Unión Internacional de Telecomunicaciones

UIT-T

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

X.782

(05/2012)

SERIE X: REDES DE DATOS, COMUNICACIONES DE SISTEMAS ABIERTOS Y SEGURIDAD

Gestión de interconexión de sistemas abiertos – Funciones de gestión y funciones de arquitectura de gestión distribuida abierta

Directrices en materia de definición de servicios web para objetos gestionados e interfaces de gestión

Recomendación UIT-T X.782

Unión Internacional de Telecomunicaciones

RECOMENDACIONES UIT-T DE LA SERIE X

REDES DE DATOS, COMUNICACIONES DE SISTEMAS ABIERTOS Y SEGURIDAD

REDES PÚBLICAS DE DATOS	
Servicios y facilidades	X.1–X.19
Interfaces	X.20–X.49
Transmisión, señalización y conmutación	X.50–X.89
Aspectos de redes	X.90–X.149
Mantenimiento	X.150–X.179
Disposiciones administrativas	X.180–X.199
INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Modelo y notación	X.200–X.209
Definiciones de los servicios	X.210–X.219
Especificaciones de los protocolos en modo conexión	X.220–X.229
Especificaciones de los protocolos en modo sin conexión	X.230–X.239
Formularios para declaraciones de conformidad de implementación de protocolo	X.240–X.259
Identificación de protocolos	X.260–X.269
Protocolos de seguridad	X.270–X.279
Objetos gestionados de capa	X.280–X.289
Pruebas de conformidad	X.290–X.299
INTERFUNCIONAMIENTO ENTRE REDES	
Generalidades	X.300–X.349
Sistemas de transmisión de datos por satélite	X.350–X.369
Redes basadas en el protocolo Internet	X.370–X.379
SISTEMAS DE TRATAMIENTO DE MENSAJES	X.400–X.499
DIRECTORIO	X.500–X.599
GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS	
Gestión de redes	X.600–X.629
Eficacia	X.630–X.639
Calidad de servicio	X.640–X.649
Denominación, direccionamiento y registro	X.650–X.679
Notación de sintaxis abstracta uno	X.680–X.699
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Marco y arquitectura de la gestión de sistemas	X.700–X.709
Servicio y protocolo de comunicación de gestión	X.710–X.719
Estructura de la información de gestión	X.720–X.729
Funciones de gestión y funciones de arquitectura de gestión distribuida abierta	**X.730–X.799**
SEGURIDAD	X.800–X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Compromiso, concurrencia y recuperación	X.850–X.859
Procesamiento de transacciones	X.860–X.879
Operaciones a distancia	X.880–X.889
Aplicaciones genéricas de la notación de sintaxis abstracta uno	X.890–X.899
PROCESAMIENTO DISTRIBUIDO ABIERTO	X.900–X.999
SEGURIDAD DE LA INFORMACIÓN Y DE LAS REDES	X.1000–X.1099
APLICACIONES Y SERVICIOS CON SEGURIDAD (1)	X.1100–X.1199
SEGURIDAD EN EL CIBERESPACIO	X.1200–X.1299
APLICACIONES Y SERVICIOS CON SEGURIDAD (2)	X.1300–X.1499

Para más información, véase la Lista de Recomendaciones del UIT-T.

Recomendación UIT-T X.782

Directrices en materia de definición de servicios web para objetos gestionados e interfaces de gestión

Resumen

La presente Recomendación define un conjunto de directrices en materia de modelización de objetos gestionados y una interfaz de gestión de redes basadas en servicios web. En conjunción con la Recomendación UIT-T Q.818, compone un marco para interfaces de gestión de redes basadas en servicios web y especifica el modo en que han de definirse las interfaces de servicios web orientadas a los servicios. Además, comprende un supuesto de aplicación procedente de los servicios web en las interfaces de gestión de redes, una serie de métodos genéricos de acceso a objetos gestionados basados en XML y técnicas de modelización de información utilizando los esquemas XML y WSDL en la esfera de los servicios web. En la presente Recomendación se facilitan definiciones WSDL y esquemas XML con miras a la determinación de ciertos tipos de datos básicos, entre ellos, los objetos gestionados genéricos y los métodos de acceso de los mismos. Las Recomendaciones UIT-T X.782 y Q.818 componen un marco para interfaces de gestión de redes basadas en servicios web con una amplia gama de aplicaciones.

Historia

Edición	Recomendación	Aprobación	Comisión de Estudio	ID único*
1.0	ITU-T X.782	2012-05-14	2	11.1002/1000/11625
1.1	ITU-T X.782 (2012) Cor. 1	2013-03-16	2	11.1002/1000/11890

Palabras clave

Esquema XML, interfaces de gestión de redes, lenguaje de descripción de servicios Web (WSDL), lenguaje extensible de marcado (XML), objetos gestionados, procesamiento distribuido, servicios web (WS).

* Para acceder a la Recomendación, sírvase digitar el URL http://handle.itu.int/ en el campo de dirección del navegador, seguido por el identificador único de la Recomendación. Por ejemplo, http://handle.itu.int/11.1002/1000/11830-en.

PREFACIO

La Unión Internacional de Telecomunicaciones (UIT) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones y de las tecnologías de la información y la comunicación. El Sector de Normalización de las Telecomunicaciones de la UIT (UIT-T) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Asamblea Mundial de Normalización de las Telecomunicaciones (AMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución 1 de la AMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

La observancia de esta Recomendación es voluntaria. Ahora bien, la Recomendación puede contener ciertas disposiciones obligatorias (para asegurar, por ejemplo, la aplicabilidad o la interoperabilidad), por lo que la observancia se consigue con el cumplimiento exacto y puntual de todas las disposiciones obligatorias. La obligatoriedad de un elemento preceptivo o requisito se expresa mediante las frases "tener que, haber de, hay que + infinitivo" o el verbo principal en tiempo futuro simple de mandato, en modo afirmativo o negativo. El hecho de que se utilice esta formulación no entraña que la observancia se imponga a ninguna de las partes.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT no ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB en la dirección http://www.itu.int/ITU-T/ipr/.

ÍNDICE

Página

Recomendación UIT-T X.782

Directrices en materia de definición de servicios web para objetos gestionados e interfaces de gestión

1 Alcance

La arquitectura de gestión de redes definida en la Recomendación [UIT-T M.3010] introduce la utilización de múltiples protocolos de gestión. Hasta la fecha, los protocolos GDMO/CMIP y GIOP/IIOP de CORBA figuraban entre las opciones disponibles en la capa de aplicación. Con arreglo a la metodología de especificación de interfaces de gestión definida en la Recomendación [UIT-T M.3020], cabe la posibilidad de introducir paradigmas tecnológicos adicionales en las interfaces de gestión de redes y añadir el servicio web/XML a la lista de paradigmas para la gestión de redes.

Las Recomendaciones UIT-T X.782 y Q.818 tienen por objeto definir un marco para la especificación de metodologías de modelización de interfaces soportadas por sistemas de gestión y elementos de red utilizando servicios web y/o esquemas XML. El alcance de la presente Recomendación comprende la provisión de las siguientes directrices o instrucciones:

– metodología de diseño de interfaces basadas en WSDL y orientadas a los servicios;
– supuestos de aplicación procedente e improcedente de los servicios web en las interfaces de gestión de redes;
– métodos genéricos de acceso a objetos gestionados;
– herencia de interfaces y objetos gestionados;
– directrices en materia de modelización de información para interfaces basadas en servicios web, y
– convenciones de estilo para especificaciones de esquemas XML y WSDL en la esfera de los servicios web.

2 Referencias

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes. En esta Recomendación, la referencia a un documento, en tanto que autónomo, no le otorga el rango de una Recomendación.

[UIT-T E.164] Recomendación UIT-T E.164 (2010), *Plan internacional de numeración de telecomunicaciones públicas*.

[UIT-T M.3010] Recomendación UIT-T M.3010 (2000), *Principios para una red de gestión de las telecomunicaciones*.

[UIT-T M.3020] Recomendación UIT-T M.3020 (2011), *Metodología para la especificación de interfaces de gestión*.

[UIT-T M.3701] Recomendación UIT-T M.3701 (2010), *Servicios comunes de gestión – Gestión de estado – Requisitos y análisis independientes del protocolo*.

[UIT-T Q.818] Recomendación UIT-T Q.818 (2012), *Servicios de gestión basados en el servicio de la web*.

[UIT-T X.701] Recomendación UIT-T X.701 (1997), *Tecnología de la información – Interconexión de sistemas abiertos – Visión general de la gestión de sistemas*.

[UIT-T X.703] Recomendación UIT-T X.703 (1997), *Tecnología de la información – Arquitectura de gestión distribuida abierta*.

[ATIS-I-000002] Especificación ATIS-I-000002 (2011), *ATIS XML Schema Development Guidelines*.

[OASIS WSN] Especificación OASIS (2006), *Web Services Base Notification v1.3*.

[OASIS UDDI] Especificación OASIS (2004), *Universal Description, Discovery and Integration (UDDI) v3.0.2*.

[W3C Primer] Recomendación W3C (2004), *XML Schema Part 0: Primer Second Edition.*

[W3C SOAP] Recomendación W3C (2007), *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).*

[W3C WSDL] Recomendación W3C (2001), *Web Services Description Language (WSDL) 1.1*.

[W3C XML] Recomendación W3C (2000), *Extensible Markup Language (XML) 1.0 Second Edition*.

[W3C XS-P1] Recomendación W3C (2004), *XML Schema Part 1: Structures Second Edition*.

[W3C XS-P2] Recomendación W3C (2004), *XML Schema Part 2: Datatypes Second Edition*.

3 Definiciones

3.1 Términos definidos en otros documentos

En la presente Recomendación se utilizan los siguientes términos definidos en otros documentos:

3.1.1 agente [UIT-T M.3020]

3.1.2 clase de objeto gestionado [UIT-T X.701]

3.1.3 gestor [UIT-T M.3020]

3.1.4 notificación [UIT-T X.703]

3.2 Términos definidos en la presente Recomendación

En la presente Recomendación no se definen términos nuevos.

4 Siglas y acrónimos

En la presente Recomendación se utilizan las siglas y los acrónimos siguientes:

B2B De empresa a empresa (*business to business*)

BPEL Lenguaje de ejecución de procesos operativos (*business process execution language*)

C2B De cliente a empresa (*customer to business*)

CMIP Protocolo de información de gestión común (*common management information protocol*)

CORBA Arquitectura de negociación de petición de objetos comunes (*common object request and broker architecture*)

DCOM Modelo de objeto de componente distribuido (*distribute component object model*)

DN Nombre distinguido (*distinguished name*)

DTD Definición de tipos de documento (*document type definition*)

EMS Sistema de gestión de elementos (*element management system*)

GDMO Directrices para la definición de objetos gestionados (*guidelines for the definition of managed objects*)

GIOP Protocolo general de negociación de petición de objetos (*general inter-orb protocol*)

IDL Lenguaje de definición de interfaz (*interface definition language*)

IIOP Protocolo Internet inter-ORB (*Internet inter-orb protocol*)

IT Tecnologías de la información (*information technology*)

LAN Red de área local (*local area network*)

MO Objeto gestionado (*managed object*)

MOC Clase de objeto gestionado (*managed object class*)

MOO Operación de objetos múltiples (*multiple object operation*)

NMS Sistema de gestión de red (*network management system*)

OOAD Análisis y diseño orientados a objetos (*object-oriented analysis and design*)

OS Sistema operativo (*operating system*)

RDN Nombre distinguido relativo (*relative distinguished name*)

SOA Arquitectura orientada a los servicios (*service-oriented architecture*)

SOAP Protocolo simple de acceso a objetos (*simple object access protocol*)

TMN Red de gestión de las telecomunicaciones (*telecommunications management network*)

UDDI Descubrimiento e integración de descripción universal (*universal description discovery and integration*)

WS Servicios web (*web services*)

WSDL Lenguaje de descripción de servicios web (*web services description language*)

WSN Notificación de servicios web (*web services notification*)

XML Lenguaje extensible de marcado (*extensible markup language*)

XSD Definición de esquema XML (*XML schema definition*)

5 Convenios

Se incluyen algunos convenios en esta Recomendación para que el lector sea consciente del objeto del texto. Aunque la mayor parte de la Recomendación es normativa, los párrafos que establecen sucintamente requisitos obligatorios que debe cumplir el sistema de gestión (que gestiona y/o es gestionado) están precedidos de una "R" en negrita entre paréntesis, seguida de un nombre corto que indica el sujeto del requisito y de un número. Por ejemplo:

(R) EJEMPLO-1 Ejemplo de requisito obligatorio.

Los requisitos que pueden ser implementados facultativamente por un sistema de gestión están precedidos por una "O" en lugar de una "R". Por ejemplo:

(O) EJEMPLO-2 Ejemplo de requisito opcional.

Las declaraciones de requisitos se utilizan para crear perfiles de cumplimiento y conformidad.

La presente Recomendación incluye ejemplos de esquemas XML y WSDL, y el Anexo A comprende ejemplos de esquemas XML y WSDL de carácter normativo que especifican los tipos de datos, las clases básicas y otros conceptos de modelización orientada a los servicios del marco. Para la redacción de los ejemplos de esquemas XML y WSDL se ha utilizado la fuente "courier" de 10 puntos:

```
<!-- Example XML schema -->
<xsd:complexType name="AType">
   <xsd:sequence>
      <xsd:element name="a" type="xsd:string"/>
      <xsd:element name="b" type="xsd:long"/>
   </xsd:sequence>
</xsd:complexType>
```

6 Visión general del marco de gestión basado en los servicios web

La industria de las tecnologías de la información ha utilizado los servicios web con profusión. El Apéndice I contiene información adicional sobre las características de la tecnología de servicio web. Los servicios web análogos a la arquitectura CORBA pueden utilizarse en el marco de las interfaces de gestión de redes.

Las Recomendaciones UIT-T X.782 y Q.818 definen un marco para la especificación de metodologías de modelización de interfaces soportadas por sistemas de gestión y elementos de red utilizando esquemas XML y WSDL.

El marco de gestión basado en los servicios web comprende los aspectos indicados *infra*.

1) Directrices en materia de definición de interfaces y objetos gestionados:
   - principios para el diseño de interfaces basadas en WSDL y orientadas a los servicios;
   - definición de objetos gestionados utilizando un esquema XML;
   - métodos de acceso a objetos gestionados;
   - herencia de objetos gestionados y operaciones de interfaz;
   - directrices en materia de modelización de información para operaciones de interfaces basadas en servicios web, y
   - convenciones de estilo para especificaciones de esquemas XML y WSDL.
2) Servicios web de apoyo a los servicios de gestión de redes:
   - definición de los servicios de notificación que utilizan el sistema de notificación basado en los servicios web de OASIS [OASIS WSN];
   - utilización del registro de servicios UDDI de OASIS [OASIS UDDI];
   - definición de "servicio de latido";
   - definición de "servicio de operaciones de objetos múltiples", y
   - definición de "servicio de contenencia".

La presente Recomendación se centra en las directrices en materia de definición de interfaces y objetos gestionados, y la Recomendación [UIT-T Q.818] se ocupa principalmente de los servicios web de apoyo a los servicios de gestión de redes. Ambas Recomendaciones conforman un marco de gestión basado en los servicios web.

7 Principios para el diseño de interfaces basadas en WSDL y orientadas a los servicios

Este apartado versa sobre cuestiones relacionadas con el diseño de interfaces que el marco en cuestión debería abordar por conducto de las interfaces orientadas a los servicios. En ese sentido, contempla los principios de modelización de objetos gestionados y orientados a los servicios y sus métodos de acceso.

Las consideraciones en materia de diseño orientado a los servicios que guardan relación con la modelización y el repertorio de WSDL y XSD hacen referencia a superclases, denominaciones de objetos gestionados e interfaces, operaciones y notificaciones orientadas a los servicios.

Los servicios web figuran entre las tecnologías orientadas a los servicios y pueden comprarse con los paradigmas tradicionales de gestión CORBA/IDL y GDMO/CMIP. El análisis y el diseño de interfaces orientadas a los objetos (OOAD) se articulan en torno al nivel de clase, es decir, encapsulan comportamientos y datos conexos en un mismo objeto que expone una o más interfaces para acceder a sus estados y atributos encapsulados. El diseño de interfaz orientada a los servicios separa el estado y los datos encapsulados del comportamiento, permitiendo un bajo grado de acoplamiento a un nivel superior. En ese contexto, el comportamiento de algunos objetos (compartidos) no se halla bajo su propio control, sino a merced de un número reducido de operaciones de interfaz predefinidas.

En las Recomendaciones UIT-T X.782 y Q.818 se define una utilización genérica y superficial de los patrones de diseño de interfaces orientadas a los servicios. Las funciones de gestión y control se precisan utilizando un enfoque articulados en torno a los servicios, lo que implica que las operaciones de las interfaces se organizan en el marco de una unidad de un servicio determinado (en conjuntos de funciones de gestión, tales como la gestión de la configuración, la gestión del rendimiento, etc.) y no de una clase de objeto de gestión individual. Esta orientación a los servicios requiere de la flexibilidad que ofrece la granularidad de acceso típica de las aplicaciones, que permite acceder a conjuntos bien definidos de tipos de entidades TMN a través de interfaces WSDL predefinidas.

El marco se basa en los siguientes principios para la definición de interfaces y modelos de información de gestión basados en WSDL y XSD y orientados a los servicios:

– Todas las interacciones de la interfaz vienen definidas como operaciones WSDL, y cada operación incluye una solicitud y una respuesta correspondiente opcional, cuando procede.

– Todas las clases de objeto gestionado (MOC) vienen definidas como un *complexType* (tipo complejo) de XML cuando su intercambio se efectúa a través de la interfaz de gestión, y cada atributo o estado de la MOC viene definido como un elemento en el *complexType*.

– La denominación de las manifestaciones de MOC se ajusta al concepto de nombre distinguido, si bien constituye una cadena que engloba toda la lista de nombres distinguidos relativos.

– En este marco se define un servicio general, que comprende cinco métodos genéricos de acceso a los objetos, a saber: *createMO* (crear MO), *deleteMO* (suprimir MO), *getMOAttribute* (obtener atributo de MO), *setMOAttribute* (establecer atributo de MO) y *getPackages* (obtener paquetes). Estos métodos genéricos de acceso a los objetos emplean pares "nombre-valor" para expresar los valores y propiedades de distintas manifestaciones de MOC.

– Otras funciones de control de la interfaz vienen definidas como operaciones de interfaz WSDL y se organizan como un servicio con granularidad de conjuntos de funciones de gestión.

– Los tipos de datos comunes vienen definidos como esquemas XML que las definiciones de interfaz típicas de las aplicaciones pueden compartir.

– Las notificaciones enviadas del agente al gestor deben ajustarse al formato y las conductas definidos en la norma [OASIS WSN]. El control de los servicios de gestión de notificaciones se define con más detalle en la Recomendación [UIT-T Q.818].

8 Definición de objetos gestionados genéricos utilizando esquemas XML

8.1 Función de los servicios web en las interfaces de gestión

Cabe definir una clase básica con miras a su empleo en la modelización de los recursos de red y, de este modo, dar soporte a los objetos de software que representan recursos gestionables. Para poder operar en este marco, las demás MOC presentes en los modelos de información deben derivarse de dicha clase básica. Además, conviene definir métodos genéricos de acceso y servicios ampliados adicionales para la creación de interfaces que permitan gestionar los objetos gestionados.

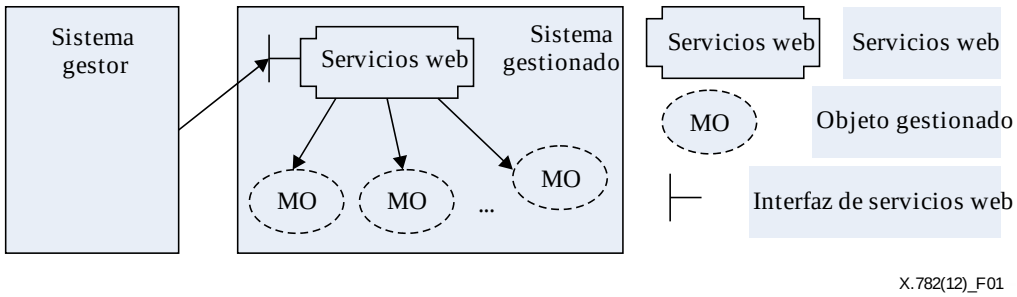

Figura 1 – Función de los servicios web

La Figura 1 ilustra el modo en que un sistema gestor accede a un sistema gestionado que da soporte a una interfaz de servicios web. La interfaz de servicios web actúa como una entidad intermedia que permite a un sistema gestor gestionar los MO correspondientes en un sistema gestionado que representa recursos gestionables.

8.2 Definición de objetos gestionados utilizando esquemas XML

Un objeto gestionado es la visión de gestión de OSI de un recurso que está sujeto a gestión, véanse una conexión o un elemento de equipo físico. Por consiguiente, los objetos gestionados constituyen una abstracción de este tipo de recursos, que representa sus propiedades a efectos de gestión. Los objetos gestionados pueden incluir atributos que proporcionan información autodescriptiva y operaciones que representan sus comportamientos. El propósito de este marco consiste en proporcionar un repertorio de capacidades para la gestión de estos objetos. Los objetos gestionados requieren ciertas metodologías de descripción de sus propiedades y comportamientos. Un objeto gestionado orientado a los servicios es una entidad gestionada que representa un recurso de servicio gestionable en términos de comportamiento y estado compartido, en cuyo contexto el estado y el comportamiento vienen separados mediante la externalización del comportamiento a una denominada "entidad gestora" (por ejemplo, un servicio y su interfaz) que asume un papel rector con respecto a los comportamientos de las entidades gestionadas que le han sido asignadas. Dada la posibilidad de separar el estado y el comportamiento de un objeto gestionado, el estado puede describirse mediante un esquema XML y el comportamiento mediante interfaces de servicios web en WSDL. La utilización de un documento XML para almacenar estados de objetos gestionados brinda una ventaja importante, a saber, que los servicios web usan esquemas XML para describir el tipo de datos de los mensajes que intercambian y que esa información de objeto gestionado basada en XML puede ser intercambiada sin modificación alguna.

8.2.1 Definición de clases de objetos gestionados genéricos

Una clase de objeto gestionado es una abstracción ulterior de una serie de objetos gestionados. Todos los recursos de red comparten ciertos atributos comunes que todas las MOC deben heredar, directa o indirectamente, de una superclase conocida como *ManagedObject* (objeto gestionado). La utilización de *ManagedObject* para definir nuevas MOC resulta más sencilla y rápida y facilita el mantenimiento. Según se indicó anteriormente, todas las MOC se describen utilizando esquemas XML. El tipo de datos y los atributos de la superclase *ManagedObject* se ilustran en los Cuadros 1 y 2, respectivamente.

Cuadro 1 – Tipo de datos de la superclase *ManagedObject*

```
<xsd:complexType name="ManagedObject_C">
  <xsd:sequence>
    <xsd:element name="objectClass" type="xsd:string"/>
    <xsd:element name="objectInstance" type="x782:NameType"/>
    <xsd:element name="packages" type="x782:PackageListType"/>
    <xsd:element name="creationSource" type="x782:SourceIndicatorType"/>
  </xsd:sequence>
</xsd:complexType>
```

Cuadro 2 – Atributos de la superclase *ManagedObject_C*

Nombre del atributo	Calificativo de soporte	Calificativo de lectura	Calificativo de escritura
objectClass	Obligatorio	Obligatorio	–
objectInstance	Obligatorio	Obligatorio	–
packages	Facultativo	Obligatorio	–
creationSource	Facultativo	Obligatorio	–

Según se indica en el Cuadro 2, la superclase *ManagedObject* se compone de cuatro atributos: *objectClass* (clase de objeto), *objectInstance* (manifestación de objeto), *packages* (paquetes) y *creationSource* (fuente de creación). Cada atributo tiene un valor asociado, cuya estructura puede ser simple o compleja. En este caso, los atributos de los objetos gestionados difieren de los atributos de la especificación de esquema XML y pueden ser simplemente asignados al elemento correspondiente del esquema XML. El atributo *objectClass* se utiliza para identificar el tipo de clase de la manifestación de objeto gestionado. El atributo *objectInstance* se utiliza para identificar unívocamente una manifestación de objeto gestionado, y el tipo de datos utiliza *NameType* (con la misma semántica del nombre distinguido) tal como se especifica en el apartado 1). Los paquetes de atributos son conjuntos de cadenas que indican las capacidades soportadas por los objetos gestionados. El atributo *creationSource* revela si el objeto gestionado ha sido creado automáticamente en un sistema gestionado o por un sistema gestor a través de una operación de gestión, o si es desconocido.

```
DN(list of xsd:string) ::= "<attriubte_name_1>=<attibute_value_1>",
             "<attribute_name_2>=<attribute_value_2>"
              …,
             " <attribute_name_n>=<attribute_value_n>"
```
(1)

Los atributos de la fórmula *supra* deberían denominar atributos de MOC.

En la cláusula A.1 se define íntegramente un esquema XML para la clase *ManagedObject_C* genérica.

(R) OBJETO-1. Todas las clases utilizadas para crear modelos de recursos en un sistema gestionado heredarán (directa o indirectamente) de la superclase *ManagedObject_C* antes descrita y vienen definidas mediante esquemas XML en la cláusula A.1. Se dará soporte a las capacidades antes descritas.

8.2.2 Herencia de objetos gestionados

Una clase de objeto gestionado puede definirse como una especialización de otra clase de objeto gestionado por medio de la herencia, al igual que sucede con el resto de los objetos gestionados que han de heredar directa o indirectamente de la superclase ManagedObject. La especialización de una clase de objeto gestionado implica que la subclase soportará todos los métodos y atributos definidos en la superclase. Habida cuenta de que los atributos de los objetos gestionados se describen utilizando esquemas XML, la herencia puede transmitirse mediante la extensión de los tipos de datos. Por ejemplo, si la MOC de equipo hereda directamente de la clase básica *ManagedObject*, el equipo puede heredar todos los atributos de *ManagedObject* por extensión y declarar por sí mismo otros atributos, tales como *userLabel* (etiqueta de usuario), según se indica en el Cuadro 3.

Cuadro 3 – Tipo de datos de *Equipment*_C por extensión

```
<xsd:complexType name="Equipment_C">
 <xsd:complexContent>
   <xsd:extension base="x782:ManagedObject_C">
     <xsd:sequence>
       <xsd:element name="userLabel" type="xsd:string"/>
       …
     </xsd:sequence>
   </ xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

Algunos objetos gestionados pueden necesitar herencias múltiples, cuyo uso no se recomienda puesto que el esquema XML solo admite herencias únicas para tipos de datos. En caso de requerirse la semántica de herencia múltiple, la MOC derivada debe recibir una herencia única de una de las MOC superiores y los atributos de la clase o las clases superiores deben agregarse manualmente.

8.2.3 Características de los paquetes

Cabe la posibilidad de utilizar paquetes para agrupar ciertas capacidades (por ejemplo, atributos conexos) o proporcionar capacidades de soporte condicional. Un paquete puede definirse como un *complexType* XSD con una "_P" como sufijo de nombre. Cuando se utiliza, el elemento puede disponer de un tipo de *complexType* que represente ese paquete, con *minOccurs="0" maxOccurs="1"* como calificativos, lo que indica que el paquete en cuestión puede ser condicional u opcional. En el Cuadro 4 se facilita un ejemplo de definición y utilización de paquetes.

Cuadro 4 – Tipo de datos de equipo por extensión

```
<!-- definition of a Package -->
<xsd:complexType name="StatePackage_P">
    <xsd:sequence>
      <xsd:element name="adminstrativeState"
type="x782:AdminstrativeStateType"/>
      <xsd:element name="operationalState" type="x782:OperationalStateype"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Equipment_C">
 <xsd:complexContent>
  <xsd:extension base="x782:ManagedObject_C">
    <xsd:sequence>
      <xsd:element name="equipmentId" type="xsd:string"/>
      <xsd:element name="userLabel" type="xsd:string"/>
      …
      <!-- usage of the Package -->
    <xsd:element name="statePackage" type="x782:StatePackage_P" minOccurs="0"
maxOccurs="1" /></xsd:sequence>
  </ xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

8.2.4 Atributos y tipos de datos comunes

En el siguiente cuadro se enumera una serie de atributos y tipos de datos comunes que este marco puede compartir.

Cuadro 5 – Atributos y tipos de datos estándar

Nombre del atributo	Tipo de datos	Descripción
administrativeState	*AdministrativeStateType*	Para obtener información más detallada, véase la Recomendación [UIT-T M.3701]
availabilityStatus	*AvailabilityStatusSetType*	Para obtener información más detallada, véase la Recomendación [UIT-T M.3701]
backedUpStatus	*BackedUpStatusType*	Para obtener información más detallada, véase la Recomendación [UIT-T M.3701]
controlStatus	*ControlStatusSetType*	Para obtener información más detallada, véase la Recomendación [UIT-T M.3701]
creationSource (Nota)	*SourceIndicatorType*	Para obtener información más detallada, véase la Recomendación [UIT-T M.3701]
externalTime	*ExternalTimeType*	
objectClass (Nota)	*ObjectClassType*	Indica una MOC

Cuadro 5 – Atributos y tipos de datos estándar

Nombre del atributo	Tipo de datos	Descripción
objectInstance (Nota)	*NameType*	Indica una manifestación de objeto gestionado
operationalState	*OperationalStateType*	Para obtener información más detallada, véase la Recomendación [UIT-T M.3701]
packages (Nota)	*StringSetType*	Indica los paquetes soportados por una manifestación de objeto gestionado
proceduralStatus	*ProceduralStatusSetType*	Para obtener información más detallada, véase la Recomendación [UIT-T M.3701]
standbyStatus	*StandbyStatusType*	Para obtener información más detallada, véase la Recomendación [UIT-T M.3701]
systemLabel	*SystemLabelType*	Indica una etiqueta para un sistema.
unknownStatus	*UnknownStatusType*	Para obtener información más detallada, véase la Recomendación [UIT-T M.3701]
usageState	*UsageStateType*	Para obtener información más detallada, véase la Recomendación [UIT-T M.3701]
NOTA – Todos los objetos gestionados heredan estos atributos.		

Las definiciones XSD detalladas para los tipos de datos antes mencionados figuran en la cláusula A.1.

9 Métodos de acceso a objetos gestionados

En el presente apartado se describe un marco de gestión de redes basado en servicios web, que facilitará un repertorio de métodos de control de recursos de red. Estos métodos proporcionan capacidades básicas para la gestión de objetos gestionados, motivo por el cual se los denomina métodos genéricos de acceso (véase el Cuadro 6). La Figura 1 ilustra el procedimiento de acceso y el marco utiliza tecnología de servicios web para intercambiar la información relativa a dichos objetos. Los servicios web separan los estados y comportamientos de los objetos gestionados y exponen sus comportamientos a través de una interfaz de servicios web. Habida cuenta de que un servicio web es un tipo de tecnología orientado a los servicios, el diseño de todos los objetos gestionados de este marco permite el acceso a los mismos a través de una interfaz única, que habrá de saber qué objeto es el objetivo real de una operación. En ese sentido, el identificador único del objeto gestionado objetivo debe proporcionarse con cada solicitud de acceso. De acuerdo con los requisitos anteriores, el Cuadro 6 prevé una serie de métodos genéricos de acceso necesarios.

Cuadro 6 – Métodos genéricos de acceso

Nombre de la operación	Parámetro de entrada	Parámetro de salida
getMOAtrributes	– *objectInstance* : Nombre – *attributeNameList* : SECUENCIA DE cadena	– *attributeNameAndValueList* : SECUENCIA DE {*attributeName, attributeType, attributeValue*} – *status* : ENUMERACIÓN
setMOAttributes	– *objectInstance* : DN – *attributeNVMList* : SECUENCIA DE {*attributeName, attributeType, attributeValue, modifyOption*}	– *status*
createMO	– *objectclass* – *objectClassInstance* – *attributeNameAndValueList*	– *status*
deleteMO	– *objectInstance*	– *status*
getPackages	– *objectInstance*	– *packages* : Lista de cadena – status

Siendo:

1) *getMOAttributes* – La operación que permite recuperar todos los valores de los atributos de un objeto gestionado, o un subconjunto de los mismos. En este caso, se utiliza el DN como primer parámetro para identificar unívocamente el objeto gestionado y una lista de nombres de atributo para su consulta. La respuesta resultante se compone de los valores de los atributos y el estado de la operación. El atributo *NameAndValueList* (lista de valores y nombres) es una lista articulada en torno a tres elementos: *attributeName* (nombre del atributo), *attributeType* (tipo de atributo) y *attributeValue* (valor del atributo). *AttributeType* indica el tipo original de *attributeValue*, y los valores de los atributos se remiten por conducto del elemento "*any*" (cualquiera) del esquema XML para los valores de tipo arbitrario. El parámetro *status* (estado) indica si la operación se ha realizado con éxito o no. Habida cuenta de que "any" se define para el tipo de datos del valor de atributo remitido al recibir la solicitud correspondiente del cliente, el servidor enviará los atributos solicitados en el marco del parámetro de salida *attributeNameAndValueList*, donde el campo *attributeValue* se codificará pasando de un elemento variable a un segmento de texto XML que la aplicación del cliente podrá decodificar con la ayuda del parámetro *attributeType*.

2) *setMOAttributes* – La operación que permite modificar los valores de los atributos de un objeto gestionado existente. Además de utilizar *objectInstance* para indicar el objeto gestionado objetivo cuyos valores serán objeto de modificación, la operación también se sirve de una lista articulada en torno a cuatro elementos, a saber, *attributeName*, *attributeType*, *attributeValue* y *modifyOption* (opción de modificación), para establecer los atributos del objeto gestionado. Los tres primeros son los atributos definidos en la cláusula anterior, y *modifyOption* indica cómo establecer los valores de los atributos del objeto gestionado correspondiente. En este caso, se trata de un tipo de enumeración compuesto por *REPLACE* (sustituir), *ADDValues* (agregar valores), *REMOVEValues* (suprimir valores) y *SETToDefault* (fijar a valor por defecto). *REPLACE* implica que el o los valores especificados del atributo se utilizarán para sustituir el o los valores actuales del mismo. *ADDValues* implica que el o los valores especificados del atributo se agregarán al o los valores actuales del mismo. *REMOVEValues* implica que el o los valores especificados del atributo deben sustraerse del o los valores actuales. *SETToDefault* implica que el atributo

recuperará sus valores predeterminados. La construcción *modifyOption* es opcional y, de no ser especificado, se asumirá REPLACE.

3) *createMO* – La operación que permite crear un objeto gestionado en el sistema gestionado. En este caso, cabe especificar la clase y el nombre del objeto creado. El parámetro *attributeNameAndValueList* (lista de valores y nombres de atributo) se utiliza para proporcionar valores de atributo, no obstante, puede omitirse y, si se omite, los atributos recuperan sus valores predeterminados.

4) *deleteMO* – La operación que permite liberar todos los recursos asociados al objeto gestionado y eliminarlo. En este caso, se utiliza un nombre distinguido para identificar el objeto gestionado objetivo y, a continuación, se remite el estado de la operación. Si no es posible eliminar el objeto gestionado objetivo ni los objetos gestionados que contiene, la operación remitirá el estado *OperationFailed* (operación fallida).

5) *getPackages* – La operación que permite remitir las capacidades del objeto gestionado objetivo (un grupo de atributos y/u operaciones). Dado que existe la posibilidad de que la manifestación de objeto gestionado no soporte todos los grupos de capacidades definidos en una MOC, el cliente utiliza esta operación para averiguar a qué capacidades puede dar soporte.

Con frecuencia se recurre a listas y estructuras tabulares para proporcionar una representación compacta y eficiente de datos potencialmente complejos que han de transmitirse como una única unidad. No obstante, ello puede entrañar problemas en términos de validación y comprensión de datos, entre otras cosas. Teniendo en cuenta que estos servicios están orientados a la web y que la herencia de servicios interfiere en su bajo grado de acoplamiento, este método no es recomendable. En ese sentido, cabe la posibilidad de emplear un patrón de diseño *singleton* o de conjunto unitario para mantener la coherencia entre los datos de los objetos gestionados.

Todos los métodos mencionados *supra* se aplican a un único objeto gestionado. Ante la posibilidad de gestionar millones de entidades, el marco debe soportar operaciones en múltiples objetos con una única invocación de método o con un número reducido de invocaciones. Esta capacidad viene proporcionada por el servicio de operación de objetos múltiples (MOO), definido en la Recomendación [UIT-T Q.818]

En la cláusula A.2 se definen íntegramente un esquema XML y una interfaz WSDL para los métodos genéricos de acceso a objetos gestionados.

(R) OBJETO-2. Todas las implementaciones de los métodos de acceso a objetos gestionados deberán admitir las operaciones antes descritas y cuyo WSDL viene definido en la cláusula A.2.

10 Herencia de objetos gestionados y operaciones de interfaz

10.1 Herencia de atributos de objetos gestionados

Una clase de objeto gestionado puede definirse como una especialización de otra clase de objeto gestionado por medio de la herencia. La especialización de una clase de objeto gestionado implica que la subclase soportará todos los métodos y atributos definidos en la superclase. Las interfaces orientadas a los servicios y basadas en los servicios web solo soportan los atributos de las clases de objeto gestionado. La introducción de la herencia de operaciones afectaría a características tales como su buen nivel de interoperabilidad y su bajo grado de acoplamiento.

Si bien los atributos de los objetos gestionados se describen utilizando esquemas XML, cabe la posibilidad de aplicar la cláusula 4.2, *Deriving Types by Extension* (derivación de tipos por extensión), de la norma [W3C Primer] para generar la herencia de atributos. Dado que los tipos de datos de los atributos de objetos gestionados básicos se definen utilizando un tipo complejo en el esquema XML, la subclase puede ampliar el tipo de datos de los objetos gestionados básicos por el valor del atributo *base* (base) en el elemento *extension* (extensión) del esquema XML.

Cuando un tipo complejo se deriva por extensión, su modelo de contenido efectivo es el modelo de contenido del tipo básico unido al especificado en la derivación de tipo. Además, estos dos últimos modelos de contenido vienen considerados como descendientes de un grupo secuencial. En el caso de *USAddress* (dirección de los EE.UU.), su modelo de contenido es el modelo de contenido de *Address* (dirección) unido a las declaraciones de un elemento de código postal y un atributo *exportCode*.

```
<complexType name="Address">
  <sequence>
   <element name="name"   type="string"/>
   <element name="street" type="string"/>
   <element name="city"   type="string"/>
  </sequence>
 </complexType>
 <complexType name="USAddress">
  <complexContent>
   <extension base="ipo:Address">
    <sequence>
     <element name="state" type="ipo:USState"/>
     <element name="zip"   type="positiveInteger"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

Por tanto, *USAddress* equivale a lo siguiente:

```
  <sequence>
   <element name="name"   type="string"/>
   <element name="street" type="string"/>
   <element name="city"   type="string"/>
   <element name="state" type="ipo:USState"/>
   <element name="zip"   type="positiveInteger"/>
  </sequence>
</complexType>
```

El modelo *supra* puede utilizarse para la herencia de elementos que admite la semántica de herencia de atributos de objeto gestionado.

10.2 Observaciones relativas a la herencia de operaciones de interfaz

Aunque la última versión de WSDL (versión 2.0) soporta la sintaxis destinada a la herencia de operaciones, no goza de una amplia acepción a escala industrial. Las versiones anteriores de WSDL (precedentes a la versión 1.1) no admiten esta sintaxis, sin embargo, cuentan con el pleno respaldo de la industria. En consecuencia, la expresión de herencias de operaciones no está permitida en la presente Recomendación. Desde un punto de vista semántico, es posible dar soporte a esta característica mediante la repetición de las definiciones de las operaciones de interfaz heredadas de las superinterfaces.

11 Directrices en materia de modelización de información para interfaces basadas en servicios web

11.1 Espacios de nombre

En el presente marco se utiliza el siguiente identificador universal de recursos (URI) para el espacio de nombre (*namespace*) objetivo:

Cuadro 5 – Espacio de nombre objetivo en el presente marco

Recomendación	Espacio de nombre objetivo
UIT-T X.782	http://www.itu.int/xml-namespace/itu-t/x.782
UIT-T Q.818	http://www.itu.int/xml-namespace/itu-t/q.818
Otras, por ejemplo, X.nnnn	http://www.itu.int/xml-namespace/itu-t/x.nnnn

Al elaborar Recomendaciones adicionales, cabe sustituir "X.nnnn" por el número de la Recomendación correspondiente en la última fila de cuadro *supra*.

11.2 complexType

Cuando se recurre a un esquema XML a fin de definir el contenido de una MOC, el *complexType* XSD se utiliza para la modelización de la clase de objeto gestionado. El *complexType* XSD contiene una secuencia que puede incluir uno o más elementos XSD. Los nombres de todos los *complexType* asociados a una MOC deben contener una "_C" como sufijo.

Cabe la posibilidad de definir otros tipos de datos de atributos comunes como *complexTypes*, no obstante, deben aplicar el sufijo "*Type*" al nombre de tipo.

11.3 Atributo

Los atributos y estados de una clase de objeto gestionado pueden definirse como los elementos del *complexType* asociados a la misma. En este caso, conviene señalar que el atributo es propiedad conceptual de una entidad modelizada a imagen de una MOC y que, por tanto, difiere de la palabra clave *attribute* (atributo) en XSD. En este marco no se utilizarán las palabras clave *attribute of element* (atributo de elemento) de XSD.

11.4 Petición

La petición se utiliza para definir mensajes relativos a interacciones entre clientes de servicios web y aplicaciones de servidores de servicios web. En este caso, el cliente del servicio web envía al servidor un mensaje de petición, que contiene todos los parámetros de entrada de una operación de un servicio web. Es posible que los parámetros de entrada procedan de múltiples partes, o que la solicitud defina únicamente una parte que combine todos los parámetros de entrada como elementos internos, contenidos en su correspondiente *complexType*.

11.5 Respuesta

La respuesta se utiliza para definir un mensaje enviado desde el servidor del servicio web al cliente. Las respuestas contienen todos los parámetros de salida, así como los valores de retorno de las operaciones de los servicios web. Es posible que los parámetros de salida procedan de múltiples partes, o que la respuesta defina una parte que combine todos los parámetros de salida como elementos internos, contenidos en su correspondiente *complexType*.

11.6 Notificación

El formato de todas las notificaciones enviadas a través de la interfaz de gestión debe ajustarse a lo estipulado en la especificación [OASIS WSN]. La Recomendación [UIT-T Q.818] contiene un encabezamiento común para todas las notificaciones y los contenidos de algunas de las notificaciones más utilizadas, en particular:

objectCreation, objectDeletion, attributeValueChange, stateChange, communicationAlarm, environmentalAlarm, equipmentAlarm, processingErrorAlarm, qualityOfServiceAlarm, Violation,integrityViolation, operationalViolation, physicalViolation, securityViolation, timeDomainViolation, relationshipChange, y las notificaciones de latido.

Los responsables de la ejecución del presente marco deben respetar las definiciones incluidas en la Recomendación [UIT-T Q.818] para las notificaciones conocidas. Todas las definiciones de notificaciones nuevas deben ajustarse a los mismos principios, así como al encabezamiento previsto.

11.7 Convenciones nominales para MOC, paquetes, atributos y tipos de datos

A fin de crear modelos basados en esquemas XML, se aplican las siguientes convenciones nominales:

– Todos los atributos de una MOC se definen como un *complexType* XSD. El nombre de la MOC debe contener una "_C" como sufijo y empezar por una letra mayúscula, de forma que el *complexType* pueda distinguirse de otras definiciones de tipos de datos normales. Por ejemplo: *ManagedObject_C, Equipment_C*.

– Un atributo de una MOC se define como un elemento del *complexType* que presenta una MOC y cuyo nombre empieza por una letra minúscula.

– Un paquete puede definirse como un *complexType* XSD, cuyo nombre contiene una "_P" como sufijo y empieza por una letra mayúscula.

– Una definición de tipo de dato normal debería contener *Type* como sufijo y empezar por una letra mayúscula para resultar más legible. Por ejemplo: AdminstrativeStateType.

– Cabe utilizar *lowerCamelCase* (letra inicial minúscula), por ejemplo *personName*, para los elementos.

– Cabe utilizar *UpperCamelCase* (letra inicial mayúscula) para definir los nombres *simpleType* y *complexType*.

– Un tipo de valor establecido (conjunto no ordenado) debe contener *SetType* como sufijo de nombre, y un tipo de valor de lista (secuencia ordenada) debe contener *ListType* como sufijo de nombre.

12 Modismos de estilo para las especificaciones con esquema XML[1]

12.1 Modelos de datos con esquema XML

12.1.1 Visión general de la utilización del esquema XML

XML permite definir datos arbitrarios utilizando etiquetas ad-hoc. Los documentos XML resultan prácticos cuando se utilizan en el marco de una estructura bien definida. El esquema XML proporciona una herramienta para la definición de reglas, semánticas y estructuras para documentos XML. Un esquema facilita la validación programática de un documento XML estructurado. Un esquema XML se define utilizando un formato XML en un archivo con una extensión .xsd.

---

[1] El texto contenido en este apartado figura originalmente en [ATIS-I-000002] y ha sido ligeramente modificado con miras a su adaptación al marco de gestión basado en los servicios web objeto de estudio.

12.1.2 Versión de especificación del esquema

El espacio de nombre para el esquema XML 1.1 http://www.w3.org/2001/XMLSchema es idéntico al del esquema XML 1.0, y el espacio de nombre de los documentos XML que contienen datos sigue siendo http://www.w3.org/2001/XMLSchema-instance. Todo ello permite a los diseñadores de esquemas elaborar documentos con un esquema XML 1.0 sin necesidad de actualizar los espacios de nombre al agregar características de la versión 1.1. El esquema XML 1.1 introduce un nuevo espacio de nombre para el control de versiones (http://www.w3.org/2007/XMLSchemaversioning).

En la presente Recomendación se utiliza la versión 1.1 del esquema XML tal y como se especifica en la cláusula 7.2 de la Recomendación [UIT-T Q.818].

12.2 Observaciones relativas al diseño de esquemas XML

12.2.1 Elaboración de un único esquema o de un repertorio de esquemas conexos

El lenguaje de definición de esquema permite a las construcciones importar esquemas de otros documentos. Al desarrollar un repertorio de esquemas conexos, las consideraciones relativas a los espacios de nombre revisten una importancia particular, especialmente, en lo tocante al modo en que el documento de datos XML resultante hace referencia a otros esquemas.

– Referencia mediante <xsd:import>: El elemento *import* permite hacer referencia a componentes de esquema de documentos de esquema con distintos espacios de nombre objetivos. Este método de diseño de esquema es el más común y se conoce asimismo como diseño de espacios de nombre heterogéneos.

– Referencia mediante <xsd:include>: El elemento *include* agrega los componentes de esquema de otros documentos de esquema que poseen el mismo espacio de nombre objetivo (o para los que no se ha especificado ningún espacio de nombre objetivo) al esquema que los contiene. Por consiguiente, este elemento permite agregar todos los componentes de un esquema incluido al esquema que lo contiene, lo que a su vez genera dos métodos de diseño diferentes.

– Método de diseño de espacios de nombre homogéneos: En este caso, a todos los esquemas conexos objeto de desarrollo se les asigna el mismo espacio de nombre objetivo.

– Método de diseño de espacios de nombre "camaleón": En este caso, se utilizan uno o más esquemas de soporte definidos sin espacio de nombre objetivo y el esquema principal incluye el o los esquemas de soporte. Los esquemas de soporte adoptan el espacio de nombre del esquema principal.

Los métodos de diseño "camaleón" y homogéneo permiten redefinir las definiciones de tipos, grupos y grupos de atributos contenidas en un esquema de soporte. La redefinición de tipos afecta a los elementos tanto del esquema incluyente como del esquema incluido. Así, los tipos redefinidos pueden generar conflictos al interactuar con tipos derivados.

En las Recomendaciones UIT-T X.782 y Q.818 se utiliza únicamente el método [xsd:import].

12.2.2 Patrones de diseño de esquemas

Los siguientes cuatro patrones de diseño estructural figuran entre los más utilizados para el diseño de esquemas XML:

– *Russian doll* (muñeca rusa): Este patrón de diseño prevé la existencia de un único elemento global en el archivo de esquema. Todos los elementos secundarios vienen definidos en el marco de esa jerarquía basada en la definición de un elemento único. Este patrón de diseño no promueve la reutilización de elementos y requiere la definición de estos últimos en un único archivo de esquema.

– *Salami slice* (loncha de salami): Este patrón de diseño prevé la definición de varios elementos raíz pero ningún *complexType*. En ese sentido, admite la reutilización de elementos individuales.
– *Garden of Eden* (jardín del edén): Este patrón de diseño prevé la exposición global de todos los tipos y elementos, y la definición de tipos complejos mediante referencias a elementos globales reutilizables. Este patrón de diseño se caracteriza por la multiplicidad de elementos raíz posibles.
– *Venetian blinds* (persianas venecianas): Este patrón de diseño prevé, en primer lugar, la creación de los tipos a partir de los cuales se construyen los elementos. Además, comprende la creación de los atributos *simpleTypes* y *complexTypes* para maximizar las posibilidades de reutilización. Este patrón de diseño se caracteriza por la existencia de un único elemento raíz.

En el presente marco, se aplica el patrón de diseño *Garden of Eden*.

12.2.3 Diseños propicios a la resiliencia

Al elaborar un esquema, uno de los objetivos principales consiste en dotarlo de un carácter resiliente a los cambios y flexible de cara a las futuras necesidades de sus usuarios. En el presente apartado se analizan algunos de los mecanismos disponibles para incrementar la flexibilidad y la extensibilidad de los modelos de esquemas XML.

12.2.3.1 Uso de comodines

El esquema de la W3C permite la utilización construcciones tales como <xsd:any> y <xsd:anyAttribute> para que un documento dado pueda contener datos XML diversos de los que exige el propio modelo de contenidos del esquema XML de definición. En consecuencia, el mecanismo de ampliación goza de un cierto grado de control, que puede especificarse utilizando los atributos *namespace* (nombre de espacio) y *processContents* (contenidos de proceso) (véanse [W3C XS-P1] y [W3C XS-P2]). Por ejemplo, el atributo *namespace* puede utilizarse para limitar los datos XML a un conjunto de espacios de nombre predefinido, lo que proporcionaría una suerte de sistema de validación que podría aplicarse a esos datos XML ampliados en un documento dado. El atributo *processContents* controla el modo en que el validador XML valida los datos XML ampliados.

El comodín <xsd:any> puede permitir a los proveedores concebir funcionalidades específicas de su gremio que no vienen definidas en el esquema de definición. Una utilización prudente de <xsd:any> y <xsd:anyAttribute> puede ayudar a crear un modelo de contenido de esquema más resiliente a los cambios y menos propenso a la rotación de esquemas. Sin embargo, a modo de advertencia, cabe señalar que <xsd:any> puede dar involuntariamente lugar a la creación de modelos de contenido no determinísticos que, a su vez, pueden causar problemas con algunos analizadores XML. Los diseñadores de esquemas habrán de tenerlo en cuenta si deciden usar <xsd:any>.

En este marco se permite la utilización de <xsd:any>, no obstante, han de proporcionarse explicaciones de uso siempre que se decida hacerlo.

12.2.3.2 Uso de la construcción *substitutionGroup*

Los modelos de contenido de los miembros del grupo de sustitución se relacionan entre sí por derivación de tipo. El elemento reemplazable se denomina elemento principal y debe definirse en el alcance general del esquema. En esencia, la construcción *substitutionGroup* (Grupo de sustitución) ayuda a crear un repertorio de elementos que puede especificarse utilizando un elemento genérico.

La construcción *substitutionGroup* puede resultar útil con objeto de:

– Establecer jerarquías de clases conexas: el establecimiento de jerarquías de clases puede permitir que los lenguajes de programación orientados a objetos aprovechen la herencia correspondiente.

– Personalizar un esquema externo para una necesidad específica: si un elemento de un esquema importado se define a escala global, cabe la posibilidad de crear un *substitutionGroup* para el mismo mediante la construcción de un tipo derivado y permitir la sustitución de ese tipo derivado en una estructura de datos compleja.

Las jerarquías de clases y el *substitutionGroup* propician un alto grado de acoplamiento entre las estructuras de datos y pueden resultar en un diseño frágil e imposible de modificar. En su lugar, pueden utilizarse construcciones tales como <xsd:choice> para crear un modelo de contenido compuesto. Un diseño compuesto puede fomentar la simplicidad y el desacoplamiento.

En la presente Recomendación se utiliza la extensión a *complexType* para presentar jerarquías de clases. En este marco no se utiliza la construcción *substitutionGroup*.

12.3 Recomendaciones para diseñadores de esquemas

12.3.1 Recomendaciones para el diseño de esquemas XML

El autor de una interfaz basada en XML debería:

– concebir esquemas compartidos siempre que sea posible.

El autor también deberá:

– Utilizar el siguiente formato de espacio de nombre para los esquemas generados por el UIT-T:

  http://www.itu.int/xml-namespace/itu-t/<ITU-T document number> o

  http://www.itu.int/xml-namespace/itu-t/<ITU-T document number>/<data model identifier>.

– Utilizar únicamente caracteres en minúsculas para los nombres de espacio de nombre.

– Tener en cuenta la posibilidad de que el documento no cambie en absoluto o lo haga ligeramente con respecto al esquema (por ejemplo, referencias actualizadas).

Por esta razón, el <ITU-T document number> (número de documento UIT-T) no incluye el número de versión del documento. Al incluirlo, se forzaría un cambio en el espacio de nombre cada vez que se actualizase el documento.

El autor debería:

– Declarar todos los tipos simples y complejos a escala global.

– Declarar los elementos y atributos a escala local. Un elemento principal engloba a todos los demás.

– Asegurarse de que el atributo de versión del esquema (número de versión menor del esquema) esté presente y aumente con cualquier cambio en el esquema. Cabe prever que el valor inicial sea "0".

– Asegurarse de que todos los esquemas tengan al menos 2 espacios de nombre: el espacio de nombre de esquema XML de la W3C y el espacio de nombre relacionado con la norma complementaria.

– Proporcionar atributos de esquema de ejemplo para la versión 1.0 del esquema de servicio de acceso a objetos gestionados.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.itu.int/xml-namespace/itu-t/x.782"
```

```
   xmlns:moas="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"
   version="1.0">
.....
</schema>
```

12.3.2 Recomendaciones para la codificación de esquemas XML

El autor debería aplicar las siguientes directrices a los tipos, elementos y nombres de rótulo de atributo:

– Utilizar nombres industriales o comerciales comunes, siempre que sea posible, para garantizar la coherencia entre los esquemas y documentos industriales o comerciales.

– Crear nombres descriptivos y evitar el uso de abreviaturas innecesarias.

– No utilizar el mismo nombre dos veces en un mismo esquema.

– Evitar los acrónimos y, de utilizarlos, mantener las mayúsculas.

– Evitar los puntos y los guiones.

– Utilizar únicamente caracteres alfanuméricos para definir elementos y nombres de tipo.

– Utilizar nombres en singular a menos que el concepto en sí sea plural.

El autor debería considerar el uso de *element* (elemento) como tipo de modelo predeterminado y aplicar las siguientes directrices:

– Si el componente puede considerarse un objeto independiente, debe convertirse en un elemento.

– Si el componente ha de reutilizarse, debe convertirse en un tipo global.

El autor debería aplicar las siguientes directrices generales:

– Declarar los elementos como opcionales a menos que sean absolutamente necesarios.

– Utilizar *minOccurs="0* en lugar de *nillable="True"*.

– Utilizar *maxOccurs*. De existir más una dimensión, cabe definirlas. De lo contrario, se utilizará *maxOccurs="unbounded*".

– Crear tantos *simplesTypes* (tipos simples) como sea posible.

– Crear un tipo global cuando el elemento haya de reutilizarse (los tipos globales se definen directamente bajo el elemento de esquema).

– Definir todos los tipos a escala global.

– Utilizar *elementFormDefault = "qualified"*.

– Utilizar *attributeFormDefault = "unqualified"*.

– Usar el sistema de codificación de caracteres UTF-8: <?xml version="1.0" encoding="UTF-8"?>

– Los elementos <*documentation*> (documentación) deben utilizarse en los casos en que sea probable la reutilización y se desee una mayor claridad (el atributo xsd:lang se utilizará para especificar el lenguaje).

– Utilizar anotaciones para describir todas las definiciones de tipos, incluido mínimamente el nombre del tipo.

– Utilizar el elemento xsd:dateTime únicamente para componentes de fecha y hora.

– Utilizar <sequence> (secuencia) o <choice> (elección) únicamente en los casos en se requiera un compositor.

El autor debería aplicar las siguientes directrices al agregar restricciones:

– Al diseñar un nuevo esquema, restringir los tipos simples nuevos y aplicar las restricciones adecuadas siempre que sea posible.

– Identificar las facetas que restringen el rango de valores.

– Elegir el tipo simple adecuado. Por ejemplo, cuando se necesita un número, se privilegia la utilización del tipo simple *nonNegativeInteger* (entero no negativo) o *positiveInteger* (entero positivo) al número entero, siempre que sea posible.

– Al definir un tipo de cadena y siempre que sea posible, utilizar *maxLength* (longitud máxima) y patrones para añadir restricciones adicionales. Por ejemplo, al definir un número de teléfono internacional de 15 dígitos, el patrón puede ser ([UIT-T E.164]):

```
<xsd:restriction base="xsd:string">
   <xsd:pattern value="([1-9][0-9]{0,2})(-[1-9][0-9]{0,4})?(-[1-
9][0-9]{0,13})(-[0-9]+)?"/>
</xsd:restriction>
```

– Cuando un tipo simple adopta únicamente un conjunto predefinido de valores, utilizar una faceta de enumeración para restringir los valores legales del tipo simple.

– Utilizar los tipos de unión en los casos en que los tipos de datos adopten dos o más conjuntos de valores diferentes que puedan restringirse de forma independiente. Por ejemplo, los códigos de estado o los códigos postales.

12.3.3 Construcciones de esquema XML que cabe evitar

El autor debería aplicar las siguientes directrices a las construcciones de esquema XML:

– No utilizar <xsd:all> (sino <sequence> o <choice>). Con <xsd:sequence> y <xsd:choice> se fuerza un orden fijo de elementos secundarios en el documento de datos.

– No utilizar instrucciones de procesamiento. Las instrucciones de procesamiento no forman parte del documento, sino que se transmiten a la aplicación con miras a la ejecución de sus funciones específicas. Las instrucciones de procesamiento no han de ajustarse a la estructura interna y, en ese sentido, desempeñan un papel limitado en las definiciones de esquema.

– No utilizar comentarios de estilo DTD o XML. Los elementos de anotación y documentación proporcionan construcciones para comentarios e información. Los comentarios de estilo XML carecen de utilidad para el procesamiento del documento de datos.

– No utilizar redefiniciones de grupo o grupos XML. La redefinición de un grupo puede generar conflictos entre el procesamiento del tipo redefinido y los datos del tipo derivado.

– No utilizar grupos de sustitución. La construcción del grupo de sustitución propicia un alto grado de acoplamiento y agrega complejidad, lo que puede resultar en diseños frágiles e imposibles de modificar.

– No utilizar valores predeterminados y/o fijos. La inclusión de atributos de esa índole tenderá a confundir a los lectores del documento de esquema por cuanto carecen de efecto.

12.4 Directrices para la ampliación de esquemas

Se entiende que los autores no pueden prever de antemano todas las necesidades futuras de sus esquemas. En ese sentido, el autor debe utilizar una definición de esquema flexible, que permita a los implementadores transportar datos privados o personalizados. En general, los autores de los esquemas deben prever la necesidad de dar soporte a datos privados mediante la inclusión de construcciones que soporten la introducción de ciertos datos de firma genérica (véase el par nombre-valor).

Sin embargo, dicha inclusión puede no ser suficiente para cubrir una necesidad más compleja, a raíz de la cual puedan requerirse, por ejemplo, políticas y aserciones de reglas de validación.

El autor del esquema debería utilizar uno de los dos mecanismos de ampliación del soporte para datos privados en un esquema existentes, los cuales consisten en:

1) prever las necesidades de ampliación de segmentos específicos del esquema y permitir a los usuarios agregar datos *OpenContent* (contenido abierto) privados; y

2) escribir la herencia utilizando la extensión <xsd:extension>.

No obstante, se desaconseja el uso de dicha extensión, ya que podría generar problemas de interpretación. Por ejemplo, si un implementador amplía un esquema normalizado utilizando *xsd:extension* para agregar atributos y elementos específicos de su gremio, cabe la posibilidad de que otros implementadores que hayan basado sus aplicaciones en el esquema normalizado original sean incapaces de analizar correctamente los datos XML resultantes.

El problema anterior puede evitarse si ambas partes utilizan el mismo esquema XML. En el presente marco puede utilizarse la ampliación, no obstante, los tipos ampliados también deberían incluirse como una norma pública.

13 Cumplimiento y conformidad

En este apartado se definen los criterios a los que deben ajustarse otros documentos normativos que aleguen el cumplimiento de estas directrices, así como las funciones que deben comprender los sistemas que aleguen ser conformes a la presente Recomendación.

13.1 Cumplimiento de los documentos normativos

Todas las especificaciones que aleguen el cumplimiento de estas directrices deberán:

1) Definir todas las clases que modelizan recursos como una derivación (directa o indirecta) de la construcción *ManagedObject_C*, descrita en la cláusula 8.2.1 y definida en el esquema XML dla cláusula A.1.

2) Soportar la herencia de atributos utilizando el mecanismo especificado en la cláusula 10.1.

3) Utilizar las definiciones para los tipos de atributos genéricos que figuran en la cláusula 8.2.4, cuando proceda.

4) Utilizar los tipos de datos comunes definidos en el esquema XML de la cláusula A.1, cuando proceda.

5) Aplicar las directrices de modelización para interfaces basadas en servicios web que figuran en la cláusula 11.

6) Aplicar las convenciones de diseño de esquemas XML especificadas en la cláusula 12.

13.2 Conformidad del sistema

Los sistemas que aleguen ser conformes a la presente Recomendación deberán:

1) soportar todas las capacidades de los métodos de acceso a objetos gestionados que figuran en la cláusula 9, así como la interfaz WSDL correspondiente, según se indica en la cláusula A.2.

13.3 Directrices para la declaración de conformidad

La declaración de conformidad debe recoger el tipo de documento y el año de publicación, con objeto de garantizar la identificación de la versión correcta del esquema XML y WSDL.

Anexo A

Definiciones comunes de esquemas XML y WSDL

(Este anexo forma parte integrante de la presente Recomendación.)

En el presente anexo se recogen definiciones comunes de interfaces WSDL, así como de ciertos tipos de datos comunes basados en esquemas XML.

A.1 Definiciones de esquemas XML para tipos de datos comunes y un objeto gestionado genérico

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML Schema Definition for common data types to be used in this framework.
       Filename : x782.xsd -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x.782"
   targetNamespace="http://www.itu.int/xml-namespace/itu-t/x.782"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified"
   version="1.0">

<xsd:simpleType name="RDNType">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:complexType name="NameType">
   <xsd:sequence>
      <xsd:element name="rdn" type="x782:RDNType" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="NameSetType">
   <xsd:sequence>
      <xsd:element name="dn" type="x782:NameType" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="UIDType">
   <xsd:sequence>
      <!-- uri indicates the namespace where the constant is defined. -->
      <xsd:element name="uri" type="xsd:string"/>
      <!-- value indicates the constant value for this item in the above
namespace. -->
      <xsd:element name="value" type="xsd:unsignedLong"/>
   </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="UIDSetType">
   <xsd:sequence>
      <xsd:element name="uid" type="x782:UIDType" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MOClassListType">
   <xsd:sequence>
      <xsd:element name="moClass" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="AdministrativeStateType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="locked"/>
      <xsd:enumeration value="unlocked"/>
      <xsd:enumeration value="suttingDown"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="OperationalStateType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="disabled"/>
      <xsd:enumeration value="enabled"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="AvailabilityStatusType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="inTest"/>
      <xsd:enumeration value="failed"/>
      <xsd:enumeration value="powerOff"/>
      <xsd:enumeration value="offLine"/>
      <xsd:enumeration value="offDuty"/>
      <xsd:enumeration value="dependency"/>
      <xsd:enumeration value="degraded"/>
      <xsd:enumeration value="notInstalled"/>
      <xsd:enumeration value="logFull"/>
   </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:complexType name="AvailabilityStatusSetType">
   <xsd:sequence>
      <xsd:element name="availableState" type="x782:AvailabilityStatusType"
minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="BackedUpStatusType">
   <xsd:restriction base="xsd:boolean"/>
</xsd:simpleType>
<xsd:simpleType name="ControlStatusType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="subjectToTest"/>
      <xsd:enumeration value="partOfServicesLocked"/>
      <xsd:enumeration value="reservedForTest"/>
      <xsd:enumeration value="suspended"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="ControlStatusSetType">
   <xsd:sequence>
      <xsd:element name="controlState" type="x782:ControlStatusType"
minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ExternalTimeType">
   <xsd:restriction base="xsd:dateTime"/>
</xsd:simpleType>

<xsd:simpleType name="ObjectClassType">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="ProceduralStatusType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="initializationRequired"/>
      <xsd:enumeration value="notInitialized"/>
      <xsd:enumeration value="initializing"/>
      <xsd:enumeration value="reporting"/>
      <xsd:enumeration value="terminating"/>
   </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:complexType name="ProceduralStatusSetType">
   <xsd:sequence>
      <xsd:element name="proceduralState" type="x782:ProceduralStatusType"
minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="SourceIndicatorType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="resourceOperation"/>
      <xsd:enumeration value="managementOperation"/>
      <xsd:enumeration value="unknown"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StandbyStatusType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="hotStandby"/>
      <xsd:enumeration value="coldStandby"/>
      <xsd:enumeration value="providingService"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="StringSetType">
   <xsd:sequence>
      <xsd:element name="value" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="SystemLabelType">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="UsageStateType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="idle"/>
      <xsd:enumeration value="active"/>
      <xsd:enumeration value="busy"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="UnknownStatusType">
   <xsd:restriction base="xsd:boolean"/>
</xsd:simpleType>
```

```
<xsd:complexType name="AttributeValueType">
   <xsd:sequence>
      <xsd:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="AttributeNameAndValueType">
   <xsd:sequence>
      <xsd:element name="attributeName" type="xsd:string"/>
      <xsd:element name="attributeType" type="xsd:string"/>
      <xsd:element name="attributeValue" type="x782:AttributeValueType"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AttributeNameAndValueSetType">
   <xsd:sequence>
      <xsd:element name="attributeNameAndValue"
type="x782:AttributeNameAndValueType" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="AdditionalTextType">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:complexType name="AnyValueType">
   <xsd:sequence>
      <xsd:element name="typeURI" type="xsd:string"/>
      <xsd:element name="value" type="x782:AttributeValueType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="AdditionalInformationSetType">
   <xsd:sequence>
      <xsd:element name="additionalInfo" type="x782:AnyValueType" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="NotificationIDType">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

```
<xsd:complexType name="NotificationIDSetType">
   <xsd:sequence>
      <xsd:element name="source" type="x782:NameType" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CorrelatedNotificationType">
   <xsd:sequence>
      <xsd:element name="source" type="x782:NameType"/>
      <xsd:element name="notifIDs" type="x782:NotificationIDSetType"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CorrelatedNotificationSetType">
   <xsd:sequence>
      <xsd:element name="notifications" type="x782:CorrelatedNotificationType"
minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType><xsd:complexType name="AttributeChangeType">
   <xsd:sequence>
      <xsd:element name="attribugteName" type="xsd:string"/>
      <xsd:element name="attributeTypeURI" type="xsd:string"/>
      <xsd:element name="oldValue" type="x782:AttributeValueType"/>
      <xsd:element name="newValue" type="x782:AttributeValueType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="AttributeChangeSetType">
   <xsd:sequence>
      <xsd:element name="attributeChange" type="x782:AttributeChangeType"
minOccurs="1" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ProbableCauseType">
   <xsd:complexContent>
         <xsd:extension base="x782:UIDType"/>
      </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="PerceivedSeverityType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="indeterminate"/>
      <xsd:enumeration value="critical"/>
      <xsd:enumeration value="major"/>
```

```
      <xsd:enumeration value="minor"/>
      <xsd:enumeration value="warning"/>
      <xsd:enumeration value="cleared"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="TrendIndicationType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="lessSevere"/>
      <xsd:enumeration value="noChange"/>
      <xsd:enumeration value="moreSevere"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ThresholdIndicationType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="up"/>
      <xsd:enumeration value="down"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="ThresholdLevelIndType">
   <xsd:sequence>
      <xsd:element name="indication" type="x782:ThresholdIndicationType"/>
      <!-- observed value -->
      <xsd:element name="low" type="xsd:float" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="high" type="xsd:float"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ThresholdInfoType">
   <xsd:sequence>
      <xsd:element name="attributeID" type="xsd:string"/>
      <xsd:element name="observedValue" type="xsd:float"/>
      <xsd:element name="thresholdLevel" type="x782:ThresholdLevelIndType"/>
      <xsd:element name="armTime" type="xsd:dateTime"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ProposedRepairActionSetType">
   <xsd:complexContent>
      <xsd:extension base="x782:UIDType"/>
   </xsd:complexContent>
</xsd:complexType>
```

```
<xsd:complexType name="SuspectObjectType">
   <xsd:sequence>
      <xsd:element name="moClass" type="xsd:string"/>
      <xsd:element name="suspectedMOInstance" type="x782:NameType"/>
      <xsd:element name="failureProbability" type="xsd:unsignedShort"
minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SuspectObjectSetType">
   <xsd:sequence>
      <xsd:element name="suspectedMO" type="x782:SuspectObjectSetType"
minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SecurityAlarmCauseType">
   <xsd:complexContent>
      <xsd:extension base="x782:UIDType"/>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SecurityAlarmDetectorType">
   <xsd:sequence>
      <xsd:element name="mechanism" type="x782:UIDType"/>
      <xsd:element name="obj" type="x782:NameType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ServiceUserType">
   <xsd:sequence>
      <xsd:element name="typeURI" type="xsd:string"/>
      <xsd:element name="value" type="x782:AttributeValueType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ServiceProviderType">
   <xsd:sequence>
      <xsd:element name="typeURI" type="xsd:string"/>
      <xsd:element name="value" type="x782:AttributeValueType"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SpecificProblemSetType">
   <xsd:complexContent>
      <xsd:extension base="x782:UIDSetType"/>
```

```
   </xsd:complexContent>
</xsd:complexType>

<!-- XML Schema Definition for generic Managed Object -->
<xsd:complexType name="ManagedObject_C">
   <xsd:sequence>
      <xsd:element name="objectClass" type="xsd:string"/>
      <xsd:element name="objectInstance" type="x782:NameType"/>
      <xsd:element name="packages" type="x782:StringSetType"/>
      <xsd:element name="creationSource" type="x782:SourceIndicatorType"/>
   </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

A.2 Definición de esquemas XML y WSDL para métodos de acceso a objetos comunes

(1) Definición de esquema XML del servicio de acceso a objetos gestionados de la UIT

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML Schema Definition for data types to be used in MO access Service
specified in this Recommendation.
       Filename : x782_MOAccessService.xsd -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x.782"
   xmlns:moas="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"
   targetNamespace="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified"
   version="1.0">
<xsd:import namespace="http://www.itu.int/xml-namespace/itu-t/x.782"
schemaLocation="x782.xsd"/>
   <xsd:complexType name="AttributeNameListType">
      <xsd:sequence>
         <xsd:element name="attributeName" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="GetMOAttributesRequestType">
      <xsd:sequence>
         <xsd:element name="objectInstance" type="x782:NameType"/>
         <xsd:element name="attributeNameList"
type="moas:AttributeNameListType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="StatusType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="OperationSucceed"/>
```

```
         <xsd:enumeration value="OperationFailed"/>
      </xsd:restriction>
   </xsd:simpleType>

   <xsd:complexType name="GetMOAttributesResponseType">
      <xsd:sequence>
         <xsd:element name="attributeNameAndValueList"
type="x782:AttributeNameAndValueSetType"/>
         <xsd:element name="status" type=" moas:StatusType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="ModifyOptionType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="REPLACE"/>
         <xsd:enumeration value="ADDValues"/>
         <xsd:enumeration value="REMOVEValues"/>
         <xsd:enumeration value="SETToDefault"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name="AttributeNVMType">
      <xsd:sequence>
         <xsd:element name="attributeName" type="xsd:string"/>
         <xsd:element name="attributeType" type="xsd:string"/>
         <xsd:element name="attributeValue" type="x782:AttributeValueType"/>
         <xsd:element name="modifyOption" type="moas:ModifyOptionType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="AttributeNVMListType">
      <xsd:sequence>
         <xsd:element name="attributeNVM" type=" moas:AttributeNVMType"
minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="SetMOAttributesRequestType">
      <xsd:sequence>
         <xsd:element name="objectInstance" type="x782:NameType"/>
         <xsd:element name="attributeNVMList" type="
moas:AttributeNVMListType"/>
      </xsd:sequence>
   </xsd:complexType>
```

```
   <xsd:complexType name="CreateMORequestType">
      <xsd:sequence>
         <xsd:element name="objectClass" type="xsd:string"/>
         <xsd:element name="objectInstance" type="x782:NameType"/>
         <xsd:element name="attributeNameAndValueList"
type="x782:AttributeNameAndValueSetType"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="GetPackagesResponseType">
      <xsd:sequence>
         <xsd:element name="status" type="moas:StatusType"/>
         <xsd:element name="packages" type="x782:StringSetType"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

(2) Definición de esquema WSDL del servicio de acceso a objetos gestionados de la UIT

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- WSDL Operation Definition for MO Access Service specified in this
Recommendation.
       Filename : x782_MOAccessService.wsdl -->
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x.782"
xmlns:moas="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"
name="MOAccessService"
targetNamespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService">
<import namespace="http://www.itu.int/xml-namespace/itu-t/x.782"
location="x782.xsd"/>
<import namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"
location="x782_MOAccessService.xsd"/>

   <wsdl:message name="getMOAttributesRequest">
      <wsdl:part name="getMOAttributesInput"
type="moas:GetMOAttributesRequestType"/>
   </wsdl:message>
   <wsdl:message name="getMOAttributesResponse">
      <wsdl:part name="getMOAttributesOutput"
type="moas:GetMOAttributesResponseType"/>
   </wsdl:message>
   <wsdl:message name="setMOAttributesRequest">
      <wsdl:part name="setMOAttributesInput"
type="moas:SetMOAttributesRequestType"/>
   </wsdl:message>
   <wsdl:message name="setMOAttributesResponse">
      <wsdl:part name="status" type="moas:StatusType"/>
```

```
   </wsdl:message>
   <wsdl:message name="createMORequest">
      <wsdl:part name="createMOInput" type="moas:CreateMORequestType"/>
   </wsdl:message>
   <wsdl:message name="createMOResponse">
      <wsdl:part name="status" type="moas:StatusType"/>
   </wsdl:message>
   <wsdl:message name="deleteMORequest">
      <wsdl:part name="objectInstance" type="x782:NameType"/>
   </wsdl:message>
   <wsdl:message name="deleteMOResponse">
      <wsdl:part name="status" type="moas:StatusType"/>
   </wsdl:message>
   <wsdl:message name="getPackagesRequest">
      <wsdl:part name="objectInstance" type="x782:NameType"/>
   </wsdl:message>
   <wsdl:message name="getPackagesResponse">
      <wsdl:part name="getPackageOutput" type="moas:GetPackagesResponseType"/>
   </wsdl:message>
   <wsdl:portType name="MOAccessServicePortType">
      <wsdl:operation name="getMOAttributes">
         <wsdl:input message="moas:getMOAttributesRequest"/>
         <wsdl:output message="moas:getMOAttributesResponse"/>
      </wsdl:operation>
      <wsdl:operation name="setMOAttributes">
         <wsdl:input message="moas:setMOAttributesRequest"/>
         <wsdl:output message="moas:setMOAttributesResponse"/>
      </wsdl:operation>
      <wsdl:operation name="createMO">
         <wsdl:input message="moas:createMORequest"/>
         <wsdl:output message="moas:createMOResponse"/>
      </wsdl:operation>
      <wsdl:operation name="deleteMO">
         <wsdl:input message="moas:deleteMORequest"/>
         <wsdl:output message="moas:deleteMOResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getPackages">
         <wsdl:input message="moas:getPackagesRequest"/>
         <wsdl:output message="moas:getPackagesResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="MOAccessServiceBinding"
type="moas:MOAccessServicePortType">
```

```
      <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getMOAttributes">
         <soap:operation soapAction="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService/getMOAttributes"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="
http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="
http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="setMOAttributes">
         <soap:operation soapAction=" http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService/setMOAttributes"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="
http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="createMO">
         <soap:operation soapAction="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService/createMO"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="deleteMO">
         <soap:operation soapAction="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService/deleteMO"/>
```

```
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getPackages">
         <soap:operation soapAction="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService/getPackages"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="MOAccessService">
      <wsdl:port name="MOAccessService" binding="moas:MOAccessServiceBinding">
         <soap:address location="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Apéndice I

Descripción general de la tecnología de servicio web y supuestos de aplicación en interfaces de gestión de redes

(Este apéndice no forma parte integrante de la presente Recomendación.)

El presente apéndice contiene una descripción general de la tecnología de servicio web.

I.1 Características de la tecnología de servicio web

Los servicios web brindan un mecanismo simplificado que permite conectar aplicaciones con independencia de la tecnología o los dispositivos utilizados e incluso de su ubicación. Esta tecnología se basa en protocolos normalizados del sector industrial, goza de soporte universal entre los proveedores y puede aprovechar el potencial de Internet y de otros mecanismos de transporte para ofrecer comunicaciones a bajo coste. La metodología de transferencia de mensajes con un bajo grado de acoplamiento admite múltiples hipótesis de conectividad e intercambio de información a través de servicios que revisten un carácter autodescriptivo y pueden descubrirse automáticamente.

A diferencia de los entornos distribuidos tradicionales, los servicios web priorizan la interoperabilidad. Si bien los entornos tradicionales suelen estar vinculados a un lenguaje de programación concreto, estos servicios son independientes de este factor. En ese sentido, ofrecen una mayor flexibilidad de cara a la elección del mecanismo de transporte, por cuanto pueden vincularse fácilmente a distintos tipos. Además, a diferencia de los entornos tradicionales, estos servicios no suelen estar vinculados a clientes o servidores concretos. En general, los servicios web se adaptan mejor a un conjunto de relaciones de granularidad gruesa y con un bajo grado de acoplamiento. Al utilizar XML, los servicios web disponen de una ventaja adicional, ya que ese lenguaje posibilita la utilización de documentos en entornos heterogéneos.

La tecnología de servicio web se basa en el esquema XML, cuyos beneficios en el marco de las aplicaciones de software se definen a continuación.

1) Buen nivel de interoperabilidad

Los servicios web son interoperativos a escala universal, pues utilizan plataformas y protocolos independientes de lenguajes tales como SOAP. Esta tecnología propicia un nuevo nivel de interoperabilidad entre aplicaciones de software. Numerosos proveedores de plataformas, diseñadores de software y proveedores de servicios públicos habilitan su software con capacidades SOAP, WSDL y UDDI.

2) Bajo grado de acoplamiento

Los servicios web son módulos de software autodescriptivos que comprenden una funcionalidad discreta. A estos servicios se accede mediante protocolos normalizados de comunicación a través de Internet, de la índole de XML y SOAP. Cabe la posibilidad de utilizar diversos lenguajes de implementación para diseñar este tipo de servicios, a los que pueden acceder toda clase de aplicaciones o servicios web. Por consiguiente, los servicios web son aplicaciones con un bajo grado de acoplamiento.

3) Utilización generalizada

Actualmente, los servicios web se utilizan de forma generalizada en el sector de los servicios de tecnologías de la información, véanse las aplicaciones de cibercomercio y de empresa a empresa. Actualmente, existen diversas plataformas estables que dan soporte al desarrollo de aplicaciones de servicios web.

4) Reutilización de software y datos

A efectos del diseño de software, los servicios web soportan el modelo basado en componentes, que permite a los diseñadores reutilizar los elementos creados por otros para ensamblar aplicaciones complejas y ampliarlas de formas innovadoras.

Esta tecnología permite la reutilización no solo del código fuente, sino también de los datos que se esconden detrás. Del mismo modo, permite la integración de esas funciones en otras aplicaciones pertinentes y su exposición a través de interfaces de servicio web.

5) Facilidad para la composición de servicios

Los servicios web permiten la definición de aplicaciones cada vez más complejas mediante la agregación progresiva de componentes en niveles superiores de abstracción. Los clientes que invocan servicios compuestos pueden verse a su vez expuestos como servicios web. El proceso en virtud del cual las funcionalidades de varios servicios web se combinan para formar un nuevo servicio web se denomina "composición de servicios". La composición de servicios puede efectuarse a partir de servicios elementales o compuestos.

Los servicios web proporcionan un mecanismo normalizado para exponer las funcionalidades de las aplicaciones como servicios y acceder a ellas. En ese sentido, cabe señalar que existen lenguajes especializados (tales como BPEL) para la definición y ejecución de procesos operativos.

6) Bajo coste

Actualmente, numerosos productos, herramientas y tecnologías admiten normas de servicios web, lo que brinda a las organizaciones diversas alternativas para reducir los costes inherentes al diseño de aplicaciones nuevas, al funcionamiento, a los factores ambientales y a la integración.

I.2 Supuestos de aplicación procedente e improcedente de los servicios web en la gestión de redes

(1) Supuestos de aplicación procedente con carácter general

En términos generales y dadas las características de los servicios web, su empleo se considera adecuado en los supuestos de aplicación descritos *infra*.

– Comunicación a través de cortafuegos

Habida cuenta de que utilizan un SOAP normalizado como protocolo de transferencia, los servicios web tienen la capacidad de atravesar con facilidad cortafuegos o servidores intermediarios que pueden estar ubicados entre distintas aplicaciones conexas. Este proceso suele resultar más complejo cuando se utiliza otro soporte intermedio de comunicación.

– Integración de aplicaciones

Es bien sabido que, con frecuencia, las aplicaciones han de ejecutarse en programas ubicados en un tipo de plataforma particular y obtener o enviar datos a otras aplicaciones instaladas en otras plataformas. Incluso se dan casos en los que es necesario integrar diversos soportes lógicos de distintos fabricantes en una misma plataforma. En el marco de los servicios web, las aplicaciones pueden exponer funciones y datos para que otras aplicaciones los utilicen de forma normalizada.

– Integración de interfaces de empresa a empresa

La integración de las transacciones comerciales entre empresas se suele denominar integración de empresa a empresa o B2B (*business-to-business*). A través de los servicios web, las empresas pueden integrar aplicaciones operativas cruciales y exponerlas a determinados proveedores y clientes. La ventaja principal de la utilización de servicios web con miras a la integración de empresa a empresa es que puede facilitar la interoperabilidad.

– Interfaz abierta con buen nivel de tolerancia a los cambios

Las interfaces de los servicios web se definen utilizando el lenguaje de descripción de servicios web (WSDL). WSDL define los servicios como repertorios de puertos o puntos extremos de red. A tal efecto, la especificación WSDL proporciona un formato XML para documentos. La definición abstracta de puertos y mensajes es independiente de su finalidad o instancia concretas, lo que permite su reutilización. De esta manera, WSDL se utiliza para describir la interfaz pública abierta al servicio web. Debido a la separación de las definiciones de los datos y las interfaces, las figuras del cliente y el servidor de las aplicaciones de los servicios web pueden diseñarse por separado y comunicarse a través de esa interfaz abierta, y el cambio de interfaces tendrá una incidencia menor en el diseño de aplicaciones de servicios web.

(2) Supuestos de aplicación improcedente con carácter general

Dado que un servicio web es una tecnología con un bajo grado de acoplamiento, especialmente concebida a efectos de la interoperabilidad de aplicaciones y la integración en entornos heterogéneos, también presenta puntos débiles tales como una menor velocidad de ejecución y menor eficacia en términos de codificación en comparación con otras tecnologías (véase CORBA o DCOM). En ciertos casos, la utilización de servicios web puede no ser la mejor opción. Por ejemplo:

– en los sistemas de equipos individuales;
– en las aplicaciones LAN isomorfas (aplicaciones que interfuncionan en un único entorno LAN con la misma plataforma y el mismo soporte intermedio subyacente), y
– en la interacción en línea con una gran cantidad de datos.

(3) Consideraciones relativas a los supuestos de aplicación de los servicios web en la gestión de redes

De acuerdo con el análisis efectuado *supra*, cabe considerar que las interfaces de dominio de gestión de red más adecuadas para la aplicación de servicios web son las siguientes:

– interfaces de empresa a empresa y/o de consumidor a empresa;
– interfaces F;
– interfaces entre sistemas operativos de alto nivel (capa de gestión de servicios o capa de gestión de operaciones), etc.

En estos casos concurren circunstancias que permiten aprovechar ampliamente los beneficios de los servicios web, especialmente, la integración de aplicaciones entre empresas con un bajo grado de acoplamiento, el acceso basado en la web y el buen nivel de extensibilidad de las nuevas aplicaciones de servicio.

La utilización de servicios web en interfaces de gestión EMS-NE puede no ser la mejor opción, ya que ambas tecnologías suelen proceder de un mismo proveedor y pueden resultar innecesarias como interfaces abiertas.

También cabe la posibilidad de utilizar servicios web en interfaces de gestión NMS-EMS, sin embargo, ello puede plantear problemas en algunos casos. Por ejemplo, CORBA cuenta con un sistema de codificación más eficaz y una velocidad de ejecución mayor en un entorno LAN.

Bibliografía

[b-UIT-T X.780] Recomendación UIT-T X.780 (2001), Directrices de la RGT para la definición de objetos gestionados mediante arquitectura de intermediario de petición de objeto común.

[b-UIT-T X.780.2] Recomendación UIT-T X.780.2 (2007), Directrices de la TMN para definir objetos CORBA de gestión orientados al servicio y objetos CORBA de fachada.

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie D	Principios de tarificación y contabilidad y cuestiones económicas y políticas de las telecomunicaciones/TIC internacionales
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedia
Serie I	Red digital de servicios integrados
Serie J	Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedia
Serie K	Protección contra las interferencias
Serie L	Medio ambiente y TIC, cambio climático, ciberdesechos, eficiencia energética, construcción, instalación y protección de los cables y demás elementos de planta exterior
Serie M	Gestión de las telecomunicaciones, incluida la RGT y el mantenimiento de redes
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de la transmisión telefónica, instalaciones telefónicas y redes de líneas locales
Serie Q	Conmutación y señalización, y mediciones y pruebas asociadas
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	**Redes de datos, comunicaciones de sistemas abiertos y seguridad**
Serie Y	Infraestructura mundial de la información, aspectos del protocolo Internet, redes de próxima generación, Internet de las cosas y ciudades inteligentes
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación

Impreso en Suiza
Ginebra, 2017