Cuestión 2 del Orden del Día: **Implantación de sistemas para el intercambio de la Información Aeronáutica y Datos Aeronáuticos (AIXM)**

Proyecto G2 del GREPECAS

(Presentada por la Secretaría)

RESUMEN

Esta nota de estudio presenta a la Reunión el estado de implantación actual del Proyecto G2 del GREPECAS y la nueva propuesta de revisión del Proyecto AIXM.

REFERENCIAS

- Anexo 15 al Convenio sobre Aviación Civil Internacional
- Reuniones Multilaterales SAM/AIM
- Informe de la Cuarta Reunión del Comité de Revisión de Programas y Proyectos del GREPECAS (CRPP/4)
- Manuales de AIXM de Eurocontrol

Objetivos estratégicos de la OACI	*A - Seguridad operacional* *B - Capacidad y eficiencia de la navegación aérea* *E - Protección del medio ambiente*

1. Antecedentes

1.1 La implantación del Proyecto G2 del GREPECAS ha sido sostenida por la Secretaría, ya que el mismo no tenía un Coordinador de Proyecto. A partir de la Reunión SAM/AIM/7, la Ing. Karina Calderón, del Perú, está desarrollando las actividades programadas como Coordinadora del Proyecto G2.

1.2 Durante todas las reuniones SAM/AIM se ha resaltado la importancia que tiene la Certificación de Calidad para la garantía de la calidad de los datos a ser transferidos, así como la importancia de la capacitación en el AIM para optimizar los resultados.

1.3 En la Reunión CRPP/4 se ha analizado la prosecución del Proyecto G2.

2. Análisis

2.1 Durante la Reunión del CRPP/4 se ha re-planteado el proyecto atendiendo a los puntos que no han sido culminados. Los términos de trabajo del Proyecto G2 se describen en el **Apéndice A** de esta nota de estudio.

2.2 La Reunión recordará que en la Reunión SAM/AIM/8, como parte de los productos AIXM a ser analizados, la Coordinación del Proyecto había seleccionado cuatro documentos para ser

tenidos en cuenta en esta fase de desarrollo. En ese sentido, se han traducido cuatros documentos de EUROCONTROL, los cuáles se mencionan a continuación:

2.3 Un documento de EUROCONTROL sobre la necesidad de un modelo de temporalidad (ver **Apéndice B** de esta nota de estudio).

2.4 Otro de los documentos seleccionados se denomina *Mapeo del UML al Esquema XML* y se basa en el Modelo Conceptual AIXM que es mantenido como un modelo de clase UML, el cual se refleja en el **Apéndice C** de esta nota de estudio.

2.5 El tercer documento traducido tiene como finalidad describir de qué manera se puede extender el modelo UML AIXM. El **Apéndice D** a esta nota de estudio presenta el documento *Generación del Esquema de Aplicación AIXM*.

2.6 El cuarto documento que se muestra en el **Apéndice E** a esta nota de estudio describe la *Identificación y Referencia de Componentes*. Conforme se desarrollan servicios para difundir información en el AIXM 5.1 a los consumidores, es esencial tener la capacidad de gestionar los vínculos entre los componentes aeronáuticos. Esto comprende los conceptos de identificación de componentes y referencia de componentes.

2.7 La SAM/AIM/8 incluyó estos documentos dentro del informe final para socializar con los expertos AIM de la Región para que efectúen el análisis correspondiente para presentar comentarios y propuestas para la elaboración del documento final del material guía para la gestión del concepto AIXM para la Región SAM.

2.8 Atendiendo a las actividades programadas y al objetivo del Proyecto G2, es necesario consultar a los Estados sobre los Planes de Acción de Implantación de los sistemas de intercambio de datos. La Secretaría ha preparado una plantilla para consultar a los Estados sobre la situación actual de implantación del AIXM, la cual se presenta como **Apéndice F** a esta nota de estudio.

2.9 La Reunión pudiera observar que, atendiendo a los entregables del proyecto, es necesario programar las fechas de las pruebas de intercambios de datos a través del AIXM. La Coordinación del Proyecto pudiera trabajar con el Coordinador OACI y los Estados para poder establecer el protocolo correspondiente atendiendo a que estas pruebas involucran a los sistemas de comunicaciones de los Estados.

3. Conclusión

3.1 De acuerdo a lo anterior, la Reunión debiera recoger los comentarios de los Estados sobre los documentos contenidos en los **Apéndices B** a **E** de esta nota de estudio, a fin de que la Coordinación del Proyecto pueda adecuar el Proyecto de Implantación del AIXM.

3.2 Asimismo, queda en evidencia la necesidad de contar con personal profesional que asesore al AIM sobre el uso de estos formatos y se capacite a los operadores para incluir la información a intercambiar.

3.3 La Reunión debe acordar con la Coordinación del Proyecto, las fechas en que se realizarían las pruebas de intercambios. Para ello se deberían tener en cuenta los Estados que se encuentran en condiciones de realizarlas y los datos a ser intercambiados.

4. **Acción sugerida**

4.1 Se invita a la Reunión a:

a) analizar la información contenida en el Apéndice A, y actualizar el Proyecto G2 en lo que sea pertinente;

b) proveer los comentarios a la Coordinación del proyecto sobre los documentos que se muestran en los Apéndices B, C, D y E a fin de adoptar esos documentos como guías para la implantación del AIXM en la Región SAM; y

c) proveer el plan de acción de implantación de los sistemas de intercambios de datos de aquellos Estados que aún no lo han hecho.

- - - - - -

APÉNDICE A

Región SAM	**DESCRIPCION DEL PROYECTO (DP)**	DP N° G2	
Programa	**Título del Proyecto**	**Fecha inicio**	**Fecha término**
AIM (Coordinador OACI del Programa: Jorge Armoa Cañete)	G2: Implantación de sistemas de intercambio de información aeronáutica (SAM) Coordinadora del Proyecto: Ing. Karina Calderón Expertos contribuyentes al proyecto: SAM/AIM/IG	01/03/12	01/12/17
Objetivo	Elaborar Plan de Acción que deben implementar los Estados para aplicar el modelo de intercambio de información/datos aeronáuticos.		
Alcance	El alcance del proyecto contempla la evaluación e identificación de los niveles de automatización asociados a la integración del modelo de intercambio de información y datos aeronáuticos en la Región por medio de encuestas, la identificación de los proveedores de bases de datos y el seguimiento sobre el avance de los SARPS en esta materia.		
Métricas	Números de Estados con Plan de Acción implantado para sistemas de intercambio de datos.		
Metas	Completar toda la documentación necesaria para los Estados antes del 31/12/16.		

Estrategia	La ejecución de las actividades del Proyecto será coordinada a través de las comunicaciones entre miembros del proyecto, el Coordinador del Proyecto y el Coordinador del Programa principalmente a través de teleconferencias (aplicación GoToMeeting). Se planifican seminarios/reuniones según las actividades del programa de trabajo. El Coordinador del Proyecto coordinará con el Coordinador del Programa la incorporación de expertos adicionales si lo ameritan las tareas y trabajos a realizarse. Se realizarán las coordinaciones CAR y SAM. Los resultados de los trabajos realizados, serán sometidos a consideración y revisión por los expertos de los Estados en forma de documento final de consolidación para su análisis, revisión, aprobación y presentación al CRPP del GREPECAS por el Coordinador del Programa.
Justificación	Integrar la información aeronáutica que permita la inter-operación de sistemas ATM manteniendo la seguridad operacional, aplicando los modelos de intercambio de información.
Proyectos relacionados	Se relaciona con el Proyecto G3 "*Implantación del sistema de gestión de calidad en las dependencias AIM en los Estados de la Región SAM*".

Entregables del Proyecto	**Relación con el Plan Regional basado en performance (PFF)**	**Responsable**	**Estado de Implantación***	**Fecha entrega**	**Comentarios**
Relevamiento de suministro de la IAIP mediante el uso de una tabla.	D-ATM	Coordinador OACI		16/03/12	Completada en fecha durante la Reunión SAM/AIM.
Distribución a los Estados relevamiento IAIP	D-ATM	Coordinador OACI		16/03/12	Completada en fecha durante la Reunión SAM/AIM.
Recolección y actualización	D-ATM	Coordinador OACI		16/03/12	Completada en fecha durante la Reunión SAM/AIM.
Recolección de experiencias en los Estados de la Región SAM AIP electrónico	D-ATM	Coordinador OACI		16/03/12	Completada en fecha durante la Reunión SAM/AIM.

Desarrollar Plan de Acción AIXM	D-ATM	Coordinador OACI		24/04/15	Completada en fecha.
Recopilación de la documentación AIXM	D-ATM	Coordinador OACI		22/05/15	Completada en fecha.
Traducción de la documentación AIXM	D-ATM	OACI		10/07/15	Completada en fecha.
Revisión de la documentación AIXM	D-ATM	Coordinador OACI		21/08/15	Completada en fecha
Validar la documentación	D-ATM	Coordinador OACI		30/11/16	
Elaboración documento describiendo pasos para las pruebas AIXM	D-ATM	Coordinador OACI		28/02/17	
Realización de las pruebas AIXM	D-ATM	Coordinador OACI		30/03/17	
Informe de resultado de las pruebas Trasmisión y recepción de datos	D-ATM	Coordinador OACI		19/05/17	
Seminario AIXM	D-ATM	Coordinador OACI		02/10/15	Completada a la fecha
Elaboración del material guía para la gestión del concepto AIXM	D-ATM	Coordinador OACI		27/12/16	
Recursos necesarios	Designación de expertos en la ejecución de algunos de los entregables. Mayor compromiso de los Estados en apoyar a los Coordinadores y expertos que están trabajando.				

**Gris* *Tarea no iniciada*
Verde *Actividad en progreso de acuerdo con el cronograma*
Amarillo *Actividad iniciada con cierto retardo pero estaría llegando a tiempo en su implantación*
Rojo *No se ha logrado la implantación de la actividad en el lapso de tiempo estimado se requiere adoptar medidas mitigatorias*

APÉNDICE B

AIXM 5

MODELO DE TEMPORALIDAD

AIXM 5

Modelo de Temporalidad

Modelo de Intercambio de Información Aeronáutica (AIXM)

Para cualquier consulta, sírvase ponerse en contacto con:

Navin VEMBAR - Navin.Vembar@faa.gov

Eduard POROSNICU - eduard.porosnicu@eurocontrol.int

Parte	Versión	Fecha de emisión de la versión	Autor de la Parte	Razón del cambio
0.1	Borrador	24 abr 2007	Equipo de Diseño	Borrador inicial
0.2	Borrador	04 jun 2007	Equipo de Diseño	Actualizado luego de las discusiones en St. Louis y Frankfurt
0.3	Borrador	10 jun 2007	Equipo de Diseño	Actualizado luego de los comentarios formulados en la reunión AIXM FG/8 y por Edna.
0.4	Propuesto	15 jul 2007	Equipo de Diseño	Se eliminó del modelo las Particiones de Tiempo (Time Slices) “estáticas”. Se reorganizó la presentación de los distintos tipos de Particiones de Tiempo
0.5	Propuesto	12 nov 2007	Equipo de Diseño	Mejorado para la primera versión pública
0.6	Propuesto	01 feb 2010	Equipo de Diseño	Describe el concepto PropertiesWithSchedule introducido en AIXM 5.1 (ver el capítulo 2.8) Incluye los diagramas UML del modelo AIXM 5.1
1.0	Publicado	15 sep 2010	Equipo de Diseño	Versión publicada, para ser utilizada como línea de base para las implementaciones del AIXM 5.1.

Indice

1. Necesidad de un modelo de temporalidad

El tiempo es un aspecto esencial en el mundo de la información aeronáutica, donde, normalmente, los cambios son notificados con mucha anticipación a su fecha de entrada en vigencia. Generalmente, se pide que los sistemas de información aeronáutica almacenen y brinden tanto la información actual como los cambios futuros. La información que ha caducado debe ser archivada para fines de investigación legal.

Para fines operacionales[1], generalmente se hace una distinción entre:

- ***los cambios permanentes*** (cuyo efecto durará hasta el próximo cambio permanente o hasta el fin de la vida útil del componente) y
- ***la situación temporal*** (cambios de una duración limitada, que se considera se superponen al estado permanente del componente).

Un cambio temporal incluye los conceptos de superposición y reversión. El cambio temporal se superpone al estado permanente del componente. Cuando el cambio temporal llega a su fin, los cambios temporales ya no son aplicables, y se regresa al estado permanente del componente.

Cabe notar que, desde el punto de vista operacional, el "estado temporal" también incluye el concepto de "componentes temporales". No obstante, desde el punto de vista del AIXM, los componentes temporales en nada difieren de los componentes normales. El componente es creado y retirado, excepto que su duración es más corta de lo normal.

A fin de satisfacer los requisitos temporales de los sistemas de información aeronáutica, el AIXM debe incluir un minucioso modelo de temporalidad que permita una representación precisa de los estados y eventos de los componentes aeronáuticos. En particular, esto deberá permitir el desarrollo e implantación de los **NOTAM digitales**. Al referirnos a NOTAM digitales, nos referimos a la sustitución del texto libre contenido en un mensaje NOTAM por hechos estructurados que permiten el procesamiento automatizado de la información.

Un modelo temporal general debería ser aplicado uniformemente a todos los tipos de componentes aeronáuticos, y el concepto de temporalidad debería abstraerse de la tarea de modelado de las propiedades del objeto. A nivel conceptual, el modelo debería describir la evolución temporal de los componentes, ya que éstos ocurren en el mundo real. Esto debería hacerse en cumplimiento con las siguientes reglas:

- Integralidad – todos los estados temporales deberían ser representables;
- Minimalismo – uso de una cantidad mínima de elementos;
- Consistencia – no re-utilización de elementos con distinto significado;
- Descontextualización - el significado de los elementos (atómicos) es independiente del contexto; no existe dependencia funcional de los elementos (atómicos) a nivel de codificación de los datos;

La especificación de intercambio de datos deberá permitir la operación del modelo conceptual. Asimismo, se puede introducir elementos de conveniencia ("vistas") en la especificación de intercambio de datos a fin de facilitar las operaciones. Esto significa que la especificación de intercambio de datos puede desviarse de la regla del "minimalismo".

[1] Por ejemplo, los sistemas que producen documentación aeronáutica impresa (AIP, cartas) tienden a ignorar la información de condición temporal; sólo los datos estáticos aparecen representados en dichos productos impresos.

2. Construcción del Modelo de Temporalidad

A fin de explicar el Modelo de Temporalidad AIXM, este capítulo adopta un enfoque por etapas, donde los elementos que conforman este modelo son agregados gradualmente, a fin de satisfacer las necesidades operacionales.

2.1 (paso 1) Propiedades que varían con el tiempo

Hay dos niveles a los que los componentes aeronáuticos se ven afectados por el tiempo:

- Cada componente tiene un inicio y un final de vida;
- Las propiedades de un componente pueden cambiar durante la vida del mismo; esto incluye la posibilidad de alta de definición de una propiedad por un período de tiempo.

El inicio y el final de la vida también pueden ser considerados como propiedades del componente (atributos). Esto resulta en la siguiente lista de propiedades de alto nivel para cualquier componente del AIXM:

- un identificador único universal;
- el inicio de vida (fecha y hora);
- el final de vida (fecha y hora);
- atributos y asociaciones que califican, cuantifican o relacionan a dicho componente de alguna manera.

Se considera que cualquier propiedad del componente puede variar con el tiempo, excepto el identificador único universal. Este es una premisa clave del Modelo de Temporalidad del AIXM.

El primer paso en la construcción del modelo de temporalidad del AIXM está representado por el siguiente diagrama, que muestra los valores de las propiedades de un componente (P1, P2, … P5) en el transcurso del tiempo.

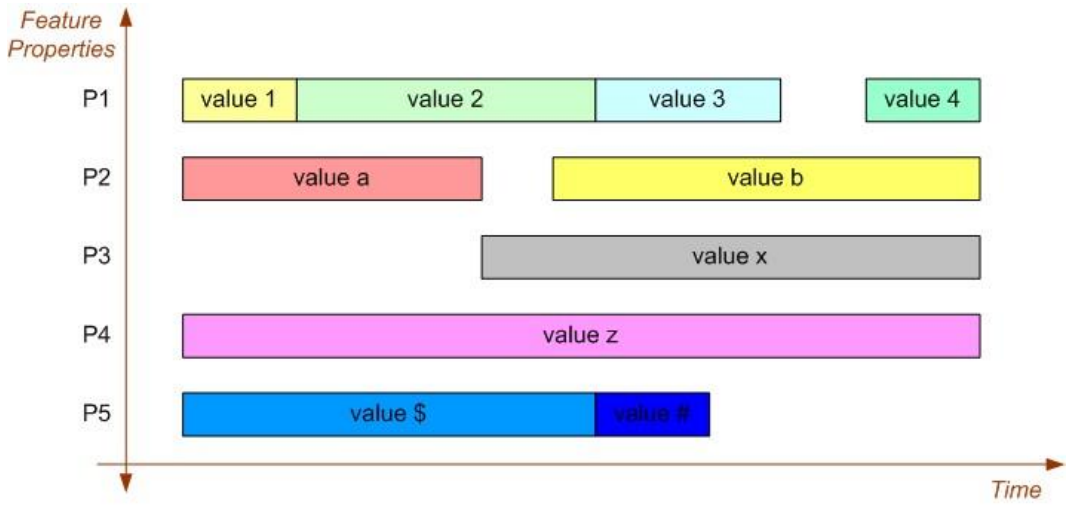


Figura 1 – Propiedades que varían con el tiempo

Discusión: ***¿Pueden variar en el tiempo las propiedades de inicio de vida y final de vida de un componente?***

A primera vista, probablemente no. Un componente es creado en un momento en el tiempo y dejará de existir en otro momento en el tiempo. Pero esto es cierto únicamente cuando se considera la historia ya conocida de un componente. Al intercambiar datos sobre el futuro, podrían darse situaciones en las cuales el inicio/final de la vida están planificados para ocurrir en una determinada fecha/hora, y esta fecha podría cambiar.

Por lo tanto, debemos incluir el inicio/final de vida de un componente en la lista de propiedades que varían con el tiempo.

2.2 (paso 2) El modelo de Fracciones de Tiempo

El modelo de temporalidad adoptado por el AIXM describe los eventos y estados de los componentes. Un evento es un cambio en una o más propiedades del componente. Un estado es una propiedad del componente cuya validez es fijada para un período de tiempo. Un evento ocurre en la transición entre estados. En el siguiente diagrama, los eventos están ubicados en los cortes verticales, mientras que los estados están representados como el conjunto de propiedades del componente entre eventos.

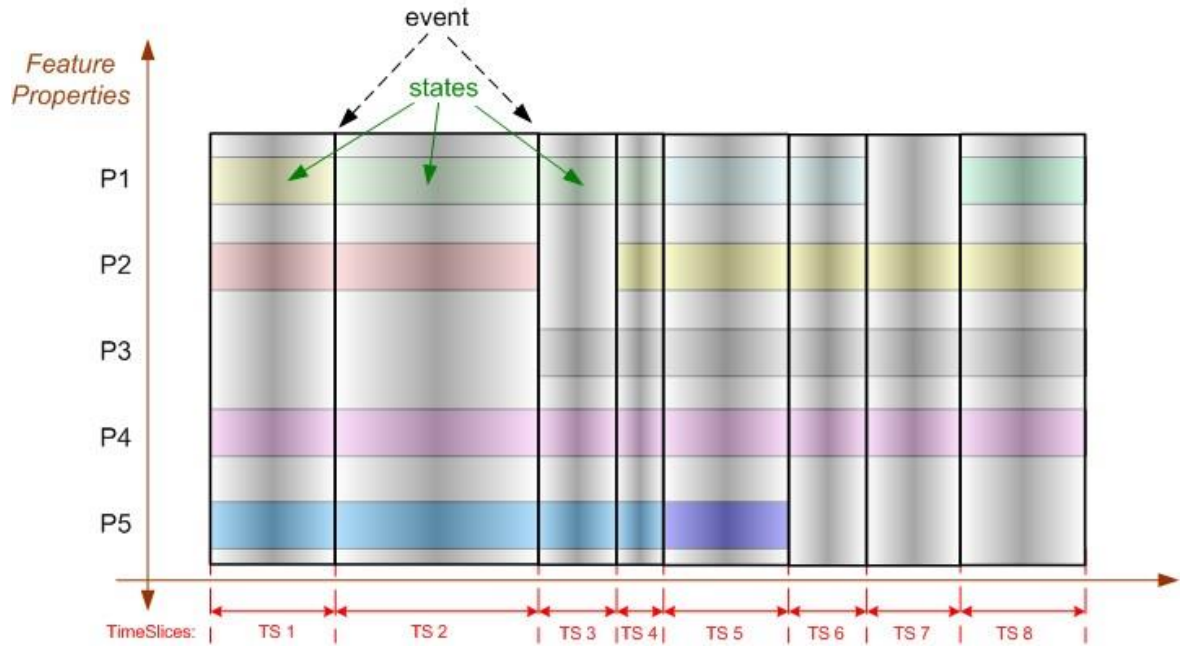


Figura 2 – Eventos y estados

A fin de describir las propiedades del componente durante los estados y los eventos, las propiedades de cada componente que varían con el tiempo están encapsuladas en un contenedor llamado "Fracción de Tiempo". La historia del componente se describe a través de Fracciones de Tiempo de "estado", cada una de los cuales contiene los valores de las propiedades que varían con el tiempo entre dos cambios consecutivos (eventos). Cada Fracción de Tiempo tiene, como máximo, un valor para cada propiedad y un período de validez especificado. En un diagrama UML, el concepto básico de Fracciones de Tiempo aparece representado a continuación:

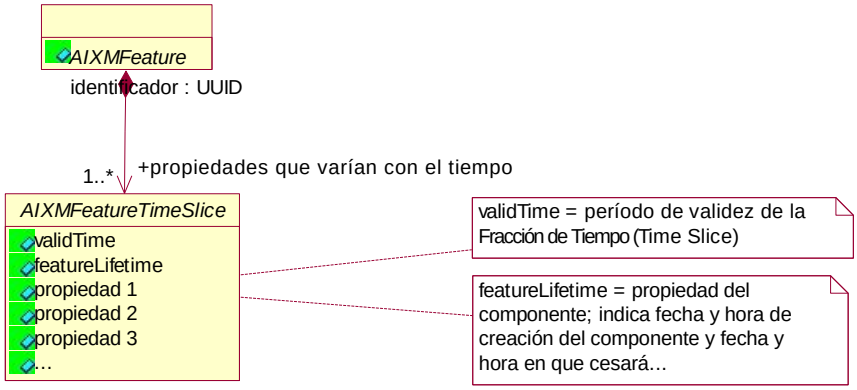


Figura 3 – AIXMFeatureTimeSlice

---

[2] El modelo de Fracciones de Tiempo del AIXM se basa en el concepto de fracciones de tiempo de la ISO 19136 (GML).

Discusión: ***¿Por qué no un período de validez para cada propiedad?***

En vez de agrupar los valores de las propiedades en Fracciones de Tiempo, otro enfoque podría ser un modelo de tiempo en el que cada propiedad tiene su propio período de validez.

El primer argumento en contra de este enfoque es que, en general, las propiedades de un componente no cambian de manera mutuamente independiente. Existen restricciones operacionales que vinculan los valores de algunas propiedades con los valores de otras propiedades. Por lo tanto, varias propiedades tendrían que ser agrupadas de todas maneras, con un período de validez común.

La segunda razón es que los cambios en el mundo aeronáutico están regulados por el ciclo AIRAC. Consecuentemente, los cambios operacionales significativos ocurren en fechas predefinidas a fin de garantizar la predictibilidad del ambiente aeronáutico y para dar tiempo a que los usuarios se ajusten a los cambios. En general, los valores de las propiedades de los componentes aeronáuticos son estables entre fechas AIRAC. Por lo tanto, el agrupamiento de las propiedades en Fracciones de Tiempo, con un mismo período de validez, constituye un modelo de tiempo simplificado, que se ajusta bien a los requisitos operacionales.

2.3 (paso 3) Eventos temporales – NOTAM digital

Los componentes aeronáuticos se pueden ver afectados por eventos temporales, tales como una ayuda para la navegación fuera de servicio, una pista cerrada, la activación de una zona restringida, etc. Todos estos eventos generan cambios temporales en los valores de una o más de las propiedades de los componentes. Al final del evento temporal, los valores de estas propiedades regresan a sus valores estáticos.

A fin de modelar los eventos temporales, tenemos que especializar el modelo de temporalidad básico definido en el paso 2, distinguiendo entre dos tipos de Fracciones de Tiempo:

- ***Línea de base*** = un tipo de Fracción de Tiempo que describe el estado del componente (el conjunto de todas las propiedades del componente) como resultado de un cambio permanente.

- ***Temporal*** = un tipo de Fracción de Tiempo que describe la superposición transitoria de un estado de un componente durante un evento temporal.

Desde el punto de vista de la "carga útil", existe una diferencia esencial entre las Fracciones de Tiempo de Línea de Base y las de carácter Temporal.
:

  - Una Fracción de Tiempo de Línea de Base incluye los valores de todas las propiedades del componente que varían con el tiempo y que han sido definidas para el período de validez de la Fracción de Tiempo; por ejemplo, en el siguiente diagrama, TS2 incluirá los valores de p1, p2, p4 y p5;
  - Una Fracción de Tiempo Temporal incluye únicamente los valores de las propiedades sujetas a un cambio temporal; por ejemplo, en el siguiente diagrama, TS "temp" incluirá únicamente p4="valor w". Por este motivo, las Fracciones de Tiempo temporales se denominan Fracciones de Tiempo con un "Delta Temporal".

  <u>*Nota*</u>: un cambio temporal también podría presentarse cuando una propiedad de un componente se vuelve temporalmente indefinida (sin valor). Para ello, las propiedades de los componentes también pueden recibir un valor 'nulo'.

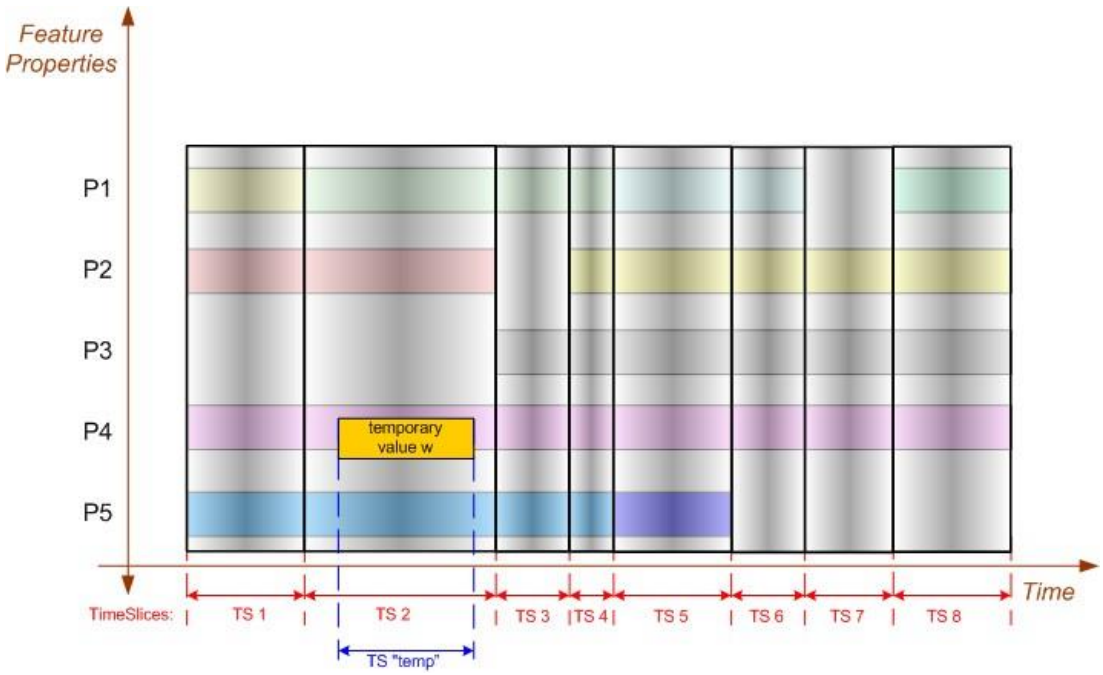


Figura 4 – Fracciones de Tiempo de LINEA DE BASE y TEMPDELTA

Una razón por la cual las Fracciones de Tiempo temporales contienen únicamente las propiedades modificadas es para evitar confusiones que podrían resultar de la superposición de los eventos temporales. Cuando varios delta temporales se traslapan en el tiempo, se necesitaría contar con reglas complicadas para decidir qué valores deberán tener las propiedades no afectadas. ¿Se debería incluir los valores de la Fracción de Tiempo de Línea de Base? ¿O se debería considerar los otros cambios temporales? Por lo tanto, el modelo resulta más claro si las Fracciones de Tiempo del Delta Temporal sólo incluyen las propiedades afectadas.

Con respecto al modelo UML, como las Fracciones de Tiempo del Delta Temporal tienen que distinguirse de las de línea de base, se requiere un atributo adicional en la clase AIXMFeatureTimeSlice. A esto se denomina "interpretación" e indica el tipo de Fracción de Tiempo – LINEA DE BASE o DELTATEMP, tal como se muestra en la siguiente figura.

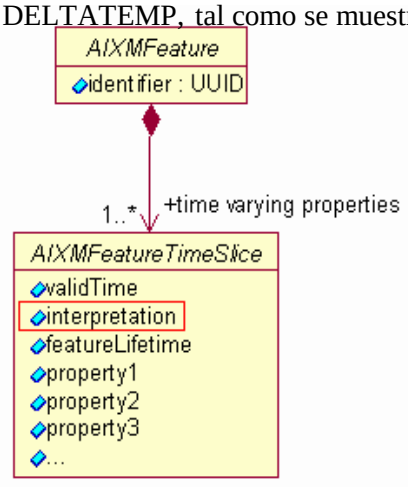


Figura 5 – FeatureTimeSlice con propiedad de "interpretación"

El beneficio esencial de las Fracciones de Tiempo DELTATEMP es que permiten la codificación del "NOTAM digital". Una Fracción de Tiempo DELTATEMP contendrá los valores de todas las propiedades del componente que, por un período de tiempo limitado, se superponen a los valores de línea de base.

Discusión: ***¿Por qué no considerar el cambio temporal como una secuencia de dos cambios permanentes?***

Si se utiliza un modelo de Fracción de Tiempo sólo con líneas de base, se tendría que dividir el intervalo TS2 en 3 fracciones de tiempo, por ejemplo TS2a, TS2b y TS2c. Según este enfoque, se modelaría la situación temporal como una secuencia de dos cambios permanentes. La desventaja de esta solución es que se perdería la información acerca de la naturaleza temporal del valor "w". Existen aplicaciones aeronáuticas, como la cartografía y la producción de las AIP, que normalmente no toman en cuenta los cambios temporales. Dichas aplicaciones necesitan saber si un valor es temporal o parte de la línea de base.

Asimismo, los eventos temporales, tales como la activación de una zona restringida, tienen vida propia: primero, se solicita la activación; luego, se planifica quizás por un intervalo de tiempo distinto al solicitado; luego se activa quizás por un período más corto al planificado, etc. A fin de modelar correctamente la vida de los eventos temporales, éstos tienen que ser modelados como tales y no escondidos detrás de cambios permanentes ficticios.

2.4 (paso 4) Situación actual – Fracciones de Tiempo SNAPSHOT (FOTO INSTANTANEA)

El modelo de temporalidad descrito hasta ahora cumple con las reglas de integridad, minimalismo, consistencia y descontextualización mencionadas al final de la sección 1. Utilizando las fracciones de tiempo BASELINE y TEMPDELTA, es posible describir las propiedades de los componentes aeronáuticos que varían con el tiempo, abarcando tanto los estados permanentes como los eventos temporales.

No obstante, el modelo es ligeramente inconveniente para una implantación en la vida real, ya que no ofrece la posibilidad de comunicar la situación actual de un componente, que resulta de la fusión de los datos de línea base con cualesquiera datos temporales válidos en ese momento. Para fines de conveniencia, necesitamos incluir en el modelo un tipo adicional de Fracción de Tiempo. Este se denominará "SNAPSHOT (FOTO INSTANTANEA)" y reflejará el resultado de la fusión de la información de LINEA DE BASE válida con todo el TEMPDELTA superpuesto, vigente en ese momento. Típicamente, una Fracción de Tiempo SNAPSHOT tendrá un Instante de Tiempo como Tiempoválido (validTime).

- SNAPSHOT = Un tipo de Fracción de Tiempo que describe el estado de un componente en un instante de tiempo, como resultado de la combinación de la Fracción de Tiempo de LINEA DE BASE válida en ese instante con todas las Fracciones de Tiempo TEMPDELTA vigentes en ese instante.

Nótese que, para un SNAPSHOT, no se deberá poblar las propiedades correctionNumber (Número de corrección) y sequenceNumber (Número de secuencia).

2.5 (paso 5) Intercambio de datos – Necesidad de Fracciones de Tiempo PERMDELTA

Otro tipo de Fracción de Tiempo que será introducido para fines de conveniencia sirve de apoyo a los sistemas que necesitan notificar a los clientes acerca de la actualización de los datos. Hay dos tipos de aplicaciones:

1. Sistemas "pull" - brindan una interfaz a través de la cual un cliente puede solicitar la información aeronáutica y extraer los resultados de la consulta;
2. Sistemas "push" – generan y transmiten al cliente las notificaciones sobre los cambios a la información aeronáutica.

Para los sistemas "push", es difícil utilizar únicamente estos tres tipos de Fracciones de Tiempo para comunicar (generar y transmitir) información acerca del futuro. Por ejemplo, ¿cómo comunicar información acerca del final (desmantelamiento) de un componente?

El empleo de las Fracciones de Tiempo de LINEA DE BASE para este fin requeriría que se comunique una 'actualización' de, por lo menos, la última Fracción de Tiempo enviada, con un valor actualizado de la propiedad 'endOfLife' (final de vida) (codificada como featureLifetime/endPosition). Esto también requeriría reglas de interpretación, tales como "si el endOfLife es igual al fin de la validez de la Fracción de Tiempo, entonces esto significa que el componente ha sido permanentemente retirado". La postergación de un retiro, lo cual es operacionalmente posible, requeriría una segunda actualización del endOfLife, lo cual podría ser difícil de interpretar.

Una solución más conveniente es incluir en el modelo de temporalidad una Fracción de Tiempo que represente los eventos de cambio permanente. A esto se le denominará Delta Permanente (PERMDELTA).

- PERMDELTA = Un tipo de Fracción de Tiempo que describe la diferencia en el estado de un componente como resultado de un cambio permanente.

El ***final de vida puede ahora ser comunicado con una Fracción de Tiempo PERMDELTA en la que featureLifetime/endPosition recibe un valor***. Simétricamente, el inicio de vida puede también ser comunicado con una Fracción de Tiempo PERMDELTA, donde la propiedad featureLifetime/startPosition y las otras prioridades del componente reciben sus valores iniciales. Modelados como eventos formales, el inicio y el final de vida pueden, en forma relativamente sencilla, ser postergados o adelantados (esto requiere un mecanismo para la actualización de una Fracción de Tiempo 'event' (evento), lo cual se analizará más adelante en este documento).

Una segunda ventaja de las Fracciones de Tiempo PERMDELTA es que los sistemas del cliente ya no necesitan comparar la Fracción de Tiempo de LINEA DE BASE previamente recibida con la nueva a fin de identificar los atributos que han cambiado. Este proceso puede tomar tiempo e, inclusive, ser propenso a errores. El originador de los datos conoce mejor la lista de propiedades cambiadas, y la Fracción de Tiempo PERMDELTA brinda la posibilidad de comunicar esta información a los clientes interesados. Esto facilita la implantación de sistemas en lo que no interesa los cambios a ciertas propiedades de los componentes. Por ejemplo, en las aplicaciones cartográficas, un PERMDELTA que afecta las propiedades que no aparecen en la carta será fácilmente ignorado.

Desde el punto de vista conceptual, una Fracción de Tiempo PERMDELTA ocurre en el límite entre cualesquiera dos Fracciones de Tiempo de LINEA DE BASE consecutivas, y contiene valores únicamente para las propiedades cambiadas.

Todos estos tipos de Fracciones de Tiempo aparecen descritos en la Figura 6.

Conceptualmente, existe una dependencia directa entre las Fracciones de Tiempo PERMDELTA y LINEA DE BASE. No obstante, esto no significa que la Fracción de Tiempo de LINEA DE BASE tenga que ser efectivamente instanciada después de cada PERMDELTA. En una implantación, es posible, por ejemplo, "acumular" Fracciones de Tiempo PERMDELTA. La instanciación de una nueva LINEA DE BASE podría ocurrir, por ejemplo, después de cada tercer PERMDELTA que afecte a un componente.

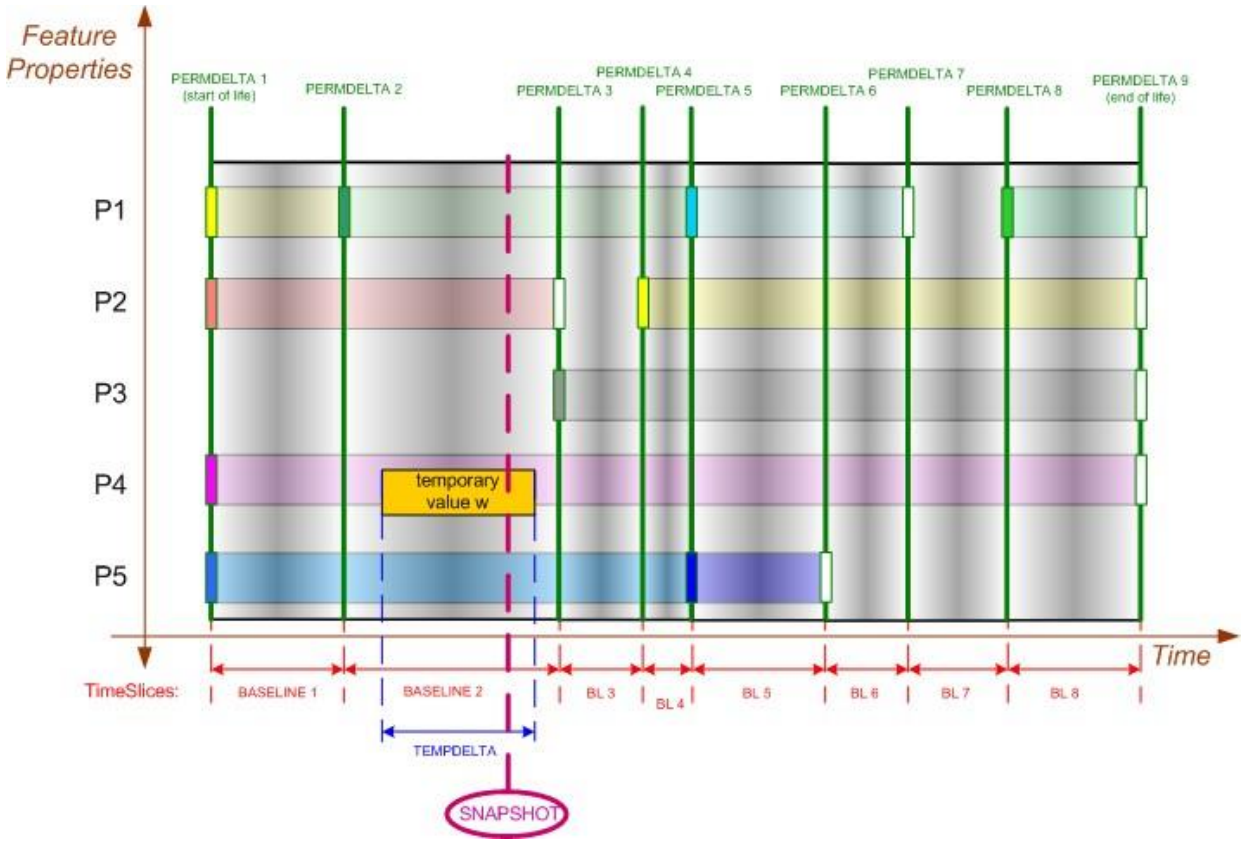


Figura 6 – Cuatro tipos de Fracciones de Tiempo

2.6 (paso 6) Intercambio de datos – correcciones

En el mundo aeronáutico, necesitamos comunicar información acerca de eventos que están planificados para llevarse a cabo en el futuro. Inevitablemente, la realidad podría ser diferente a lo inicialmente planificado, y podría resultar necesario actualizar la información ya comunicada.

Como en el caso del AIXM, las propiedades de un componente están encapsuladas en Fracciones de Tiempo. Esto significa que necesitamos un mecanismo para actualizar/corregir una Fracción de Tiempo de componente previamente comunicada. Primero, se requiere una clave para identificar la Fracción de Tiempo en cuestión. Para ello, ***se introduce en el modelo un atributo de "número de secuencia", que sirve para identificar, en forma singular, cada Fracción de Tiempo*** dentro de un componente.

Si es necesario corregir una Fracción de Tiempo previamente comunicada, se proporcionará una actualización a la Fracción de Tiempo, con el mismo número de secuencia, pero un número de corrección más alto. Como resultado, si existe más de una Fracción de Tiempo con el mismo número de secuencia relacionado a un determinado componente, se considerará válida la que tenga el número de corrección más alto.

La representación UML del modelo final de Fracción de Tiempo de los Componentes AIXM 5 aparece a continuación:

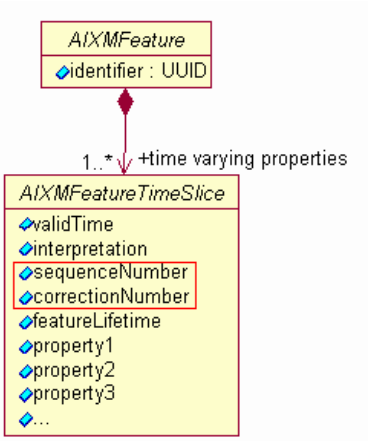


Figura 7 – Modelo completo de AIXMFeatureTimeSlice

En resumen, los tipos de Fracción de Tiempo utilizados en el AIXM son los siguientes:

- ***LINEA DE BASE = un tipo de Fracción de Tiempo que describe el estado del componente (el conjunto de todas las propiedades del componente) como resultado de un cambio permanente.***
- ***PERMDELTA = un tipo de Fracción de Tiempo que describe la diferencia en el estado de un componente como resultado de un cambio permanente.***
- ***TEMPDELTA = un tipo de Fracción de Tiempo que describe la superposición del estado de un componente durante un evento temporal***.
- ***SNAPSHOT*** = un tipo de Fracción de Tiempo que describe el estado de un componente en un instante, como resultado de la combinación de la Fracción de Tiempo de LINEA DE BASE (válida en ese instante) con todas las posibles Fracciones de Tiempo TEMPDELTA vigentes en ese instante.

Discusión: ***¿Cuál era el modelo de temporalidad de las versiones pasadas del AIXM?***

AIXM 3.x y 4.x brindan apoyo de temporalidad limitado. Es posible codificar el estado del componente en un momento dado (Mensaje AIXM-Snapshop) y comunicar las líneas de base (AIXM-Update). AIXM 3.x y 4.x no apoyan la codificación directa de la información sobre el estado temporal; tendría que hacerse en forma de una secuencia de dos líneas de base, una cambiando las propiedades y la segunda, revirtiendo a la situación anterior. Pero esto no permite distinguir entre los cambios permanentes reales y la información sobre el estado temporal.

Asimismo, AIXM 3.x y 4.x incorporan la temporalidad en el mensaje de intercambio y no en el componente mismo. En consecuencia, la temporalidad era una propiedad del mensaje y no una propiedad del componente aeronáutico. Las propiedades del mensaje describen de qué manera los sistemas receptores deberían interpretar el contenido del mensaje.

La limitada capacidad para transmitir información de tiempo utilizando los mensajes Update y Snapshot en AIXM 3.x y 4.x han llevado al desarrollo de este Concepto de Temporalidad más completo del AIXM 5, a nivel del componente.

2.7 Propiedades con horario

El Modelo de Temporalidad descrito hasta ahora funciona bien para los componentes que tienen propiedades con valores constantes durante su tiempo de validez. En algunos casos, una o dos propiedades de un componente pueden tener su propia variación cíclica en el tiempo, de conformidad con un horario establecido. Por ejemplo, una ayuda para la navegación puede estar operativa durante el día y fuera de servicio durante la noche; se podría activar un espacio aéreo restringido todos los días entre las 09:00 y las 17:00; etc.

A fin de modelar estas situaciones, se ha introducido en el AIXM 5.1 el concepto de "propiedades con horario". La idea es asociar las propiedades que tienen valores de variación cíclica con un "Timesheet" (cronograma) que describa los períodos en que cada valor es aplicable para dichos atributos. El concepto de Cronograma (Timetable/Timesheet) ya existía en el AIXM 3.x y 4.x. Fue heredado como tal por el AIXM 5.0, donde se encontró que, a veces, se superponía al concepto de Fracciones de Tiempo. Por lo tanto, fue necesario reformular el papel de los Cronogramas en el AIXM 5.1 (ver Propuesta de Cambio 5.1-35, que brinda un análisis más detallado de la necesidad de contar con horarios en el AIXM).

Discusión: ***¿Necesita realmente el modelo apoyar los horarios?***

Es obvio que los horarios existen en el dominio de los datos aeronáuticos. La pregunta es si el concepto de Fracciones de Tiempo es suficiente como para cubrir también dichas situaciones.

Teóricamente, el modelo de Fracciones de Tiempo sin horarios puede ser utilizado para un componente (por ejemplo, una ayuda para la navegación) que tiene una propiedad (por ejemplo, las horas de funcionamiento) que cambia cíclicamente (por ejemplo, está operativa todos los días entre las 06:00 y las 22:00). Pero esto significa que se codifica una LINEA DE BASE dedicada o un TEMPDELTA cada vez que cambia el operationalStatus de operativo a no operativo. Esto generaría 730 Fracciones de Tiempo de LINEA DE BASE o una Fracción de Tiempo de LINEA DE BASE y 365 Fracciones de Tiempo TEMPDELTA en un año, lo que constituye un gran inconveniente. Asimismo, el aspecto "cíclico" no sería visible de inmediato.

Por lo tanto, es necesario complementar el modelo de Fracciones de Tiempo con un concepto de "horarios", que permite modelar directamente la variación cíclica de los valores de una o más propiedades del componente.

A nivel de componente, todas las propiedades que varían de acuerdo a un horario establecido deben ser aisladas en una clase por separado, tal como se ilustra a continuación con la clase NavaidOperationalStatus (estado operacional de las ayudas para la navegación). Esta clase hereda de una clase abstracta denominada "PropertiesWithSchedule" (propiedades con horario).

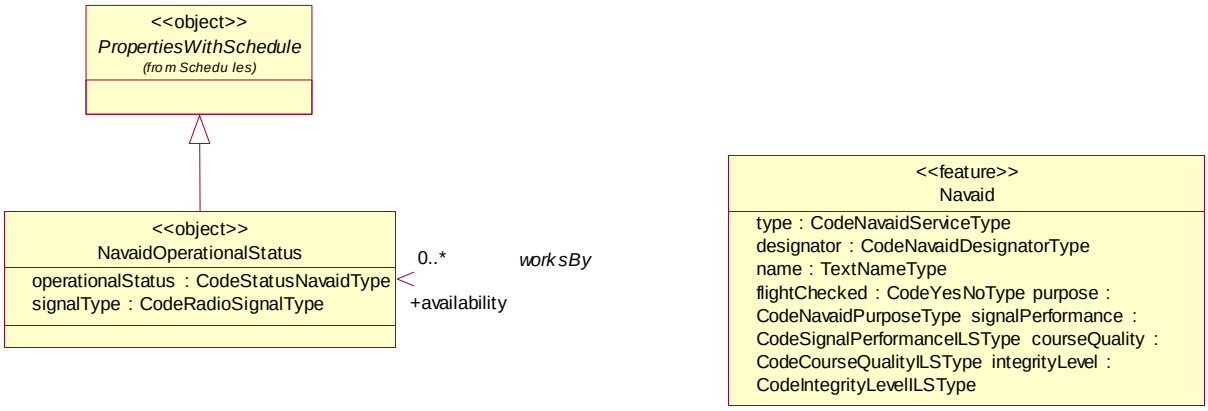


Figura 8 – Modelo de propiedades con horario

Luego, se asocia la clase abstracta PropertiesWith Schedule (propiedades con horario) al(los) Cronograma(s).

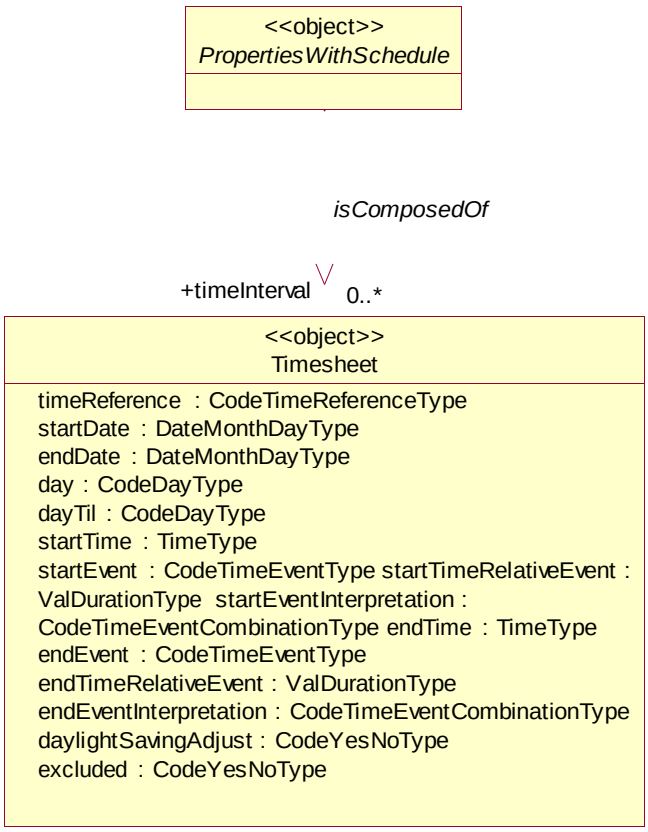


Figura 9 – Clase de cronograma

La clase de Cronograma contiene el sistema de referencia de tiempo (UTC-12 a UTC +14), el indicador de ahorro de luz diurna, y brinda la posibilidad de incluir/excluir fechas y horas especificadas. Por ejemplo, puede representar:

- un solo período de tiempo repetitivo; por ejemplo, "cada lunes de 10:00 a 16:00";
- un solo bloque de tiempo que abarca varios días; por ejemplo, "desde cada lunes a las 10:00 hasta el jueves al atardecer"
- un intervalo de fechas; por ejemplo, "cada año, del 15 de octubre al 15 de mayo";
- etc…

Discusión: ***¿Existe alguna opción a la introducción del concepto de "propiedades con horario"?***

Otra solución podría ser incluir "horarios" en el concepto de Fracciones de Tiempo y hacer que un horario pueda ser utilizado por cualquier componente. Esto tendría dos desventajas.

Si un atributo, como, por ejemplo, el valor de una distancia declarada, tiene un valor durante el día y otro valor durante la noche, cada uno de los dos valores tendría que ser parte de una Línea de Base diferente. Cada una de las Líneas de Base tendría un horario que indique cuándo son aplicables. Pero las dos Líneas de Base tendrían períodos de validez que se traslapan. Esto complicaría significativamente el concepto de Temporalidad del AIXM. El análisis también demuestra que, con frecuencia, los horarios sólo se refieren a uno o dos atributos. La aplicación del horario a nivel del componente enmascararía este importante aspecto.

Por lo tanto, se considera que la introducción del concepto de atributo con horario es el enfoque más conveniente.

La introducción de PropertiesWithSchedule (propiedades con horario) requiere reglas claras para poder interpretar las varias combinaciones que pueden ocurrir entre tipos de Fracciones de Tiempo, su validez y los horarios de sus propiedades. El riesgo es que el valor de una propiedad puede quedar sin definir si los horarios asociados con una LINEA DE BASE deja "vacíos". En las actuales operaciones AIS, para que las propiedades cambien de valor de acuerdo con un horario, es bastante común especificar únicamente el valor "principal", como, por ejemplo, "operacional", "activo", etc. Por ejemplo, se indica que la "ayuda para la navegación opera todos los días de 06:00 a 22:00", pero no se indica explícitamente cuál es el estado de la ayuda para la navegación entre las 22:00 y las 06:00. El personal operacional asumirá que la ayuda para la navegación no está operando entre las 22:00 y las 06:00.

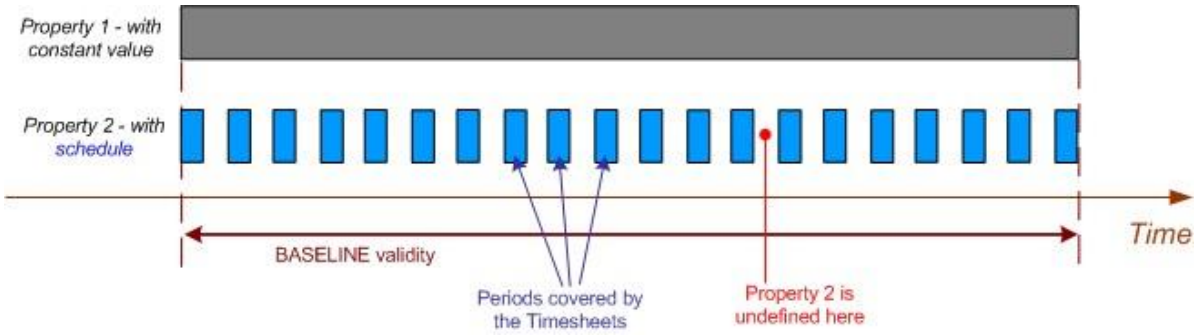


Figura 10 – Horario con vacíos

Debido a que las máquinas no pueden establecer "premisas", para fines del procesamiento de datos digitales, es más seguro también indicar explícitamente las horas "no operativas" de manera que los horarios asociados con los valores de la propiedad no dejen vacío alguno. Por lo tanto, se recomienda que las Fracciones de Tiempo de LINEA DE BASE contengan únicamente propiedades plenamente definidas, con horario, indicando explícitamente el valor de la propiedad en cualquier momento durante el período de validez de la Fracción de Tiempo. Si hay uno o más cronogramas asociados con una propiedad con horario, entonces el valor de la propiedad deberá ser considerado como no definido durante cualquier momento no cubierto por un Cronograma.

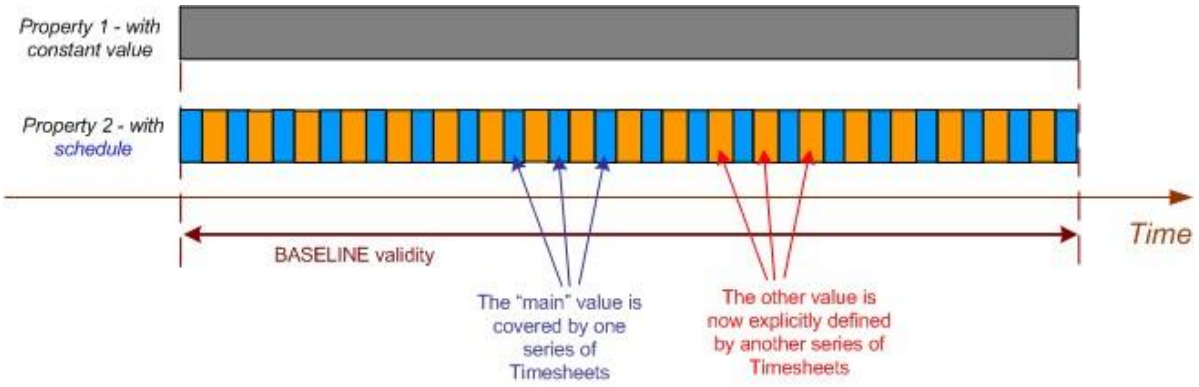


Figura 11 – Propiedad plenamente definida, con horario

Los Cronogramas que dejan vacíos pueden también presentarse en las Fracciones de Tiempo TEMPDELTA. Por definición, cualquier propiedad contenida en un TEMPDELTA anula el valor de la propiedad de LINEA DE BASE equivalente, mientras dure la validez del TEMPDELTA. Por lo tanto, como principio general, los tiempos codificados en los Cronogramas contenidos en las Fracciones de Tiempo TEMPDELTA también reemplazan totalmente los tiempos codificados en los Cronogramas de LINEA DE BASE equivalentes.

La situación es simple y clara si el TEMPDELTA no cuenta con Cronogramas o si estos Cronogramas abarcan todo el período de vigencia del TEMPDELTA, como se muestra en los siguientes diagramas:

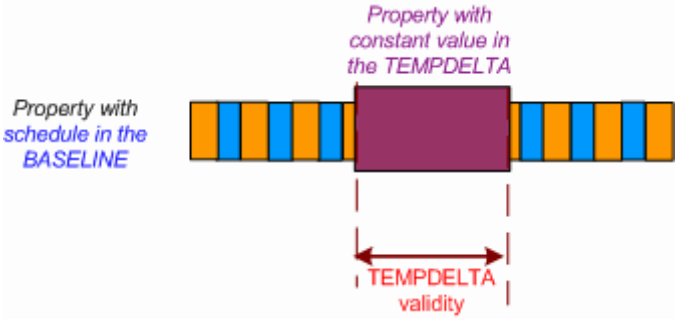


Figura 12 – El valor constante de TEMPDELTA reemplaza al horario de LINEA DE BASE

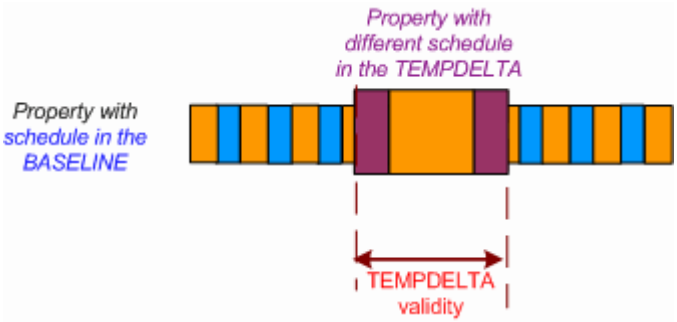


Figura 13 – El horario TEMPDELTA reemplaza al horario de LINEA DE BASE

Una situación que puede llevar a dificultades de interpretación es donde los Cronogramas asociadas con el TEMPDELTA dejan vacíos (períodos en que el valor de la propiedad no está explícitamente definido), como en el siguiente diagrama:

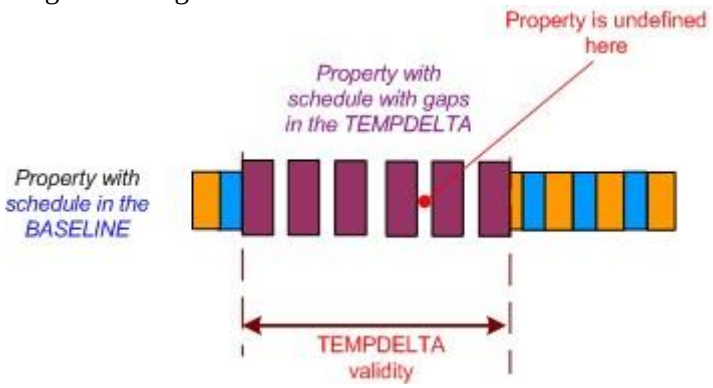


Figura 14 – Los vacíos en el horario TEMPDELTA dejan a la propiedad indefinida

La tentación sería considerar que la situación de LINEA DE BASE se aplica a los vacíos dejados en los Cronogramas asociados al TEMPDELTA. Pero esto estaría en conflicto con el principio general según el cual los valores de TEMPDELTA reemplazan totalmente a los valores de LINEA DE BASE. Por lo tanto, si un horario TEMPDELTA deja vacíos (períodos en los que el valor no está explícitamente indicado), entonces se considerará que la propiedad tiene un valor no especificado durante dichos períodos de tiempo.

En base a los ejemplos anteriores, se recomienda que los horarios TEMPDELTA no dejen períodos no especificados (vacíos) durante la vigencia del TEMPDELTA.

2.8 La temporalidad aplicada al modelo abstracto

El modelo UML AIXM contiene un conjunto de clases abstractas que son utilizadas como plantillas para los componentes y objetos definidos en el AIXM. La aplicación del concepto de Fracciones de Tiempo, tal como se describe en este documento, causaría la división de toda clase UML que representa a un componente en una clase principal y una clase "FeatureTimeSlice", tal como se muestra en el siguiente diagrama.

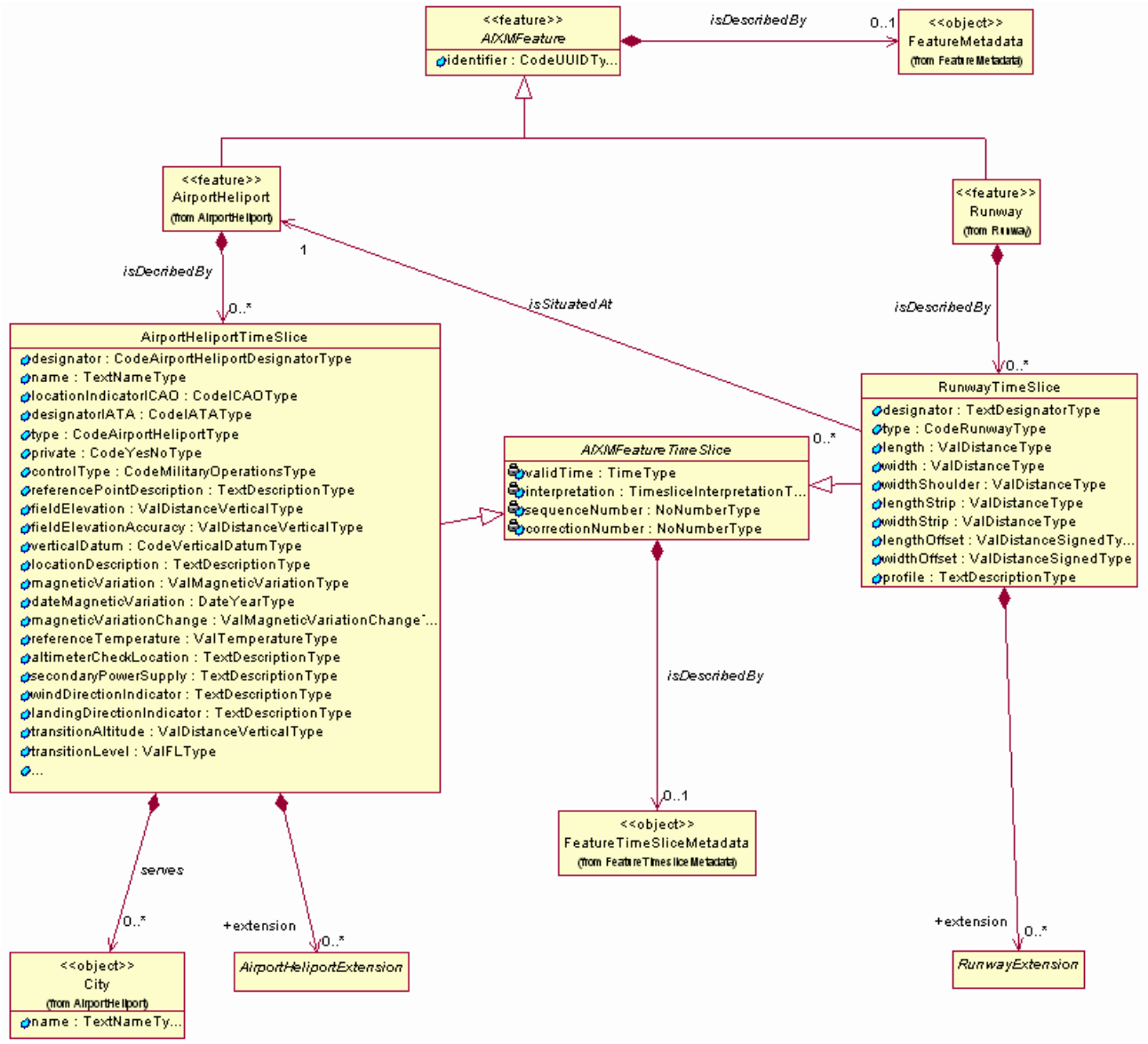


Figura 15 – Modelo ampliado con clases de Fracciones de Tiempo explícitas

El diagrama UML muestra cómo cada `<<feature>>` (componente) se deriva de la clase abstracta AIXMFeature. Los componentes concretos son descritos por las Fracciones de Tiempo que tienen propiedades. La Fracción de Tiempo se deriva de la clase abstracta AIXMFeatureTimeSlice.

El diagrama también muestra que cada componente del AIXM puede tener FeatureMetadata (metadatos de componente) y que cada Fracción de Tiempo puede tener FeatureTimeSliceMetadata (metadatos de la Fracción de Tiempo del componente). Finalmente, cada Fracción de Tiempo puede contener una Extensión. El mecanismo de extensión le permite a cada usuario del AIXM 5 definir y utilizar sus propios atributos y clases específicos, además de los AIXM básicos.

El diagrama anterior es bastante complejo. Si se aplica a todo el conjunto de clases del AIXM, podría perjudicar la capacidad de lectura de los diagramas UML, ya que habría que añadir una clase "Fracción de Tiempo" ("TimeSlide") por separado y las necesarias asociaciones para cada clase de componente (<<feature>>). **Por lo tanto, el Equipo de Diseño ha decidido brindar un modelo UML AIXM simplificado, sin una herencia visible de todos los componentes del AIXMFeature abstracto y sin clases visibles de *SomeFeature*TimeSlice**. No obstante, se asume que existe la división en clases SomeFeatureTimeSlice cuando se hace la conversión del modelo UML al Esquema XML AIXM.

3. Aspectos de aplicación

3.1 Fracciones de Tiempo de LINEA DE BASE con una duración indeterminada

Los cambios operacionalmente significativos en el dominio de la información aeronáutica están regulados por el ciclo AIRAC. Generalmente, cuando se comunica un cambio permanente, no se sabe cuándo ocurrirá el siguiente cambio permanente. Por lo tanto, desencadena la codificación de una LINEA DE BASE con un final de validez desconocido. Esto se expresa en GML como "<gml:endPosition indeterminatePosition="unknown"/>". Esta LINEA DE BASE cubrirá el período hasta el siguiente cambio permanente. Implícitamente, cuando ocurre el próximo cambio, la LINEA DE BASE anterior recibe un final de validez y debe ser actualizada/corregida.

La situación se puede representar como en el siguiente diagrama. La primera LINEA DE BASE, creada al comienzo de la vida del componente, inicialmente tiene final de validez desconocido. En este diagrama, está representada como "BASELINE 1", asumiendo que tiene un número de secuencia =1.

Cuando ocurre el cambio permanente "PERMDELTA 2", termina la validez de la LINEA DE BASE inicial y es reemplazada por una nueva LINEA DE BASE. A fin de representar plenamente la historia del componente, se instancia una versión corregida de la primera LINEA DE BASE (con el mismo número de secuencia=1 y también un número de corrección=1), esta vez con una fecha de finalización de validez conocida. La recién creada LINEA DE BASE tiene el número de secuencia=2 y aún no tiene corrección alguna.

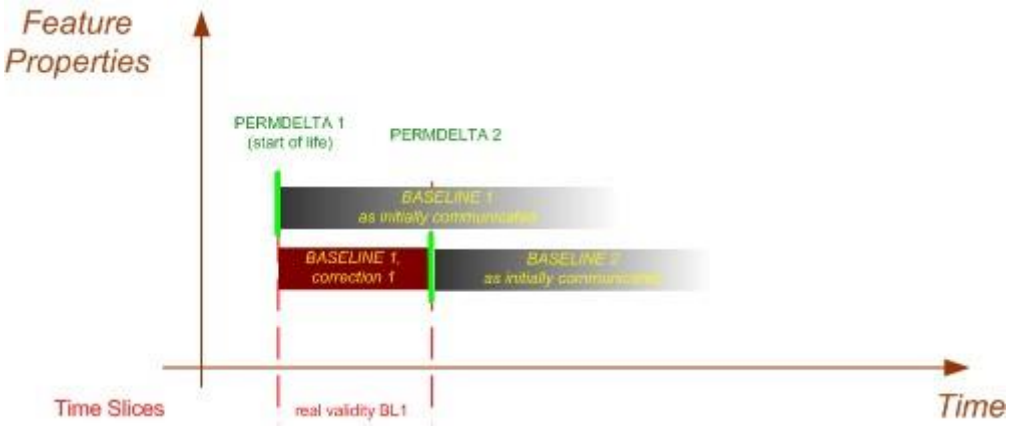


Figura 16 – Corrección de la LINEA DE BASE anterior, como resultado de un PERMDELTA

3.2 Valores de los números de secuencia

Como se explicó en 2.6, el número de secuencia se utiliza básicamente como identificador de la Fracción de Tiempo, a fin de aplicar una corrección. Por lo tanto, el número de secuencia deberá ser singular por tipo de Fracción de Tiempo, y debería ser persistente. No está permitido cambiar el número de secuencia de una Fracción de Tiempo, ya que esto podría romper el vínculo con una Fracción de Tiempo de corrección, y no existe un mecanismo en el AIXM para notificar el cambio de un número de secuencia.

Un aspecto secundario es que los números de secuencia también pueden ser utilizados para brindar cierta información cronológica acerca del momento en que se emitió dicha Fracción de Tiempo (no del orden en que adquiere validez!).

Por lo tanto, se recomienda que los números de secuencia sean asignados empezando con el "1" y se vayan incrementando en unidades de 1 ("2", "3", "4", etc.) cada vez que se codifica una nueva Fracción de Tiempo de ese tipo:

- El PERMDELTA inicial que crea al componente tendrá el número de secuencia=1 y la primera LINEA DE BASE resultante tendrá también el número de secuencia=1;
- El segundo PERMDELTA (el primer cambio del componente luego de su creación) tendrá el número de secuencia=2 y la LINEA DE BASE resultante tendrá también el número de secuencia=2, etc.
- Luego, el primer TEMPDELTA que ocurre tendrá el número de secuencia=1, el siguiente tendrá el número de secuencia=2, etc.

El resultado de esta recomendación aparece ilustrado en la Figura 17 – Historia completa de un componente.

3.3 Final de la vida útil del componente

Tal como se explicó en 2.5, las Fracciones de Tiempo PERMDELTA fueron introducidas para facilitar la notificación del final de la vida/desmantelamiento/retiro de un componente. Esto será codificado como un PERMDELTA que cambia la propiedad featureLifetime/../endPosition (de la Fracción de Tiempo de LINEA DE BASE válida al momento del retiro) de "indeterminado" a un valor de fecha y hora preciso. La fecha de entrada en vigencia del PERMDELTA deberá ser igual al valor del final de la vida. En este caso, no se incluye ninguna otra propiedad del componente en el PERMDELTA, ya que este PERMDELTA no dará como resultado el establecimiento de una nueva LINEA DE BASE, sino, simplemente, una corrección a la última LINEA DE BASE activa. Extendiéndolo a la historia completa del componente, la corrección de las LINEAS DE BASE inicialmente comunicadas hasta el final de la vida del componente puede representarse como se ilustra en el siguiente diagrama.

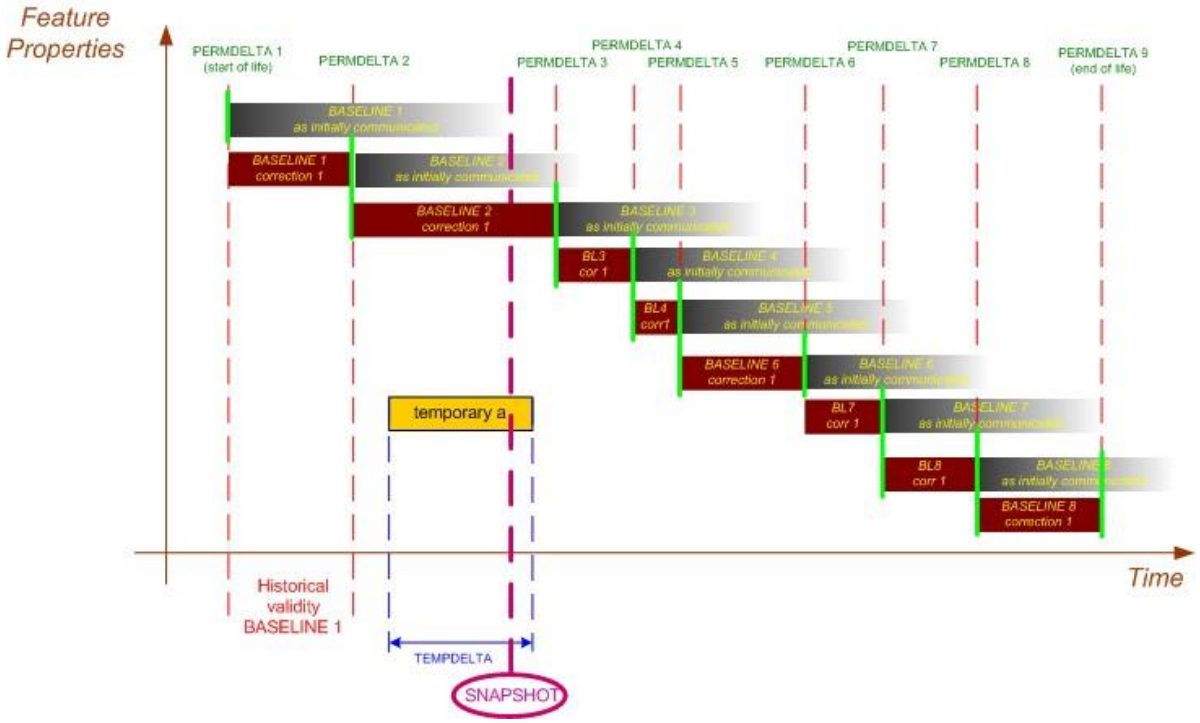


Figura 17 – Historia completa de un componente

3.4 "Delta" para propiedades complejas

Muchos componentes del AIXM tienen propiedades complejas conformadas por cero o más clases de elementos (representadas como clases agregadas en el modelo UML, 0..*). Por ejemplo, un AeropuertoHelipuerto tiene una AirportHeliportAvailability (Disponibilidad de AeropuertoHelipuerto) asociada, "composedOf" (conformada por) cero o más AirportHeliportUsage (usos de AeropuertoHelipuerto).

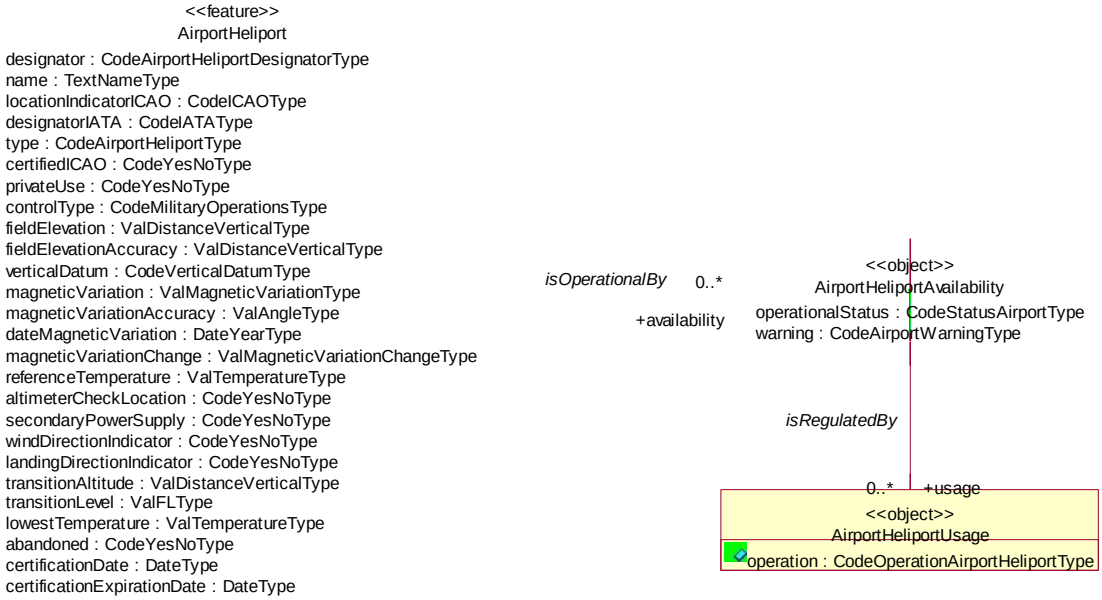


Figura 18

La pregunta es: ¿Qué deberían contener las Fracciones de Tiempo PERMDELTA o TEMPDELTA para dichas situaciones?

Por definición, las Fracciones de Tiempo "delta" deberán contener estrictamente los valores de las propiedades del componente que se ven afectadas, y esta regla se aplica únicamente a los componentes. Los objetos son considerados como tipos complejos de una propiedad de componente y tienen que ser incluidos en su totalidad en una Fracción de Tiempo "delta", en caso que haya cambiado la propiedad encapsulante del componente. Esto se explicará más adelante con un ejemplo.

Las propiedades de un componente son todos los atributos del componente y todas las asociaciones para las que el componente tiene navegabilidad (indicadas como una flecha que va de la clase del componente a otra clase). Por ejemplo, en el diagrama de clases anterior, las propiedades del componente AirportHeliport son todos atributos (designador, nombre, …, fecha de expiración de certificado (certificationExpirationDate)) y la propiedad de "disponibilidad", dada por el papel que desempeña la clase AirportHeliportAvailability en la asociación isOperationalBy. La propiedad de "disponibilidad" del espacio aéreo es compleja, conformada por varios AirportHeliportUsage. Si ocurre un cambio temporal o permanente dentro de AirportHeliportAvailability (por ejemplo, la modificación de uno de sus AirportHeliportUsage), entonces el AirportHeliportAvailability modificado deberá ser incluido en su totalidad en la Fracción de Tiempo TEMPDELTA o PERMDELTA.

3.5 "Delta" para propiedades de ocurrencia múltiple

Una regla equivalente se aplica a las propiedades de componente que ocurren múltiples veces. En el UML, dichas propiedades están encapsuladas en un Objeto, el cual está relacionado con la clase del componente mediante una asociación 0..*. Por ejemplo, un AirportHeliport puede brindar servicios a 0..* ciudades, tal como se indica en el siguiente diagrama. Esto significa que la propiedad "atiende" ("serves") del compuesto AirportHeliport tiene el potencial de ocurrir múltiples veces.

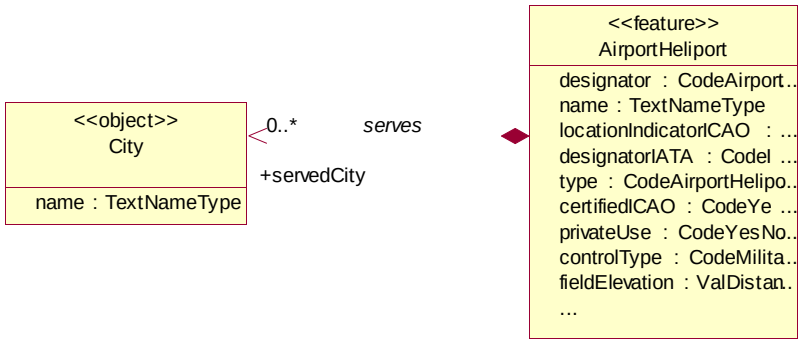


Figura 19

La regla es que, en una Fracción de tiempo PERMDELTA o TEMPDELTA, se indicará las propiedades de ocurrencia múltiple, incluyendo todas las ocurrencias. Por lo tanto, si un AirportHeliport atiende, por ejemplo, a dos ciudades y esto tiene que cambiarse en forma permanente a tres ciudades, las tres propiedades "servedCity" ("ciudad atendida") deben ser incluidas en un PERMDELTA.

3.6 Identificación del componente afectado por una Fracción de Tiempo DELTA

Una Fracción de Tiempo se codifica siempre como un elemento hijo de un componente. Como todo componente AIXM tiene la propiedad 'gml:identifier', esto debería ser suficiente para este fin. Supuestamente, este es un identificador único universal (del tipo UUID), que brinda una clave inequívoca para cada componente del AIXM.

No obstante, es probable que los identificadores únicos universales no existan por un tiempo. En esta situación, hay dos posibilidades:

- Utilizar la propiedad gml:identifier para codificar un identificador único local (una clave artificial) específico para el originador de los datos. En este caso, las Fracciones de Tiempo PERMDELTA y TEMPDELTA pueden ser operacionalmente recibidas sólo por el mismo originador que proporcionó los datos de LINEA DE BASE. Si se utiliza PERMDELTA/TEMPDELTA de otra fuente de datos, inevitablemente se rompería la cadena, ya que se utilizaría identificadores diferentes.
- O, además de la Fracción de Tiempo PERMDELTA o TEMPDELTA, incluir en el archivo AIXM una Fracción de Tiempo SNAPSHOT, conteniendo algunas propiedades (una "clave natural") que sean suficientes para identificar al componente. El hecho que el SNAPSHOT contiene sólo algunas propiedades naturales clave y no todas la propiedades no se contradice con la definición de una Fracción de Tiempo SNAPSHOT, ya que un SNAPSHOT representa la visión de un sistema en particular con respecto a ese elemento, que puede ser una visión incompleta. El receptor de los datos tendrá que interrogar a su sistema local e identificar el componente que tiene los mismos valores en ese momento, identificándolo así como el objetivo de la actualización.

3.7 Cancelación de una Fracción de Tiempo (cambios abandonados)

Para los sistemas de información aeronáutica que funcionan en modo "*push*", el principal medio para generar y brindar información acerca de un cambio son las Fracciones de Tiempo PERMDELTA y TEMPDELTA. La pregunta es qué procedimientos deberán aplicarse en caso de un cambio en la planificación, como, por ejemplo:

- Abandono de la puesta en servicio/desmantelamiento de un componente (antes de su fecha de entrada en vigencia)
- Abandono de un cambio permanente (antes de su fecha de entrada en vigencia)
- Abandono de un cambio temporal (antes de su fecha de entrada en vigencia)

Como ya se indicó (ver 2.6), la postergación/adelanto de un evento requiere una corrección de una Fracción de Tiempo, utilizando la propiedad de Número de secuencia (sequenceNumber) como clave para identificar la Fracción de Tiempo en cuestión. También se utilizará el Número de secuencia para identificar el PERMDELTA o TEMPDELTA que necesita ser abandonado. A fin de indicar claramente que el cambio contenido en la Fracción de Tiempo ha sido cancelado, la propiedad gml:validTime estará vacía y el atributo nilReason indicará "inaplicable". Por ejemplo, para cancelar un PERMDELTA de algún componente (SomeFeature) con el Número de secuencia (sequenceNumber) "23", se debe emitir un segundo PERMDELTA con el mismo Número de secuencia y un Número de corrección más alto, tal como se muestra a continuación:

TimeSlice (inicial)
- validTime = timeInstant…
- *interpretation = PERMDELTA*
- sequenceNumber = 23
- featureLifetime/beginPosition = same timeInstant…
- property 1
- property 2
- property 3
- property 4

5

TimeSlice (corrección)
- validTime : nilReason="inapplicable
- *interpretation = PERMDELTA*
- sequenceNumber = 23
- correctionNumber = 1
- featureLifetime/beginPosition: nilReason="inapplicable

Nótese que esta cancelación de Fracción de Tiempo no afecta a los sistemas '*pull'*, como son los servicios Web o el WFS, donde el sistema proporciona la información más actualizada en respuesta a una solicitud en línea del cliente. Supuestamente, el cliente no debe referirse a los resultados de una consulta anterior ni comparar los resultados con dichos resultados.

3.8 Superposición y corrección de Fracciones de Tiempo

El Número de secuencia y el Número de corrección son utilizados para resolver e interpretar las Fracciones de Tiempo superpuestas. Consideremos el escenario ilustrado en la siguiente figura, donde la propiedad de estado *(Status property)* de un componente es cambiada repetidamente a lo largo de varios intervalos de tiempo superpuestos. Cada cambio temporal recibe un Número de secuencia. En el ejemplo, una de las Fracciones de Tiempo es corregida, resultando en un Número de secuencia duplicado y un Número de corrección diferente.

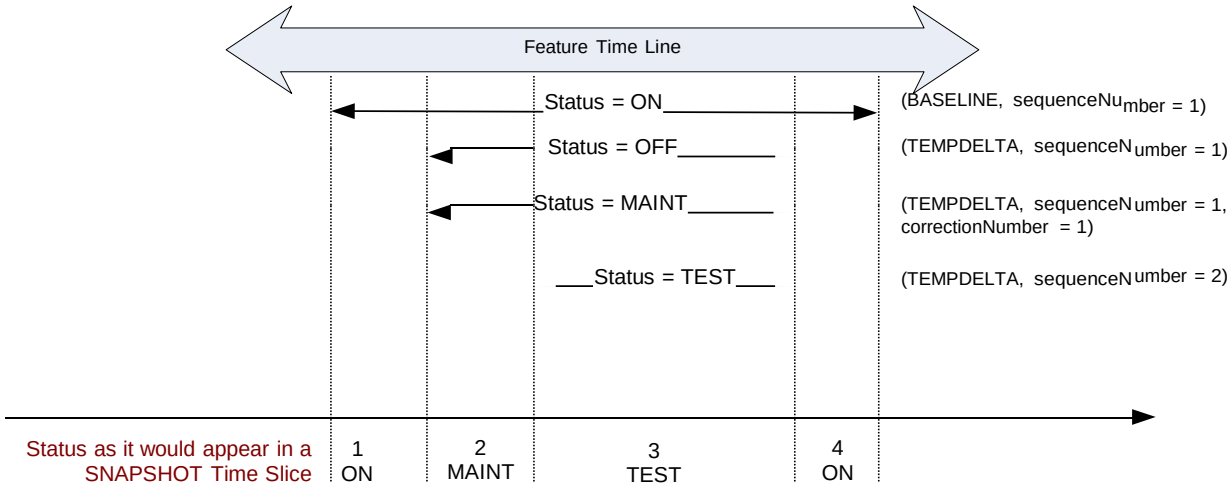


Figura 20. Ejemplo de correcciones y superposición de Fracciones de Tiempo

En los límites de cada evento temporal, podemos identificar transiciones a distintas versiones del componente. La combinación del tipo y Número de secuencia de la Fracción de Tiempo puede ser utilizada para identificar inequívocamente el valor de la propiedad de estado del componente a cada momento.

A fin de determinar el valor de una propiedad en un determinado momento, o inclusive en el transcurso de un determinado intervalo de tiempo, se debería aplicar las siguientes reglas:

1. Identificar la LINEA DE BASE vigente en ese momento, observando su Tiempo de Validez (validTime). En caso existan varias, todas deberían tener el mismo Número de secuencia y distintos Números de corrección. Elegir la que tiene el Número de corrección más alto;
2. Identificar todos los TEMPDELTA vigentes en el momento especificado;
3. Clasificar los TEMPDELTA por Número de secuencia, en orden ascendente;
4. Aplicar los TEMPDELTA al componente, del Número de secuencia más bajo al más alto.
    a. Cuando dos o más deltas tienen el mismo Número de secuencia, aplicar el delta con el Número de corrección más alto.

La posibilidad de resolver la superposición de los TEMPDELTA utilizando el Número de secuencia y el Número de corrección muestra cómo se puede comunicar las cancelaciones y las correcciones. En este ejemplo, el Número de secuencia=1 de TEMPDELTA es utilizado inicialmente para comunicar que el Status del componente (feature Status) = OFF. Posteriormente, se transmite una corrección de Fracción de Tiempo utilizando el mismo Número de secuencia = 1 pero con un Número de corrección = 1; corrige el estado del componente al Status = MAINT. No obstante, el estado final está dado posteriormente por el TEMPDELTA con un Número de secuencia 2, que indica el Status del componente = TEST.

3.9 Otras consideraciones relacionadas con la implantación

El modelo temporal conceptual descrito en la sección anterior brinda considerable flexibilidad para los sistemas que implementan la temporalidad. Un sistema que intentara implantar plenamente el modelo de temporalidad AIXM sería muy complejo. Sin embargo, no se requiere que los sistemas que implementan el AIXM apoyen todos los tipos de Fracciones de Tiempo. Por ejemplo:

- Algunos sistemas podrían sólo almacenar datos de Fracciones de Tiempo de LINEA DE BASE e ignorar cualquier cambio temporal. Algunos ejemplos de estos sistemas son la publicación AIP, los editores de cartas impresas y los sistemas basados en ARINC 424.
- Algunos sistemas podrían sólo transmitir y almacenar cambios temporales. Algunos ejemplos de estos sistemas son los sistemas NOTAM. No obstante, dichos sistemas necesitan referirse a la fuente de los datos de LINEA DE BASE.
- Algunos sistemas podrían únicamente requerir *snapshots* periódicos que indicquen el estado actual del sistema. Un ejemplo es un sistema de monitoreo pasivo diseñado para reportar el estado del sistema a intervalos de tiempo seleccionados.
- Algunos sistemas podrían querer un nuevo *"snapshot"* luego de cada cambio, sin hacer distinción entre un cambio temporal y un cambio permanente. Ejemplos de esto incluyen a los sistemas de gestión de tránsito y de procesamiento de planes de vuelo.
- Se puede desarrollar sistemas que puedan procesar e interpretar todos los componentes temporales y proporcionar a los usuarios Fracciones de Tiempo de Línea de Base, Delta y *Snapshots* en cualquier momento.

El AIXM contiene un modelo temporal completo; sin embargo, como se muestra en los ejemplos, es responsabilidad de los sistemas interactuantes el negociar los requisitos específicos de intercambio de datos temporales, así como integrar la temporalidad en sus subsistemas internos.

4. Ejemplos de uso

4.1 Ejemplo de ayuda para la navegación

La Figura 21 ilustra el modelo temporal, mostrando un cambio en la frecuencia de transmisión para una ayuda para la navegación (VOR AML, de 112.0 MHz a 113.2 MHz). Normalmente, este cambio debería ocurrir en una fecha del ciclo AIRAC. Generalmente, el cambio requiere que la ayuda para la navegación esté fuera de servicio por un cierto tiempo; luego, que esté a prueba en la nueva frecuencia. Actualmente, el estado temporal es comunicado a través de mensajes NOTAM.

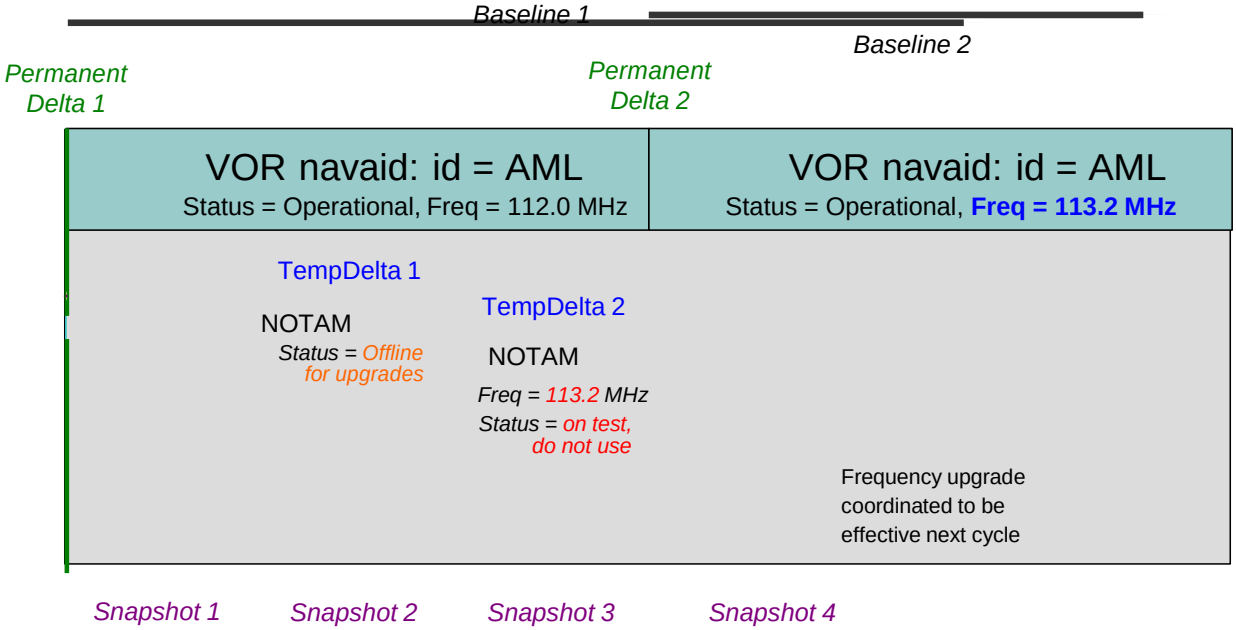


Figure 21

En base a este diagrama, podemos identificar los siguientes componentes temporales:

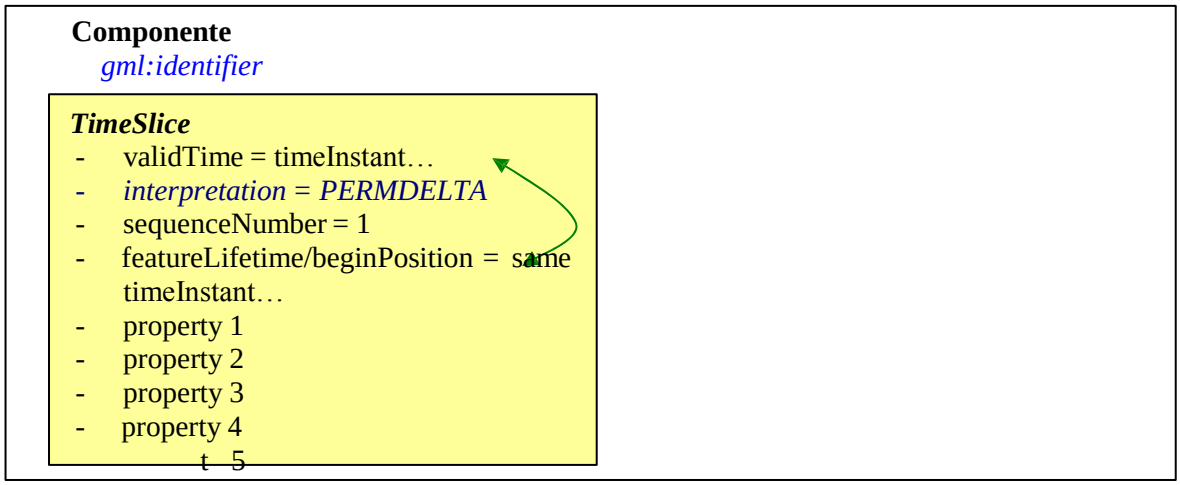
- El diagrama muestra dos Fracciones de Tiempo de LINEA DE BASE. La primera línea de base tiene una frecuencia de ayuda para la navegación de 112.0 MHz y está vigente desde hace un tiempo; la segunda línea de base tiene una nueva frecuencia de 113.2 MHz y está vigente desde la fecha de ciclo AIRAC.
- Un PERMDELTA puede ser utilizado para describir el cambio permanente de estado, que es el cambio de frecuencia del VOR AML. Para fines de integridad, también se muestra el anterior PERMDELTA que precede a la LINEA DE BASE (1).
- Cada evento transitorio puede ser expresado como un TEMPDELTA que cambia el Estado Operacional de la ayuda para la navegación y, eventualmente, la frecuencia.
- En base a las Fracciones de Tiempo PERMDELTA y TEMPDELTA mostradas en el diagrama, pueden existir cuatro versiones distintas del "estado actual del componente". Cada versión del "estado actual" comienza y termina en el límite de un Delta Permanente o Temporal, y puede ser presentado como una Fracción de Tiempo del tipo SNAPSHOT.

Dependiendo de la implantación temporal utilizada por los sistemas que realizan el intercambio, se puede utilizar distintos métodos para comunicar los cambios del componente. En aras de una normalización a nivel mundial, el resto de esta sección brinda algunas recomendaciones. Estas son

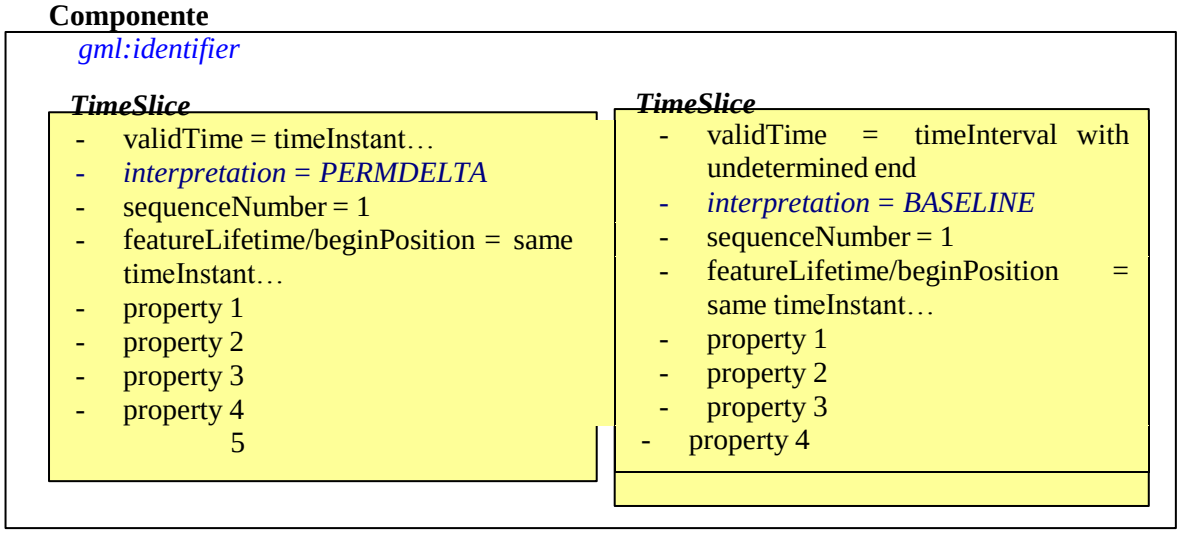
pertinentes especialmente para las aplicaciones del tipo *"push"*, las cuales generan y proporcionan notificaciones en la forma de Fracciones de Tiempo TEMPDELTA y PERMDELTA.

4.2 Creación de componentes (puesta en servicio)

El inicio de la vida de un componente (también conocido como "puesta en servicio") está modelado como un PERMDELTA, que le da un valor inicial a la propiedad startOfLife y a todas las otras propiedades definidas del componente. El tiempo de validez (validTime) del PERMDELTA deberá ser la fecha y hora de entrada en vigencia en que el componente es puesto en servicio.



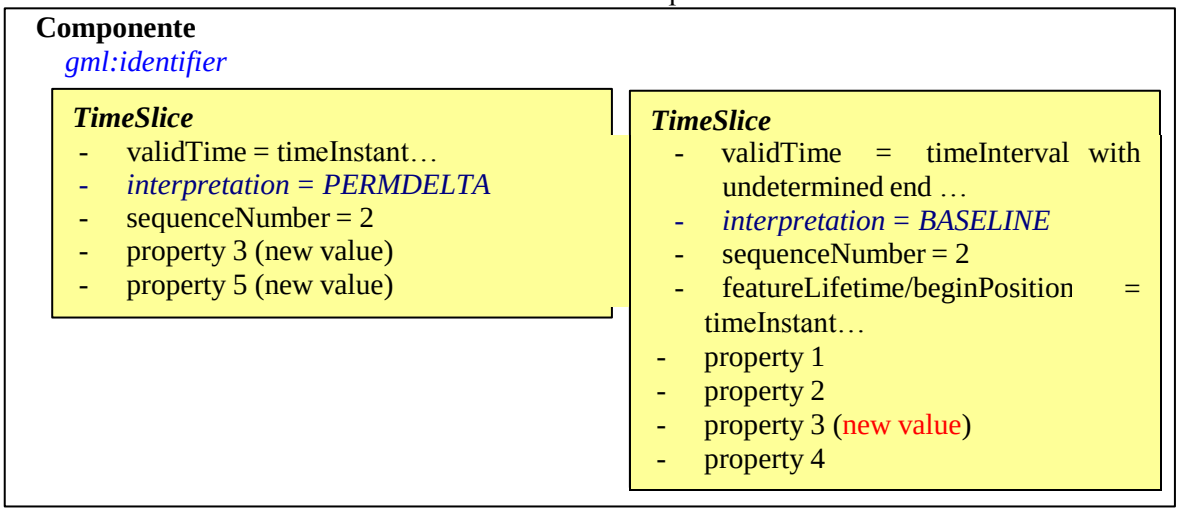
Opcionalmente, si así lo solicita el usuario, también se puede incluir una Fracción de Tiempo de LINEA DE BASE que contenga los mismos valores de propiedades que el PERMDELTA (ya que son el resultado del PERMDELTA). El tiempo de validez (validTime) de la LINEA DE BASE deberá ser un Intervalo de tiempo (timeInterval), con un final "indeterminado".



4.3 Cambio permanente

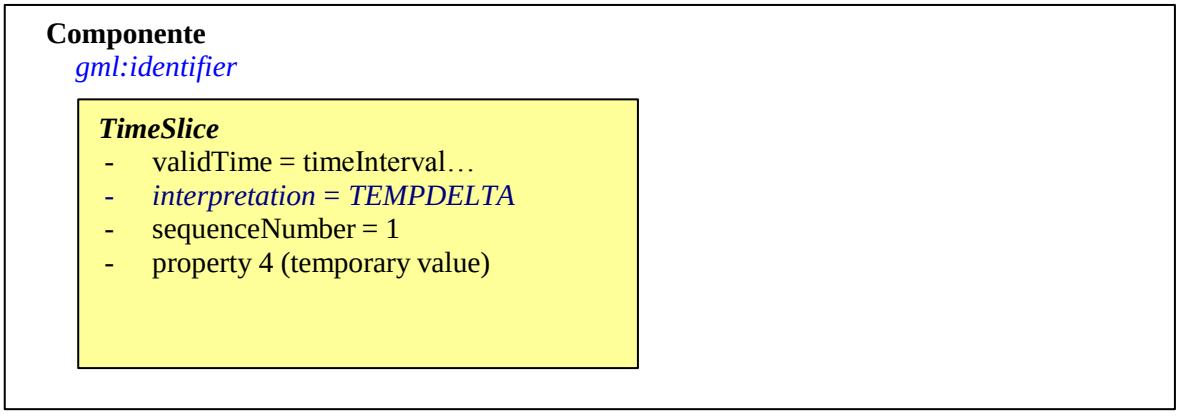
Un cambio permanente es modelado como una Fracción de Tiempo PERMDELTA, que contiene todas las propiedades que cambian de valor. El Tiempo de validez (validTime) del PERMDELTA deberá ser la fecha y hora de entrada en vigencia del cambio.

Opcionalmente, se puede incluir una Fracción de Tiempo de LINEA DE BASE que contenga todas las propiedades que tienen un valor resultante luego del PERMDELTA. El Tiempo de validez de la nueva LINEA DE BASE deberá ser un Intervalo de tiempo con un final "indeterminado".

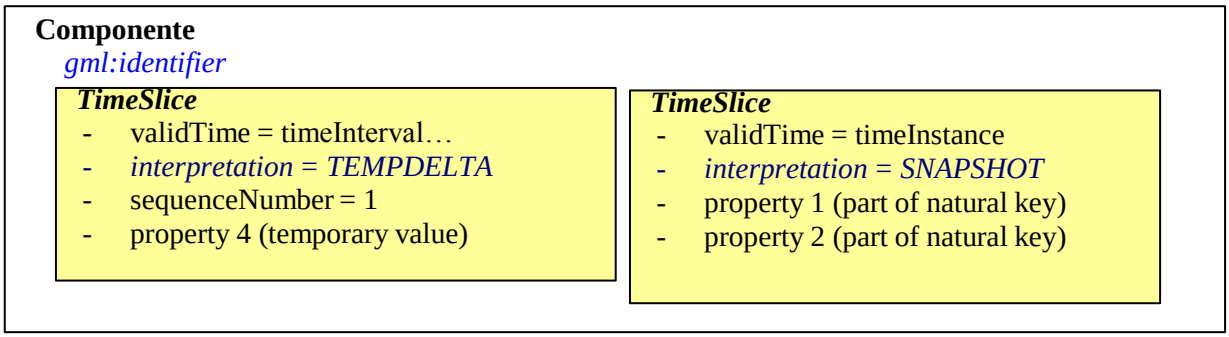


4.4 NOTAM digital

Un estado temporal de un componente está codificado como una Fracción de Tiempo TEMPDELTA que contiene todas las propiedades que cambian de valor en forma temporal. El Tiempo de validez del PERMDELTA deberá indicar el comienzo y el final de la vigencia del estado temporal. El final puede ser indeterminado.

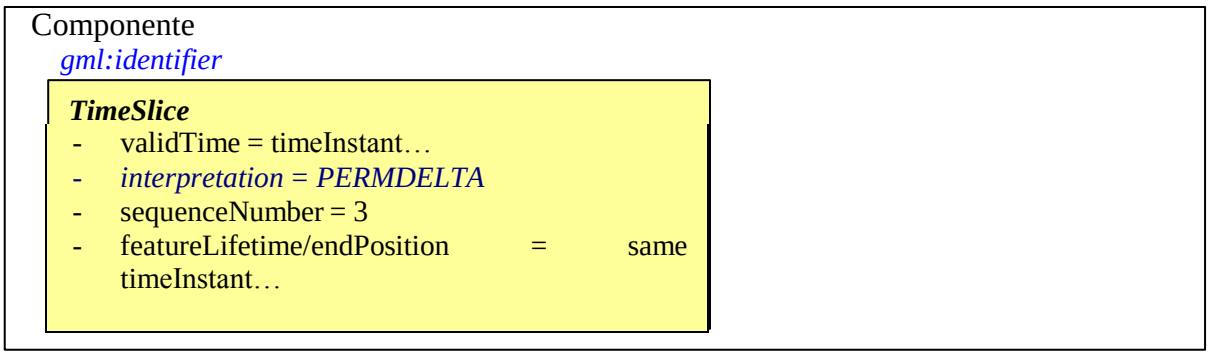


Opcionalmente, se puede incluir una Fracción de Tiempo SNAPSHOT en el conjunto de datos (utilizado como "clave natural").

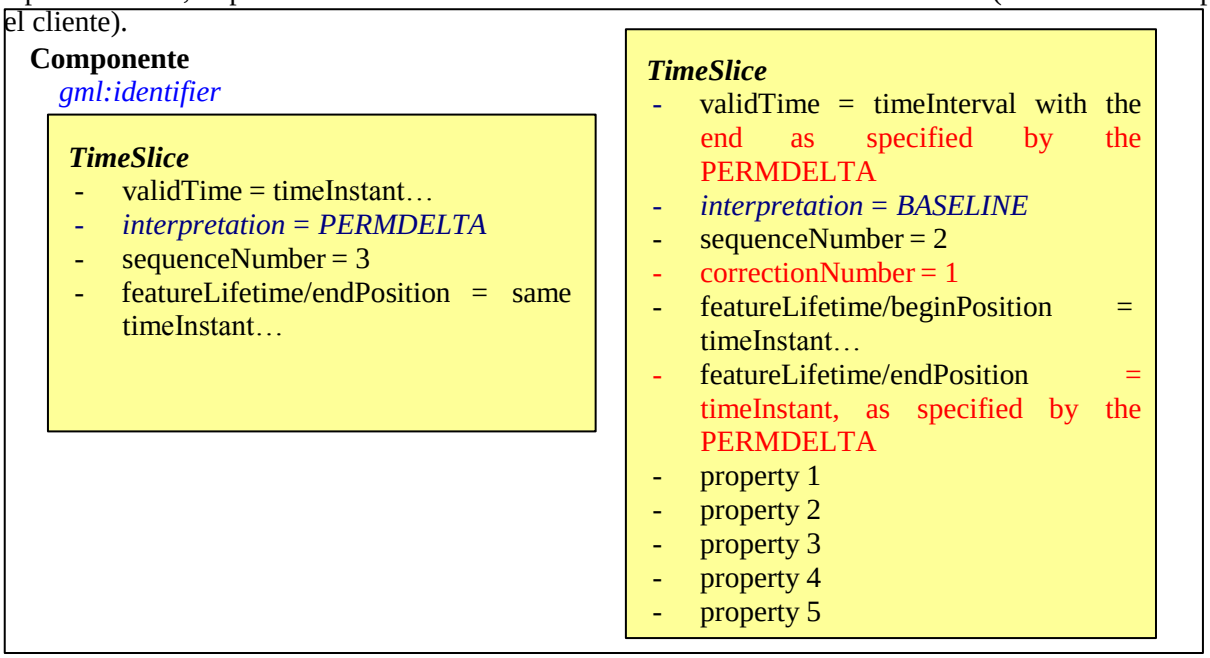


4.5 Final de la vida útil (retiro del servicio)

El final de la vida útil de un componente (también conocido como "retiro permanente" o "desmantelamiento") está modelado como un PERMDELTA, que le da un valor al featureLifetime/endPosition.



Opcionalmente, se puede incluir la corrección de la LINEA DE BASE más reciente (de ser solicitado por el cliente).



4.6 Historias completas de componentes

El modelo de Fracción de Tiempo puede ser utilizado para transmitir la historia de un componente, transmitiendo la secuencia de cambios que ocurren en la propiedad del componente. La historia del componente puede ser la historia pasada o la historia futura.

La Figura 22 muestra un ejemplo de historia de una ayuda para la navegación VOR ficticia. La ayuda para la navegación tiene los siguientes eventos:

- Enero 7, 2006: Puesta en servicio
- Enero 23 – Feb 18, 2006: Cambio temporal de frecuencia
- Feb 11 – Mar 9, 2006: Fuera de línea en forma temporal
- Feb 22, 2006: Cambio en la variación magnética
- Mar 27, 2006: Cambio en la frecuencia

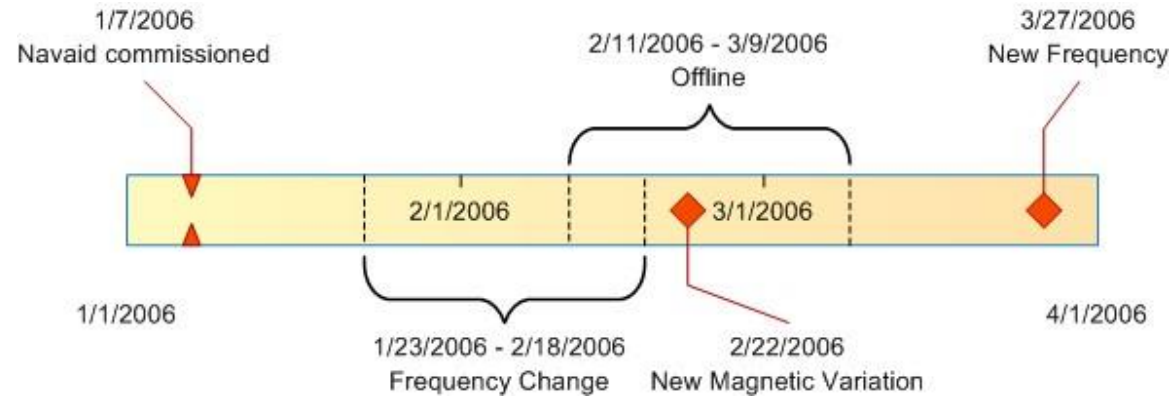


Figura 22: Ejemplo ficticio – historia de una ayuda para la navegación VOR

Utilizando el modelo de Fracción de Tiempo, podemos representar la historia de una ayuda para la navegación VOR como una serie de cinco Fracciones de Tiempo, tal como se muestra en la Figura 23. Se utiliza tres Fracciones de Tiempo para representar los estados y dos para representar los eventos temporales. Nótese que los eventos superpuestos se codifican como Fracciones de Tiempo separadas. Las Fracciones de Tiempo PERMDELTA no aparecen en este ejemplo.

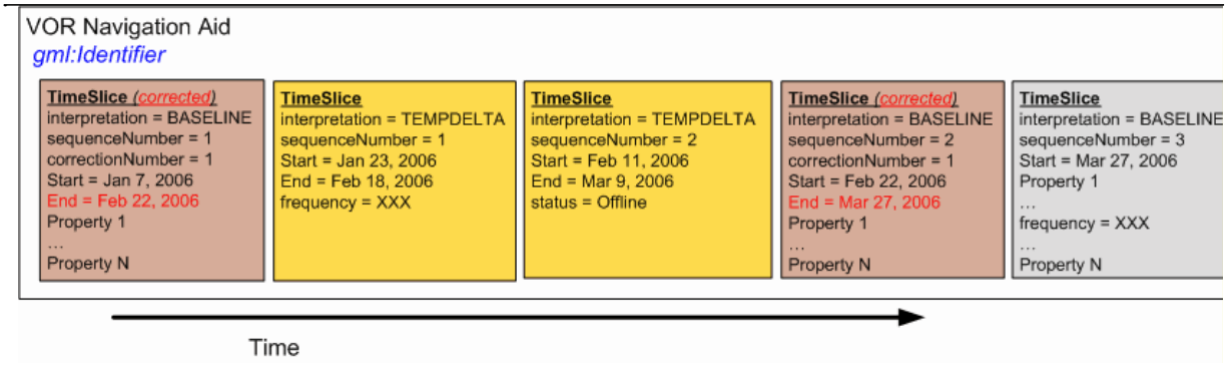


Figura 23: Fracciones de Tiempo para la historia de la ayuda para la navegación VOR

Este enfoque de modelado de la historia es equivalente al enfoque recomendado para GML 3.2 [4]. En implantaciones reales del modo de Fracción de Tiempo AIXM, la comunicación de historias puede generar mensajes muy extensos. Estos mensajes extensos podrían ser un problema para algunos sistemas con recursos limitados. Si bien los problemas de implantación están fuera del alcance de este documento de diseño, queremos indicar que se debería sopesar la desventaja del tamaño del mensaje en comparación con el valor de la normalización y cumplimiento con el GML. Generalmente, el valor de la normalización puede superar la pérdida de eficiencia del mensaje.

Referencias

1. Aeronautical Information Exchange Model (AIXM), Exchange Model goals, requirements and design, diciembre de 2006, www.aixm.aero
2. Aeronautical Information Conceptual Model, Edition 1.0, Ref. AIS.ET2.ST01.2000-02, 01 octubre de 1997 (Eurocontrol Extranet, OneSky Teams)
3. "Dynamic Features" Tim Wilson and David Burggraf. September 29, 2005. Contract deliverable to FAA from Galdos Systems Inc.
4. GML: Geography Markup Language. Ron Lake, David S. Burggraf, Milan Trninic, Laurie Rae. Wiley 2004.
5. Temporal Features, James Ressler, Northrop Grumman TASC, OPENGIS PROJECT DOCUMENT #06-076
6. Geographic information - Geography Markup Language (GML), ISO 19136:2007(E) 2007-03-12
7. AIXM Primer. 4.5 draft 2 Edition. EATMP-xxxxxx-xx. Nov. 28, 2005. EUROCONTROL.
8. Anexo 15 del Convenio de Aviación Civil Internacional – Servicios de Información Aeronáutica. 12ª. Edición. OACI. Julio de 2004.

APÉNDICE C

AIXM

MAPEO DEL UML AL ESQUEMA XML

AIXM

Mapeo del UML al Esquema XML

Modelo de Intercambio de Información Aeronáutica (AIXM)

Para cualquier consulta, sírvase ponerse en contacto con:

Brett BRUNK - brett.brunk@faa.gov

Eduard POROSNICU - eduard.porosnicu@eurocontrol.int

Versión No.	Fecha de emisión	Autor	Razón del cambio
0.1	2006/06/22	Brett Brunk	Primera edición
	2006/08/25	Barb Cordell	Incorporar la sección sobre modelado de datos
	2006/09/06	Vamshi Reddy	Incorporar la sección sobre tipos de datos
	2007/01/30	Barb Cordell	Actualizar para Versión Candidata 1
0.2	2007/07/31	Scott Wilson	Actualizar para Versión Candidata 2
0.3	2007/09/20	Eddy Porosnicu	Modelo modificado en las unidades de medición
0.4	2008/01/15	Eddy Porosnicu	Reglas de asociaciones modificadas: de aquí en adelante, basadas en los nombres del papel que desempeñan.
1.0	2008/03/15	Eddy Porosnicu	Cambios editoriales para la primera versión pública.
1.1	2010/02/04	Eddy Porosnicu Hubert Lepori	Actualizada para el AIXM 5.1

INDICE

1 Alcance

1.1 Introducción

El Modelo Conceptual AIXM es mantenido como un modelo de clase UML. El formato de intercambio AIXM es codificado como una serie de esquemas XML. Existe una vínculo directo entre el Modelo Conceptual AIXM y el Esquema XML AIXM.

Este documento describe cómo el Modelo Conceptual AIXM es convertido en un Esquema XML AIXM. El proceso de conversión aparece ilustrado en una serie de ejemplos del esquema XML AIXM 5.

1.2 Referencias

1. Geographic Information – Spatial Schema. ISO 19107. Primera edición, 2003-05-01
2. ISO 19136:2007 - Geographic information -- Geography Markup Language (GML)
3. UML 2.0 In a Nutshell. Dan Pilone. O'Reilly Media Inc. 2005.
4. AIXM Temporality Model, www.aixm.aero (ver Descargas)

2 Convenciones de modelado UML AIXM

2.1 Tipos de diagramas

El modelo utiliza dos tipos de diagramas:

Diagramas de clases – Son utilizados para representar los componentes, propiedades, relaciones y herencias entre componentes;
Diagramas de paquetes – Son utilizados para dividir el modelo en módulos e identificar las dependencias entre los conjuntos de clases.

2.2 Estereotipos

Las clases se diferencian por sus estereotipos. Los estereotipos son utilizados para definir mejor y extender los conceptos UML normalizados. Los principales estereotipos son `<<feature>>`, `<<object>>`, `<<choice>>`, `<<datatype>>` y `<<codelist>>`.

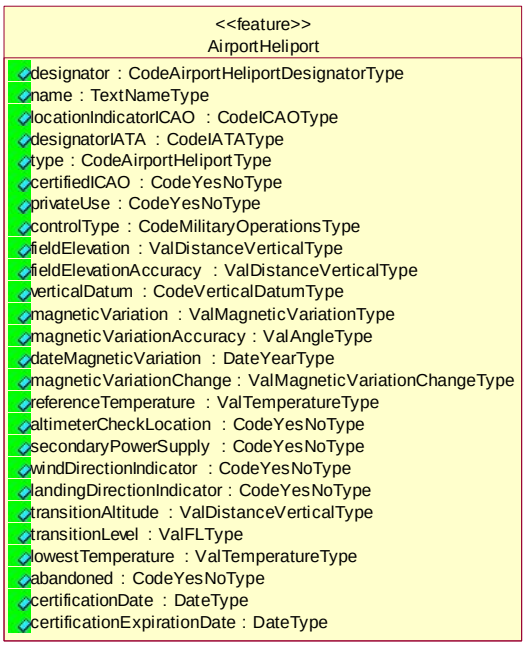
2.3 Clases abstractas

Además, algunas clases son abstractas. Las clases abstractas son designadas colocando el nombre de clase en *itálicas.* Una clase abstracta no puede ser materializada en una implantación como la de un documento XML. Más bien, las clases abstractas son utilizadas como clases de base en una jerarquía de herencia. Por ejemplo, la clase abstracta AIXMFeature describe las propiedades básicas de un componente AIXM. Cada componente específico AIXM, como una Pista, hereda[1] de la clase abstracta AIXMFeature.

2.4 Componentes

Los componentes describen las entidades del mundo real y son fundamentales para el AIXM. Los componentes AIXM pueden ser concretos y tangibles, o abstractos y conceptuales, y pueden cambiar con el tiempo. Los componentes están representados como clases con un estereotipo `<<feature>>`. Algunos ejemplos son: Pista y AeropuertoHelipuerto.

Los componentes AIXM son dinámicos. Los objetos Timeslice son utilizados para describir los cambios que afectan al componente AIXM a través del tiempo. Los objetos Timeslice y la temporalidad son analizados ampliamente en un documento por separado sobre la Temporalidad AIXM.



[1] Sírvase consultar la sección 3.1, El Modelo Abstracto, que explica por qué esta herencia no es visible en el UML.

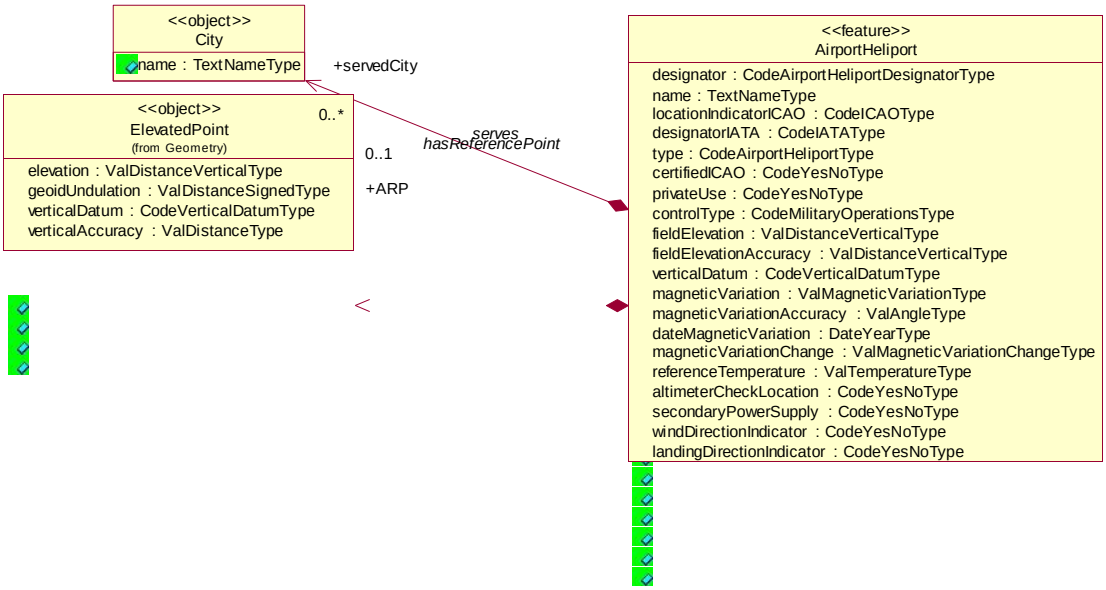
2.5 Objetos

Los objetos son abstracciones de las entidades del mundo real o, más frecuentemente, de las propiedades de dichas entidades, las cuales no existen fuera de un componente. Hay dos razones para crear un objeto en el AIXM:

Cuando una propiedad tiene un multiplicidad superior a uno (como la ciudad que es atendida por un AeropuertoHelipuerto), o

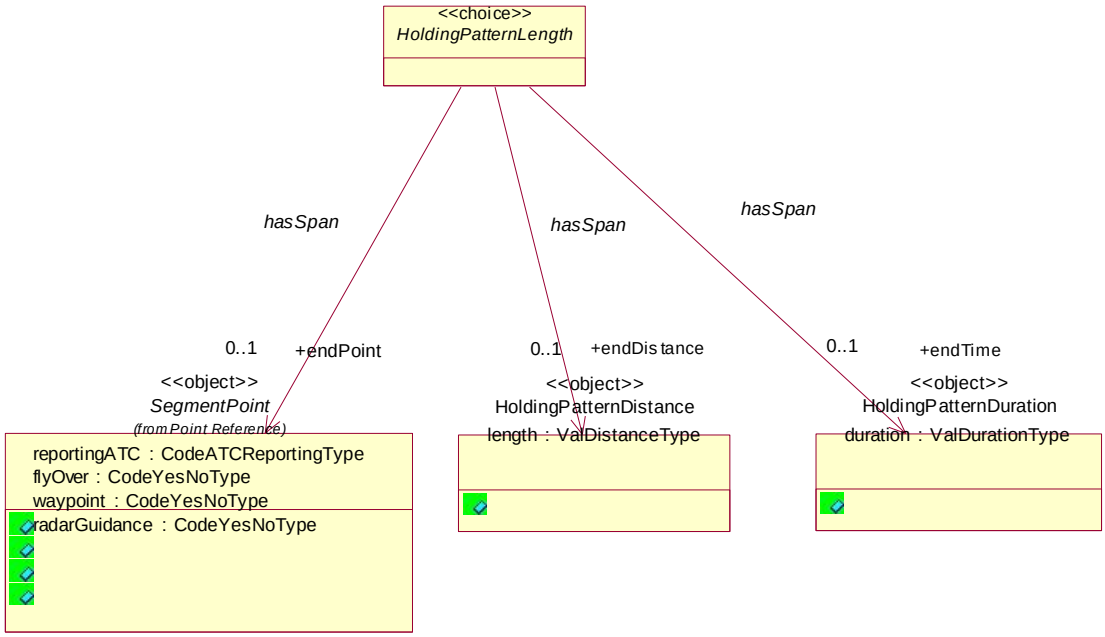
El objeto tiene sus propios atributos que son reutilizados a través del modelo (como ElevatedPoint).

En ambos casos, la propiedad está representada como un objeto con la debida relación de composición UML, como se muestra a continuación.



2.6 Opciones

Algunas clases están marcadas como `<<choice>> (opción)`. Estas se utilizan para modelar las relaciones XOR. Por ejemplo, la longitud de un Circuito de Espera puede expresarse utilizando un HoldingPatternDistance, un HoldingPatternDuration o un SegmentPoint, que definen el final de un tramo de salida.



2.7 Propiedades

Las propiedades son los atributos y relaciones que caracterizan al componente u objeto. En el UML:

Los atributos son utilizados para describir propiedades simples de un componente u objeto; las relaciones son utilizadas para describir las asociaciones con los componentes u objetos. Si una propiedad tiene una multiplicidad mayor a uno, es descrita utilizando una relación UML con cardinalidad.

2.7.1 Atributos

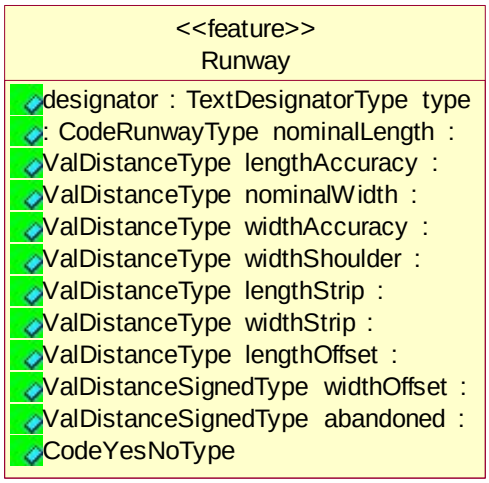
Las propiedades simples de cardinalidad aparecen como atributos en el diagrama UML.

Un atributo tiene el siguiente formato:
Visibility / stereotype name : type multiplicity
(Visibilidad/nombre de estereotipo: multiplicidad de tipo)

Para el AIXM 5, se utiliza los siguientes valores:

Visibilidad – Público
/ – no se utiliza
Estereotipo – No se utiliza
Nombre – nombre de la propiedad
Tipo – tipo propiedad
Multiplicidad – generalmente, no se especifica; por motivos relacionados con el modelo de temporalidad AIXM, en la implantación, se debería asumir que todas las propiedades son opcionales [0..1]

A manera de ilustración, el componente Pista tiene varias propiedades simples (por ejemplo, el designador y el tipo). A estas categorías se les asigna un datatype (tipo de datos); por ejemplo, el atributo del designador es del tipo TextDesignatorType.

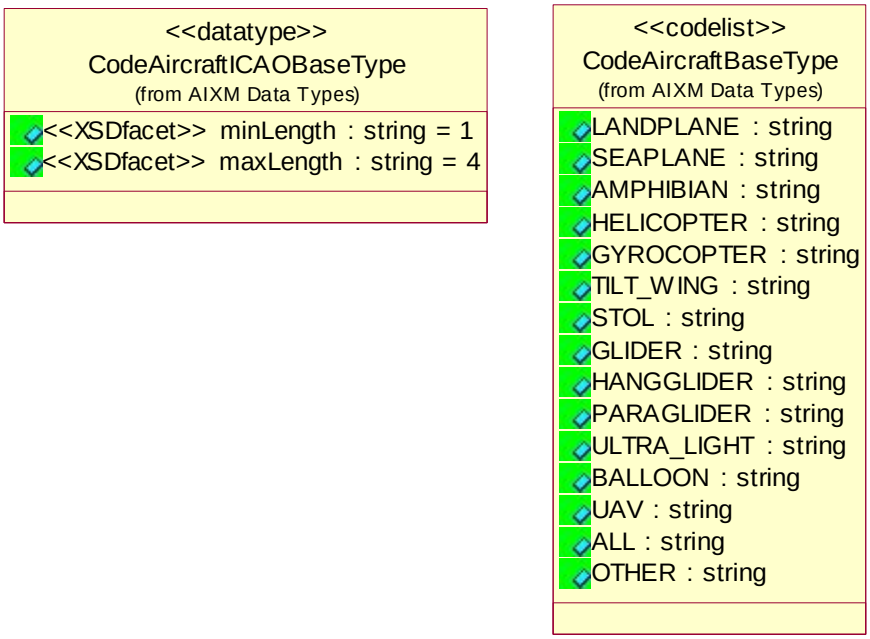


2.7.1.1 Tipos de datos

El modelo UML enumera los tipos de datos utilizados a través del AIXM. A estos se les asigna uno de los siguientes estereotipos:

<<datatype>> - Este es el tipo de datos básico que especifica un patrón a ser utilizado.

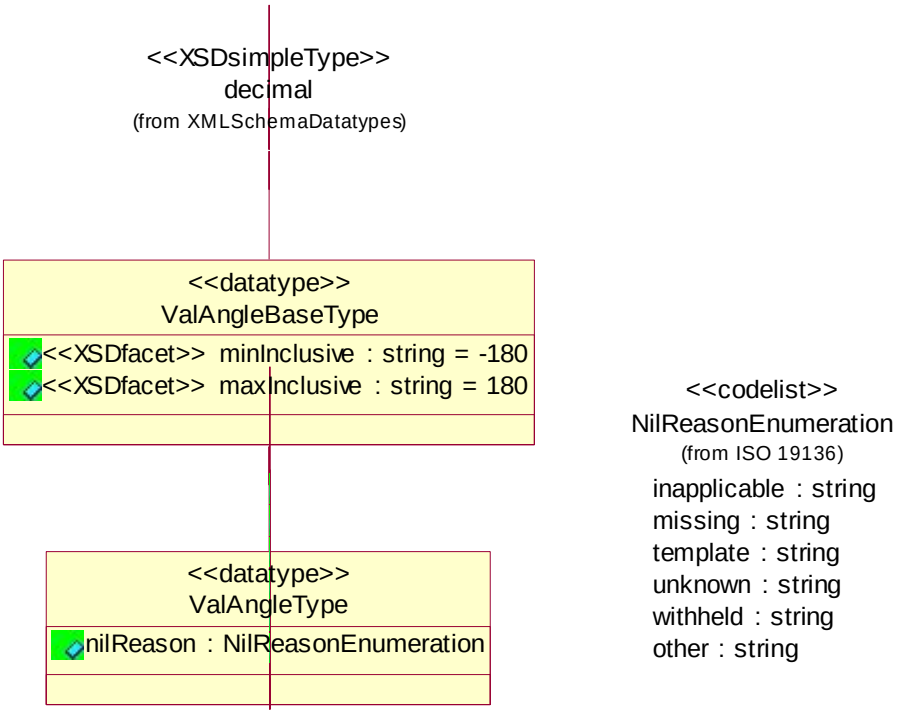
<<codelist>> - Este es el tipo de datos que codifica una lista predefinida de valores. La <<codelist>> incluye el valor OTRO, el cual puede ser expandido utilizando texto libre en mayúsculas (“OTHER:*MY_VALUE*”) para permitir valores no soportados.



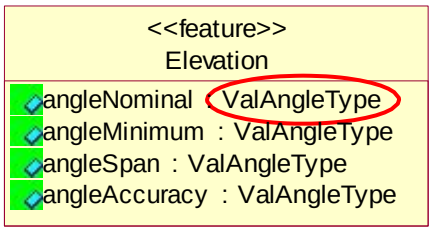
Todos los tipos de datos utilizados para escribir las propiedades AIXM simples definen un nilReason, que se usa para indicar el motivo por el cual existe un valor nulo. Esto se obtiene en el AIXM 5.1 mediante la introducción de:

Un tipo base, el cual contiene la información medular del "negocio", como, por ejemplo, un rango de valores para <<datatype>>, o la lista de valores de sarta para <<codelist>>.

Un tipo de datos derivado, que explícitamente declara el atributo nilReason, y que se utiliza para escribir las propiedades simples AIXM correspondientes.

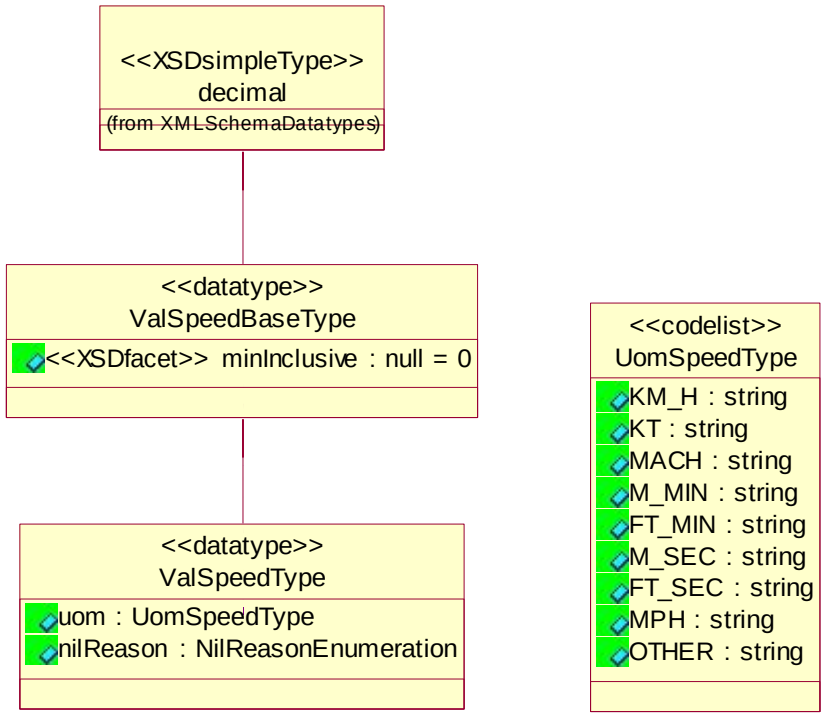


En el ejemplo anterior, el tipo base utilizado para representar un ángulo se denomina ValAngle**BaseType**. Se deriva del decimal y define el rango de valores permitidos para un porcentaje de ángulo ([-180;180]). El tipo de datos derivado ValAngle**Type** es una herencia de ValAngleBaseType e incluye el nilReason, digitado con NilReasonEnumeration. Siempre se utiliza ValAngle**Type** para digitar los porcentajes especificados en los componentes AIXM u objetos AIXM.



Hay un conjunto limitado de tipos de datos definido en el modelo UML AIXM 5.1 que no se utiliza para digitar directamente las propiedades simples AIXM, pero son clases básicas de las que heredan varios tipos de datos AIXM. Estos tipos de datos son: AlphaType, AlphaNumericType, Character1, Character2, Character3. Estos no requieren un atributo nilReason, y, en consecuencia, no se define tipos BaseType correspondientes en el modelo UML AIXM.

Asimismo, ciertos `<<datatype>>` podrían tener una Unidad de Medida asociada. Esto aparece indicado en el modelo con la inclusión de un atributo "uom" al mismo nivel que el atributo nilReason, es decir, en la definición de la clase `<<datatype>>` derivada. Típicamente, el tipo del atributo uom es una clase `<<codelist>>`, tal como se muestra a continuación:



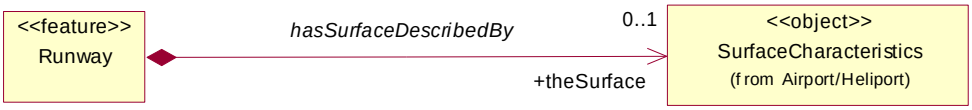
Nótese que los tipos <<codelist>> que representan Unidades de Medida no requieren un nilReason. En consecuencia, no se crea un tipo base para uom.

2.7.2 Relaciones

Cuando una propiedad tiene una multiplicidad mayor a uno, no puede ser descrita en UML con un atributo. En tal caso, la propiedad es descrita utilizando una relación UML que especifique la cardinalidad, la cual es siempre navegable en una y sólo una dirección. El nombre de la propiedad compleja está dado por el nombre del papel que desempeña la clase objetivo.

2.7.2.1 Relaciones con objetos

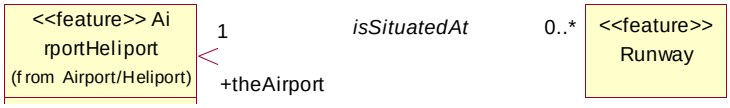
Las relaciones con objetos están representadas por la asociación de composición (*agregación por valor*) UML normalizada. La composición es una forma de agregación donde el todo tiene una fuerte titularidad sobre las partes y una duración de vida coincidente con las mismas. La parte queda suprimida cuando el todo es suprimido.



El ejemplo anterior muestra que el <<componente>> (<<feature>>) Pista tiene una propiedad llamada *theSurface.* Esta propiedad es modelada en UML utilizando una asociación de composición entre el <<feature>> Pista y un objeto que representa las características de una superficie geométrica.

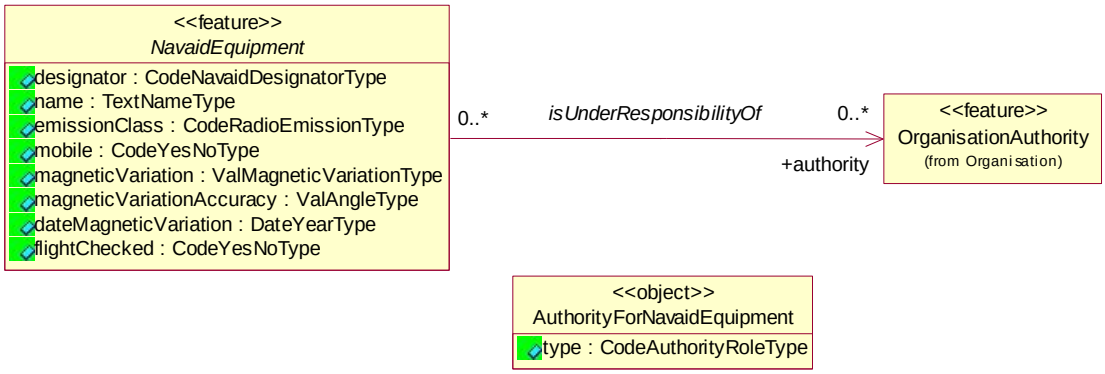
2.7.2.2 Relaciones con componentes

Las relaciones con componentes son descritas mediante una asociación UML normalizada. Todas las asociaciones son navegables sólo en una dirección. Esto muestra que las dos clases están relacionadas, pero que sólo una clase sabe que la relación existe. En el siguiente ejemplo, el componente Pista sabe acerca del AeropuertoHelipuerto, pero el AeropuertoHelipuerto no sabe acerca de la Pista.



2.7.2.3 Clases de asociaciones

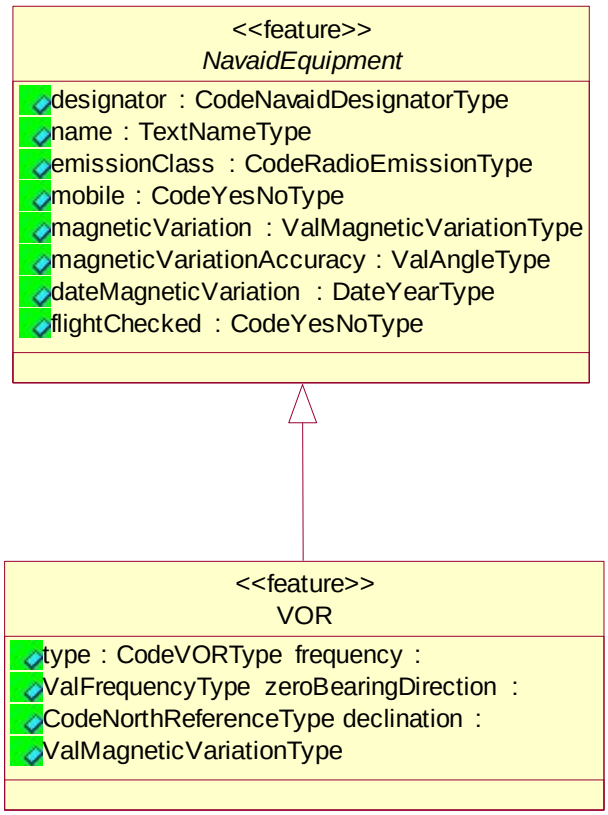
Cuando se necesita información acerca de una relación, se utiliza una clase de asociación UML. La clase de asociación se conecta con la relación mediante una línea punteada.



2.8 Herencia

La herencia se refiere a la capacidad de una clase (la clase especializada o hija) de heredar las propiedades de otra clase (la clase generalizada o progenitora), y luego agregar nuevas propiedades propias. En el AIXM, los Componentes sólo deben heredar de otros Componentes y los Objetos sólo deben heredar de otros Objetos. No se permite múltiples herencias.

En el siguiente ejemplo, el VOR es un tipo de NavaidEquipment (equipo de ayuda para la navegación).



2.9 Denominación

Los nombres de los Componentes, Objetos y Opciones se digitan en UpperCamelCase; por ejemplo, NavaidEquipment.

Los nombres de las propiedades simples (es decir, los atributos) se digitan con las iniciales en mayúsculas, excepto la primera (lowerCamelCase), por ejemplo widthShoulder. Las relaciones se digitan en lowerCamelCase pero como verbos en tiempo presente, por ejemplo, isSituatedAt (estáUbicadoEn). Los nombres del Papel de la Relación también se digitan en lowerCamelCase, y son sustantivos que expresan el papel que desempeña la clase en la asociación.

Los nombres de los tipos de datos se digitan con las iniciales en mayúsculas (UpperCamelCase) y terminan con ‘Type’; por ejemplo, CodeAircraftType.

3 Otros Aspectos del Modelo

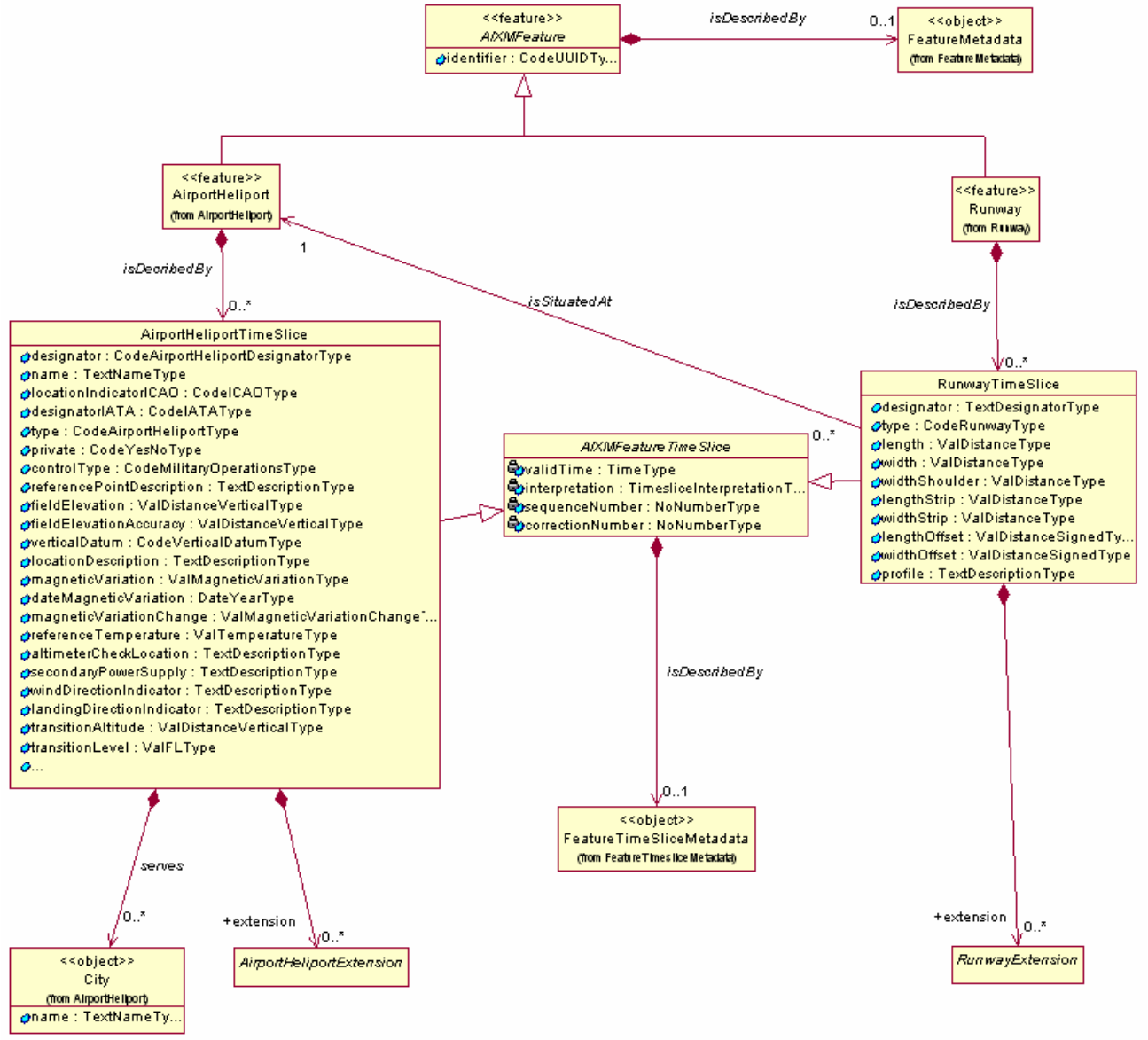
A fin de simplificar el modelo UML, se ha adoptado algunas medidas convenientes. Algunos elementos no aparecen en todos los diagramas, y algunas relaciones son 'asumidas'.

3.1 El Modelo Abstracto

El modelo debería contener un conjunto de clases abstractas AIXM, que son utilizadas como bloques de construcción para el Esquema XML AIXM. No obstante, para fines de simplicidad, estas relaciones no aparecen en cualquier diagrama y, realmente, no existen en el UML. Sólo se asume que existen, al convertir el modelo UML al Esquema XML AIXM.

3.1.1 La clase AIXMFeature y AIXMFeatureTimeSlice

El UML que aparece a continuación muestra cómo todos y cada uno de los `<<feature>>` heredan de la clase abstracta AIXMFeature. Los componentes concretos son descritos por TimeSlices (Fracciones de Tiempo), las cuales están compuestas por propiedades. El TimeSlice hereda de la clase abstracta AIXMFeatureTimeSlice.



El diagrama anterior es bastante complejo. Si se aplica a todo el conjunto de clases AIXM, podría afectar la legibilidad de los diagramas UML. Por lo tanto, el Equipo de Diseño ha

decidido proporcionar un modelo UML simplificado, sin una visible herencia del AIXMFeature en todos los componentes y sin clases visibles SomeFeatureTimeSlice.

No obstante, todas estas relaciones y clases deben ser mapeadas en el esquema XML AIXM.

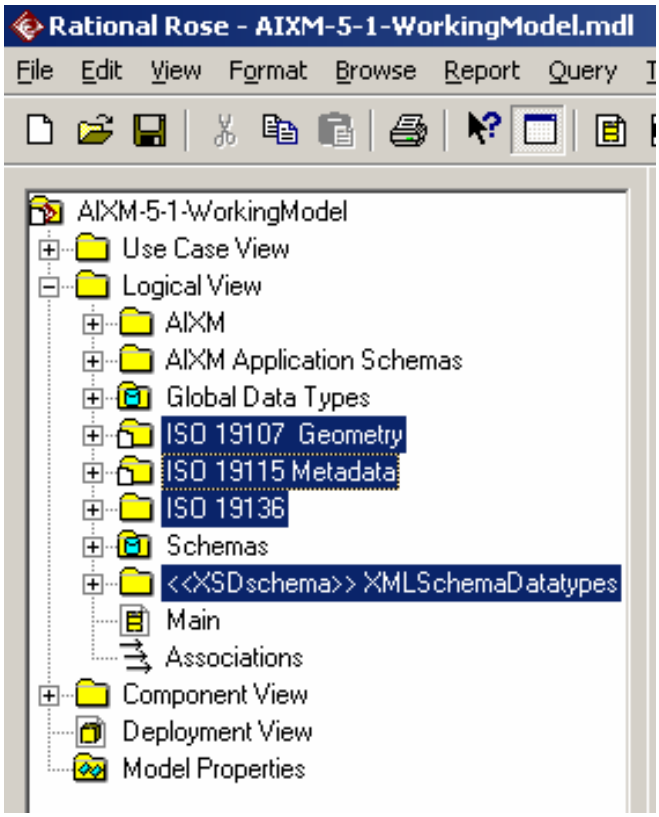
3.1.2 Metadatos

El diagrama también muestra que cada componente AIXM y cada TimeSlice son descritos por metadatos. El esquema XML AIXM incorpora los elementos de metadatos de la ISO 19139 – ver 3.2.2.

3.1.3 Extensión

Finalmente, cada TimeSlice puede contener una Extensión. El mecanismo de Extensión le permite a cada usuario del AIXM5 definir y utilizar sus propios atributos y clases específicos.

3.2 Paquetes externos



3.2.1 <<XSDschema>> XMLSchemaDatatypes

El paquete Datatypes del esquema XSD declara los tipos de datos específicos XSD referenciados por los tipos de datos AIXM al momento de generar el esquema XML (XSD) AIXM. No obstante, estos vínculos XSD no significan que el AIXM es "dependiente" de la especificación del Esquema XML. Los tipos simples XSD predefinidos (como sarta, decimal, unsignedInt, etc.) referenciados por el AIXM son suficientemente genéricos y mapeables a los tipos de datos simples de muchas otras normas de codificación de datos.

3.2.2 Metadatos de la ISO 19115

Este paquete contiene algunas conexiones básicas del modelo AIXM con los elementos de metadatos de la ISO 19115 (MD_Metadata, MD_Constraints …).

3.2.3 Geometría de la ISO 19107

Este paquete contiene algunas conexiones básicas entre el modelo AIXM y los elementos de la geometría de la ISO 19107 (GM_Point, GM_Surface …).

3.2.4 ISO 19136

Este paquete contiene algunas conexiones básicas entre el modelo AIXM y los elementos específicos del GML, que no son parte de la ISO 19107. Prácticamente, el paquete contiene únicamente el tipo de datos NilReasonEnumeration, utilizado para indicar la razón del valor nulo.

4 Mapeo al Esquema XML AIXM

4.1 AIXM - archivos medulares XSD

El formato medular de intercambio AIXM está conformado por tres archivos principales:

AIXM_AbstractGML_ObjectTypes.xsd: el archivo hace referencia al Esquema de Metadatos de la ISO19139 y define los constructos básicos Feature/Object del AIXM.

- AbstractAIXMFeatureType / AbstractAIXMFeature
- AbstractAIXMTimesliceType / AbstractAIXMTimeslice
- AbstractAIXMObjectType
- AbstractAIXMPropertyType, que define la nilReason para todas las propiedades complejas AIXM

AIXM_Datatypes.xsd: Este archivo contiene la representación XML de todos los tipos de datos definidos en el modelo UML AIXM.

AIXM_Features.xsd: Este archivo contiene la representación XML de todos los componentes AIXM con todas sus propiedades (simples y complejas).

Los capítulos aquí contenidos especifican las reglas que rigen el mapeo entre el modelo UML AIXM y el Esquema XML AIXM.

4.2 AIXM es GML

El modelo de intercambio AIXM es una norma de intercambio XML basada en un subconjunto del Lenguaje de Marcado Geográfico (Geography Markup Language - GML). Esencialmente: los Componentes AIXM son componentes GML;

Los Objetos AIXM son objetos GML;

El AIXM aplica el concepto objeto-propiedad GML.

4.3 El Modelo Objeto-Propiedad GML

El modelo objeto-propiedad GML explica parte de la complejidad del mapeo del UML AIXM al XSD. Significa que ningún objeto GML puede aparecer como el hijo inmediato de un objeto GML. Consecuentemente, ningún elemento puede ser, al mismo tiempo, objeto GML y propiedad GML.

El modelo objeto-propiedad prohíbe la codificación de un objeto directamente dentro de un componente; por ejemplo,

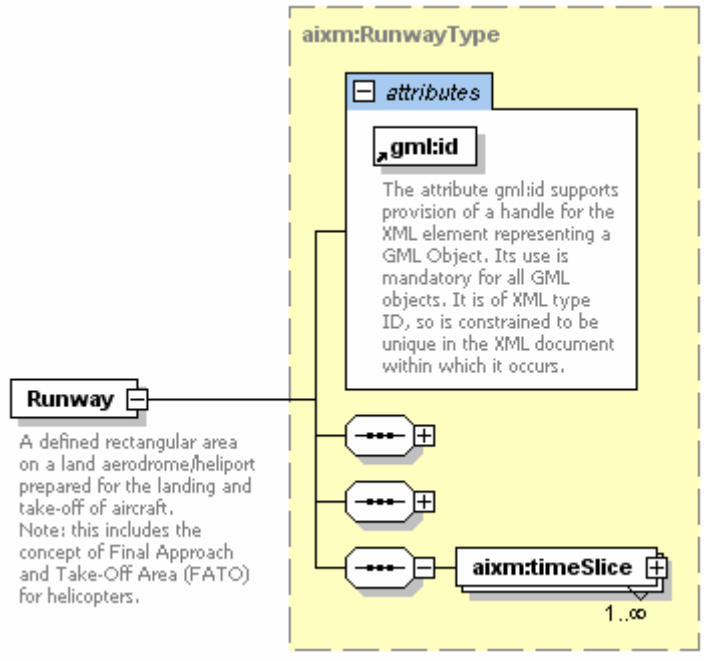
```
<AirportHeliport> <!-- feature -->
 <ElevatedPoint> <!-- object -->
```

Más bien, en un esquema de aplicación GML que cumple con las normas, una asociación entre dos componentes (o entre un componente y un objeto) se realiza sobre la propiedad de un componente; por ejemplo,

```
<AirportHeliport> <!-- feature -->
 <hasReferencePoint> <!-- property -->
  <ElevatedPoint> <!-- object -->
```

La dirección de la flecha de asociación de los diagramas UML (la navegabilidad) determina cuál de los dos socios en la asociación tiene la propiedad que asocia al otro.

En el Esquema XML AIXM, se codifica el modelo objeto-propiedad declarando un tipo, y luego asignándole propiedades (atributos y relaciones) a ese tipo. El tipo define al objeto.



4.4 Mapeo de la Herencia

Dentro del Esquema XML AIXM, la herencia implica dos características:

1. Sustituibilidad. El componente u objeto más general puede ser sustituido por una especialización. En el esquema XML, esto se logra utilizando grupos de sustitución.
2. Herencia de propiedades. El componente especializado hereda todas las propiedades del componente más general. En el esquema XML, esto se logra incluyendo las propiedades de la clase general en la clase especializada.

4.5 Mapeo del nombre de las clases

El nombre de la clase UML es utilizado para los nombres de los elementos en el Esquema XML.

4.6 Mapeo de componentes

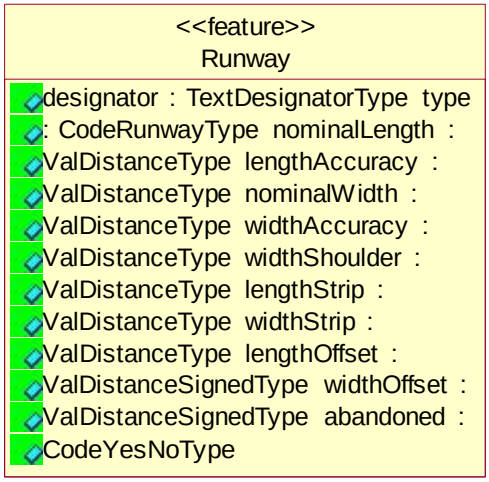
Para cada componente AIXM en el UML, se crea las siguientes entidades del esquema XML:

*Feature*PropertyType
*Feature Feature*Type
*Feature*TimeSlicePropertyType
*Feature*TimeSlice.
*Feature*TimeSliceType
*Feature*PropertyGroup
Abstract*Feature*Extension

La dirección en que los distintos tipos y elementos son usados en la definición del esquema (e.g., Feature usa FeatureType)

4.6.1 Ejemplo de un mapeo

Se utilizará el componente Pista (que se muestra a continuación) para ilustrar el mapeo. El ejemplo se centrará en las propiedades (mostradas como atributos).



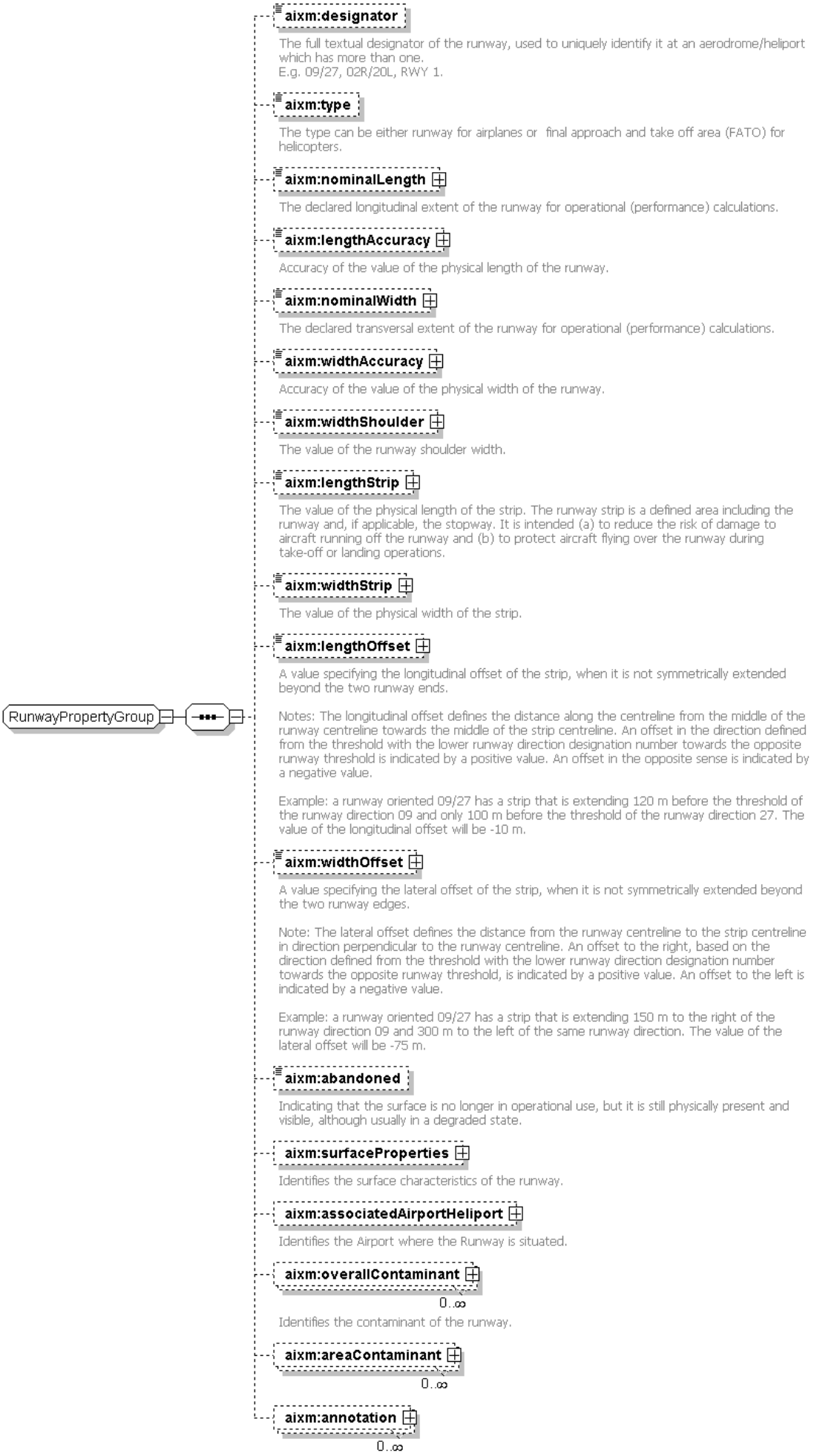
4.6.1.1 RunwayPropertyGroup

Para cada componente, se genera un grupo (XSD) de Esquema XML que contiene todas las propiedades (atributos y relaciones) del componente.

El orden en que se declara los elementos del grupo es el siguiente:

1. (de ser aplicable) sólo en el caso de clases derivadas, primero se inserta el grupo de propiedades de la superclase;
2. luego, todos los elementos que corresponden a los atributos de la clase, en el orden en que aparecen en el diagrama de la clase UML;
3. luego, todos los elementos que corresponden a los nombres de los papeles de asociación, en orden aleatorio;
4. por último, la propiedad de “anotación” – nótese que, para las clases derivadas, sólo se define esta propiedad en la superclase; por lo tanto, aparecerá en el grupo de propiedades de la superclase
   .

A continuación, se presenta un ejemplo de RunwayPropertyGroup en forma gráfica y como extracto del XSD. Muestra claramente cómo se mapea los atributos del UML al XSD, y cómo se crea la relación ‘associatedAirportHeliport’.

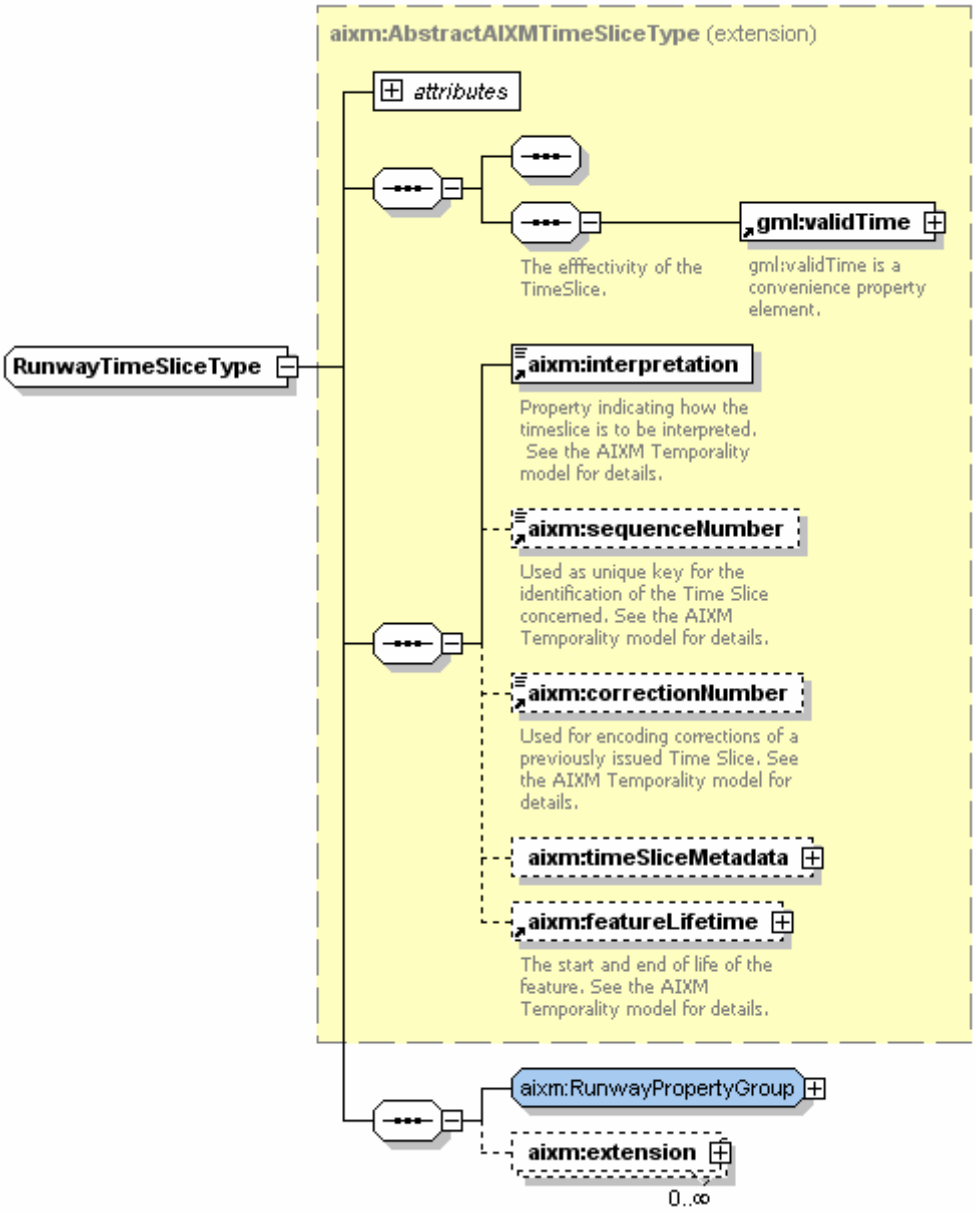

RunwayPropertyGroup
aixm:designator
The full textual designator of the runway, used to uniquely identify it at an aerodrome/heliport which has more than one.
E.g. 09/27, 02R/20L, RWY 1.
aixm:type
The type can be either runway for airplanes or final approach and take off area (FATO) for helicopters.
aixm:nominalLength
The declared longitudinal extent of the runway for operational (performance) calculations.
aixm:lengthAccuracy
Accuracy of the value of the physical length of the runway.
aixm:nominalWidth
The declared transversal extent of the runway for operational (performance) calculations.
aixm:widthAccuracy
Accuracy of the value of the physical width of the runway.
aixm:widthShoulder
The value of the runway shoulder width.
aixm:lengthStrip
The value of the physical length of the strip. The runway strip is a defined area including the runway and, if applicable, the stopway. It is intended (a) to reduce the risk of damage to aircraft running off the runway and (b) to protect aircraft flying over the runway during take-off or landing operations.
aixm:widthStrip
The value of the physical width of the strip.
aixm:lengthOffset
A value specifying the longitudinal offset of the strip, when it is not symmetrically extended beyond the two runway ends.
Notes: The longitudinal offset defines the distance along the centreline from the middle of the runway centreline towards the middle of the strip centreline. An offset in the direction defined from the threshold with the lower runway direction designation number towards the opposite runway threshold is indicated by a positive value. An offset in the opposite sense is indicated by a negative value.
Example: a runway oriented 09/27 has a strip that is extending 120 m before the threshold of the runway direction 09 and only 100 m before the threshold of the runway direction 27. The value of the longitudinal offset will be -10 m.
aixm:widthOffset
A value specifying the lateral offset of the strip, when it is not symmetrically extended beyond the two runway edges.
Note: The lateral offset defines the distance from the runway centreline to the strip centreline in direction perpendicular to the runway centreline. An offset to the right, based on the direction defined from the threshold with the lower runway direction designation number towards the opposite runway threshold, is indicated by a positive value. An offset to the left is indicated by a negative value.
Example: a runway oriented 09/27 has a strip that is extending 150 m to the right of the runway direction 09 and 300 m to the left of the same runway direction. The value of the lateral offset will be -75 m.
aixm:abandoned
Indicating that the surface is no longer in operational use, but it is still physically present and visible, although usually in a degraded state.
aixm:surfaceProperties
Identifies the surface characteristics of the runway.
aixm:associatedAirportHeliport
Identifies the Airport where the Runway is situated.
aixm:overallContaminant
0..∞
Identifies the contaminant of the runway.
aixm:areaContaminant
0..∞
aixm:annotation
0..∞
Generated by XmlSpy
www.altova.com

```
<group name="RunwayPropertyGroup">
   <sequence>
     <element name="designator" type="aixm:TextDesignatorType"
nillable="true" minOccurs="0"/>
     <element name="type" type="aixm:CodeRunwayType" nillable="true"
minOccurs="0"/>
     <element name="nominalLength" type="aixm:ValDistanceType"
nillable="true" minOccurs="0"/>
     <element name="lengthAccuracy" type="aixm:ValDistanceType"
nillable="true" minOccurs="0"/>
     ...
     <element name="associatedAirportHeliport"
type="aixm:AirportHeliportPropertyType" nillable="true" minOccurs="0"/>
     ...
     <element name="areaContaminant" type="aixm:
RunwayContaminationPropertyType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
     <element name="annotation" type="aixm:NotePropertyType" nillable="true"
minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</group>
```

4.6.1.2 RunwayTimeSliceType

Las propiedades de un componente o del objetivo de cualquier relación de componente pueden variar durante la vida del componente. Esta temporalidad puede ser expresada en el GML utilizando componentes dinámicos y colecciones de componentes. La propiedad TimeSlice de un componente dinámico contiene una o más TimeSlices del Componente que capturan la evolución del componente a través del tiempo. Un objeto gml:TimeSlice contiene las propiedades dinámicas del componente.

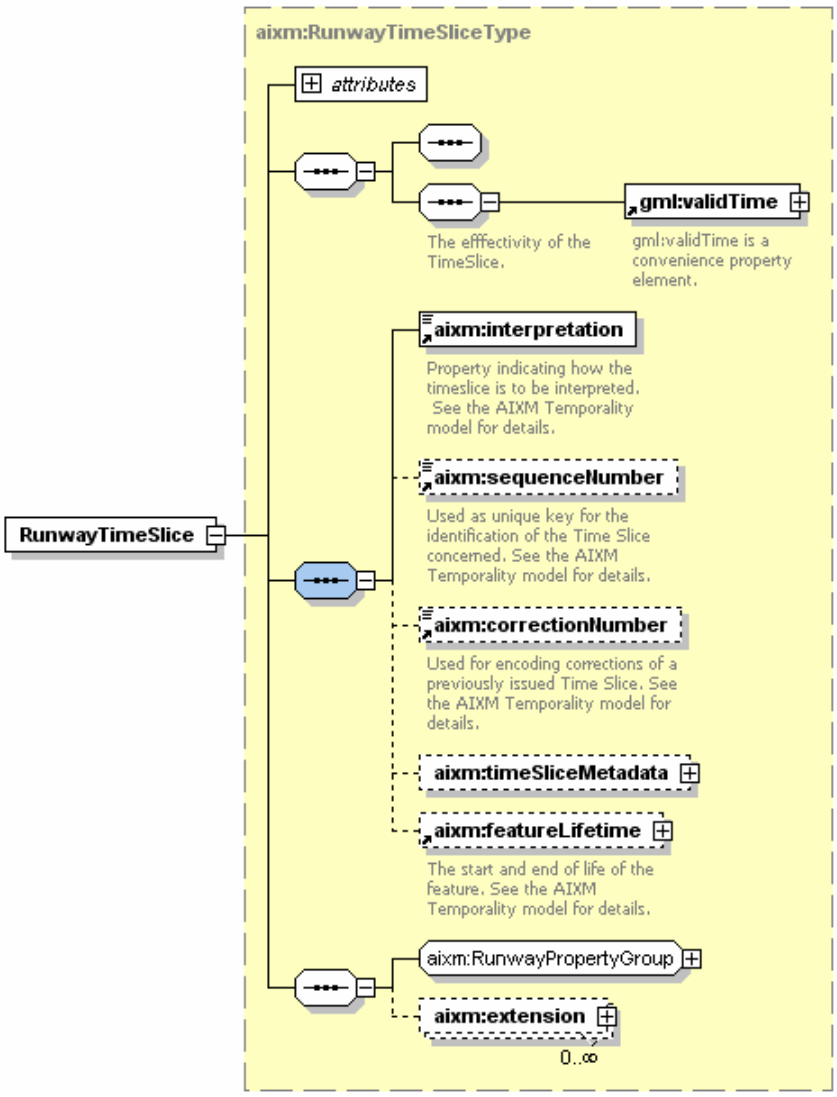
Para cada componente, se crea una propiedad TimeSlice que contiene una gama de objetos TimeSlice del componente. Este ejemplo muestra el RunwayTimeSliceType que encapsula a todas las propiedades de la Pista (RunwayPropertyGroup creado anteriormente) que cambian con el tiempo.




```
<complexType name="RunwayTimeSliceType">
 <complexContent>
  <extension base="aixm:AbstractAIXMTimeSliceType">
   <sequence>
    <group ref="aixm:RunwayPropertyGroup"/>
    <element name="extension" minOccurs="0" maxOccurs="unbounded">
     <complexType>
      <sequence>
       <element ref="aixm:AbstractRunwayExtension"/>
      </sequence>
      <attributeGroup ref="gml:OwnershipAttributeGroup"/>
     </complexType>
    </element>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

4.6.1.3 RunwayTimeSlice

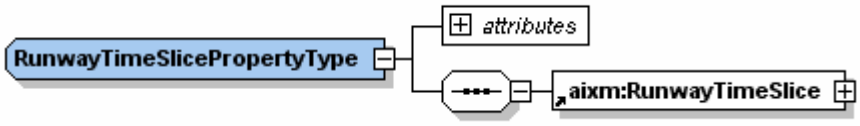
El objeto *Feature*TimeSlice es del tipo *Feature*TimeSliceType. Continuando con el ejemplo, el elemento RunwayTimeSlice es del tipo RunwayTimeSliceType.




```
<element name="RunwayTimeSlice" type="aixm:RunwayTimeSliceType"
substitutionGroup="gml:AbstractTimeSlice"/>
```

4.6.1.4 RunwayTimeSlicePropertyType

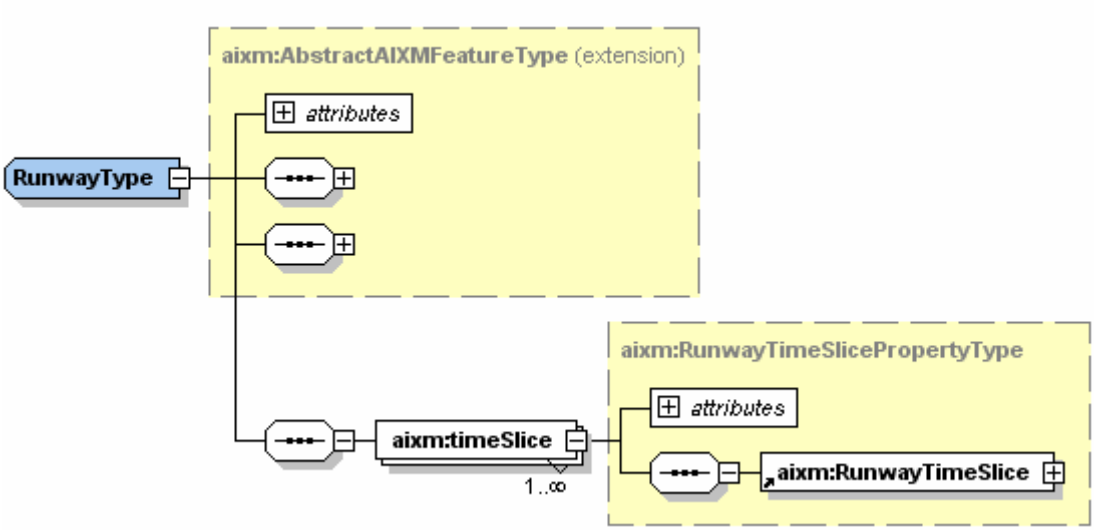
Se crea un tipo de propiedad GML que contiene un objeto *Feature*TimeSlice.




```
<complexType name="RunwayTimeSlicePropertyType">
 <sequence>
  <element ref="aixm:RunwayTimeSlice"/>
 </sequence>
 <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

4.6.1.5 RunwayType

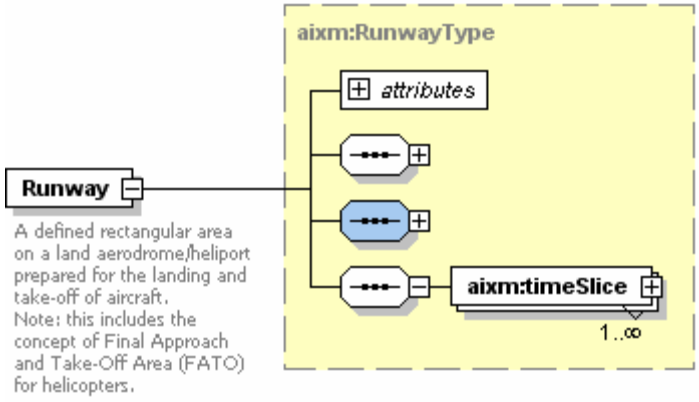
Continuando con el modelo objeto-propiedad, se crea el tipo de componente Pista extendiendo el AbstractAIXMFeatureType con el objeto RunwayTimeSlice creado anteriormente.




```
<complexType name="RunwayType">
 <complexContent>
  <extension base="aixm:AbstractAIXMFeatureType">
   <sequence>
    <element name="timeSlice" type="aixm:RunwayTimeSlicePropertyType"
maxOccurs="unbounded"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

4.6.1.6 Runway (Pista)

A continuación, el componente Runway (Pista) es definido por el RunwayType.




```
<element name="Runway" type="aixm:RunwayType"
substitutionGroup="aixm:AbstractAIXMFeature">
 <annotation>
  <appinfo>RWY</appinfo>
  <appinfo><gml:description>A defined rectangular area on a land
aerodrome/heliport prepared for the landing and take-off of aircraft. Note:
this includes the concept of Final Approach and Take-Off Area (FATO) for
helicopters.</gml:description></appinfo>
 </annotation>
</element>
```

4.6.1.6.1 RunwayPropertyType

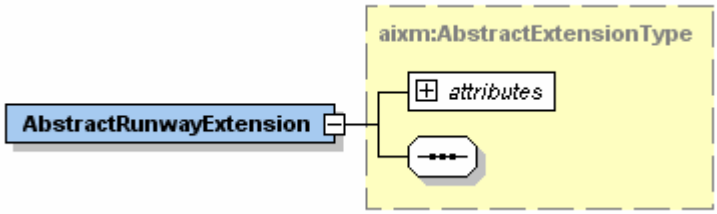
Cuando una propiedad de un componente es una relación, la relación debe estar asociada a otro componente u objeto. Esto se logra con la creación del *Feature*PropertyType, en este caso, el RunwayPropertyType.

En el AIXM, cuando la relación o asociación apunta hacia otro componente, el componente es referenciado utilizando el atributo `xlink:href` (siempre es una codificación "remota"). Cuando la relación apunta a un objeto, el objeto es incluido dentro del progenitor. (Los objetos no pueden existir sin el progenitor.) Debido a que Runway (Pista) es un componente, se crea el RunwayPropertyType con el atributo xlink:href .


```
<complexType name="RunwayPropertyType">
 <attributeGroup ref="gml:OwnershipAttributeGroup"/>
 <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

4.6.1.7 RunwayExtension

Todos los Componentes y Objetos pueden ser extendidos. Se crea una relación con un elemento abstracto XML que actúa como la raíz para todas las extensiones. A continuación, se presenta un ejemplo de la extensión del componente Runway. El elemento AbstractRunwayExtension utiliza el AbstractExtensionType, como se muestra a continuación.




```
<element name="AbstractRunwayExtension" type="aixm:AbstractExtensionType"
abstract="true" substitutionGroup="aixm:AbstractExtension"/>
```

4.7 Mapeo de Objetos

Los objetos AIXM son codificados como objetos GML. En su mayor parte, las entidades del esquema XML son creadas de la misma manera que los Componentes, siguiendo el modelo objeto-propiedad. No obstante, es importante recordar que los objetos AIXM no existen fuera de un componente y, por lo tanto, son parte del TimeSlice del componente. No se crea tipos y elementos TimeSlice para los objetos.

Las siguientes entidades del esquema XML son creadas para cada Objeto AIXM:

*Object*PropertyGroup
*Object Object*Type
*Object*PropertyType
Abstract*Object*Extension

*Object*Type es un tipo complejo que extiende al AbstracAIXMObjectType.

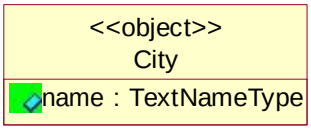
```
<complexType name="NavaidEquipmentDistanceType">
 <complexContent>
  <extension base="aixm:AbstractAIXMObjectType">
   <sequence>
    <group ref="aixm:NavaidEquipmentDistancePropertyGroup"/>
    <element name="extension" minOccurs="0" maxOccurs="unbounded">
     <complexType>
      <sequence>
       <element ref="aixm:AbstractNavaidEquipmentDistanceExtension"/>
      </sequence>
      <attributeGroup ref="gml:OwnershipAttributeGroup"/>
     </complexType>
    </element>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

El tipo *Object*PropertyType es un tipo complejo que extiende al aixm:AbstractAIXMPropertyType.

```
<complexType name="NavaidEquipmentDistancePropertyType">
 <complexContent>
  <extension base="aixm:AbstractAIXMPropertyType">
   <sequence>
    <element ref="aixm:NavaidEquipmentDistance"/>
   </sequence>
   <attributeGroup ref="gml:OwnershipAttributeGroup"/>
  </extension>
 </complexContent>
</complexType>
```

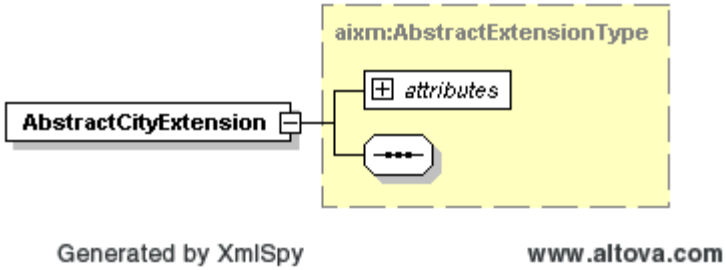
4.7.1 Ejemplo de mapeo

El ejemplo utilizará el objeto City (Ciudad) ilustrado a continuación. El objeto representa la ciudad que es atendida por un AeropuertoHelipuerto.



4.7.1.1 AbstractCityExtension

Un elemento abstracto XML actúa como la raíz para toda extensión del objeto Ciudad. Las extensiones de objetos son definidas de la misma manera que las extensiones de Componentes.

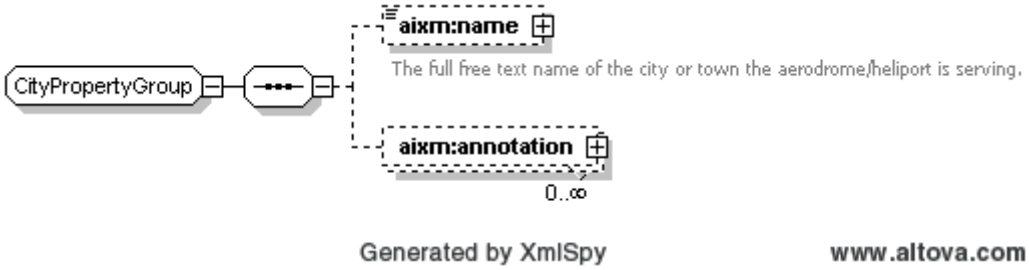



```
<element name="AbstractCityExtension" type="aixm:AbstractExtensionType"
abstract="true" substitutionGroup="aixm:AbstractExtension"/>
```

4.7.1.2 CityPropertyGroup

Se crea un grupo XSD que contiene las propiedades del objeto Ciudad, nuevamente en forma similar a los Componentes

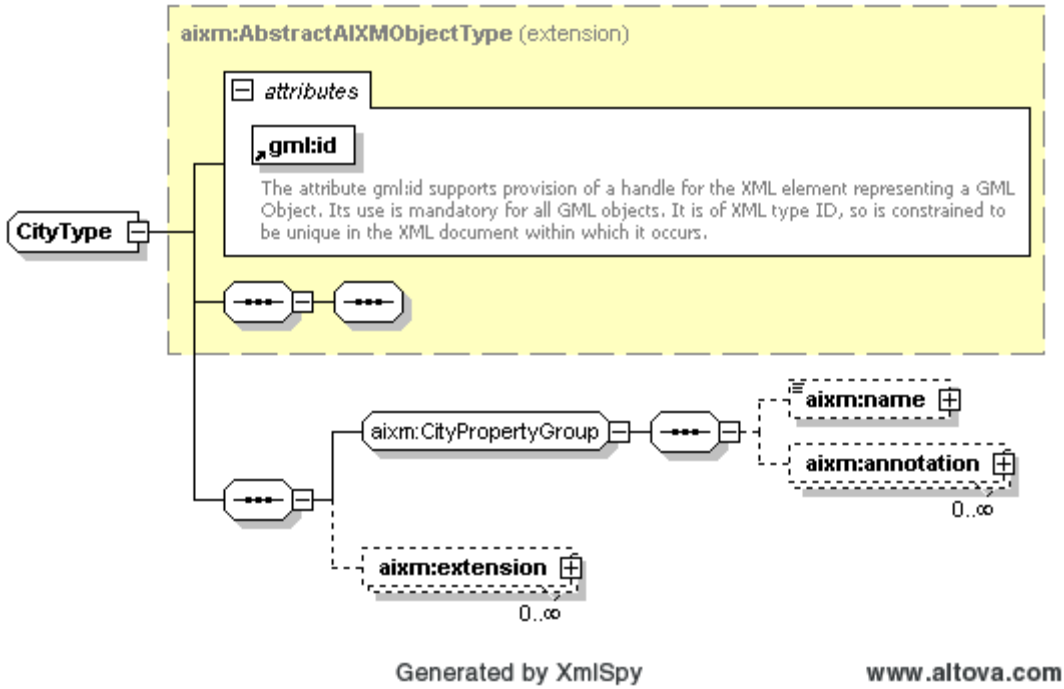
.




```
<group name="CityPropertyGroup">
  <sequence>
    <element name="name" type="aixm:TextNameType" nillable="true"
minOccurs="0">
      <annotation>
        <appinfo>AIXM 4.5</appinfo>
        <appinfo><gml:description>The full free text name of the city or town
the aerodrome/heliport is serving.
        </gml:description></appinfo>
      </annotation>
    </element>
    <element name="annotation" type="aixm:NotePropertyType" nillable="true"
minOccurs="0" maxOccurs="unbounded">
    </element>
  </sequence>
</group>
```

4.7.1.3 CityType

La definición de CityType utiliza el CityPropertyGroup y la extensión, y extiende el AbstractAIXMObjectType

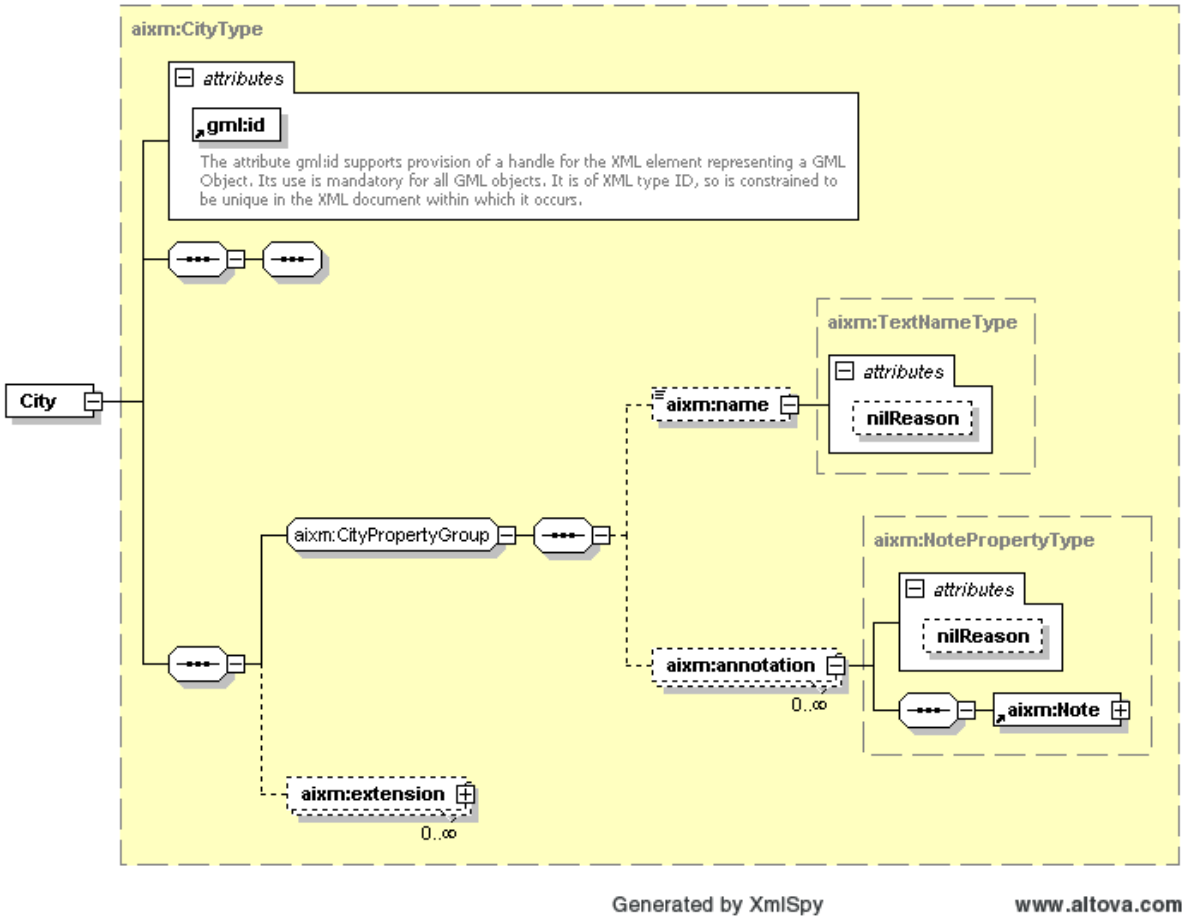



```
  <complexType name="CityType">
    <complexContent>
      <extension base="aixm:AbstractAIXMObjectType">
        <sequence>
```

```
        <group ref="aixm:CityPropertyGroup"/>
        <element name="extension" minOccurs="0" maxOccurs="unbounded">
         <complexType>
          <sequence>
           <element ref="aixm:AbstractCityExtension"/>
          </sequence>
          <attributeGroup ref="gml:OwnershipAttributeGroup"/>
         </complexType>
        </element>
       </sequence>
      </extension>
    </complexContent>
  </complexType>
```

4.7.1.4 City (Ciudad)

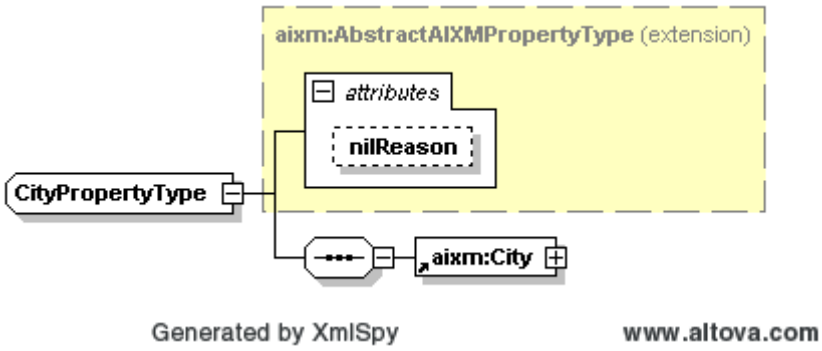
Luego, se define el objeto Ciudad como un elemento XSD del tipo CityType.




```
<element name="City" type="aixm:CityType">
  <annotation>
    <appinfo><gml:description>A city or location that may be served by an
airport/heliport.</gml:description></appinfo>
  </annotation>
</element>
```

4.7.1.5 CityPropertyType

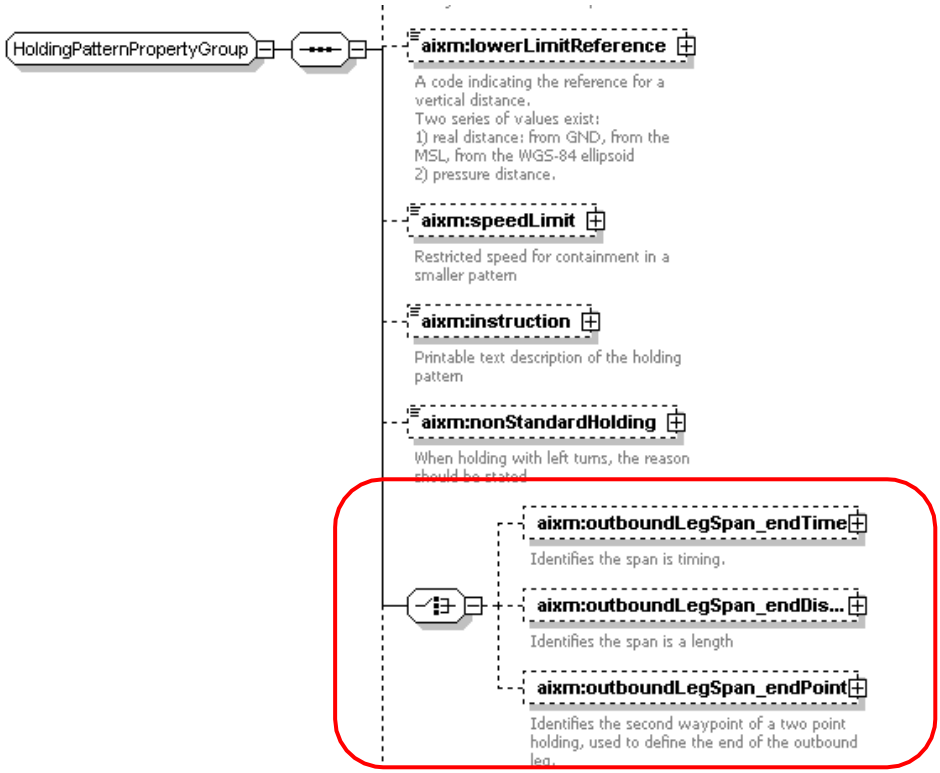
Se crea un tipo complejo XSD que representa un tipo de propiedad GML. Un Componente utiliza este elemento para incluir al objeto *City* (Ciudad) en vez de referenciarlo (using `xlink:href`), ya que el objeto no existe sin el progenitor.




```
<complexType name="CityPropertyType">
  <complexContent>
    <extension base="aixm:AbstractAIXMObjectType">
      <sequence>
        <element ref="aixm:City"/>
      </sequence>
      <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </extension>
  </complexContent>
</complexType>
```

4.8 Mapeo de opciones

Las clases marcadas con el estereotipo <<choice>> (opción) no aparecen en el Esquema XML. Más bien, se crea la opción de elementos.



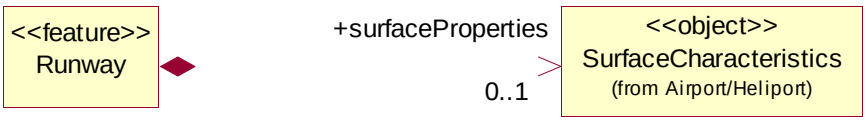
El nombre del elemento es la concatenación del papel de la clase <<choice>> con el papel que desempeña la clase objetivo de cada rama de opción, separados por "_".

```
<group name="HoldingPatternPropertyGroup">
  <sequence>
    ............
     <element name="nonStandardHolding" type="aixm:CodeYesNoType"
nillable="true" minOccurs="0">
```

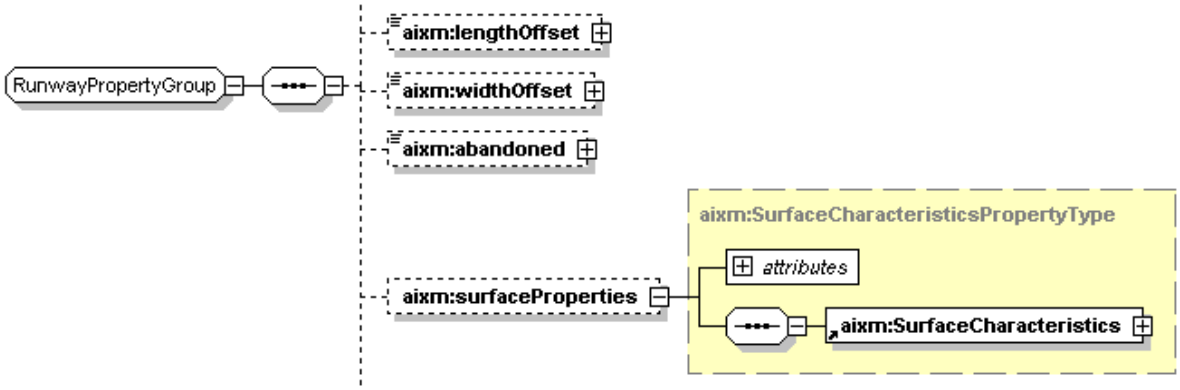
```
      <annotation>
        <appinfo>
          <gml:description>ndicates whether the HoldingPattern is non-
standard, for example because it uses left-hand turns.</gml:description>
        </appinfo>
      </annotation>
    </element>
    <choice>
      <element name="outboundLegSpan_endTime"
type="aixm:HoldingPatternDurationPropertyType" nillable="true"
minOccurs="0">
        <annotation>
          <appinfo>
            <gml:description>Span is timing</gml:description>
          </appinfo>
        </annotation>
      </element>
      <element name="outboundLegSpan_endDistance"
type="aixm:HoldingPatternDistancePropertyType" nillable="true"
minOccurs="0">
        <annotation>
          <appinfo>
            <gml:description>span is length</gml:description>
          </appinfo>
        </annotation>
      </element>
      <element name="outboundLegSpan_endPoint"
type="aixm:SegmentPointPropertyType" nillable="true" minOccurs="0">
        <annotation>
          <appinfo>
            <gml:description>The second waypoint of a two point holding, used
to define the end of the outbound leg.</gml:description>
          </appinfo>
        </annotation>
      </element>
    </choice>
   ............
  </sequence>
</group>
```

4.9 Mapeo de las relaciones con objetos

Las relaciones son codificadas mediante la creación de un elemento XML con el mismo nombre que el nombre del papel en el modelo UML. Es del tipo *Object*PropertyType.



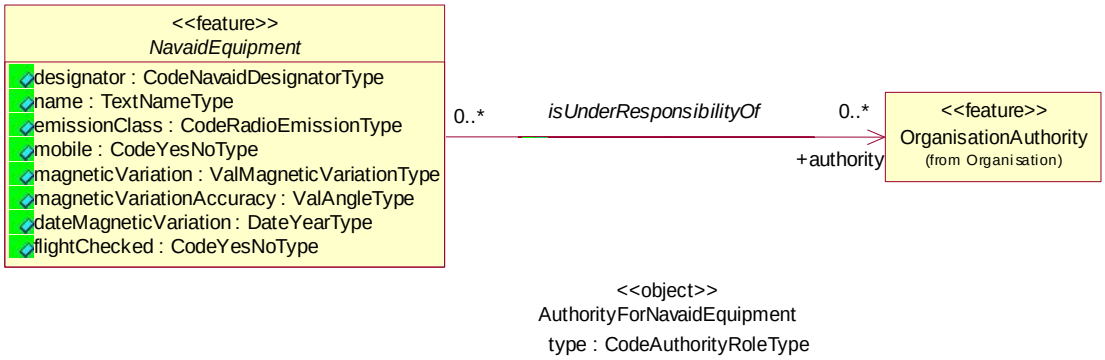
En este ejemplo, el objeto SurfaceCharacterisitcs es una propiedad de la Pista (Runway). La propiedad “surfaceProperties” de la Pista es definida como del tipo SurfaceCharacteristicsPropertyType.



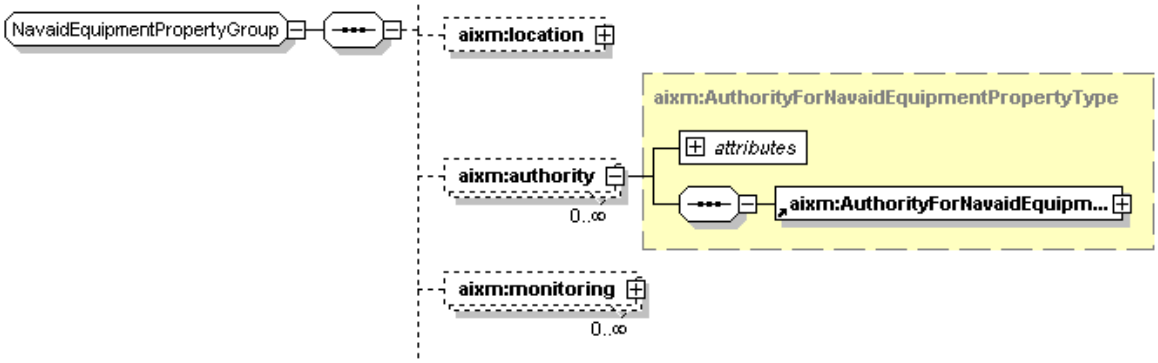

```
<element name="surfaceProperties"
type="aixm:SurfaceCharacteristicsPropertyType" minOccurs="0"/>
```

4.9.1 Mapeo de asociaciones con clases de asociaciones

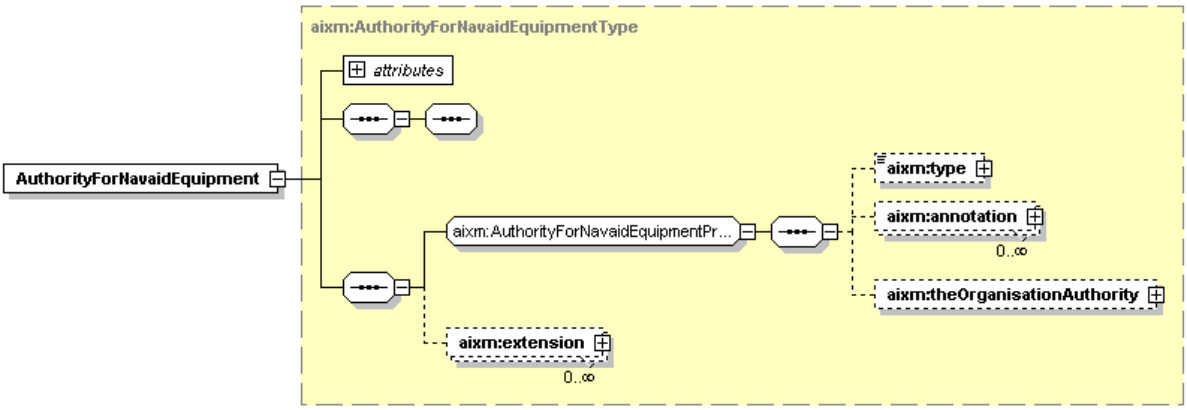
En el UML que aparece a continuación, el componente NavaidEquipment tiene una relación con el componente OrganisationAuthority feature. Esta relación contiene propiedades definidas en la clase AuthorityForNavaidEquipment.



Cuando se mapea esto en el XSD, se crea una propiedad 'authorityForNavaidEquipment' en el componente NavaidEquipment, tal como se muestra a continuación. El nombre de esta propiedad es derivado automáticamente del nombre de la clase de asociación, mediante la conversión al estilo lowerCamelCase. La dirección de la flecha es importante. Si la dirección hubiera apuntado hacia el NavaidEquipment, se hubiera creado la propiedad en el componente OrganisationAuthority.

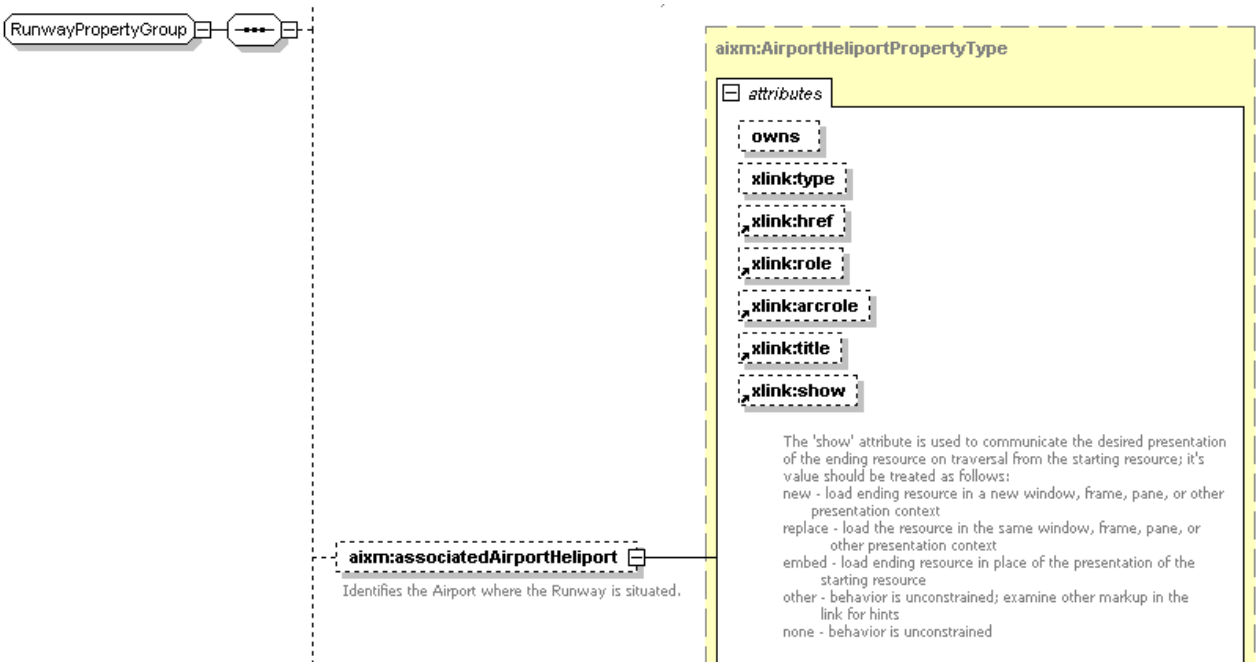


Luego, se requiere un segundo paso para completar el XSD. En este caso, se agrega un elemento llamado 'theOrganisationAuthority' en la definición del AuthorityForNavaidEquipmentPropertyGroup, en base al papel que desempeña la clase OrganisationAuthority en esta asociación.



4.10 Mapeo de las relaciones con componentes

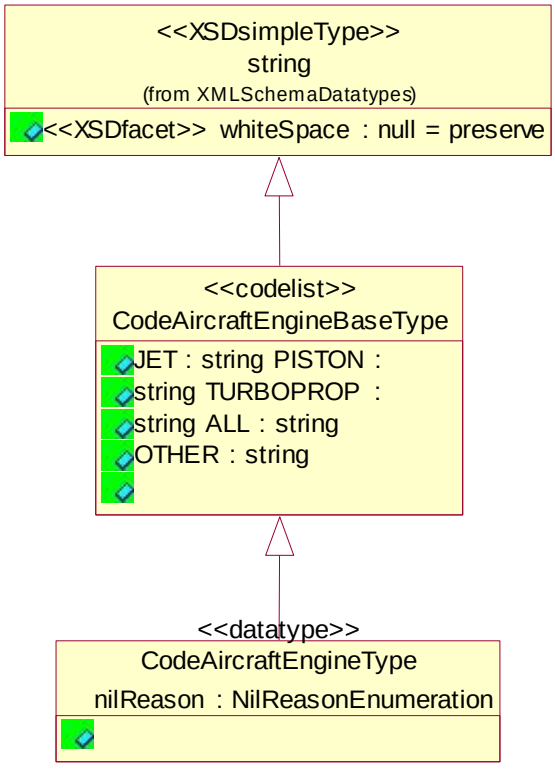
En el AIXM, las relaciones con los componentes son descritas por referencia, utilizando `xlink:href`. Se utiliza el nombre del papel UML para el nombre del elemento XML, y el elemento XML es del tipo *Feature*PropertyType.



4.11 Mapeo de los tipos de datos

4.11.1 <<codelist>>

Los codelists (listas de códigos) están dados por el estereotipo `<<codelist>>`. Como se puede observar en el diagrama a continuación, por cada tipo de <<codelist>>, también hay una clase <<datatype>>, la cual define el atributo nilReason.



Primero, se convierte la clase <<codelist>> en un simpleType en el XSD:

```
<simpleType name="CodeAircraftEngineBaseType">
   <annotation>
     <appinfo><gml:description>A code indicating the type of aircraft engine
(for example, jet, piston, turbo).</gml:description></appinfo>
   </annotation>
   <union>
     <simpleType>
      <restriction base="xsd:string">
       <enumeration value="JET">
         <annotation>
          <appinfo><gml:description>Jet Engine</gml:description></appinfo>
         </annotation>
       </enumeration>
       <enumeration value="PISTON">
         <annotation>
          <appinfo><gml:description>Piston
Engine</gml:description></appinfo>
         </annotation>
       </enumeration>
       <enumeration value="TURBOPROP">
         <annotation>
          <appinfo><gml:description>Turbo Propeller
Engine</gml:description></appinfo>
         </annotation>
       </enumeration>
```

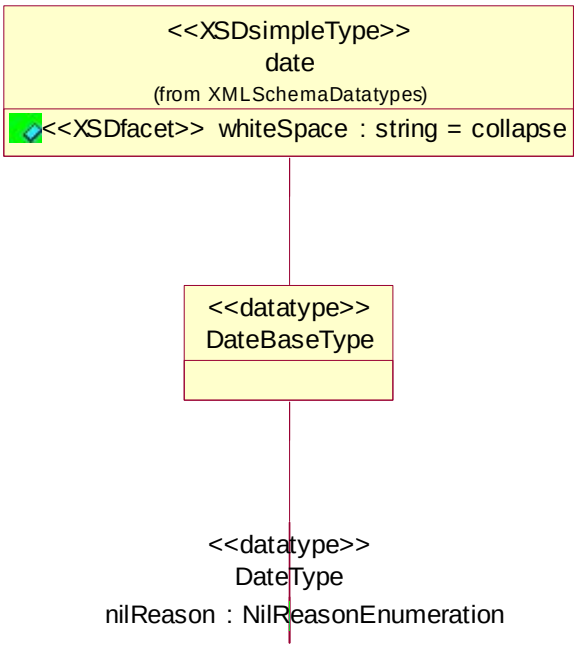
```
        <enumeration value="ALL">
          <annotation>
            <appinfo><gml:description>All aircraft engine
types.</gml:description></appinfo>
          </annotation>
        </enumeration>
      </restriction>
    </simpleType>
    <simpleType>
      <restriction base="string">
        <pattern value="OTHER(:(\w|_){1,58})?"/>
      </restriction>
    </simpleType>
  </union>
 </simpleType>
```

Nótese que los tipos de datos simples están declarados como la unión entre los valores enumerados, declarados en el modelo UML (con excepción del valor “OTHER”) y una sarta con el patrón “OTHER(:(\w|_){1,58})?”. Esto permite que los tipos de datos del <<codelist>> incluyan valores que no tienen el respaldo de la lista de enumeración. Por ejemplo, un tipo de motor eléctrico podría ser codificado como “OTHER:ELECTRIC”.

Asimismo, se define un tipo complejo, que incluye la declaración del atributo nilReason:

```
 <complexType name="CodeAircraftEngineType">
  <simpleContent>
   <extension base="aixm:CodeAircraftEngineBaseType">
    <attribute name="nilReason" type="gml:NilReasonEnumeration"/>
   </extension>
  </simpleContent>
 </complexType>
```

4.11.2 <<datatype>> - caso por defecto



En cuanto al <<codeList>>, el mapeo del <<datatype>> utilizado para digitar propiedades simples (ver 2.7.1.1) consta de dos pasos.

El primer paso es la creación del simpleType correspondiente al BaseType.

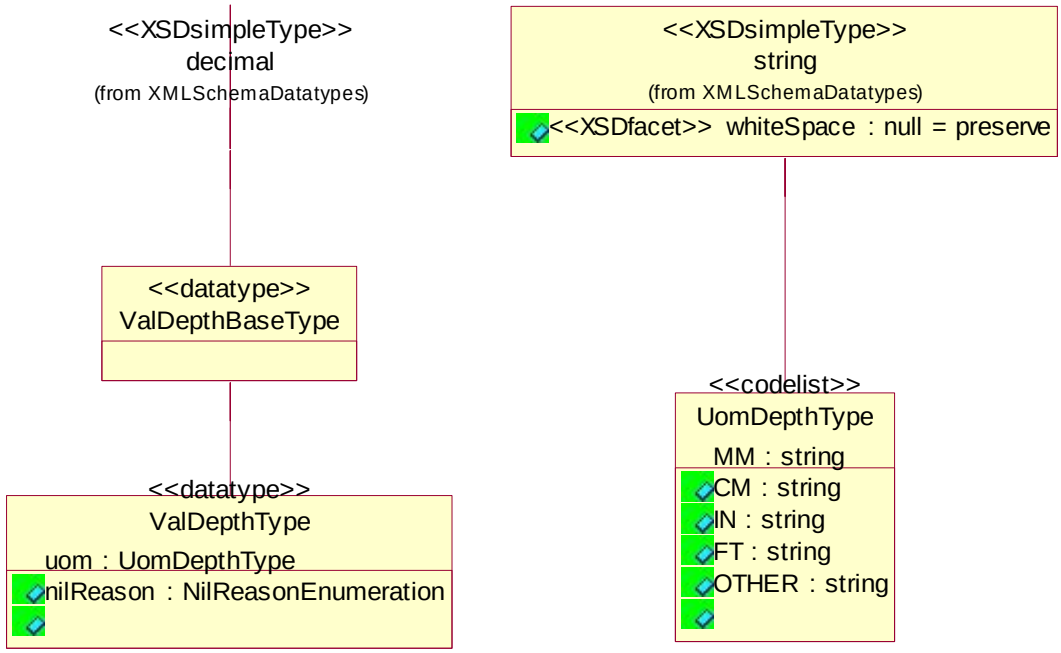
```
<simpleType name="DateBaseType">
  <restriction base="xsd:date">
  </restriction>
</simpleType>
```

El segundo paso es la creación del complexType que define al atributo nilReason.

```
  <complexType name="DateType">
    <simpleContent>
      <extension base="aixm:DateBaseType">
        <attribute name="nilReason" type="gml:NilReasonEnumeration "/>
      </extension>
    </simpleContent>
  </complexType>
```

4.11.3 <<datatype>> con Unidad de Medida

Existe una Unidad de Medida (UOM) para muchos tipos de datos que adoptan valores numéricos. Esto ha sido modelado como un atributo uom en la clase <<datatype>>.



El mapeo XSD de los tipos de uom sigue las mismas reglas que con cualquier otro <<codelist>>, excepto que no se necesita ningún tipo complejo con el nilReason.

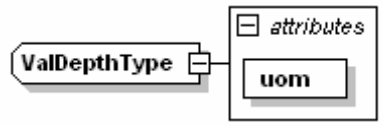
```
  <simpleType name="UomDepthType">
    <union>
      <simpleType>
        <restriction base="xsd:string">
          <enumeration value="MM">
          </enumeration>
          <enumeration value="CM">
          </enumeration>
          <enumeration value="IN">
          </enumeration>
          <enumeration value="FT">
          </enumeration>
        </restriction>
      </simpleType>
      <simpleType>
```

```
      <restriction base="string">
       <pattern value="OTHER:\w{2,58}"/>
      </restriction>
     </simpleType>
    </union>
  </simpleType>
```

En un segundo paso, se genera la clase ValDepthBaseType como un tipo simple, tal como se describe en 4.11.2.

```
<simpleType name="ValDepthBaseType">
  <restriction base="xsd:decimal"/>
</simpleType>
```

Luego, se agrega el atributo uom al complexType ValDepthType, luego de la definición del atributo nilReason.



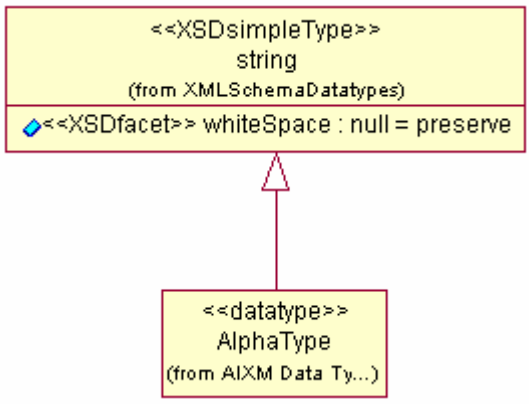

```
<complexType name="ValDepthType">
  <simpleContent>
    <extension base="aixm:ValDepthBaseType">
      <attribute name="nilReason" type="gml:NilReasonEnumeration"/>
      <attribute name="uom" type="aixm:UomDepthType" use="required"/>
    </extension>
  </simpleContent>
</complexType>
```

4.11.4 Casos particulares

4.11.4.1 <<datatype>> sin BaseType

Los 5 tipos de datos enumerados en 2.7.1.1 se mapean directamente a los datatypes (tipos de datos) integrados, definidos por la especificación del esquema XML. Los tipos de datos por defecto son `string`, `float`, `double`, etc, que son considerados como `simpleTypes`.

El AlphaType es un ejemplo apropiado.




```
<simpleType name="AlphaType">
 <restriction base="xsd:string">
  <pattern value="[A-Z]*"/>
 </restriction>
</simpleType>
```

4.11.4.2 <<datatype>> XHTMLBaseType

<<datatype>> XHTMLBaseType representa un documento XHTML estructurado que cumple con http://www.w3.org/1999/xhtml. Debería ser mapeado en el XML de la siguiente manera:

```
<complexType name="XHTMLBaseType">
  <sequence>
    <any namespace="http://www.w3.org/1999/xhtml" minOccurs="1"
maxOccurs="unbounded" processContents="skip"/>
  </sequence>
</complexType>
```

APÉNDICE D

AIXM

GENERACIÓN DEL ESQUEMA DE APLICACIÓN AIXM

AIXM

Generación del Esquema de Aplicación AIXM

Modelo de Intercambio de Información Aeronáutica (AIXM)

Para cualquier consulta, sírvase ponerse en contacto con:

Brett BRUNK - brett.brunk@faa.gov

Eduard POROSNICU - eduard.porosnicu@eurocontrol.int

Versión No.	Fecha de emisión	Autor	Razón del cambio
0.1	2007/12/20	Barb Cordell / Paul Heffley	Primera edición
0.2	2008/01/08	Barb Cordell / Paul Heffley	Versión actualizada
1.0	2008/03/10	Eddy Porosnicu	Primera versión pública Modificaciones editoriales
1.1	2010/02/04	Hubert Lepori	Versión actualizada - AIXM 5.1

INDICE

1 Alcance

1.1 Introducción

La finalidad de este documento es describir de qué manera se puede extender el modelo UML AIXM para apoyar las necesidades de una determinada Comunidad de Interés (COI):

Definir los mensajes que son necesarios y, eventualmente, restringir el contenido de estos mensajes a un sub-conjunto de componentes AIXM;

Extender los componentes AIXM existentes con nuevos atributos o asociaciones, o definir nuevos componentes que sean pertinentes únicamente para la comunidad.

La conversión de UML al esquema XML para extensiones aparece ilustrada en una serie de ejemplos de las extensiones de Esquemas de Aplicación AIXM 5.

1.2 Antecedentes

El modelo conceptual y la norma de datos AIXM son mantenidos como un modelo UML. El AIXM fue desarrollado para ser extensible, permitiendo una mayor flexibilidad para su uso a nivel internacional. Cada componente y lista de códigos puede ser extendido para satisfacer las necesidades individuales de la Comunidad de Interés (COI) AIXM.

Si no está familiarizado con el documento de mapeo del UML AIXM al XSD AIXM, por favor revíselo antes de leer este documento. Una buena comprensión del documento de mapeo ayudará a entender las extensiones AIXM.

Los componentes describen las entidades del mundo real y son fundamentales en el AIXM. Los componentes del AIXM pueden ser concretos y tangibles, o abstractos y conceptuales, y pueden cambiar con el tiempo [7]. Los componentes se representan como clases con un estereotipo `<<feature>>`. Algunos ejemplos son Pista y AeropuertoHelipuerto.

Los componentes del AIXM son componentes dinámicos. Los objetos de Fracción de Tiempo *(Timeslice)* son utilizados para describir los cambios que afectan al componente AIXM a través del tiempo. Los objetos de Fracción de tiempo y la temporalidad son materia de discusión en un documento por separado sobre la Temporalidad en el AIXM.

Los objetos son abstracciones de las entidades del mundo real o, más frecuentemente, de las propiedades de dichas entidades, que no existen fuera de un componente. Un objeto es creado por dos motivos en el AIXM:

Cuando una propiedad tiene una multiplicidad mayor a uno (como la ciudad atendida por un AeropuertoHelipuerto), o

El objeto tiene sus propios atributos que son reutilizados a través del modelo, como un PuntoElevado (ElevatedPoint).

Algunas clases están marcadas como `<<choice>>`. Estas son utilizadas para modelar las relaciones XOR. Por ejemplo, la proyección horizontal de un VolumendeEspacioAéreo (AirspaceVolume) puede ser una Superficie (Surface), un CorredordeEspacioAéreo (AirspaceCorridor) o una forma igual a la de otro EspacioAéreo (Airspace).

Las propiedades son los atributos y relaciones que caracterizan a un componente u objeto. En el UML:

Los atributos son utilizados para describir las propiedades simples de un componente u objeto; las relaciones son utilizadas para describir las asociaciones con componentes u objetos. Cuando una propiedad tiene una multiplicidad mayor a uno, es descrita utilizando una relación UML con cardinalidad.

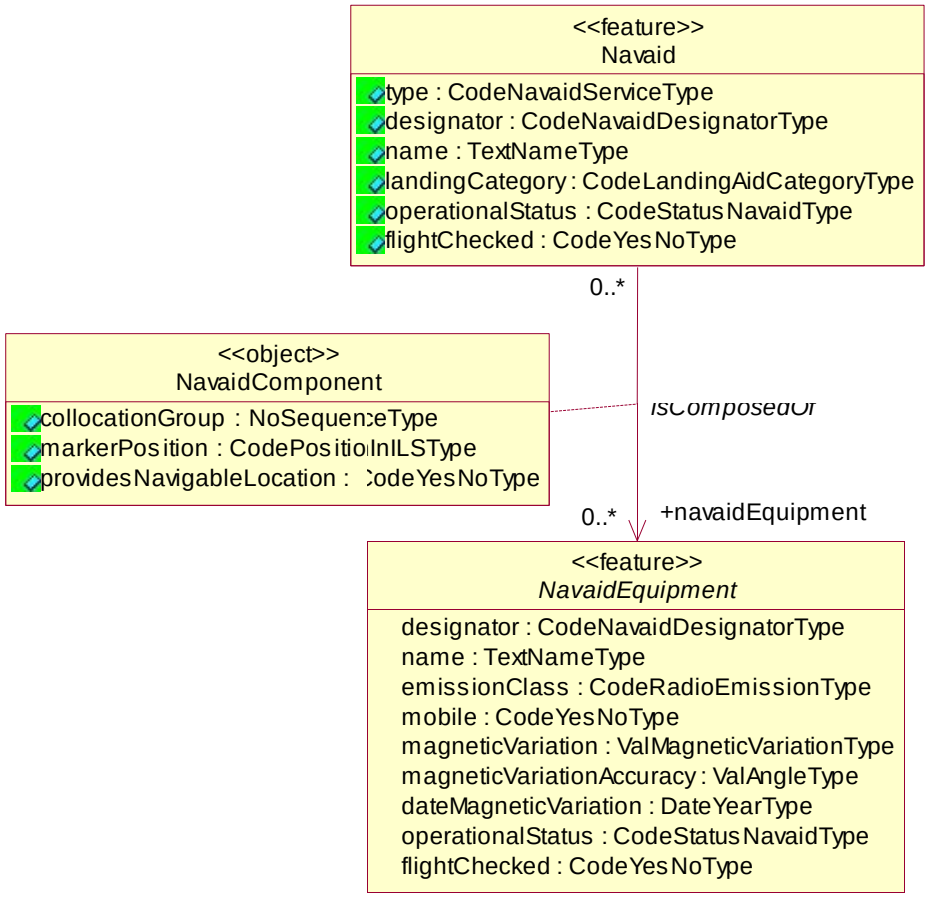
- Las relaciones con los objetos están representadas por la asociación normalizada de la composición UML (*agregación por valor).* La composición es una forma de agregación donde el todo tiene una fuerte titularidad y una vida coincidente con respecto a las partes. La parte queda suprimida cuando se suprime el todo.
- Las relaciones con los componentes son descritas mediante una asociación normalizada UML. Todas las asociaciones son navegables en una sola dirección. Esto demuestra que las dos clases están relacionadas, pero sólo una clase sabe que la relación existe.

El modelo UML enumera los tipos de datos utilizados en el AIXM. A estos se les da uno de los dos siguientes estereotipos:

<<datatype>> - Es el tipo de datos básicos, que especifica el patrón a ser utilizado.

<<codelist>> - Es un tipo de dato que codifica una lista de valores predefinida. El <<codelist>> incluye el valor OTRO (OTHER), que puede ser expandido con algo de texto libre en mayúsculas ("OTHER:*MY_VALUE*") para apoyar los valores que no son soportados.

Cuando se requiere información sobre una relación, se utiliza una clase de asociación UML. Esta clase de asociación se conecta a la relación con una línea de Clase de Asociación *(Association Class).*



La herencia se refiere a la capacidad de una clase (la clase especializada o clase hija) de heredar las propiedades de otra clase (la clase generalizada o clase madre), y luego agregarle nuevas propiedades propias. En el AIXM, los Componentes sólo deben heredar de otros Componentes, y los Objetos sólo pueden heredar de otros Objetos. No se permite una herencia múltiple.

Nota importante**: Sólo se permite la herencia desde clases "Abstractas".** Los guiones de UML a XSD no permiten la herencia de clases no abstractas. El mecanismo de "extensión"

que se explica en este documento brinda la oportunidad de extender una clase AIXM existente agregándole nuevas propiedades, con la ventaja que la clase extendida permanece vigente frente al esquema medular AIXM.

1.3 Objetivo

El modelo medular AIXM define los componentes normalizados de la información aeronáutica. A fin de utilizar el AIXM para una aplicación específica, una Comunidad de Interés (COI) deberá ponerse de acuerdo en cuanto a cómo se habrá de intercambiar y comunicar las instancias de los componentes del AIXM en la comunidad. Esto se puede lograr utilizando ya sea un servicio web predefinido (como el WFS), que permite la provisión directa de componentes individuales AIXM o colecciones de componentes AIXM, o definiendo mensajes personalizados AIXM con propiedades específicas y abarcado una lista selecta de componentes AIXM.

En la definición de los Esquemas de Aplicación AIXM, la COI podría también desear extender el AIXM medular con propiedades y componentes adicionales. Algunos principios que regular dichas extensiones incluyen:

> Una extensión de un componente AIXM existente debería permanecer vigente frente a la definición del elemento XSD AIXM medular con el mismo nombre (para ello, se inserta el elemento AbstractSomeFeatureExtension en el XSD AIXM medular). Una consecuencia es que no es posible extender las clases <<datatype>>. Sólo se puede extender los <<codelist>>.
>
> Un componente y objetos adicionales deberán seguir las convenciones de modelado del AIXM medular (estereotipos, denominación, tipos de datos, etc.)

Nota importante: Es responsabilidad de la COI asegurarse que las extensiones no dupliquen componentes y propiedades ya existentes en el modelo medular. Una vez definidas dichas extensiones, la COI podría compartirlas con la comunidad AIXM a nivel mundial. Para ello, el esquema de aplicación puede ponerse a disposición del público en www.aixm.aero.

1.4 Referencias

1. Geographic Information – Spatial Schema. ISO 19107. Primera edición, 2003-05-01
2. ISO 19136:2007 - Geographic information -- Geography Markup Language (GML)
3. UML 2.0 In a Nutshell. Dan Pilone. O'Reilly Media Inc. 2005.
4. AIXM UML to XML Schema Mapping, www.aixm.aero (ver descargas)
5. AIXM Temporality Model, www.aixm.aero (ver descargas)

2 Extendiendo los componentes/objetos AIXM

2.1 Paquete UML para extensiones

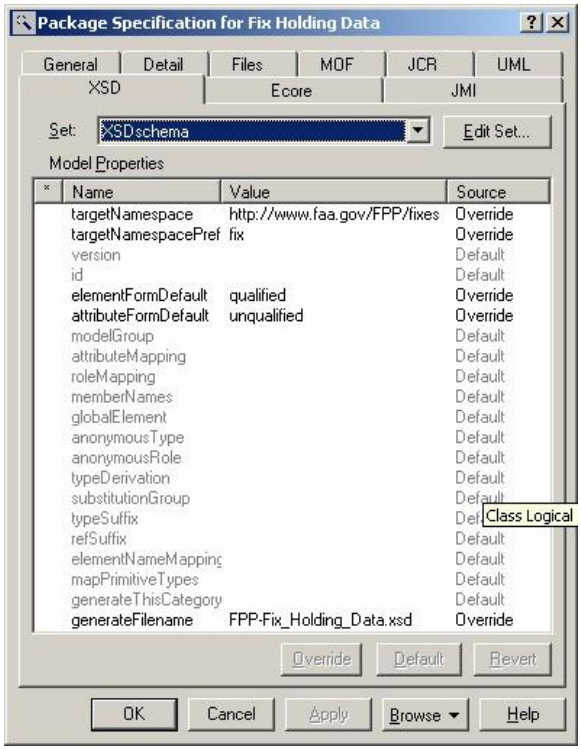
Para extender el AIXM, es necesario crear un nuevo paquete dentro del paquete de Esquemas de Aplicación AIXM. Este paquete contendrá toda la información necesaria para la extensión.

2.1.1 Estructura del paquete

Se utiliza distintos tipos de sub-paquetes para controlar la generación de los esquemas XML apropiados (XSD). El sub-paquete Extensión contiene las extensiones de los componentes y objetos medulares AIXM. Si la extensión requiere nuevos tipos de datos, entonces se crea un segundo sub-paquete, los Tipos de Datos de Extensión, que contiene cualesquiera nuevos tipo de dato y listas de códigos que se requieran. Los sub-paquetes finales que se requieren son los paquetes de mensajes. Puede que se requiera múltiples paquetes, dependiendo de la cantidad de distintos esquemas de mensajes requeridos. La mayoría de los Paquetes de Esquemas de Aplicación tendrán por lo menos un paquete de solicitud y un paquete de respuesta.

2.1.2 Especificaciones y espacios de nombres (namespaces) del paquete

El paquete de extensión debe contar con los atributos de herramientas XSD apropiados de manera que el guión pueda generar los espacios de nombres correctamente. A continuación, se presenta un ejemplo de cómo se define estos atributos para el sub-paquete Fix Holding Data.



Cada nuevo paquete de Esquema de Aplicación utilizado para generar esquemas XML debe tener cinco propiedades. Estas propiedades aparecen resaltadas a continuación con la Fuente como 'Override'.

Para modificar el valor de estas propiedades, abrir la Especificación del Paquete (Package Specification) y navegar a la pestaña XSD. Los valores de las propiedades targetNamespace y targetNamespacePrefix están determinados por la COI y son utilizados de conformidad con otros esquemas relacionados, y determinarán si se incluye o importa una importación externa.

Asimismo, designar la propiedad generateFilename como aplicable de manera que el esquema reciba un nombre consistente cada vez que es generado con los guiones UML-a-XSD.

Lo que sigue fue generado para el paquete del Esquema de Aplicación de Datos de Retención de Puntos de Referencia (*Fix Holding Data Applicatio Schema)*.

```
<schema xmlns:fix="http://www.faa.gov/FPP/fixes"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:aixm="http://www.aixm.aero/schema/5.1"
xmlns:dpshare="http://www.faa.gov/avnis/shared"
targetNamespace="http://www.faa.gov/avnis/fixes"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
```

2.2 Paquete de extensión UML

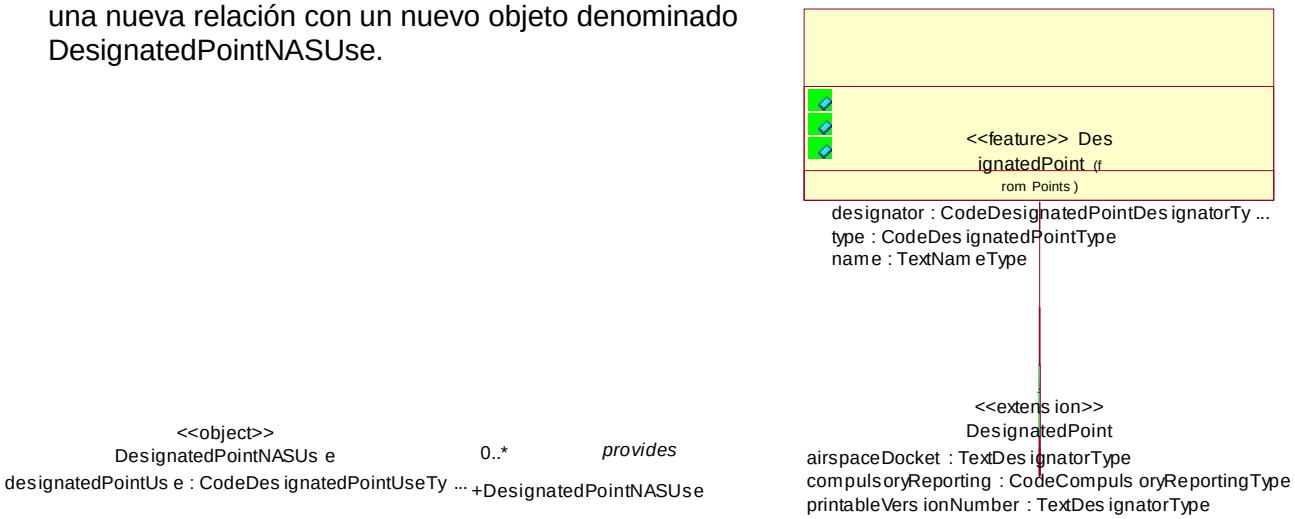
2.2.1 Generalidades

Se puede extender un componente u objeto creando una clase con el mismo nombre que el componente AIXM medular y dándole un estereotipo <<extension>>. Esta nueva clase puede contener:

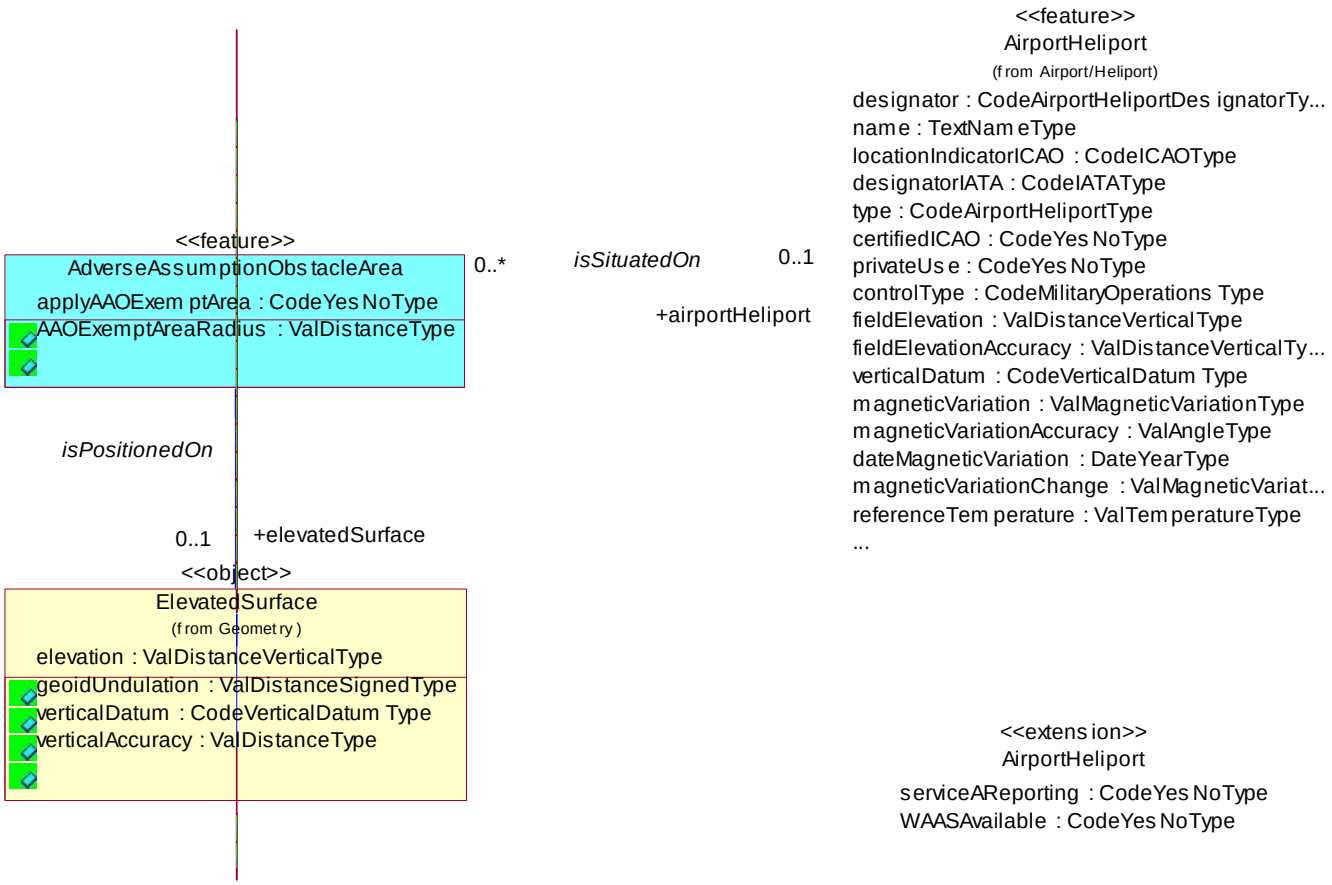
- Nuevos atributos
- Nuevas asociaciones. ***Nota importante:*** Si está involucrada una clase medular AIXM, la navegabilidad de la asociación siempre debería ir de la clase <<extension>> hacia la clase medular AIXM.

Asimismo, es posible crear, en las extensiones, clases totalmente nuevas (componentes y objetos) que no extienden las clases medulares AIXM existentes. La única regla es seguir las convenciones de modelado UML AIXM descritas al inicio del documento. Esto le permitirá al guión AIXM-FeatureGenerator.ebs generar correctamente los elementos XML para estas nuevas clases. Esta situación no aparece descrita en detalle en este documento, ya que no requiere acción especial alguna. Simplemente, hay que seguir las convenciones de modelado UML.

El siguiente ejemplo muestra la convención de modelado utilizada para extender el componente DesignatedPoint. El ejemplo agrega un nuevo atributo a DesignatedPoint y una nueva relación con un nuevo objeto denominado DesignatedPointNASUse.



También, se puede crear asociaciones entre nuevos componentes u objetos y los componentes u objetos medulares AIXM, tal como se ilustra a continuación en la asociación entre AdverseAssumptionObstacle y AirportHeliport. La nueva asociación debería ser creada en el paquete Application Schema package y con dirección al Componente AIXM Medular y no con dirección a una extensión (en caso exista). Esta acción garantiza que la relación esté debidamente representada en el Esquema XML.



Use las reglas aprobadas para los elementos medulares AIXM para generar nuevos componentes u objetos. Algunas reglas aplicables a las nuevas clases de *extensión* son:

El estereotipo de la clase de extensión debe ser <<extension>>.

El nombre de la clase de extensión para la extensión debe coincidir con la clase que se está extendiendo. (Cuando se utiliza Rational Rose, es posible hacerlo creando una nueva clase en el menú de navegación, a la izquierda, y cambiar el nombre de dicha clase; sólo entonces, arrastrar y colocar la clase en el diagrama.)

La clase de extensión debe ser una clase especializada que extiende la clase base correspondiente. Se agrega los atributos de la clase de extensión a la clase de extensión, de la misma manera que con la clase regular AIXM (los Tipos de Datos son materia de discusión posteriormente en este documento).

2.2.2 Generación de esquemas XML

Utilizar el guión AIXM-FeatureGenerator.ebs para generar el esquema de extensión XML, donde el guión activa la generación de un elemento de extensión mediante el reconocimiento del estereotipo <<extension>>. La generación de la extensión sigue las reglas de generación AIXM.

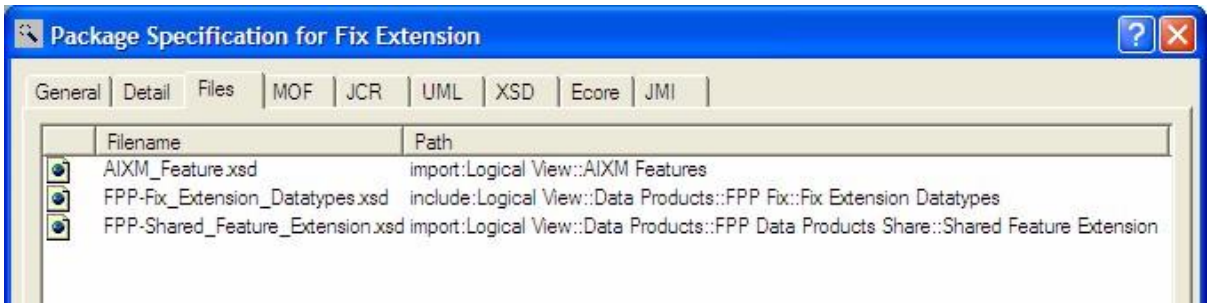
Si se introduce nuevos tipos de datos o listas de códigos al guión, se debe ejecutar primero AIXM-DataTypeGenerator.ebs en el Paquete de Tipo de Datos asociado.

2.2.2.1 Esquemas importados e incluidos

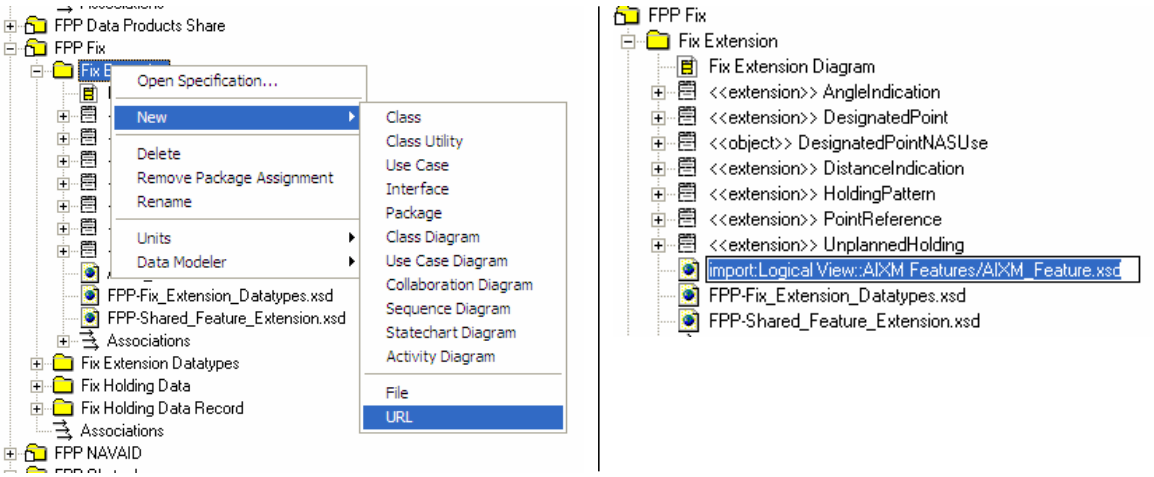
Cada sub-paquete del esquema de aplicación debe incorporar los esquemas XML importados de los esquemas medulares AIXM (AIXM Core Schemas). Asimismo, si se introduce nuevos tipos de datos o listas de códigos, se debe incluir el esquema de dicho sub-paquete. A veces, es necesario utilizar objetos comunes de un paquete 'compartido' para aumentar la re-utilización de los objetos. Se debería incorporar estos elementos también, mediante la importación del esquema.

No es necesario generar estos esquemas para que el guión corra en Rational Rose, pero, si no son creados en la estructura de la carpeta cuando se abre el esquema, tendrá errores. El guión, AIXM-DataTypeGenerator.ebs, es usado para crear el esquema en el sub-paquete de Tipo de Datos asociado.

Estos esquemas vinculados, básicamente URL, pueden ser incorporados ingresándolos en la pestaña Archivos (Files) de la Especificación del Paquete.



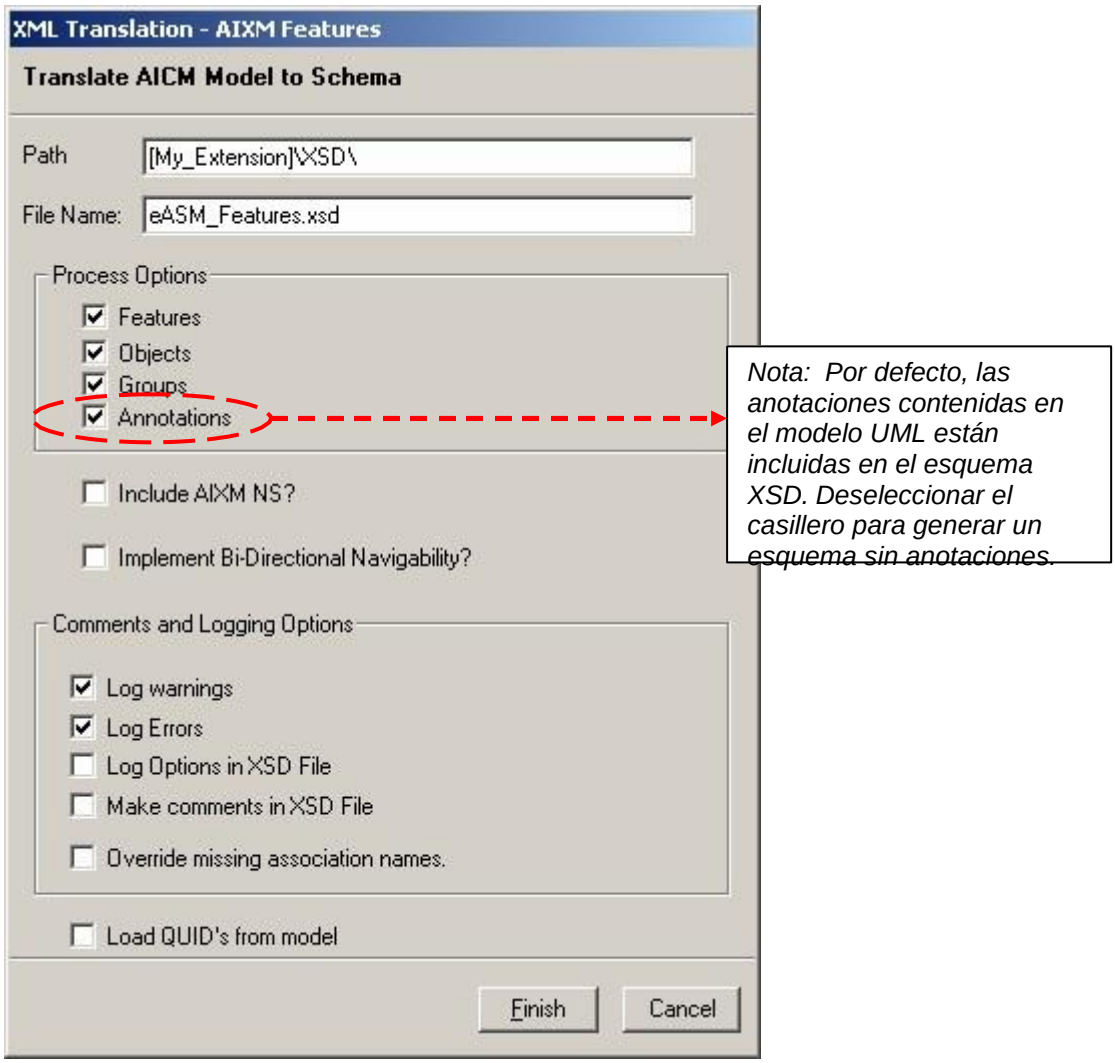
También pueden ser agregados en la ventana de navegación, ingresando la trayectoria completa de la ubicación del esquema dentro del modelo.



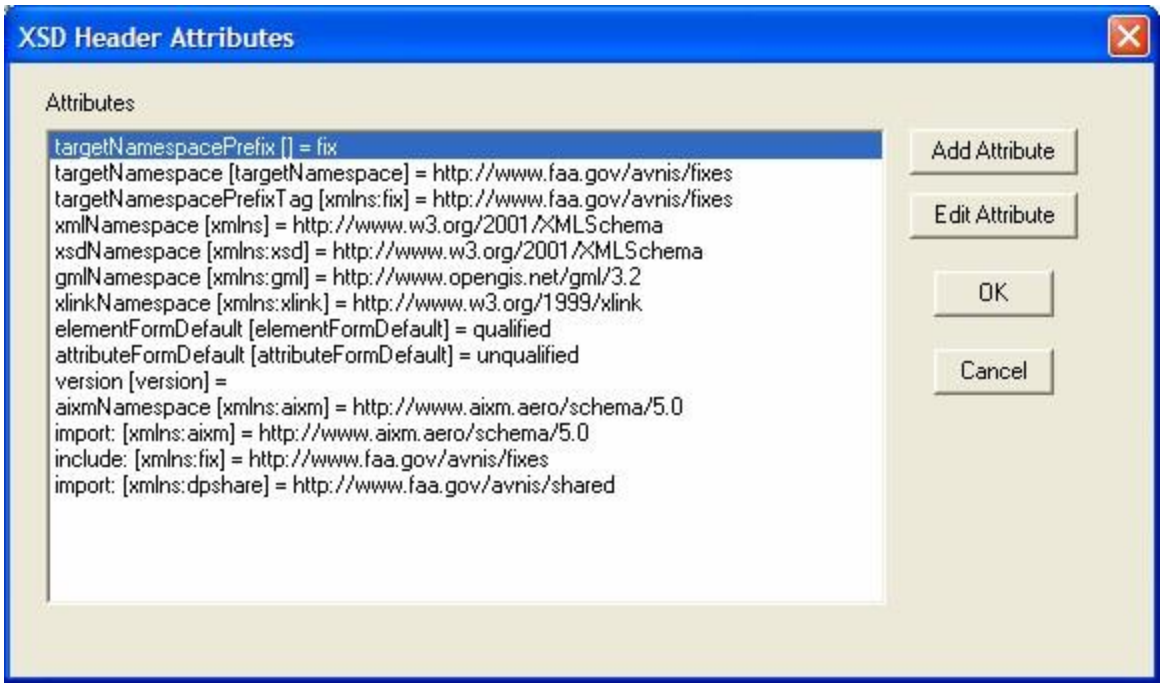
2.2.2.2 Ejecución del guión

Es necesario definir algunas opciones específicas al ejecutar el guión en Rational Rose, pero la mayor parte de las opciones tendrán el valor por defecto. En la siguiente imagen, nótese que el Nombre del Archivo es tomado de la propiedad generateFilename definida previamente en la especificación del paquete.

Los guiones para la generación de esquemas son utilizados tanto para los esquemas medulares AIXM como para los esquemas de aplicación. Se marca el casillero de 'Include AIXM NS' cuando se ejecuta guiones para paquetes que no son parte del conjunto Medular AIXM ya que el Espacio de Nombre AIXM es incluido automáticamente para estos esquemas. Asimismo, marcar el casillero de 'Load QUID's from model' cuando se ha agregado nuevas clases al modelo desde que se ejecutó el guión por última vez, lo cual garantiza que los identificadores de elementos sean correctamente reconocidos.



Luego de seleccionar 'Finish' y de regenerar los QUID (de ser necesario), se abrirá un diálogo para permitir la adición o edición de los atributos del Encabezado XSD (XSD Header). No debería ser necesario hacer cambio alguno, pero tomar nota de los atributos de targetNamespace generados a partir de las especificaciones del paquete definidas previamente. Luego de seleccionar 'OK', se generará el archivo del esquema XML, el cual puede encontrarse en el directorio en el que se ejecutó el guión.

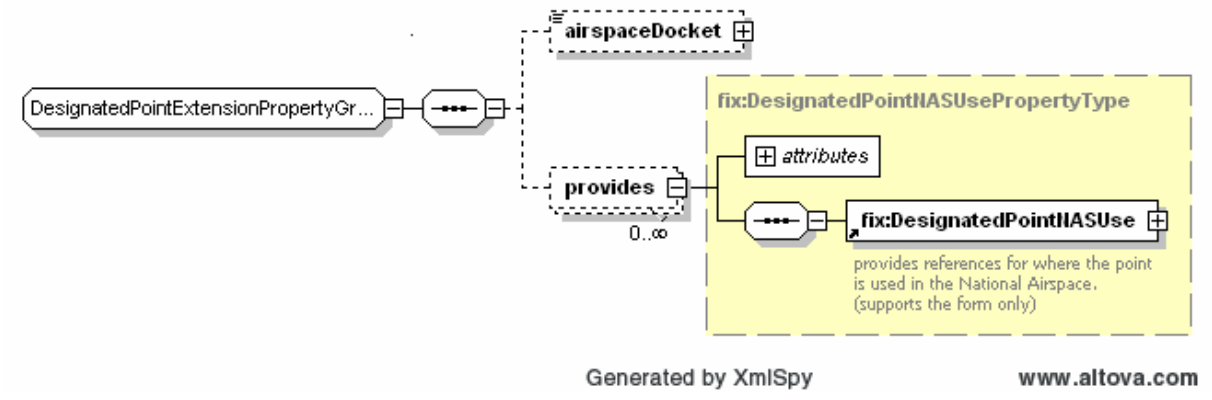


2.2.2.3 Producto del esquema XML

Las clases con el estereotipo <<extension>> generan tres elementos relacionados para esta clase.

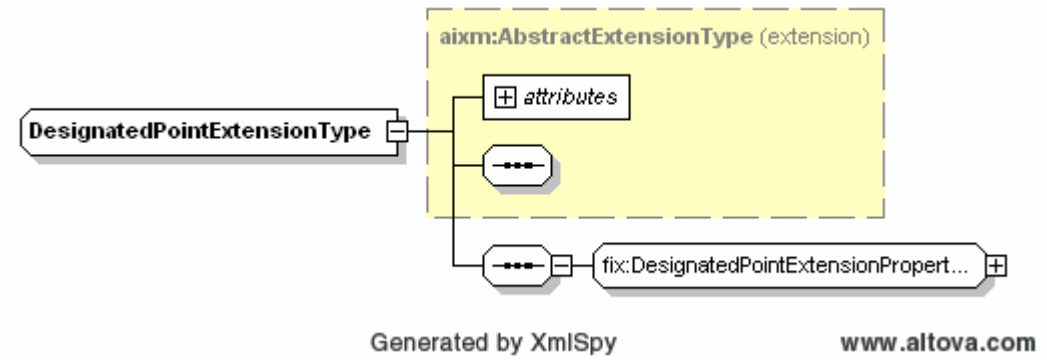
<classname>ExtensionPropertyGroup
<classname>ExtensionType
<classname>Extension

El <classname>ExtensionProperyGroup contiene las propiedades (elementos y relaciones) de la Extensión.

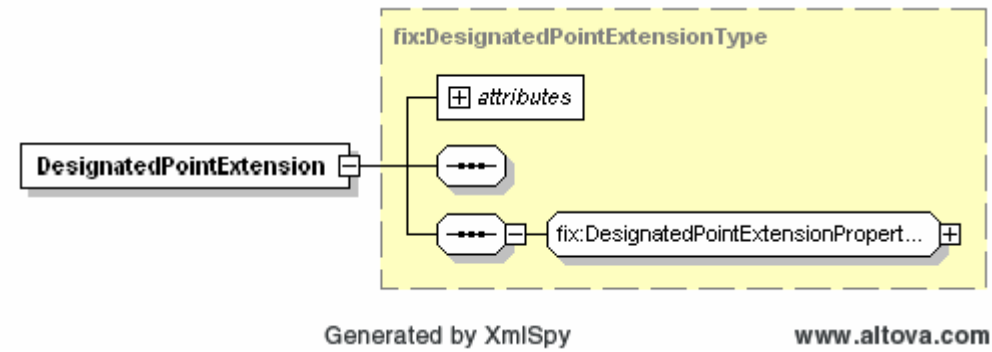


(

El elemento <classname>ExtensionType es generado como un tipo complejo del EsquemaXML (XMLSchema <complexType.>) y extiende el tipo base aixm:AbstractExtensionType.



Se genera el elemento <classname>Extension como un <element> del Esquema XML. El elemento de Extensión no puede estar sólo; sólo puede existir como una extensión del elemento base AIXM. El elemento de Extensión no tiene una fracción de tiempo. El atributo substitutionGroup del elemento de Extensión es el substitutionGroup de la extensión de tipo base. Los elementos de Extensión no son extensibles.

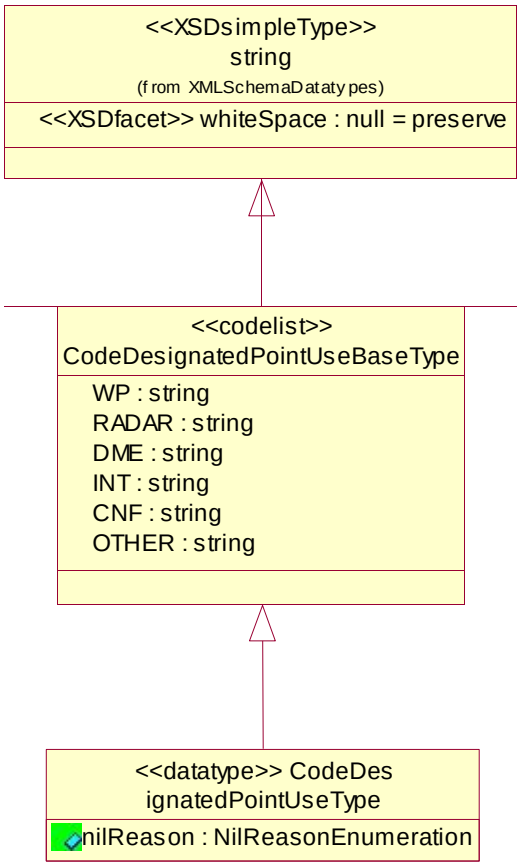


2.3 Paquete de extensión del tipo de datos

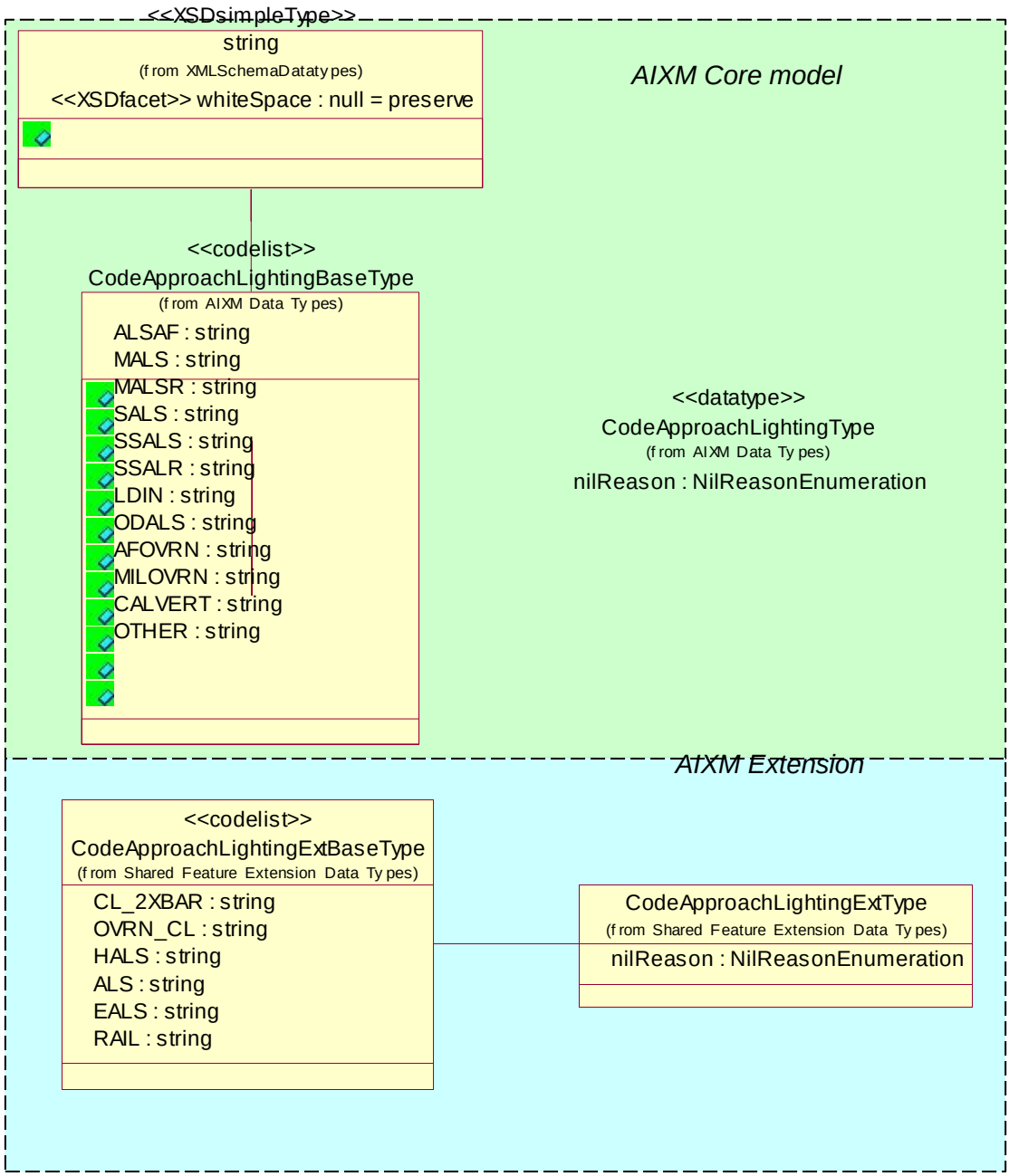
2.3.1 Generalidades

Una extensión, un nuevo componente o clase de objeto puede requerir tipos de datos o listas de códigos adicionales para capturar los valores válidos para los nuevos atributos agregados a la clase. Para agregar nuevos tipos de datos o listas de códigos, crear un sub-paquete de Tipo de Datos (Data Type) que contenga cualquier nuevo tipo de datos que fuera necesario.

En el siguiente ejemplo, se define un <<codelist>> en un paquete de extensión. Se denomina CodeDesignatedPointUseBaseType, tiene una generalización a la clase de 'sarta' y hereda los atributos básicos de una variable de sarta XSD. Se crea el <<datatype>> CodeDesignatedPointBaseType con la propiedad nilReason, tal como se especifica en [4]. Esta es la configuración más común para las listas de códigos.



Los <<codelists>> medulares del AIXM también pueden ser extendidos en el sub-paquete de Tipo de Datos (Data Type). Extender una lista de código mediante la creación de una clase con el mismo nombre que la lista de código y dándole un estereotipo <<codelist>>.



Se debe hacer un análisis detallado para asegurarse que la lista extendida de valores sigue estando normalizada. No deberá duplicar valores que ya existen en el <<codelist>> medular (incluyendo el valor OTRO), pero con otros nombres/abreviaturas.

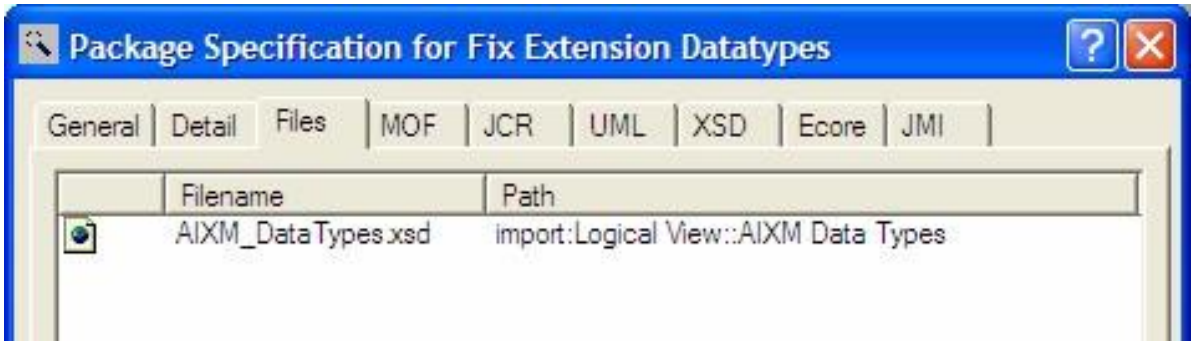
En caso se requiera valores adiciones para una lista de código medular AIXM para el intercambio de datos a nivel internacional, entonces será necesario actualizar el modelo medular AIXM.

2.3.2 Generación de esquemas XML

Utilizar el guión AIXM-DataTypeGenerator.ebs para generar el esquema XML de extensión de tipo de datos. Los Tipos de Datos son generados como un <simpleType> de XMLSchema, con las facetas, Patrones y/o listas de código apropiados definidos.

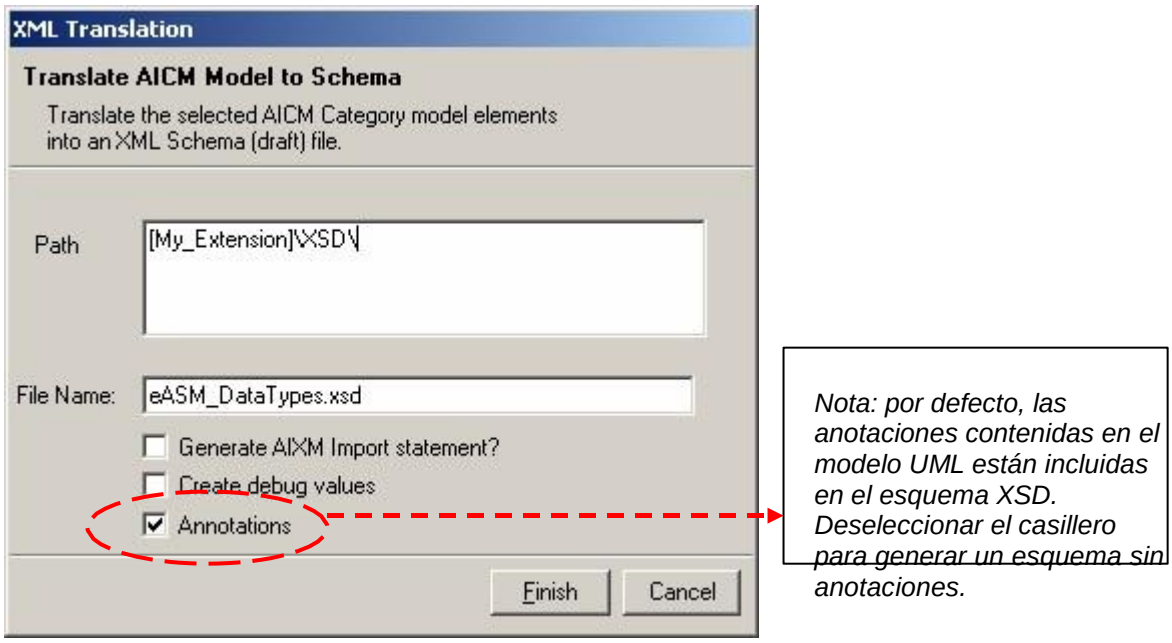
2.3.2.1 Esquemas importados e incluidos

Cada sub-paquete de tipo de datos debe incorporar el esquema medular de Tipo de Datos AIXM. No es necesario generar este esquema para que el guión se ejecute en Rational Rose, pero, si no se genera el esquema de Tipo de Datos Básicos AIXM en la estructura de la carpeta donde se abre el esquema, tendrá errores.



2.3.2.2 Ejecución del guión

En la siguiente imagen, nótese que el Nombre de Archivo (File Name) es tomado de la propiedad generateFilename definida previamente en la especificación del paquete. No obstante, la ruta (Path) indica la ubicación del archivo del modelo UML, la cual debería ser cambiada apropiadamente. Cuando se ejecuta guiones para paquetes que no son parte del conjunto medular AIXM, se selecciona el casillero correspondiente a 'Include AIXM NS', ya que el Espacio de nombre (Namespace) del AIXM es automáticamente incluido para estos esquemas.

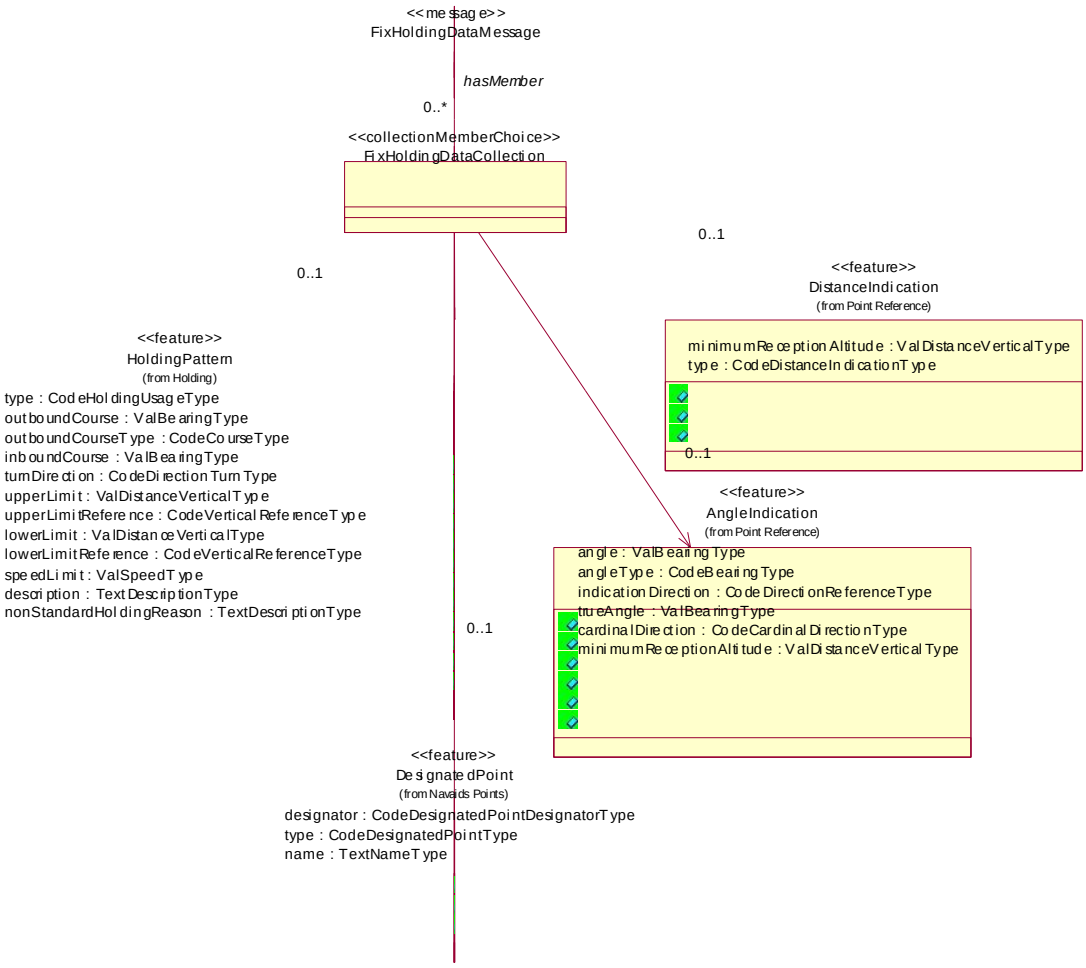


2.4 Paquete de mensaje UML

2.4.1 Generalidades

El paquete de mensaje es utilizado para generar un Esquema XML para los mensajes de solicitud y respuesta. La siguiente ilustración es un ejemplo del Mensaje de Respuesta FixHoldingData. Este mensaje incluye las extensiones descritas anteriormente, si bien éstas no aparecen en el diagrama.

Se modela un mensaje en UML, utilizando el objeto de clase con un estereotipo <<message>>. En este ejemplo, el mensaje es una colección limitada de componentes AIXM con extensiones. Esto se modela relacionando la colección de componentes <<collectionMemberChoice>> al mensaje a través de la relación “hasMember”.

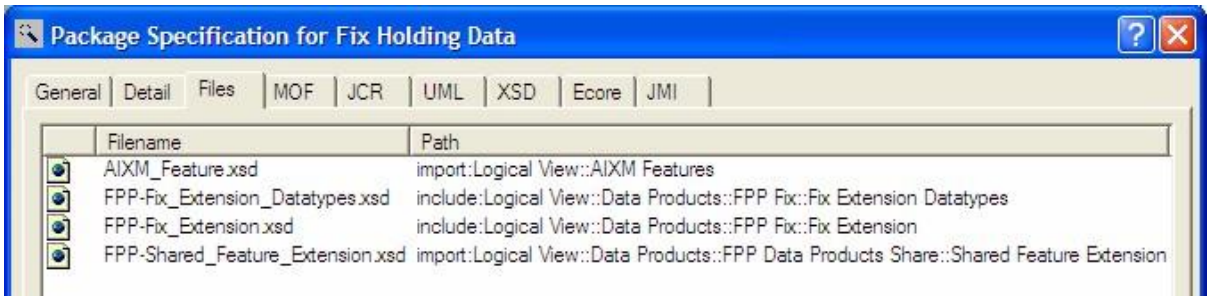


2.4.2 Generación del esquema XML

Utilizar el guión AIXM-ApplicationSchemaGenerator.ebs para generar el mensaje XSD. El guión activa la generación del elemento del mensaje, mediante el reconocimiento del estereotipo <<message>>.

2.4.2.1 Esquemas importados e incluidos

El sub-paquete del mensaje UML agrupa a todos los elementos relacionados, creados en procesos anteriores, tales como extensiones y tipos de datos. Como en los casos anteriores, se debe incluir el esquema medular AIXM, así como cualesquiera otros esquemas referenciados (es decir, objetos compartidos o comunes que son utilizados en múltiples esquemas de aplicación).



A continuación, se muestra la acumulación de los Esquemas XML importados e incluidos.

```
<import namespace="http://www.opengis.net/gml/3.2"
  schemaLocation="./ISO_19136_Schemas/gml.xsd"/>
<import namespace="http://www.aixm.aero/schema/5.1"
  schemaLocation="./AIXM_Feature.xsd"/>
<import namespace="http://www.w3.org/1999/xlink" schemaLocation="./xlink/xlinks.xsd"/>
<import namespace="http://www.faa.gov/avnis/shared" schemaLocation="./FPP-
  Shared_Feature_Extension.xsd"/>
<include schemaLocation="FPP-Fix_Extension_Data_Types.xsd"/>
<include schemaLocation="FPP-Fix_Extension.xsd"/>
```

2.4.2.2 Ejecución del guión

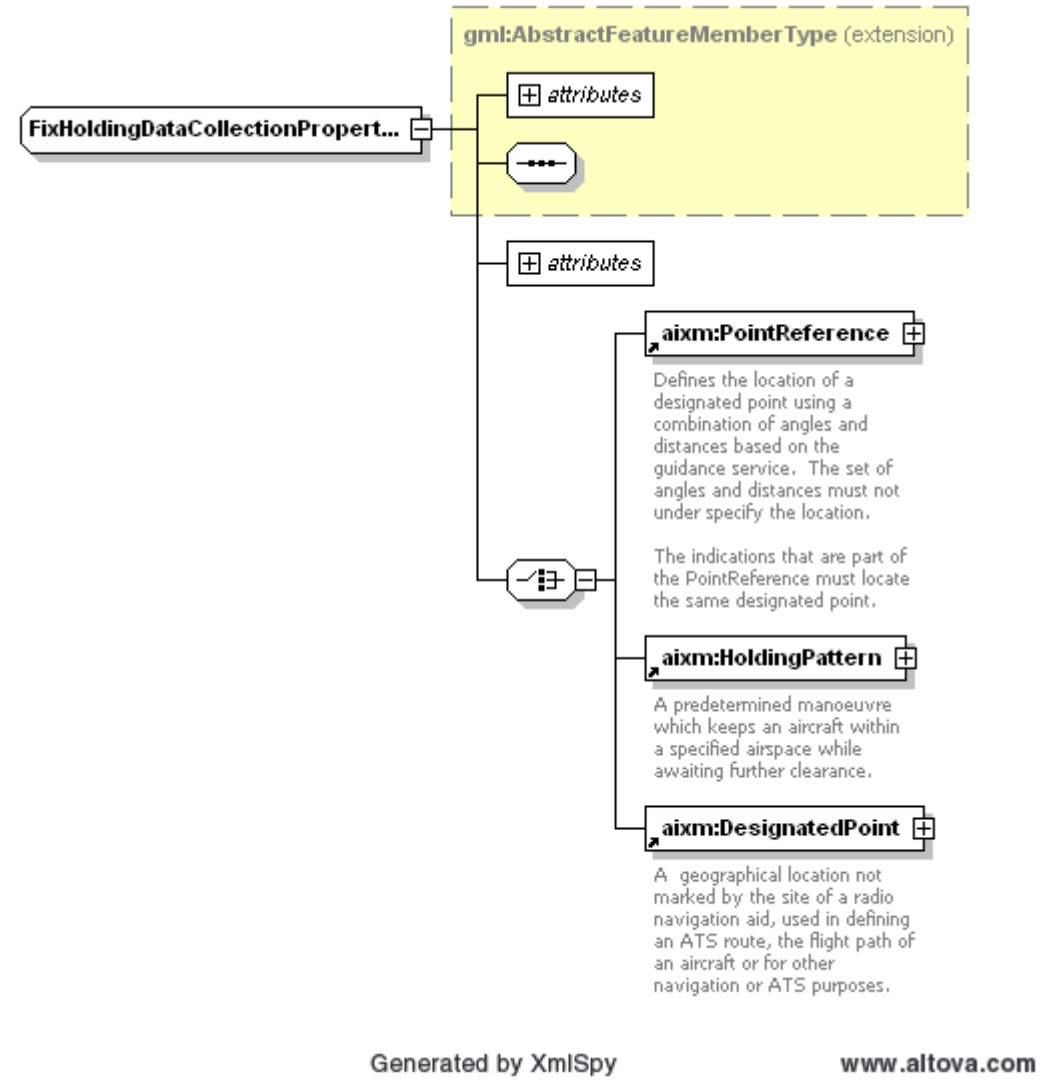
Seguir los procedimientos indicados en la sección 2.2.2.2.

2.4.2.3 Producto del esquema XML

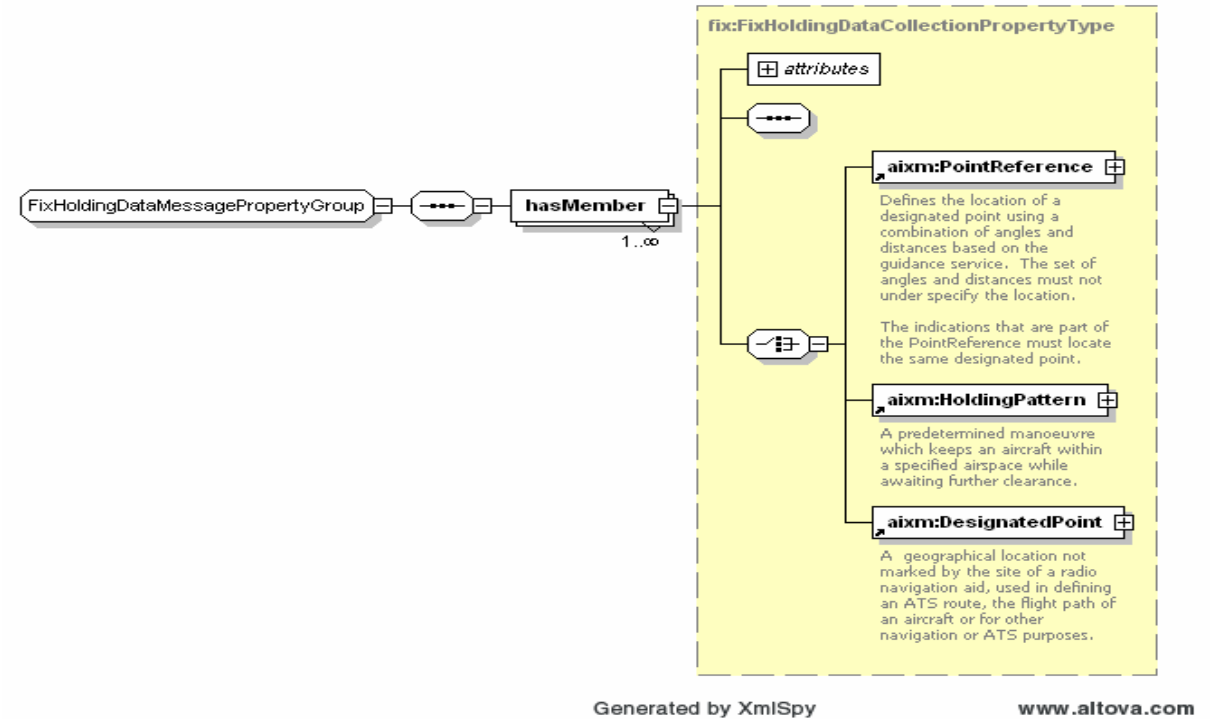
Las clases con el estereotipo <<message>> a continuación de la respuesta de recolección de componentes AIXM generan cuatro elementos relacionados para dicha clase.

<classname>CollectionPropertyGroup
<classname>MessagePropertyGroup
<classname>MessageType
<classname>Message

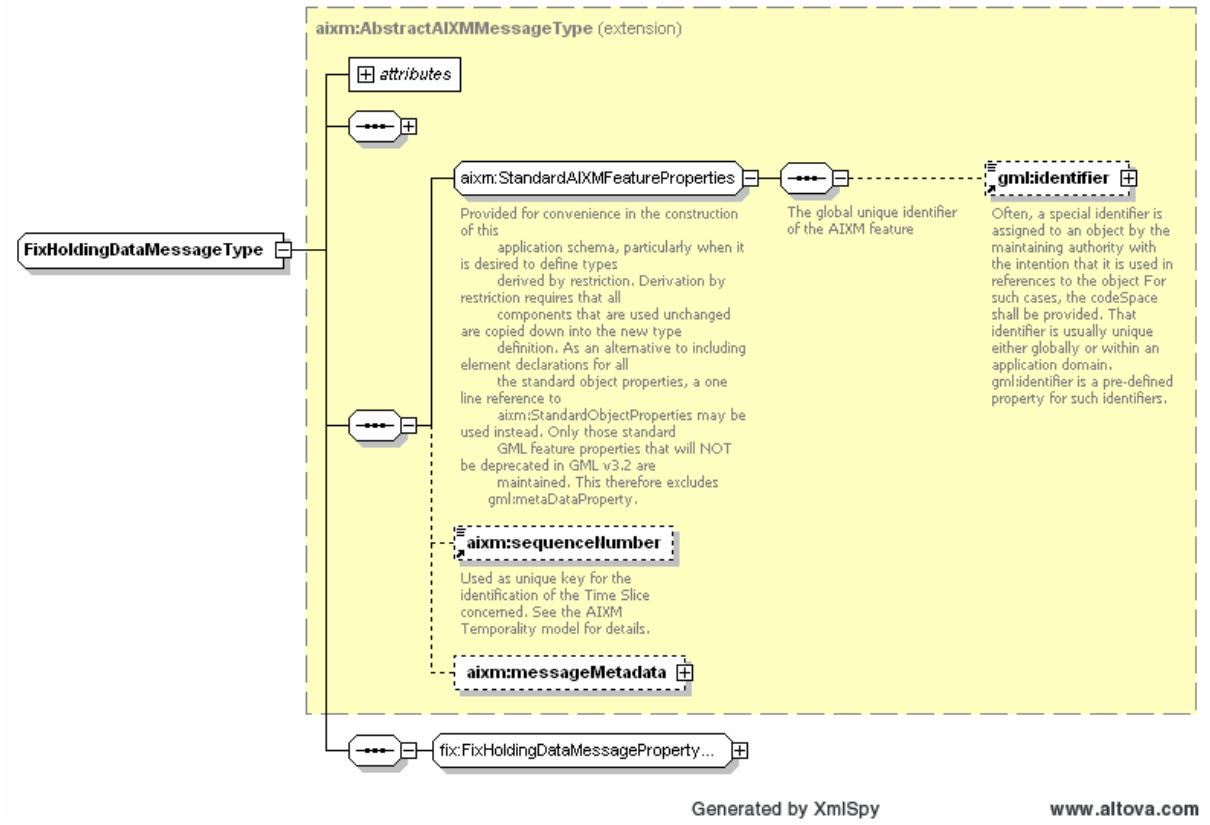
Se genera el <classname>CollectionPropertyGroup como un Esquema XML <complexType>, el cual extiende gml:AbstractFeatureMemberType, e incluye un <choice> entre todos los componentes a los que apunta.



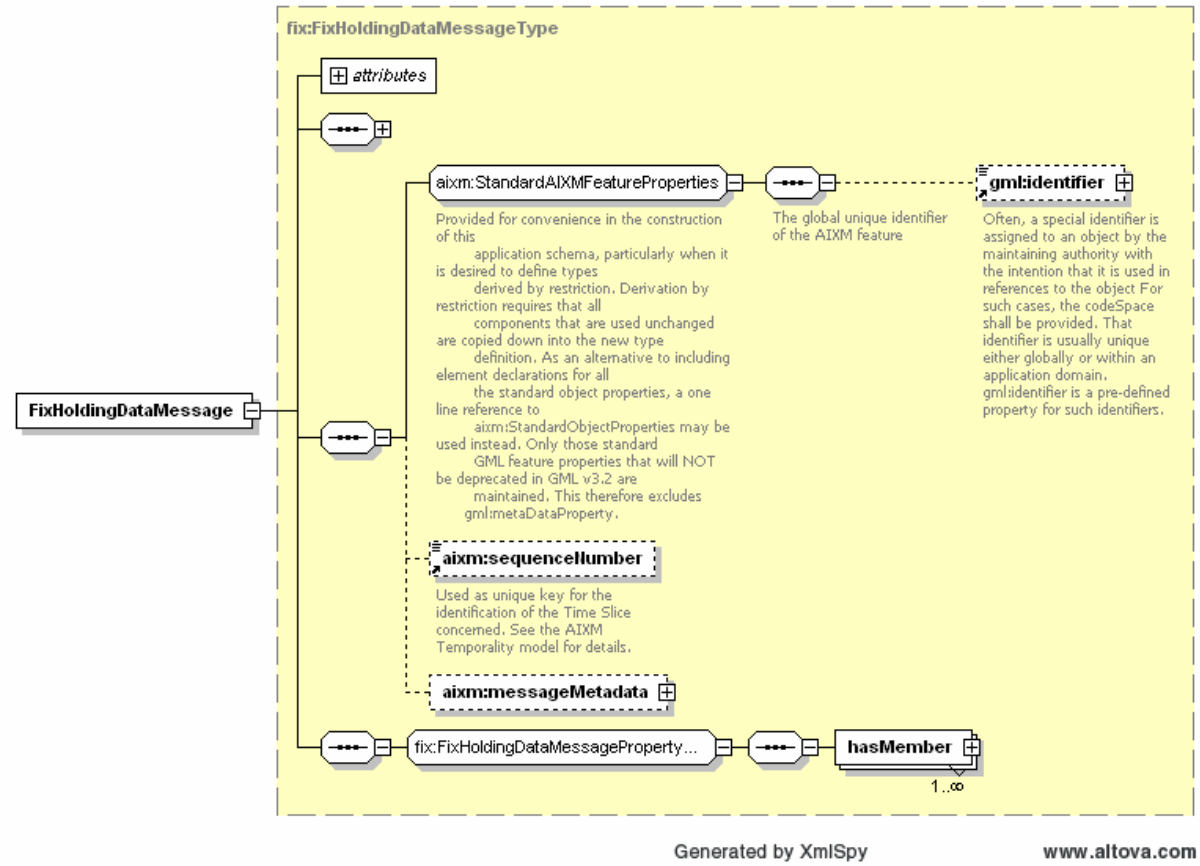
Se genera <classname>MessagePropertyGroup como un XMLSchema <group>, el cual contiene las propiedades (elementos y relaciones) del Mensaje.



Se genera el elemento <classname>MessageType como un XMLSchema <complexType.> y extiende el tipo base aixm:AbstractAIXMMessageType.



Se genera el elemento <classname>Message como un XMLSchema <element>.
Las asociaciones son tratadas como objetos. Estos son incluidos en el esquema.



APÉNDICE E

AIXM 5

IDENTIFICACIÓN Y REFERENCIA DE COMPONENTES

USO DE XLINK:HREF Y UUID

AIXM 5

Identificación y Referencia de Componentes

- uso de xlink:href y UUID -

Modelo de Intercambio de Información Aeronáutica (AIXM)

Para cualquier consulta, sírvase ponerse en contacto con:

Deborah COWELL - deborah.cowell@faa.gov

Eduard POROSNICU - eduard.porosnicu@eurocontrol.int

Versión No.	Fecha de emisión de la versión	Autor	Razón del cambio
0.1 Primera	2009	Equipo de diseño	Primera versión
0.2 Propuesta	2010	Equipo de diseño	Actualización
0.3 Propuesta	06 oct 2010	Equipo de diseño	Actualización de referencias abstractas
0.4 Propuesta actualizada	22 nov 2010	Eurocontrol-FAA Equipo de diseño AIXM Con el aporte de los miembros del Foro AIXM	Se incluyó guías para la generación de UUID. Actualizada en base a los comentarios y recomendaciones recibidos de los miembros del Foro AIXM y durante los seminarios AIXM-XML de 2010.
0.5 Propuesta actualizada	21 dic 2010		Actualizada luego de los últimos comentarios del Foro AIXM. Uso de URN “uuid”. Uso de URN únicamente para referencias abstractas.
0.6 Actualizada	21 feb 2011		Actualizada luego de las discusiones fuera de línea con los miembros del Foro AIXM más activos (en este tema).

1.0	Publicada	29 abr 2011		Publicación final. Se agregó una oración en la sección 2.2 para resaltar la importancia que tiene que el UUID sea atribuido por la verdadera fuente autorizada de los datos de un componente.

Indice

1 Alcance

1.1 Introducción

El Modelo de Intercambio de Información Aeronáutica (AIXM) es un esquema de aplicación GML 3.2 cuyo objetivo es permitir el intercambio de información aeronáutica de máquina a máquina en un formato estructurado. Conforme se desarrollan servicios para difundir información en el AIXM 5.1 a los consumidores, es esencial tener la capacidad de gestionar los vínculos entre los componentes aeronáuticos. Esto comprende los conceptos de identificación de componentes y referencia de componentes.

El esquema AIXM 5.1 utiliza el esquema XLink unido al GML 3.2 para representar una referencia entre dos componentes. Conjuntamente con la norma XLink, se puede utilizar la norma XPointer para tratar los elementos individuales XML de los mensajes. Este documento define un número de casos normalizados sobre cómo se debería utilizar los XLinks dentro de un mensaje AIXM 5.1 y cómo estos XLinks deberían ser resueltos por las aplicaciones.

El esquema AIXM 5.1 también utiliza identificadores únicos universales (UUID) como identificadores artificiales para los componentes AIXM. En realidad, no identifican al componente en sí, sino a los datos que representan a dicho componente en los sistemas digitales de gestión de la información aeronáutica. Este documento brinda orientación con respecto a los algoritmos que pueden ser utilizados para la generación de estos valores UUID.

1.2 Referencias

[XLINK]	Especificación XLink v1.0 http://www.w3.org/TR/xlink
[XPTR]	Especificación XPointer http://www.w3.org/TR/xptr
[XPTH]	Especificación XPath http://www.w3.org/TR/xpath
[UUID]	Identificador único universal (teoría) http://en.wikipedia.org/wiki/Universally_unique_identifier
[UUID-AIXM]	Análisis del identificador único universal (UUID), por Robert DeBlanc, MITRE, en apoyo de la FAA
[UUID-ISO]	ISO/IEC 9834-8, que es también UIT-T Rec x.667 http://www.itu.int/ITU-T/studygroups/com17/oid.html

1.3 Premisas y dependencias

Este documento asume que los sistemas en cuestión se están comunicando utilizando mensajes y conjuntos de datos que cumplen con la versión de esquemas AIXM 5.0, 5.1 ó posterior.

2 Identificación de componentes (UUID)

2.1 La propiedad gml:identifier

Cada componente AIXM es identificado mediante el uso de la propiedad ***identificador (identifier)***, la cual es heredada del AIXMFeature abstracto. En el esquema XML AIXM, esto se vincula a la propiedad gml:identifier, que todos los componentes AIXM heredan de gml:DynamicFeatureType.

De acuerdo con el Concepto de Temporalidad AIXM, la propiedad *identifier* es la única propiedad que no varía con el tiempo; por lo tanto, está ubicada fuera del objeto complejo TimeSlice, el cual encapsula todas las propiedades del componente que pueden cambiar con el tiempo. La propiedad gml:identifier puede ser transmitida junto con cualquier TimeSlice (Fracción de Tiempo) del componente, lo cual permite identificar al componente al cual pertenece el TimeSlice.

Hay dos requisitos esenciales para la propiedad de identificador:

1. ***ser único*** – debería existir una razonable confianza en que el identificador nunca será utilizado intencionalmente por alguien para otro fin;
2. ***ser universal*** – se debería utilizar el mismo identificador en todos los sistemas para identificar un determinado componente AIXM.

2.2 Uso del UUID

El primer requisito puede ser satisfecho mediante el uso de Identificadores Unicos Universales (UUID). Los algoritmos de generación de UUID pueden garantizar que el riesgo que el mismo valor UUID sea generado por otro sistema, para otro componente, sea extremadamente bajo. El Apéndice 1 de este documento proporciona información acerca de dichos algoritmos.

En cuanto al segundo requisito, es importante observar que el identificador no identifica a un componente. Identifica los datos que alguien tiene sobre un componente! A fin de obtener el máximo beneficio del UUID, éstos deberían ser generados por el originador primario (fuente autorizada) de dichos datos de componente.

Idealmente, todas las partes involucradas deberían tener los mismos datos sobre un determinado componente. No obstante, como pueden existir múltiples fuentes de información “seudo-primarias” para el mismo dato, o debido a que se puede romper o duplicar la cadena de transmisión de datos digitales, esto no se puede garantizar, por lo menos en el corto plazo. El uso garantizado del mismo gml:identifier en todos los sistemas para un determinado componente AIXM es un requisito para el proceso de gestión de la información; por lo tanto, se tiene que hacer a través de las reglas de proceso. Desde esta perspectiva, los UUID pueden indicar la continuidad y coherencia de la cadena de datos. Si dos sistemas utilizan el mismo UUID para un componente, esto es una indicación que:

- sus datos provienen de la misma fuente (podría ser uno de los dos sistemas, o un tercero), o
- existen procesos para garantizar la consistencia de los datos entre los dos sistemas.

Por lo tanto, es posible que existan dos o más conjuntos de datos (lista de TImeSlices) para el mismo componente AIXM, en dos sistemas diferentes, con distintos valores de gml:identifier. Cuando se fusionan los datos de distintas fuentes en un solo sistema, el dueño de dicho sistema podría verse en la necesidad de identificar y fusionar datos duplicados de un componente, en base a las propiedades reales del componente y no en gml:identifier.

En términos generales, la ventaja más importante del uso del UUID como gml:identifier en el AIXM es para fines de desarrollo de soporte lógico. Es mucho más sencillo y menos susceptible a error el escribir un código que utiliza el UUID para la identificación y referencia de los componentes, comparado con cualquier combinación de "claves naturales".

.

2.3 Versión y codeSpace (espacio de código) del UUID

En base al análisis presentado en el Apéndice 1, ***se recomienda el uso de la versión 4 del UUID, basada en la generación de números al azar, para el AIXM***. A continuación, se presenta un ejemplo de gml:identifier, utilizando un valor UUID.

```
<gml:identifier
     codeSpace="urn:uuid:">a82b3fc9-4aa4-4e67-8def-
aaea1ac595j</gml:identifier>
```

El Apéndice 1, sección A.1.9, brinda información acerca de la capacidad de generación de UUID en el soporte lógico común.

Cabe notar que se utiliza un Nombre de Recurso Uniforme (URN) como codeSpace para el gml:identifier. La ISO/IEC 9834-8 ó RFC 4122 (http://www.ietf.org/rfc/rfc4122.txt) proporcionan el valor "urn:uuid:" codeSpace UUID.

2.4 La propiedad gml:id

Cada objeto GML debe tener un valor gml:id que sirva como identificador único local dentro del conjunto de datos XML. Los componentes AIXM, como también son objetos GML, también deben tener un valor gml:id. Asimismo, todos los otros objetos GML dentro del componente (TimeSlice, gml:TimePeriod, gml:Point, aixm:SurfaceCharacteristics, aixm:AirspaceVolume, etc.) también deben tener un valor gml:id, como se muestra en el siguiente ejemplo:

```
<aixm:Airspace gml:id="…">
  <gml:identifier
     codeSpace="urn:uuid:">a82b3fc9-4aa4-4e67-8def-
aaea1ac595j</gml:identifier>
  <aixm:timeSlice>
    <aixm:AirspaceTimeSlice gml:id="…">
      <gml:validTime>
        <gml:TimePeriod gml:id="…">
          <gml:beginPosition>2010-06-29T17:31:00</gml:beginPosition>
          <gml:endPosition>2010-06-29T19:00:00</gml:endPosition>
        </gml:TimePeriod>
      </gml:validTime>
      <aixm:interpretation>BASELINE</aixm:interpretation>
      <aixm:sequenceNumber>1</aixm:sequenceNumber>
      <aixm:type>D</aixm:type>
      <aixm:geometryComponent>
        <aixm:AirspaceGeometryComponent gml:id="…">
          <aixm:theAirspaceVolume>
            <aixm:AirspaceVolume gml:id="…">
              <aixm:upperLimit uom="FT">500</aixm:upperLimit>
              <aixm:upperLimitReference>MSL</aixm:upperLimitReference>
              <aixm:lowerLimit uom="FT">GND</aixm:lowerLimit>
              <aixm:lowerLimitReference>MSL</aixm:lowerLimitReference>
              <aixm:horizontalProjection>
              <aixm:Surface gml:id="…">
                <gml:patches>
                  <gml:PolygonPatch>
                    <gml:exterior>
              ...
```

```
</aixm:Airspace>
```

El valor gml:id debe cumplir con las mismas reglas que cualquier otro atributo ID XML: ser único dentro del archivo XML, empezar con una letra, etc.

Se recomienda que el gml:id de los componentes AIXM (como aixm:Airspace, aixm:Runway, etc.) también utilicen el valor UUID, con el prefijo "uuid.". Como los UUID son universalmente únicos, también son localmente únicos y, por lo tanto, los perfectos candidatos para gml:id. Atención: esta recomendación sólo es válida para el nivel de componente AIXM, no para los niveles inferiores, como aixm:AirspaceTimeSlice, etc. El uso del UUID del componente como gml:id facilitará la implantación de referencias concretas xlink:href, utilizando la sintaxis directa '#ID', tal como se explica en 3.1.

Aplicado al ejemplo anterior, estas recomendaciones dan los siguientes valores gml:id:

```
<aixm:Airspace gml:id="uuid.a82b3fc9-4aa4-4e67-8def-aaea1ac595j">
  <gml:identifier
     codeSpace="urn:uuid:">a82b3fc9-4aa4-4e67-8def-
aaea1ac595j</gml:identifier>
  <aixm:timeSlice>
    <aixm:AirspaceTimeSlice gml:id="ID00001">
      <gml:validTime>
        <gml:TimePeriod gml:id="ID00002">
          <gml:beginPosition>2010-06-29T17:31:00</gml:beginPosition>
          <gml:endPosition>2010-06-29T19:00:00</gml:endPosition>
        </gml:TimePeriod>
      </gml:validTime>
      <aixm:interpretation>BASELINE</aixm:interpretation>
      <aixm:sequenceNumber>1</aixm:sequenceNumber>
      <aixm:type>D</aixm:type>
      <aixm:geometryComponent>
        <aixm:AirspaceGeometryComponent gml:id="ID00003">
          <aixm:theAirspaceVolume>
            <aixm:AirspaceVolume gml:id="ID00004">
              <aixm:upperLimit uom="FT">500</aixm:upperLimit>
              <aixm:upperLimitReference>MSL</aixm:upperLimitReference>
              <aixm:lowerLimit uom="FT">GND</aixm:lowerLimit>
              <aixm:lowerLimitReference>MSL</aixm:lowerLimitReference>
              <aixm:horizontalProjection>
              <aixm:Surface gml:id="ID00005">
                <gml:patches>
                  <gml:PolygonPatch>
                    <gml:exterior>
              ...

</aixm:Airspace>
```

3 Referencia de componente (xlink:href)

3.1 Introducción

El Esquema XML AIXM establece asociaciones entre los componentes AIXM mediante el uso de XLinks [XLINK].

La recomendación general es utilizar el gml:identifier (UUID) del componente AIXM referenciado. Esto apoya muchas soluciones comerciales listas para usar, ya que se basa enteramente en las normas XLink, XPointer y XPath. Es fundamental que cada valor XLink apunte al componente de interés correcto. Por ejemplo, un valor xlink que identifica un espacio aéreo debe llevar el identificador de un componente de espacio aéreo y no el de una pista u otro componente, mensaje, etc. No obstante, esto no se puede garantizar con el esquema XML y tiene que ser resuelto como parte de las reglas de validación de los datos.

Si no se dispone del UUID, se puede utilizar una búsqueda de clave natural. En general, puede que la verbosidad de tal solicitud aumente, ya que será necesario consultar un TimeSlice del componente AIXM para encontrar la clave.

Desde la perspectiva del objetivo Xlink, hay tres casos de interés:

1. Referencias concretas locales dentro de un conjunto de datos;
2. Referencias concretas externas resueltas a través de servicios web;
3. Referencias abstractas a ser resueltas por la aplicación prevalente.

Estas aparecen descritas en mayor detalle en las siguientes sub-secciones.

3.2 Referencias concretas locales dentro de un mensaje

En algunos casos, los servicios que producen datos AIXM 5.1 brindarán conjuntos de datos en los que están incluidos todos los componentes referenciados. En vez de una referencia por gml:identifier, se podría utilizar en este caso una más simple referencia local al atributo gml:id. Por definición, el gml:id es único dentro de un archivo XML. Por lo tanto, cuando se utiliza para referencias locales, resulta inequívoco. Los atributos gml:id se definen como identificadores en el esquema y pueden ser indexados durante el análisis sintáctico (parsing).

A continuación, se brinda un ejemplo. Nótese que aquí se aplica la recomendación formulada en 2.4, lo cual significa que el gml:id del componente EspacioAéreo está, en realidad, basado en el valor UUID del gml:Identifier.

```
<aixm:Airspace gml:id="uuid.:a82b3fc9-4aa4-4e67-8def-aaea1ac595j">
  <gml:identifier
     codeSpace="urn:uuid:">a82b3fc9-4aa4-4e67-8def-
aaea1ac595j</gml:identifier>

    ...

</aixm:Airspace>
...
<aixm:AirTrafficControlService gml:id="uuid.d4d33081-54ad-4c1a-9519-
b5b67de561ae">
  <aixm:timeSlice>
    <aixm:AirTrafficControlServiceTimeSlice
gml:id="AirTrafficControlService01_TS1">
      <gml:validTime>
        <gml:TimePeriod gml:id="AirTrafficControlService01_TS1_TP1">
          <gml:beginPosition>2008-01-01T00:00:00</gml:beginPosition>
          <gml:endPosition indeterminatePosition="unknown"/>
```

```
        </gml:TimePeriod>
      </gml:validTime>
      <aixm:interpretation>BASELINE</aixm:interpretation>
      <aixm:type>ACS</aixm:type>
      <aixm:clientAirspace xlink:href="#uuid.a82b3fc9-4aa4-4e67-8def-
aaea1ac595j"/>
    </aixm:AirTrafficControlServiceTimeSlice>
  </aixm:timeSlice>
</aixm:AirTrafficControlService>
```

3.3 Referencias externas concretas

Cuando se expone la información sobre los componentes a través de los servicios web, un método que se puede utilizar para resolver XLinks es a través de un Localizador Universal de Recursos (URL).

En este caso, la expectativa es que el consumidor del mensaje pueda seguir el URL proporcionado en el XLink directamente y el *hashtag* identificará el componente dentro del recurso resultante vinculado a la referencia. Si se aplica la recomendación formulada en 2.4 y el componente objetivo tiene un gml:id basado en el UUID, entonces la referencia concreta puede ser codificada utilizando simplemente la sintaxis de referencia '#ID', como en el siguiente ejemplo:

```
      <aixm:clientAirspace
xlink:href="http://aim.faa.gov/services/AirspaceService#uuid.a82b3fc9-
4aa4-4e67-8def-aaea1ac595j"/>
```

Una solución más general, pero también mucho más compleja, es utilizar xpointer. Nuevamente, la combinación del URL y el XPointer debería resultar en la etiqueta vinculada al componente al que se hace referencia.

```
      <aixm:clientAirspace
xlink:href="http://aim.faa.gov/services/AirspaceService?get=a82b3fc9-4aa4-
4e67-8def-
aaea1ac595j#xmlns(ns1=http://www.opengis.net/gml/3.2)xmlns(ns2=http://www.
aixm.aero/schema/5.1)xpointer(//ns2:Airspace[ns1:identifier='a82b3fc9-
4aa4-4e67-8def-aaea1ac595j'])"/>
```

Nótese que el URL arriba mencionado es independiente de la implementación. Puede identificar un servidor WFS (*Web Feature Server*), pero también puede ser una implementación de un servicio web sencillo que devuelve un determinado conjunto de componentes en base a la consulta del usuario. En tanto la resolución del URL produzca un documento XML que contenga el componente AIXM, lo arriba indicado es una referencia válida.

El enfoque para los dos primeros casos (referencias concretas locales o externas) se basa puramente en la norma XLink y XPointer: para resolver la referencia, no se requiere una lógica específica para la aplicación. No obstante, los implementadores deberían aprovechar las estrategias de *"caching"* para evitar estar resolviendo continuamente los componentes para los que ya cuentan con definiciones.

3.4 Referencias abstractas

Xlink adopta un "enfoque basado en los recursos", donde el documento o fragmento XML es un recurso que puede ser referenciado desde cualquier lugar. Asume que el recurso está disponible en la web en una sola copia definitiva. En el dominio de la información aeronáutica, el componente es una entidad que puede ser referenciada a nivel mundial, pero existen muchas representaciones de dicho componente en circulación. Muchas de estas representaciones están en mensajes AIXM; otras están en bases de datos o en

aplicaciones. Es por esto que el uso de referencias concretas es inusual y es necesario considerar también el uso de referencias abstractas.

Las referencias abstractas encajan muy bien con el paradigma GML, ya que ofrecen una referencia a la idea abstracta del componente y no a una de sus representaciones. La aplicación se encarga de resolver la referencia abstracta a una referencia física, y puede enfrentar el problema de múltiples copias en todos los mensajes, datos de bases, etc., de los que tiene conocimiento.

Dejando la resolución a cargo de la aplicación también permite que la aplicación se adapte a su contexto. Por ejemplo, si la aplicación opera fuera de línea, podría optar por usar una copia del componente almacenada localmente. Una aplicación en línea podría recurrir al servicio web definitivo para buscar al componente.

En estos casos, el xlink:href debería utilizar un nombre de recurso uniforme (URN) en vez de un URL; nótese que los URN, a diferencia de los URL, no pueden ser utilizados directamente para encontrar un recurso. En caso que se le presente un URN al consumidor, es responsabilidad de la aplicación consumidora el resolver la referencia.

Nada en el uso de un URN implica disponibilidad del componente referenciado o su ubicación; puede ser que el componente esté definido localmente dentro del mensaje, sea accesible en forma remota a través de un servicio web, o directamente a través del acceso a la base de datos.

En general, se seguirá los siguientes tres pasos:

1. El destinatario de los datos utilizará el identificador del componente referenciado para buscar en su base de datos local.
2. Si no existe dicho componente en su conjunto de datos a nivel local, se haría la búsqueda del componente referenciado en el conjunto de datos entrante.
3. Si no se encuentra el componente en la forma arriba indicada, el sistema haría la búsqueda en las fuentes de datos conocidas para resolver la referencia.

Este tipo de referencia está limitada en cuanto a su apertura, ya que requiere la lógica de la aplicación para su resolución. En tal sentido, se espera que su uso vaya disminuyendo con el tiempo, ya que el dominio de la información aeronáutica está evolucionando hacia soluciones basadas en servicios web. Al utilizar las normas concretas antes mencionadas, cualquier sistema que cumpla con las normas podrá resolver las referencias AIXM sin una lógica de aplicación adicional.

3.4.1 Uso del UUID

Se recomienda que el URN esté basado en el UUID del componente referenciado, como en el siguiente ejemplo.

```
    <aixm:clientAirspace xlink:href="urn:uuid:a82b3fc9-4aa4-4e67-8def-
aaea1ac595j"/>
```

En este caso, la aplicación consumirá el localizador basado en el URN e, internamente, descubrirá la definición del espacio aéreo referenciado a través de registros de datos, codificación manual u otros métodos específicos para los sistemas involucrados. El comienzo del URN debería coincidir con el codeSpace (Espacio de Código) utilizado para el gml:identifier.

3.4.2 Uso de claves naturales

Cuando no hay UUID disponibles, se podría utilizar un URN específico para el AIXM, con claves naturales, como en el siguiente ejemplo:

```
<aixm:clientAirspace xlink:href="urn:aixm:Airspace(gml:timePosition=2010-
04-07T09:00;aixm:type=D;aixm:designator=EBD25A)"/>
```

Aparte de ser mucho más complejos que aquéllos basados en el UUID, la desventaja de los URN basados en claves naturales es que requieren un código específico para descodificar e identificar al componente objetivo. Por lo tanto, se debería utilizar el URN con claves naturales únicamente durante los períodos de transición y a una escala limitada. Por ejemplo, podría ser una solución utilizar un URN basado en claves naturales entre sistemas que almacenan los datos en formatos heredados y que no tienen la posibilidad de trabajar con valores UUID.

Se deberá aplicar la siguiente regla en la composición del URN "aixm":

urn:aixm:***Feature***(timePosition=***time_value***;***property*_name**=***property_value***;…)

donde:

- ***Feature*** es el nombre de un componente AIXM, tal como se define en el esquema XML AIXM; por ejemplo: AeropuertoHelipuerto, EspacioAéreo, Pista, etc.
- ***time_value*** es un valor de fecha y hora UTC en el formato aaaa-mm-ddThh:mm e indica el momento en que el BASELINE TImeSlice (Fracción de Tiempo de LINEA DE BASE) del componente AIXM tenía el(los) ***property_value*(s)** que aparecen en la composición del URN;
- ***property_name*** es el nombre de la propiedad de un componente AIXM que compone una clave natural para dicho componente; el nombre de la propiedad deberá deletrearse de conformidad con el esquema XML AIXM, exactamente como aparece como elemento hijo del TimeSlice del componente AIXM.

  Se asume que todas las propiedades de clave natural son elementos hijos directos del TimeSlice del componente o que existen sólo una vez en el árbol XML (como el gml:pos de una ayuda para la navegación).

- ***property_value*** es el valor de la propiedad identificada por ***property_name****,* tal como se define en el TimeSlice de LINEA DE BASE de dicho componente, vigente en la fecha y hora especificadas por el ***time_value***;

  En algunas situaciones, esto incluye propiedades que no tienen un valor directamente, pero tienen un atributo xlink:href que apunta hacia otro componente. En este caso, el URN del componente referenciado deberá ser utilizado como property_value. Un ejemplo típico es el componente Pista (Runway), para el cual la clave natural incluye el AeropuertoHelipuerto (AirportHeliport) en el cual está ubicada. El URN se verá como en el siguiente ejemplo:

```
urn:aixm:Runway(gml:timePosition=2010-12-
20T16:32;aixm:designator=02%2F20;aixm:associatedAirportHelipor
t=urn:aixm:AirportHeliport(gml:timePosition=          2010-12-
20T16:32;aixm:designator=EBBR))
```

  Nótese que el valor de una propiedad de clave natural de componente AIXM podría contener caracteres que no están permitidos en la composición del URN, tal como se explica en la Sintaxis URN (RFC 2141) y tienen que ser reemplazados con su código hexadecimal, con el prefijo "%". Tal es el caso del carácter "/", que, típicamente, es utilizado en los designadores de pista. Por lo tanto, en el ejemplo anterior, el designador de pista "02/20" fue codificado como "02%2F20", donde "%2F" es la representación hexadecimal de "/".

Para que el URN sea válido en términos del RFC 2141 (Sintaxis URN), el URN “aixm” tendría que estar registrado ante la Autoridad de Números Asignados por Internet *(Internet Assigned Numbers Authority).* Hasta entonces, será considerado como un URN no normalizado (experimental).

3.5 Uso del xlink:title

El valor del atributo xlink:title en un xlink es una descripción legible del valor referenciado. En este caso, sería una descripción legible del componente aeronáutico referenciado.

Se sugiere el uso del xlink:title, especialmente en aquellos casos en que el componente referenciado es definido en forma remota. El título debería ser un nombre legible del componente que pueda ser utilizado internamente por las aplicaciones para fines de visualización. Se desalienta el uso de xlink:title para la identificación automática del componente.

```
      <aixm:clientAirspace xlink:href="urn:uuid:a82b3fc9-4aa4-4e67-8def-
aaea1ac595j" xlink:title=”Gabbs North MOA”/>
```

A.1. Algoritmos UUID

A.1.1 Introducción

Este Apéndice analiza las cuatro principales versiones del UUID; cómo son generadas, su eficiencia y su carga computacional. En base a esta comparación, se ha determinado que la versión 4 UUIDI, basada en la generación de números aleatorios, es la más eficiente. La tendencia en la industria es hacia el uso de la versión 4, con la notable excepción de Oracle, que sigue en la versión 1. La Figura 7 contiene un resumen de una encuesta referente a las versiones del UUID/Identificador Unico Universal (GUID) que tienen soporte de los principales productos y bibliotecas de soporte lógico.

A.1.2 Definición y Singularidad

Un UUID es un número de 128 bitios, codificado ya sea con un número aleatorio, el producto de una función hash criptográfica, o una combinación de número aleatorio y el momento de generación. Los UUID son elaborados convencionalmente o presentados en su forma 'canónica', que es una secuencia de 32 dígitos hexadecimales agrupados en una secuencia de 8, 4, 4, 4 y 12 dígitos; a continuación, se ofrece un ejemplo.

```
550e8400-e29b-41d4-a716-446655440000
```

El dígito ubicado más a la izquierda de la sarta representa los cuatro bitios más importantes del UUID.

La finalidad del UUID aparece mejor descrita en http://en.wikipedia.org/wiki/UUID : "*El propósito de los UUID es permitir que los sistemas distribuidos identifiquen de manera singular la información, sin necesidad de una coordinación central significativa. ..... …..Cualquiera puede crear un UUID y utilizarlo para identificar algo con una razonable confianza que el identificador nunca será utilizado intencionalmente por alguien para otro fin.*" El UUID está definido en tres normas compatibles: ISO/IEC 11578:1996, UIT-T Rec. X.667 | ISO/IEC 9834-8:2005 y IETF RFC 4122.

La razón para utilizar un campo numérico tan grande como [1-2^{122}] es que el UUID es ***universal***, es decir, la probabilidad de encontrar un UUID duplicado dentro del universo de la informática en el futuro previsible debe ser muy baja.

A.1.3 Formato y versiones

El formato UUID aparece ilustrado en la Figura 1. Seis de los 128 bitios son utilizados para especificar el tipo de UUID, dejando 122 para que lleven un número aleatorio, el producto de una función hash criptográfica o una combinación de una dirección MAC de Tarjeta de Interfaz de Red (Network Interface Card - NIC) y la hora de generación, dependiendo de la versión del UUID. Los seis bitios de control están ubicados en forma complicada, y ni siquiera son contiguos, haciendo que la generación e interpretación de los UUID sean más complicadas de lo necesario. Los dígitos están numerados empezando por el dígito ubicado más a la izquierda de la forma canónica (sarta). El tipo de UUID está definido primero por el valor del campo variante (YY), que ocupa los dos bitios más significativos del 17º. dígito hexadecimal.

Nota: El *RFC 4122, párrafo 4.1.1, muestra el campo variante como poseedor de tres bitios. Para todos los UUID normalizados, el bitio menos significativo de estos tres es irrelevante; es utilizado como el bitio superior en la secuencia de reloj en los UUID de versión 1, o es parte del campo aleatorio o hash en los UUID de versión 3,4 ó 5.*

Un valor de 2 en el campo variante indica que el UUID es una de las versiones normalizadas. Un valor cero del bitio más significativo del campo variante significa que el UUID fue generado por una estación de trabajo del Sistema Computacional de Red Apollo (*Apollo Network Computing System*). Un valor de 3 del campo variante indica un UUID que fue utilizado por **.**COM en versiones de Microsoft Windows antes de Windows 2000. El campo de V*ersión* (VVVV) está codificado en el 13er dígito, permitiendo hasta 15 versiones de UUID. Actualmente, existen cinco versiones, numeradas del 1 al 5.

Dígitos hexade-cimales	1	2	3	4	5	6	7	8
1 a 8	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
9 a 16	XXXX	XXXX	XXXX	XXXX	**VVVV**	XXXX	XXXX	XXXX
17 a 24	**YY**XX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
25 a 31	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX

Figura 1 – Formato general del UUID

Las normas UUID no requieren que los UUID generados por un determinado sistema sean todos de la misma versión. Un sistema distribuido cerrado (es decir, que no es universal) podría explotar esto, permitiendo la generación de cualquiera de las cuatro principales versiones. Así, el grado de singularidad de los UUID se cuadruplicaría, de un campo de' [1-2^{122}] a un campo efectivo de [1-2^{124}].

A.1.4 Versión 1 del UUID

El formato de la versión 1 es el más antiguo y el más complicado, pero aún es usado extensamente; la codificación aparece en la Figura 2. Los 60 bitios rotulados como 't' representan la hora en que se creó el UUID, en incrementos de 100 nanosegundos ($1x10^{-7}$). Los 48 bitios rotulados como 'm' son la dirección MAC (universalmente única) de una de las Tarjetas de Interfaz de Red (Network Interface Cards - NIC) en el sistema; cuando esto no está disponible, se genera un número aleatorio o seudo-aleatorio de 48 bitios. Los 14 bitios de la secuencia de reloj rotulados como 'c' son utilizados para reducir la posibilidad de generación de un valor UUID duplicado luego de haber ajustado la hora del reloj en sentido contrario (luego de un corte eléctrico) o de cambiar la tarjeta NIC. El campo permite 2^{14} ó >16,000 reajustes durante el transcurso de la vida del sistema. En el caso improbable que se conozca la secuencia del reloj justo antes del evento, ésta puede ser utilizada para la recuperación, y simplemente se incrementa; en caso contrario, la secuencia del reloj será reiniciada con un número aleatorio.

Dígitos hexade-cimales	1	2	3	4	5	6	7	8
1 a 8	tttt	tttt	tttt	tttt	tttt	tttt	tttt	tttt
9 a 16	tttt	tttt	tttt	tttt	**0001**	tttt	tttt	tttt
17 a 24	**10**cc	cccc	cccc	cccc	mmmm	mmmm	mmmm	mmmm
25 a 31	mmmm	mmmm	mmmm	mmmm	mmmm	mmmm	mmmm	mmmm

Figura 2 – Formato del UUID Versión 1 – Hora y dirección

El campo de tiempo en la versión 1 del UUID indica el tiempo transcurrido, en incrementos de 100 nanosegundos, desde el inicio del calendario Gregoriano en octubre de 1582. El valor máximo (sin signo) de 2^{60} nanosegundos equivale, aproximadamente, a 3,663 años. Actualmente, el conjunto de bitios de más alto orden en un campo de tiempo de la versión 1 de UUID es el bitio no. 57. El bitio no. 58 y mayores no serán utilizados hasta después de 2444.

A.1.5 Versión 2 del UUID

La versión 2 del UUID es utilizada en el Ambiente Computacional Distribuido POSIX de Interfaz del Sistema Operativo Portátil IEEE (*IEEE Portable Operating System Interface POSIX Distributed Computing Environment*). El formato es muy similar a la versión 1.

A.1.6 Versión 4 del UUID

La Figura 3 muestra la codificación de un UUID versión 4. Los 122 bits rotulados como 'r' comprenden un número seudo-aleatorio o, de preferencia, un número aleatorio de calidad criptográfica.

Dígitos hexade-cimales	1	2	3	4	5	6	7	8
1 a 8	rrrr	rrrr	rrrr	rrrr	rrrr	rrrr	rrrr	rrrr
9 a 16	rrrr	rrrr	rrrr	rrrr	**0100**	rrrr	rrrr	rrrr
17 a 24	**10**rr	rrrr	rrrr	rrrr	rrrr	rrrr	rrrr	rrrr
25 a 31	rrrr	rrrr	rrrr	rrrr	rrrr	rrrr	rrrr	rrrr

Figura 3 – Formato de un UUID versión 4 – Número aleatorio

Luego que un sistema distribuido comienza a generar UUID de versión 4, si se utiliza una fuente de números aleatorios de calidad criptográfica, la probabilidad que genere un UUID que sea el duplicado de otro ya existente p(n;d) está dada por la expresión $1 - e^{-nxn/2d}$, donde el número ya generado es 2^n y d es la cantidad de bitios ocupados por el número aleatorio. La probabilidad de colisión será mayor si se utilizan números seudo-aleatorios generados por el sistema.

Como ejemplo, luego de haber generado 243 (≈8.8 billones) UUID, la probabilidad que el siguiente UUID sea el duplicado de uno ya existente es ≈ $7.276x10^{-12}$. La Figura 4 brinda los p(n;d) para valores de n: 36, 41, 43 y 46 y valores de d: 90, 106 y 122 {calculados con el Lenguaje Estadístico R}. La cuarta columna de la tabla muestra que, si se toma 4 octetos del campo del número aleatorio para otro fin, se generaría una significativa probabilidad de colisión.

	d = 122	d = 106	d = 90
n = 36	$4.44x10^{-16}$	$2.91x10^{-11}$	$1.19x10^{-6}$
n = 41	$4.55x10^{-13}$	$2.98x10^{-8}$	$1.95x10^{-3}$
n = 43	$7.28x10^{-12}$	$4.77x10^{-7}$	$3.08x10^{-2}$
n = 46	$4.66x10^{-10}$	$3.05x10^{-5}$	$8.65x10^{-1}$

Figura 4 – Probabilidad de colisión para los UUID Versión 4 (Número Aleatorio)

A.1.7 Versiones 3 y 5 del UUID

La codificación de un UUID versión 3 ó 5 aparece en la figura 5. Los bitios rotulados como 'h' son el resultado de una función hash criptográfica MD5 en los UUID de versión 3 y de una función hash criptográfica SHA-1 en los UUID versión 5. Los valores hash de 128 bitios (MD5) y 160 bitios (SHA-1) son truncados a los 122 bitios disponibles en el UUID.

Dígitos hexade-cimales	1	2	3	4	5	6	7	8
1 a 8	hhhh	hhhh	hhhh	hhhh	hhhh	hhhh	hhhh	hhhh
9 a 16	hhhh	hhhh	hhhh	hhhh	**0011/0101**	hhhh	hhhh	hhhh
17 a 24	**10**rr	hhhh	hhhh	hhhh	hhhh	hhhh	hhhh	hhhh
25 a 31	hhhh	hhhh	hhhh	hhhh	hhhh	hhhh	hhhh	hhhh

Figura 5 – Formato de un UUID Versión 3 ó 5

Se puede encontrar información sobre el algoritmo MD5 en http://www.w3.org/TR/1998/REC-DSig- label/MD5-1_0, y sobre el algoritmo SHA-1 en http://www.itl.nist.gov/fipspubs/fip180-1.htm. Ambos algoritmos han sido 'descifrados' por ataques sistemáticos; en el caso del MD5, dentro de 2^{43} operaciones hash y, en el caso del SHA-1, dentro de 2^{63} operaciones hash. Esto está muy por debajo del número de operaciones en que se garantiza ocurrirá una colisión, 2^{64} y 2^{80} , respectivamente. No obstante, esto no tiene implicancias para el uso de números hash en los UUID, donde son utilizados más por conveniencia que por seguridad, aparte de sugerir que la singularidad de un UUID generado con cualquiera de estos algoritmos será inferior a aquélla de un UUID versión 4, creado con un número aleatorio de calidad criptográfica.

A.1.8 Comparación de versiones del UUID

Eficiencia en el uso de los bitios

El algoritmo de tiempo utilizado en los UUID versión 1 es ineficiente en su granularidad de 100 nanosegundos. El bitio no. 56 del campo de tiempo de 60 bitios no fue utilizado hasta algún momento de 1986, y el bitio no. 57 no será utilizado hasta 2444. Esto significa que, efectivamente, los 4 bitios superiores no están siendo utilizados. Asimismo, el campo de la secuencia de tiempo de 14 bitios en el UUID versión 1 sólo se utiliza cada vez que el sistema es activado, permitiendo 2^{14} ó >16,000 reajustes durante la vida del sistema. Un número inferior de bitios proporcionaría una cantidad suficiente de reajustes, dado que el reciclado de este campo no produciría UUID con una hora de generación ambigua.

A diferencia del UUID versión 1, en los UUID de las versiones 3, 4 y 5, todos los 122 bitios disponibles son utilizados para contener ya sea un número aleatorio o el producto de una función hash.

Carga computacional

En abril de 2007, en una prueba en que se comparó el tiempo que tomaba generar UUID de las versiones 1, 3 y 4 {http://johannburkard.de/blog/programming/java/Java-UUID-generators-compared.html}, se obtuvo los siguientes tiempos promedios de generación de 1 millón de UUID con soporte lógico Java normalizado:

Versión 1 basada en el tiempo	5,432 milisegundos
Versión 3 basada en MD5	40, 788 milisegundos
Versión 4 basada en números aleatorios	48, 900 milisegundos

Figura 6 – Tiempo promedio de generación de los UUID

Estos resultados muestran que las versiones 3 y 4 del UUID tienen una carga de procesamiento sustancialmente más elevada que los UUID de la versión 1: por un factor de aproximadamente 7.5 para la versión 4 y un factor de aproximadamente 9 para la versión 3.

El algoritmo SHA-1 genera un hash de 160 bitios comparado con los 128 bitios generados por el algoritmo MD5. Cuando se utilizan en un UUID, ambos hashes son truncados a 122 bitios, de manera que cualquier ventaja en cuanto a seguridad o aleatoriedad que pudiera tener el SHA-1 con respecto al MD5 se pierde cuando se utiliza en un UUID.

Singularidad

El campo de sello de tiempo en el UUID versión 1 no es un número aleatorio. Cada bitio cambia a una frecuencia que es la mitad del bitio precedente menos significativo; el bitio no. 48 y superiores cambian con una frecuencia de menos de una vez al año. Más significativo es la dedicación de 14 bitios al campo de secuencia de reloj, que sólo se incrementa cada vez que un sistema es activado nuevamente. Consecuentemente, los UUID versión 1 son sustancialmente menos singulares que los UUID versión 3 ó 4.

La singularidad de los UUID de versión 3 ó 5 no puede ser mayor que la singularidad de los UUID de versión 4, y podría ser inferior, dado el hecho que ambos pueden verse comprometidos luego de una cantidad de usos que está muy por debajo del número al cual se podría esperar una duplicidad ("ataque con fuerza bruta"): 2^{43} en vez de 2^{64} en el caso del MD5 y 2^{63} en vez de 2^{80} en el caso del SHA-1.

A.1.9 Apoyo que brinda el soporte lógico común

A veces, es difícil establecer qué versiones del UUID cuentan con el soporte de una determinada aplicación de soporte lógico, ya que muchos usuarios e, inclusive, anuncios publicitarios de productos, no parece saber de la existencia de cinco versiones diferentes. Típicamente, la producción de un generador de UUID se describe únicamente como varios grupos de dígitos hexadecimales. Notorio es el caso de Oracle en este sentido, que simplemente define su UUID como un valor crudo de 16 bytes. En el caso de Microsoft, fue difícil desentrañarlo, ya que los UUID son utilizados ampliamente dentro de los paquetes Microsoft; sólo una de las muchas interfaces en el Modelo de Objetos Componentes de Microsoft (*Microsoft Component Object Model - COM*) y, ahora, en el marco de .NET, se utiliza el tipo de UUID incluido en este estudio. Microsoft cambió de utilizar la versión 1 a la versión 4, con la introducción de .NET y Windows 2000.

Si bien un paquete de aplicación en particular puede no soportar una determinada versión de UUID, en muchos casos, se puede agregar una extensión extraída de bibliotecas como el Proyecto de Soporte Lógico Abierto basado en UNIX (*UNIX-based Open Source Software Project - OSSP*) para apoyar dicha versión.

	Versión 1	Versión 3	Versión 4	Versión 5	Variante
Microsoft .COM					**3**
Microsoft Windows 2000			**X**		
Oracle	**X**				
Java (JUG)	**X**	**X**	**X**	**X**	
Java J2SE5		**X**	**X**		

JavaScript **uuid.js**	**X**				
Linux (ossp)	**X**	**X**	**X**	**X**	
MySQL **Versión 1**	**X**				
MySQL **actual**			**X**		
PostgreSQL			**X**		
Apache **(Proyecto Jakarta)**	**X**	**X**	**X**	**X**	
OpenPKG	**X**	**X**	**X**		
Python	**X**	**X**	**X**	**X**	
Ruby	**X**	**X**	**X**	**X**	
C++ (ossp)	**X**	**X**	**X**	**X**	

Figura 7 – Versiones de UUID que cuentan con apoyo del soporte lógico común