frente a la relación obtenida mediante un atributo de referencia.

Property (Propiedad)

Sin. de →atributo.

Protocol (Protocolo)

Entidad de todas las interfaces de clase, mensajes y reglas de comportamiento de un conjunto de sistemas que contribuyen a la solución de una tarea de aplicación.

Receiver (Receptor)

Def. cf. →transferencia de datos.

Sin. Sistema objetivo.

Reference attribute (Atributo de referencia)

→Relación que sólo es conocida a través del primer →objeto de cada par de objetos de la →relación.

Sin. Relación →unilateral.

Reference system (Sistema de referencia)

→ Sistema de coordenadas, que aparece al final de una serie de sistemas de coordenadas y conversiones donde se produce exactamente una fecha geodésica y que se encuentra al comienzo de la serie..

Referential integrity (Integridad referencial)

Regla, que determina lo que debe suceder con una relación de objeto, o con los objetos en cuestión, si se suprimen los objetos involucrados o la propia relación..

Relationship (Relación)

Conjunto de pares de →objetos (o de forma general, objetos de n tuplas también conocidos como objetos de →relación). El primer objeto de un par pertenece a una primera →clase A, el segundo objeto a una segunda →clase B. La atribución de objetos a dichos pares será predefinida; por lo tanto, sólo debe ser descrito, es decir, modelado. Distinguimos entre la →relación propiamente dicha (es decir, →asociación, →agregación, composición), la relación de herencia y el atributo de referencia.

Nota 1: Como demuestra el concepto de vista, es por otra parte posible calcular tales asignaciones por medio de algoritmos, por ejemplo, sobre la base de valores de atributo.

Nota 2: Cf. →relación de objeto.

Nota 3: Para una relación real se o propiamente dicha, se definen tanto →fuerza como→ cardinalidad

Relationship access (Acceso a la relación)

Condiciones y posibilidades para hacer →referencia a objetos de relación y por estos medios→ también objetos de clases (ordinarias)→ a través de →caminos.

Relationship object (Objeto de relación)

Def. cf. →relación.

Replicate (Reproducir)

Para copiar mediante el cual el objeto copiado no puede →ser alterado de forma independiente del original.

Nota: Término que se utiliza principalmente en relación con la→ actualización incremental.

Representation description (Descripción de representación)

→esquema conceptual, que describe la asignación de →símbolos gráficos a o→bjetos y se compone de temas→ gráficos. Los →objetos pueden ser seleccionados en una vista.→

Sin. modelo gráfico, descripción gráfica.

Nota 1: Una descripción representación en→ INTERLIS 2 se compone de temas →gráficos cada uno de los cuales corresponde a un tema de datos (depends on). Un tema gráfico →es un conjunto de definiciones →gráficas (no de→ clases!).

Nota 2: La descripción representación en sí también puede contener esquemas→ de datos→ (por ejemplo, clases que describen la situación de textos).

Role (Función, papel, role)

Significado de los →objetos de una →clase dentro de una →relación.

Nota: En una →relación real, el papel de cada clase →involucrada se describe por su nombre, su →fuerza y su →cardinalidad. Un →atributo de referencia describe el papel de la →clase con este →atributo. Dentro de una→ herencia de relación, el papel de cada clase están implícitamente definidos.

Nota de la traducción al español: Se puede usar role, pero es preferible usar términos propios del idioma como función (tarea que lleva a cabo, por ejemplo, un extremo de la relación). Si función provoca indefinición o repetición respecto al uso de la palabra para hablar de método, puede usarse papel, pero la expresión no debe incluir el verbo “jugar”: podrá representar, tener o hacer un papel, pero no jugar un papel (anglicismo de “play a role”, pero en español el papel no se juega, si no que se interpreta, se tiene o se hace).

Run time parameter (Parametros de tiempo de ejecucion)

→Parámetro cuyo→ valor se suministra en tiempo de ejecución por un sistema de tratamiento, evaluación o representación.

Nota: Ejemplos de ello son la escala de representación, o el →datum.

Schema (Esquema)

Sin. de → esquema de datos.

Schema element (Elemento de esquema)

esquema parcial de un esquema conceptual de datos→ que posee un nombre.

Nota: Todos→ los elementos de modelado son los elementos del esquema.

Secondary database (Base de datos secundaria)

Copia del estado de la →base de datos a partir de una →base de datos primaria.

Nota: Por lo general, una base de datos secundaria no se puede encontrar en el mismo →sistema que la base de datos →primaria.

Sender (Remitente)

Def. cf.→ transferencia de datos.

Set of entities (Conjunto de entidades)

Sin. de → clase.

Sign (Signo)

Letra o dígito o marca o símbolo en blanco o puntuacion.

Signature (Firma)

Al describir la llamada de una →operación, que consiste en el nombre de la→ operación, sus →tipos de datos y los posibles nombres de sus→ parámetros y posiblemente indicación de un tipo de retorno de datos.

Simple closed line string (Línea poligonal cerrada simple)

Cadena de →línea cuya equivalencia asignada →es inyectiva, con la excepción de su →punto de inicio y el →punto final, que coinciden.

Simple inheritance (Herencia Simple)

Def. cf.→ herencia.

Simple line string (Cadena de Linea simple)

→cadena de línea cuya equivalencia →asignada también es inyectivo.

Singular point (Punto singular)

→ Punto que, junto con su entorno, puede ser deformado en un conjunto de →hélice plana, el punto mismo está en su centro.

SN

Abrev. de →Norma Suiza

Software interface (Interfaz de software)

Sin. de → interfaz de clase.

Space (Espacio)

espacio euclídeo tridimensional.

Specialization (Especialización)

→Papel de la →subclase de una →relación de herencia, a menudo también sinónimo de →herencia.

Sin. extension.

Nota 1: Cf. →especialización de clase y →atributo de especialización.

Nota 2: Dado que será necesario para la descripción de una especialización de →clase o →atributo que para la →superclase o el atributo de texto más original, a menudo en lugar de hablar de la→ extensión de la especialización.

Stable identification (Identificación estable)

→Identificación que es independiente del tiempo, es decir, que no puede ser alterado durante el ciclo de vida de un →objeto. Una vez que se ha s→uprimido un objeto, su identificación estable ya no puede ser utilizada.

Nota: Cf. → object identification.

Standard (Estándar)

Una norma “de jure” (o norma corta) es un reglamento técnico establecido por comités nacionales o internacionales de normalización. Un estándar "de facto" es un reglamento técnico, generalmente reconocido y mayoritario, pero menos vinculante que un estándar "de jure".

Nota 1: Una ley es una norma superior a los estándares de jure y de facto.

Nota 2: sinonimo alemán para el estándar "de facto" es "estándar". En Inglés estándar se utiliza para 'de facto' o estándares de jure.

Starting point of a curve segment (Punto inicial de un segmento de curva)

La representación de uno de los puntos extremos al establecer el →mapeo o asignación que define el segmento de→ la curva.

String (Cadena)

Sucesión (es decir, conjunto ordenado) →de signos.

Structure (Estructura)

Conjunto de →elementos de estructura con las mismas propiedades y→ operaciones. Sólo estas →operaciones se permite que no alterarán los datos de los →elementos de estructura. Cada propiedad es descrito por un →atributo, cada →operación con su →firma.

Nota 1: Estructuras se producen ya sea dentro o LIST- BAG-atributos →(subestructura) o existir sólo temporalmente como resultado de →funciones.

Nota 2: Cf. → class element.

Structure attribute (Atributo de la estructura)

→atributos con tipo de datos INTERLIS-2 tipo BAG o LIST.

Nota: A diferencia de la definición de una→ composición por medio de la clase de→ asociación, con una estructura de elementos de atributo →de la estructura no se puede hacer referencia, es decir, fuera del →objeto a cuya estructura atribuir valor al que pertenecen, no tienen identidad.

Structure element (Elemento de estructura)

Los datos de un objeto del mundo real con las →operaciones que podrían ser ejecutadas con estos datos, sin embargo, sin la autorización de los altera, y sin la →identificación de objetos.

Nota: Un elemento de estructura es el →ejemplo de una →estructura.

Structured domain (Dominio estructurado)

Elemento del lenguaje en INTERLIS 2 para la descripción de →atributos compuestos tales como →fecha u hora.

Subclass (sub clase)

Def. cf. → inheritance relationship.

Substructura (Sub estructura)

→Dominio que se ha definido por medio de una →estructura.

Nota: Cf. →atributo de estructura.

Super class (Super clase)

Def. cf. → inheritance relationship.

Surface (Superficie)

Unión F de un número finito de elementos de →superficie que es continua y cumple con la siguiente condición: por cada →punto P de la superficie existe un entorno que puede deformarse en un polígono plano (es decir, permite el mapeo homeomorph). Si en el curso de un punto de→ deformación P tal debe ser colocado en el límite del polígono, se convierte en un punto de F límite, de lo contrario sigue siendo un punto interior de F.

Surface element (Elemento superior)

Un elemento de superficie es un subconjunto del →espacio, que es el conjunto de imágenes de un→ mapeo suave y inyectiva de un polígono regular planar.

Symbol (Simbolo)

syn ambigua. para→ el símbolo gráfico, símbolo de idioma o símbolo semiótico.

Symbology (Simbología)

Subconjunto de elementos de un sistema →de signos cartográfica que consiste en→ símbolos gráficos, fuentes, diagramas, medios tonos.

Nota: Cf. → symbol library.

Symbol attribute (Atributo de símbolo)

Sin. para la →elaboración de reglas.

Symbol library (Libreria de símbolos)

Colección de → símbolos gráficos, estructurado de acuerdo con un modelo de →simbología.

Nota 1: Una biblioteca de símbolos siempre es una contenedor, es decir, →un archivo XML.

Nota 2: En la mayoría de los casos una biblioteca de símbolos es un aspecto concreto, la recogida específica de usuario de →símbolos gráficos.

Symbology model (Modelo de simbología)

esquema →conceptual →que describe la estructura de datos de →símbolos gráficos y sus →parámetros.

Nota 1: modelos de simbología siempre exigen →contratos.

Nota 2: En el Apéndice J del Manual de Referencia 2-INTERLIS Versión encontrará una sugerencia para un modelo de simbología extendida.

Nota 3: Cf. → symbol library.

Symbol object (Simbolo de objetos)

Sin. for → graphic symbol.

Synchronization (Sincronización)

Ajuste automático y regular de los estados de →base de datos de dos → Base de datos.

System (Sistema)

Totalidad de todos los componentes (hardware y software) que forman un sistema de procesamiento de datos y que se sometan a un uso determinado.

Table (Tabla)

→Clase de cuyos objetos →es imposible definir explícitamente las →operaciones.

Target system (Objetivo del sistema)

Sin. para el →receptor.

TID

Abr. para →la identificación de transferencia.

Topic (Tema)

Conjunto de clases cuyos→ datos en un cierto sentido pertenecen juntos, por ejemplo, que tienen una relación, pertenecen a la misma autoridad de procesamiento de datos o poseer un ritmo similar de actualizaciones.→ Los casos de los temas son →baskets (receptores).

Nota 1: En →UML, un tema s→e describe por un →paquete de debajo→ de un modelo de datos descrito, con el significado adiciona→l que este paquete (a) posee→ su propio espacio de nombres

y (b) puede depender →(ser una extensión) de otros paquetes. Un paquete de UML, asignado a un tema, puede contener otros paquetes →(anidados).

Nota 2: Beware: Con capa,→ en términos de CAD una expresión comúnmente utilizada para "superficie", nos referimos a una colección de datos gráficos. Un tema puede comprender varias capas (gráficos), →además de los datos temáticos estructurados adicionales.

Transfer (Transferir)

Sin. para → transferencia de datos.

Transfer community (Transferir comunidad)

Comunidad de →emisores y →receptores que ambos participan en un→ transferencia de datos.

Transfer file (transferencia de archivos)

→File preparado para→ la transferencia de datos en un →formato de transferencia→ apropiado.

Transfer format (formato de transmisión)

Sistema de campos de datos dentro de un archivo de transferencia.

Sin. format.

Transfer identification (Identificación de transferencia)

Def. cf. i→dentificación.

Abr. TID.

Transformation (Transformación)

→Mapeo de un sistema de →coordenadas (resp. De su espacio)→ a otro sistema de coordenada→s (resp. A su espacio→), donde la regulación de asignación (fórmula) se basa en hipótesis y los →parámetros se establecen por medio de análisis de la mayoría de estadística de las mediciones en ambos →sistemas coordinados.

Tuple (Tupla)

Sin. for → object.

Type (Tipo)

Ambiguous Sin. para→el tipo de datos (es decir, →de dominio),→ interfaz de clase, y→ la firma.

Type name (Tipo nombre)

→Nombre de la categoría que consiste en clases de →temas, l→as asociaciones, →las vistas, →as definiciones gráficas, →baskets, →unidades, →funciones, →tipos de formularios línea, →dominios, →estructuras.→

UML

Abr. para Unified Modeling Language.

Def. cf. www.omg.org/.

Unit (Unidad)

Elemento básico de una escala de medición (por ejemplo: metros, segundos).

Update (Actualización)

Una o varias →enmiendas sobre una base de →datos primaria. Por medio de una actualización de la base de →datos primaria se transfiere de una →base de datos de estado al siguiente.

Nota: Varias →modificaciones en la base de datos →primaria pueden ocurrir en paralelo al mismo tiempo. En el caso de modificaciones→ paralelas→, la base de datos principal debe garantizar la consistencia de los resultados.

User interface (Interfaz de usuario)

Interfaz gráfica de un programa de ordenador.

Sin. interfaz, interfaz gráfica de usuario.

Nota: Véase también →la interfaz de la clase y la →interfaz de datos.

Value (Valor)

→Elemento de datos de un →dominio.

Vertex (Vértice)

Sin. for → corner.

View (Ver)

→Clase cuyos →objetos son creados mediante la combinación y selección (para ser exactos por las →operaciones de vista)→ objetos de otras →clases de vista.

Nota 1:→ Los objetos de una vista no son "originales" en el sentido de que no se corresponden directamente a un objeto del mundo real. Así, una especie de punto de vista es una clase →virtual.

Nota 2: Cf. → class element.

View base class (Clase base para vistas)

→Vlass cuyos →objetos participar en la formación de una →vista.

View operation (Operacion de Vistas)

Reglamento para la definición de un nuevo objeto→ a partir de los →objetos de clases vista base, respectivamente.→ de vista →básicas.

Nota: Operaciones de Vista en → INTERLIS 2 son proyecciones, uniones, agrupaciones e inspecciones. A continuación, el conjunto de objetos puede ser restringido por medio de selecciones.

View projection (Ver proyección)

→Clase cuyos→objetos son determinados por complementar→ atributos seleccionados a partir de→ objetos de otra →clase, ver o ver proyecciones→. En particular, es posible definir más (virtual) →atributos cuyos →valores son determinados por →funciones.

Nota 1:→ Son posibles extensiones de vista de las proyecciones. Sin embargo, sus objetos →siempre serán subconjuntos del objeto-conjunto de la →clase básica,→ vista básica o proyección vista básica.

Nota 2: Cf. → class element.

Visibility domain of a name (Dominio de la visibilidad de un nombre)

Conjunto de todos los →espacios de nombres de los que se puede hacer referencia al nombre de manera no cualificada. El dominio de visibilidad del nombre consiste en su dominio de definición y de los espacios de nombres de su categoría de nombre en todos los elementos de →modelado que están jerárquicamente subordinados al elemento de modelado de su dominio de definición.

Nota: Además del →espacio de nombres de su dominio de definición, se puede definir un nombre nuevo en cada espacio de nombres de su dominio de visibilidad. Así, este espacio de nombres se convertirá en el nuevo dominio de un nombre. Este nuevo →dominio de definición y su campo de visibilidad asignado "anulan" parte del dominio de visibilidad original en la medida en que en este subdominio (que forma un subárbol de la jerarquía de elementos de modelado) sólo se aplicará la nueva definición / significado del nombre.

Apendice L (informativo)

Indice.....

.....

A

ABSTRACT 24, 25, 26, **28**, 30, 31, 34, 36, 37, 41, 43, 48, 54, 55, 57, 65, 67, 69, 70, 80, 81, 92, 93, 94, 130, 133, 150, 157, 158, 159, 160
ACCORDING 25, **70**, 72
AGGREGATES 25, **61**, 62, 65
Aggregation 66, **67**
AGGREGATION 25, 65, **67**
Alias **78**
AlignmentType 37, **40**
ALL 25, **39**, 59, 61, 63, 67, 85, 112
AND 25, **60**, 61
ANY 25, 29, **44**, 45, 92
ANYCLASS 25, **32**, 62, 63, 64, 93
ANYSTRUCTURE 25, **32**, 33, 62, 63, 64, 66, 93, 94
ARCS 25, 47, **49**, 92, 127, 128, 157, 158
ArcSegment **88**, 89
AREA 25, **49**, 53, 54, 58, 59, 66, 67, 75, 86, 89, 90, 127, 128
Argument **61**
ArgumentType **63**
AS 25, **28**, 29, 30, 34
ASSOCIATION 25, 30, **33**, 34, 68, 101, 128, 133, 149, 150, 151, 159, 160, 161, 162
AssociationDef 28, **34**
AssociationPath **61**
AssociationRef 32, **34**
AT .. 25, **27**, 70, 71, 72, 80, 81, 92, 101, 102, 112, 128, 130, 132, 140, 148, 156, 157, 158
Attribute 84, 85, **86**, 88
ATTRIBUTE 25, **30**, 34, 45, 57, 64, 67, 87, 93, 112, 162
AttributeDef 30, **31**, 34, 67
AttributePath 45, 59, **61**, 69, 70
AttributePathConst 37, **45**
AttributePathType 37, **45**
AttributePathTypeValue 86, **87**
AttributeRef **61**
ATTRIBUTES 25, **49**, 101
AttributeValue **86**
AttrType 31, **32**
AttrTypeDef **31**, 45, 57, 58, 63

B

BAG 8, 25, **31**, 32, 33, 58, 59, 62, 64, 65, 66, 75, 86, 88, 93, 133
BagValue 86, **88**
BASE 25, **67**
BaseAttrRef **42**
BASED .. 25, **42**, 43, 68, 69, 71, 94, 112, 113, 132, 140, 148
BaseExtensionDef 65, **67**
BaseType **37**
Contenedor **83**
CONTENEDOR 25, **28**, 43, 56, 71, 72, 94, 112, 140, 156
BINARY 25, **45**
BLACKBOX 25, **45**, 87
BlackboxType 37, **45**
BlackboxValue 86, **87**
BOOLEAN 25, **40**, 64, 68, 87, 92, 93, 132, 133, 159, 161
BooleanType 37, **40**
Boundaries **89**, 90
Boundary 89, **90**
BY 25, **67**

C

Cardinality 31, **34**
CARDINALITY 25, **34**
CIRCULAR .. 25, **39**, 41, 42, 43, 44, 94, 101, 132, 140, 148, 157, 158
CLASS . 7, 23, 24, 25, 29, **30**, 43, 45, 57, 58, 59, 68, 70, 71, 72, 80, 81, 87, 93, 101, 102, 127, 128, 133, 140, 148, 149, 156, 157, 158, 159, 160, 161, 162
ClassConst 37, **45**
ClassDef 24, 27, 28, **30**
ClassOrAssociationRef **32**, 60
ClassOrStructureDef **30**
ClassOrStructureRef **30**, 32, 45
ClassRef **30**, 32, 57, 69
ClassType 37, **45**
ClassTypeValue 86, **87**
ClippedBoundaries **89**, 90
ClippedSegment 88, **89**
CLOCKWISE 25, **42**
Comment 77, 78, **79**
ComposedUnit **55**
CondSignParamAssignment **69**
Constant **37**, 61, 70
CONSTRAINT 25, 58, 59, **60**, 66, 95, 133, 149, 150, 161
ConstraintDef 30, 34, **59**, 60, 65
CONSTRAINTS 25, **60**, 128
ConstraintsDef 28, **60**
CONTINUOUS 25, **31**, 43, 94, 132, 148
CONTRACTED .. 25, **27**, 70, 71, 72, 92, 112, 130, 132, 140, 157, 158
ControlPoints **49**
COORD . 25, **44**, 70, 75, 88, 91, 92, 101, 128, 156, 157, 158
CoordinateType 37, **44**
CoordValue 86, **88**
COUNTERCLOCKWISE 25, **42**, 140, 157, 158

D

DataSection 77, **83**
Dec **24**, 26, 42, 49, 59
DecConst **42**, 55
DEFINED 25, **60**, 61
Definitions **28**
Delentry **78**, 83
DeleteObject 83, **84**
DEPENDS 25, **28**, 29, 32, 71, 112, 128, 140
DERIVED 25, **34**, 68
DerivedUnit **55**
Digit **23**, 24

DIRECTED 25, 47, **49**, 66, 94
DOMAIN .. 25, 36, **37**, 38, 39, 40, 41, 43, 44, 45, 68, 70, 71, 92, 94, 101, 132, 147, 148, 156, 157, 158
DomainDef 27, 28, **37**
DomainRef 28, 30, 32, 34, **37**, 39, 42, 49
DrawingRule 68, **69**

E

EmbeddedLink 84, **85**
EmbeddedLinkStruct **85**
END ... 25, **27**, 28, 30, 33, 34, 43, 48, 57, 58, 59, 60, 65, 66, 68, 69, 70, 71, 72, 73, 80, 81, 93, 94, 95, 101, 102, 112, 113, 127, 128, 129, 131, 132, 133, 140, 148, 149, 150, 151, 156, 157, 158, 159, 160, 161, 162
Entries **78**
EnumAssignment **70**
EnumElement **39**
Enumeration **39**
EnumerationConst 37, **39**, 70
EnumerationType 37, **39**
EnumRange **70**
ENUMTREEVAL 25, **63**, 64, 93
EnumTreeValueType 37, **39**
ENUMVAL 25, **63**, 64, 93
EnumValue 86, **87**
EQUAL 25, **67**
EXISTENCE 25, **59**
ExistenceConstraint 58, **59**
Explanation **24**, 27, 55, 63
Expression 59, **60**, 61, 67, 69
EXTENDED ... 24, 25, 26, 28, **30**, 31, 33, 34, 36, 43, 57, 65, 67, 69, 72, 81, 93, 94, 102, 133, 148, 149, 161, 162
EXTENDS 25, 26, **28**, 30, 33, 34, 36, 37, 39, 41, 43, 48, 54, 55, 56, 57, 65, 69, 70, 72, 80, 81, 92, 93, 94, 102, 128, 130, 131, 132, 133, 140, 148, 149, 150, 157, 158, 159, 160
EXTERNAL 15, 25, **32**, 34, 36, 128

F

Factor 31, 34, 60, **61**, 63, 67, 69, 70
FINAL . 24, 25, 26, **28**, 30, 31, 34, 37, 38, 39, 40, 48, 56, 57, 65, 67, 69, 92, 94
FIRST 25, **61**, 66, 95
Float **24**
FORM 25, **50**, 89, 92
FORMAT 25, **42**, 43, 94, 132, 148
FormatDef **42**
FormationDef 65, **66**
FormattedConst 37, **42**
FormattedType 37, **42**
FormattedValue 86, **87**
FROM 25, **34**, 68
FUNCTION 25, 55, **63**, 64, 68, 93, 131, 132, 133
FunctionCall **61**
FunctionDef 27, 28, **63**

G

GlobalUniqueness **59**
GRAPHIC 25, 30, **69**, 71, 72, 112, 113, 140
GraphicDef 28, **69**
GraphicRef **69**

H

HALIGNMENT 25, **40**, 87, 92, 157
HeaderSection 77, **78**
HexDigit **23**, 24
HIDING 25, **34**, 36

I

IMPORTS 25, **27**, 28, 71, 72, 80, 81, 102, 112, 128, 132, 140, 156, 158
IN 25, 49, **59**, 70, 72
INHERITANCE 25, **42**, 43, 94
InnerBoundary **89**, 90
Inspection 61, 62, 66, **67**
INSPECTION 25, 53, 61, 66, **67**, 112
INTERLIS 7, 8, 9, **10**, 11, 12, 13, 14, 16, 17, 18, 19, 20, 25, 26, 27, 28, 30, 36, 37, 38, 43, 50, 54, 56, 57, 66, 70, 91, 92, 95, 101, 102, 112, 124, 125, 127, 130, 132, 133, 136, 138, 139, 140, 143, 147, 148, 157, 158
INTERLIS2Def **26**
IntersectionDef **49**

J

Join **66**
JOIN 25, 65, **66**, 68

L

LAST 25, **61**
Letter **23**
LINE 25, **49**, 50, 89, 92, 101
LineAttr 88, **89**, 90
LineAttrDef **49**, 53
LineForm **49**
LineFormSegment **89**
LineFormType **49**
LineFormTypeDef 27, **50**
LineType 37, **49**
Link 83, **84**
LIST 8, 25, **31**, 32, 33, 57, 58, 62, 66, 75, 86, 88, 93, 94, 95, 149, 159, 160
ListValue 86, **88**
LNBASE 25, **42**
LOCAL 25, 58, **60**
LocalUniqueness **59**, 60

M

MANDATORY .. 7, 25, **31**, 37, 48, 57, 59, 66, 70, 93, 94, 95, 101, 102, 132, 133, 140, 148, 149, 150, 157, 158, 159, 160, 161, 162
MandatoryConstraint **59**
MetaDataBasketDef 27, 28, **56**
MetaDataBasketRef **56**
METAOBJECT 25, 29, 38, 56, **57**, 70, 90, 93, 140
MetaObjectRef 42, **56**, 70
Model **78**
MODEL ... 25, 26, **27**, 30, 55, 57, 70, 71, 72, 78, 80, 81, 92, 101, 102, 112, 128, 130, 132, 140, 148, 156, 157, 158
ModelDef 26, **27**
Models **78**, 79
MTEXT 25, 37, **38**, 64, 86, 93, 96
MTextValue **86**

N

Name .. **23**, 27, 28, 30, 31, 34, 37, 39, 42, 45, 49, 50, 55, 56, 57, 58, 60, 61, 63, 65, 67, 69, 70
NAME 25, **38**, 57, 86, 90, 92, 93
NO 25, 29, **30**, 34
NOT 25, **60**, 133
NULL 25, 65, **66**
Number **24**
NUMERIC .. 25, 41, **42**, 48, 57, 64, 70, 92, 93, 94, 133, 148, 157
NumericConst 37, **42**
NumericType 37, **42**, 44
NumericValue 86, **87**

O

Object 83, **84**
OBJECT 25, **63**
ObjectOrAttributePath 60, **61**
OBJECTS 25, 43, **56**, 59, 63, 64, 93, 94, 112, 140, 156
OF 25, **27**, 31, 39, 43, 45, 46, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 71, 72, 75, 78, 79, 80, 82, 85, 86, 88, 90, 93, 94, 95, 112, 113, 128, 133, 140, 149, 156, 159, 160
OID 17, 20, 25, 28, 29, 30, 32, 34, **44**, 45, 75, 76, 83, 84, 85, 92, 100, 124, 125, 126
OIDAttributeValue 86, **90**
OidSpace **78**
OidSpaces **78**, 79
OIDType 37, **44**
ON 25, **28**, 29, 32, 42, 43, 68, 69, 71, 94, 112, 113, 128, 132, 140, 148
OR 25, 33, 34, 59, **60**, 61, 65, 66
ORDERED 25, 34, **39**, 40, 71, 92
OTHERS 25, **39**, 87
OuterBoundary **89**, 90
OVERLAPS 25, 48, **49**, 101, 127, 128

P

PARAMETER 25, 30, 56, 57, **58**, 61, 69, 70, 72, 93, 133, 140, 148, 157, 158, 161, 162
ParameterDef 30, **57**
PARENT 25, **61**, 62
PathEl **61**
PI 25, **42**, 101, 131, 148, 157, 158
PlausibilityConstraint 58, **59**
POLYLINE 25, 47, **49**, 66, 75, 86, 88, 91, 94, 101, 157, 158
PolylineValue 86, **88**, 90
PosNumber **24**, 34, 38, 42, 44, 61
Predicate **60**
Projection **66**
PROJECTION 25, 65, **66**
Properties **24**, 28, 29, 30, 31, 32, 34, 36, 37, 56, 57, 65, 67, 68, 69

R

REFERENCE 7, 25, **32**, 90, 159, 160
ReferenceAttr **32**
ReferenceAttribute 88, **90**
RefSys **42**, 57
REFSYSTEM ... 25, 26, **27**, 43, 55, 56, 57, 93, 94, 132, 148, 156
Relation **60**
RenamedViewableRef 34, **66**, 67
REQUIRED 25, 49, **59**
RestrictedClassOrAssRef **32**, 34, 61, 63
RestrictedClassOrStructureRef **32**
RestrictedStructureRef **31**, 32
RESTRICTION 25, **32**, 33, 34, 45, 64, 93, 159
Role 84, **85**
RoleDef **34**
ROTATION 25, **44**, 71, 101, 156, 158
RotationDef **44**
RunTimeParameterDef 27, **58**

S

Scaling **24**
SegmentSequence 88, **89**
Selection 65, **67**, 69
SET 25, 58, 59, **60**
SetConstraint 59, **60**
SetOrderPos 83, **86**
SIGN 25, **56**, 70, 71, 72, 112, 140
SignParamAssignment 69, **70**
StartSegment **88**, 89
STRAIGHTS 25, 47, **49**, 92, 101, 127, 128, 157, 158
StraightSegment **88**, 89
String **23**, 27, 38, 42
STRUCTURE 25, **30**, 43, 45, 48, 57, 59, 63, 64, 66, 75, 86, 87, 88, 93, 94, 95, 101, 132, 148, 159, 160
StructureDef 27, 28, **30**
StructureRef **30**, 32, 42
StructureValue 86, **88**, 89
SUBDIVISION 25, **31**, 43, 94, 132, 148
SURFACE 25, **49**, 53, 54, 58, 59, 64, 75, 86, 89, 90, 91, 93, 101, 157, 158
SurfaceValue 86, **89**, 90
SYMBOLOGY 25, 26, **27**, 55, 70, 72, 140, 157, 158

T

Tagentry **78**, 82
Term **60**
Term1 ..... **60**
Term2 ..... **60**
TEXT 25, 36, 37, **38**, 45, 58, 64, 70, 71, 81, 86, 92, 93, 101, 128, 148, 149, 151, 156, 157, 158, 159, 160
TextConst **37**, 38
TextType 37, **38**, 44
TextValue **86**
THATAREA 25, **61**, 62, 66
THIS 25, **61**, 62
THISAREA 25, **61**, 62, 66
TO 7, 25, **32**, 33, 90, 159, 160
TOPIC 25, **28**, 30, 65, 70, 71, 72, 75, 80, 81, 83, 93, 101, 102, 112, 127, 128, 133, 140, 148, 156, 157, 158
TopicDef 27, **28**
TopicRef **28**, 56
Transfer 74, 75, **77**, 78, 79, 83, 86, 88, 90
TRANSIENT 25, 31, **65**, 67, 75
TRANSLATION 25, **27**, 78, 79, 80, 82
Type 32, **37**
TYPE 25, 26, **27**, 92, 130

U

UNDEFINED 25, **37**, 59, 61, 64, 67
Union 66, **67**
UNION 25, 65, **67**
UNIQUE 7, 25, 57, 58, **59**, 93, 127, 128
UniqueEl **59**, 60, 67
UniquenessConstraint 58, **59**
UNIT 25, 41, 43, 54, **55**, 92, 94, 101, 130, 148, 157, 158
UnitDef 27, 28, **55**
UnitRef 42, **55**
UNQUALIFIED 25, **27**, 28
URI 25, 27, **38**, 86, 92, 124

V

Valentry **78**, 82
VALIGNMENT 25, **40**, 87, 92, 157
VERSION 25, **27**, 80, 81, 92, 101, 102, 112, 128, 130, 132, 140, 148, 156, 157, 158
VERTEX 25, **49**, 101, 127, 128, 157, 158
VIEW 25, 28, 30, **65**, 68, 75, 83, 112
ViewableRef 45, 59, 61, **67**, 69
ViewAttributes 65, **67**
ViewDef 28, **65**
ViewRef 63, **65**

W

WHEN 25, **70**, 72
WHERE 25, 59, 60, **67**, 68, 69, 72, 112, 113

WITH 25, **49**, 101, 127, 128, 157, 158
WITHOUT 25, 48, **49**, 101, 127, 128

X

XML **45**
XML-Any **76**, 87
XML-base64Binary **76**, 87
XML-ID **76**, 83, 84, 85, 86, 90
XML-NcName **76**
XML-NormalizedString **76**, 77, 86, 87
XML-String **76**, 77, 79, 86
XML-Value **76**, 78, 79, 83, 84
XML-ValueDelimiter **76**

El índice ofrece las palabras reservadas en mayúsculas (véase el capítulo 2.2.7 Símbolos especiales y palabras reservadas), las definiciones de sintaxis del lenguaje de descripción con texto normal (véase el capítulo 2 Descripción del lenguaje) y las definiciones de sintaxis de la transferencia (en cursiva véase el capítulo 3 de transferencia secuencial). El número de páginas impresas en negrita indica el pasaje en el manual de referencia que ofrece la definición más completa de un término.