NORMA
BRASILEÑA

ABNT NBR 15606-5

Segunda edición
19.04.2011

Válida a partir de
19.05.2011

Televisión digital terrestre — Codificación de dados y especificaciones de transmisión para radiodifusión digital
Parte 5: Ginga-NCL para receptores transportables – Lenguaje de aplicación XML para codificación de aplicaciones

ICS 33.160.01

ISBN 978-85-07-02811-6

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS

Número de referencia
ABNT NBR 15606-5:2011
118 páginas

ABNT
Av.Treze de Maio, 13 - 28º andar
20031-901 - Rio de Janeiro - RJ
Tel.: + 55 21 3974-2300
Fax: + 55 21 2220-1762
abnt@abnt.org.br
www.abnt.org.br

Impresso en Brasil

Índice

Página

Prefacio

La Associação Brasileira de Normas Técnicas (ABNT) es el Fórum Nacional de Normalización. Las Normas Brasileñas, cuyo contenido es responsabilidad de los Comités Brasileños (ABNT/CB), de los Organismos de Normalización Sectorial (ABNT/ONS) y de las Comisiones de Estudios Especiales (ABNT/CEE), son elaboradas por Comisiones de Estudio (CE), formadas por representantes de sus sectores implicados de los que forman parte: productores, consumidores y neutrales (universidades, laboratorios y otros).

Los Documentos Técnicos ABNT se elaboran de acuerdo con las reglas de Directivas ABNT, Parte 2.

La Associação Brasileira de Normas Técnicas (ABNT) llama la atención sobre la posibilidad de que algunos de los elementos de este documento pueden ser objeto de derechos de patente. La ABNT no debe ser considerada responsable por la identificación de cualesquiera derechos de patente.

La ABNT NBR 15606-5 fue elaborada por la Comisión de Estudio Especial de Televisión Digital (ABNT/CEE-00:001.85). El Proyecto circuló en Consulta Nacional según Edicto nº 12, de 13.12.2007 a 11.02.2008, con el número de Proyecto 00:001.85-006/5. El Proyecto de Enmienda 1 circuló en Consulta Nacional según Edicto nº 01, de 13.01.2011 a 14.03.2011, con el número de Proyecto de Enmienda ABNT NBR 15606-5.

En caso que surja cualquier duda con relación a la interpretación de la versión en español siempre deben prevalecer las prescripciones de la versión en portugués.

Esta Norma está basada en los trabajos del Fórum del Sistema Brasileiro de Televisão Digital Terrestre, según establece el Decreto Presidencial nº 5.820, de 29/06/2006.

La ABNT NBR 15606, bajo el título general “Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital”, está previsto que contenga las siguientes partes:

— Parte 1: Codificación de datos;

— Parte 2: Ginga-NCL para receptores fijos y móviles – Lenguaje de aplicación XML para codificación de aplicaciones;

— Parte 3: Especificación de transmisión de datos;

— Parte 4: Ginga-J – Ambiente para la ejecución de aplicaciones imperativas;

— Parte 5: Ginga-NCL para receptores portátiles – Lenguaje de aplicación XML para codificación de aplicaciones.

Esta segunda edición incorpora la Enmienda 1 de 10.06.2011 y cancela y sustituye la edición anterior (ABNT NBR 15606-5:2008).

Esta versión en español es equivalente a la ABNT NBR 15606-5:2011, de 19.04.2011.

Esta versión en español fue publicada en 14.06.2011.

Introducción

La Associação Brasileira de Normas Técnicas (ABNT) llama la atención para el hecho de que la exigencia de conformidad con este documento ABNT puede involucrar el uso de una patente relacionada a NCL, tal como mencionado en 5.1.

La ABNT no toma posición con respecto a evidencias, validez y alcance de estos derechos de patente.

El propietario de este derecho de patente aseguró a ABNT que él está preparado para negociar licencias sobre términos y condiciones razonables y no discriminatorias con los solicitantes. Sobre esto, una declaración del propietario de esta patente está registrada en ABNT. Informaciones se pueden obtener en:

Pontificia Universidad Católica de Río de Janeiro, Departamento de Transferencia de Tecnología

Rua Marquês de São Vicente, 225 – Gávea, 22451-900 - Rio de Janeiro - RJ - Brasil.

ABNT llama la atención para la posibilidad de que algunos de los elementos de este documento ABNT puedan ser objeto de otros derechos de patente además de los identificados anteriormente. La ABNT no se debe considerar responsable por la identificación de cualesquiera derechos de patente.

Esta Norma estandariza un lenguaje de aplicación XML que permite a los autores escribir presentaciones multimedia interactivas. Este componente de ABNT NBR 15606 es parte de las especificaciones de codificación de datos para el sistema brasileño de televisión digital terrestre (SBTVD) y comprende la especificación del lenguaje utilizado por la máquina de presentación Ginga-NCL del *middleware* SBTVD, denominado Ginga.

Por medio de ese lenguaje, denominado NCL (*Nested Context Language* - Lenguaje de Contextos Anidados), un autor puede describir el comportamiento temporal de una presentación multimedia, asociar *hyperlinks* (interacción del usuario) a objetos de media, definir alternativas para presentación (adaptación) y describir el layout de la presentación en múltiples dispositivos.

Esta Norma se destina fundamentalmente a las entidades que están especificando terminales y/o estándares basados en el Ginga. También se destina a los desarrolladores de aplicaciones que utilizan las funcionalidades del Ginga y de sus API. El *middleware* Ginga tiene como objeto garantizar la interoperabilidad de las aplicaciones en diferentes implementaciones de plataformas que lo soportan.

Las aplicaciones Ginga se clasifican en dos categorías, dependiendo si la aplicación inicialmente procesada posee contenido de naturaleza declarativa o imperativa. Esas categorías de aplicaciones son denominadas aplicaciones declarativas y aplicaciones imperativas, respectivamente. Los ambientes de aplicación se clasifican del mismo modo en dos categorías, dependiendo si procesan aplicaciones declarativas o imperativas, siendo entonces denominados Ginga-NCL y Ginga-J, respectivamente.

Una implementación únicamente Ginga-NCL es permitida para receptores portátiles, siendo que, en ese caso, la implementación del ambiente Ginga-J es opcional. Al ser implementado solamente con el Ginga-NCL, el *middleware* Ginga brinda soporte a códigos imperativos a través del lenguaje Lua.

Esta Norma no especifica la forma de implementación de los ambientes de aplicación en un receptor.

Televisión digital terrestre — Codificación de dados y especificaciones de transmisión para radiodifusión digital Parte 5: Ginga-NCL para receptores transportables – Lenguaje de aplicación XML para codificación de aplicaciones

1 Alcance

Esta parte de la ABNT NBR 15606 especifica un lenguaje de aplicación XML denominado NCL *(Nested Context Language),* el lenguaje declarativo del *middleware* Ginga, la codificación y la transmisión de datos para radiodifusión digital.

2 Referencias normativas

Los documentos indicados a continuación son indispensables para la aplicación de este documento. Para las referencias fechadas, se aplican solamente las ediciones citadas. Para las referencias sin fecha, se aplican las ediciones más recientes del documento citado (incluyendo enmiendas).

ABNT NBR 15603-2:2007, *Televisión digital terrestre — Multiplexación y servicios de información (SI) — Parte 2: Estructura de datos y definiciones de la información básica de SI*

ABNT NBR 15606-1, *Televisión digital terrestre — Codificación de datos y especificaciones de transmisión para radiodifusión digital — Parte 1: Codificación de datos*

ABNT NBR 15606-2: 2007, *Televisión digital terrestre — Codificación de datos y especificaciones de transmisión para radiodifusión digital — Parte 2: Ginga-NCL para receptores fijos y móviles — Lenguaje de aplicación XML para codificación de aplicaciones*

ISO/IEC 13818-1:2008, *Information technology — Generic coding of moving pictures and associated audio information: Systems*

ECMA 262, *ECMAScript language specification*

3 Términos y definiciones

Para los efectos de esta parte de la ABNT NBR 15606, se aplican los siguientes términos y definiciones.

3.1
ambiente de aplicación
contexto o ambiente de *software* en el cual se procesa una aplicación

3.2
ambiente de aplicación declarativa
ambiente que brinda soporte al procesamiento de aplicaciones declarativas

NOTA Un formateador (*user agent*) NCL es un ejemplo de ambiente de aplicación declarativa.

3.3
ambiente de aplicación imperativa
ambiente que brinda soporte al procesamiento de aplicaciones imperativas

3.4
API DON
API que define la estructura lógica de un documento XML y la forma de acceder, o manejar, un documento XML

NOTA Esta API es una interfaz independiente de plataformas y lenguajes y sigue el Modelo DOM (*Document Object Model*).

3.5
aplicación
información que expresa un conjunto específico de comportamientos observables

3.6
aplicación declarativa
aplicación que utiliza principalmente, y como punto de partida, información declarativa para expresar su comportamiento

NOTA Una instancia de documento NCL es un ejemplo de aplicación declarativa.

3.7
aplicación híbrida
aplicación híbrida declarativa o aplicación híbrida imperativa

3.8
aplicación híbrida declarativa
aplicación declarativa que contiene contenido de objeto activo

NOTA Un documento NCL con un Java Xlet embutido es un ejemplo de aplicación híbrida declarativa.

3.9
aplicación híbrida imperativa
aplicación imperativa con contenido declarativo

NOTA Un Java Xlet que crea y causa la exhibición de una instancia de documento NCL es un ejemplo de aplicación híbrida imperativa.

3.10
aplicación nativa
función intrínseca implementada por una plataforma receptora

NOTA Una exhibición en *closed caption* es un ejemplo de aplicación nativa.

3.11
aplicación imperativa
aplicación que utiliza principalmente, y como punto de partida, informaciones imperativas para expresar su comportamiento

NOTA Un programa en Java es un ejemplo de una aplicación imperativa.

3.12
almacenamiento persistente
memoria disponible que puede ser leída o grabada por una aplicación y puede ser mantenida por más tiempo que el tiempo de vida de la misma aplicación

NOTA El almacenamiento persistente puede ser volátil o no volátil.

3.13
atributo
parámetro que representa la modalidad de una propiedad

3.14
atributo de un elemento
propiedad de un elemento XML

3.15
audio principal
audio básico
flujo básico de audio cuya component_tag es igual a 0x10 para el receptor *full-seg* y 0x83 ó 0x85 para el receptor *one-seg*

3.16
autor
persona que especifica documentos NCL

3.17
canal de interactividad
canal de retorno
mecanismo de comunicación que suministra conexión entre el receptor y un servidor remoto

3.18
carácter
"letra" específica u otro símbolo identificable

EJEMPLO “A”

3.19
carrusel de datos
método que envía cualquier conjunto de datos en forma cíclica, para que esos datos se puedan obtener, vía radiodifusión, en un intervalo de tiempo tan largo como sea necesario

[ISO/IEC 13818-6:2001]

3.20
ciclo de vida de una aplicación
caracteriza el período de tiempo, desde el momento en que la aplicación es cargada hasta el momento en que es destruida

3.21
codificación de caracteres
mapeo entre un valor de entrada entero y el carácter textual, representado por ese mapeo

3.22
contenido de objeto activo
tipo de contenido que toma la forma de un programa ejecutable

NOTA Un Xlet Java compilado es un ejemplo de contenido de objeto activo.

3.23
contenido NCL
conjunto de informaciones que consiste en un documento NCL y en un grupo de datos, incluyendo objetos (de media o de ejecución), que acompañan el documento NCL

3.24
digital storage media command and control
DSM-CC
método de control que suministra acceso a un archivo o flujo en servicios digitales interactivos

[ISO/IEC 13818-6:2001]

3.25
document type definition
DTD
declaración que describe un tipo de documento XML

3.26
ECMAScript
lenguaje de programación definido en la ECMA 262

3.27
elemento
unidad de estructuración del documento delimitada por *tags*

NOTA Por lo general, un elemento es delimitado por una *tag* inicial y una *tag* final, excepto un elemento vacío que es delimitado por una *tag* de elemento vacío.

3.28
elemento *property*
elemento NCL que define un nombre de propiedad y su valor asociado

3.29
entidad de la aplicación
unidad de información que expresa alguna parte de una aplicación

3.30
evento
ocurrencia en el tiempo que puede ser instantánea o tener duración mensurable

3.31
exhibidor de mídia
media player
componente de uno ambiente de aplicación que decodifica o ejecuta uno tipo específico de contenido

3.32
eXtensible HTML
XHTML
versión extendida del HTML como aplicación XML

NOTA En la especificación XHTML, un documento HTML es reconocido como aplicación XML.

3.33
herramienta de autoría
herramienta para ayudar a los autores a crear documentos NCL

3.34
fuente
mecanismo que permite la renderización específica de un carácter

EJEMPLO Tiresias, 12 puntos.

NOTA En la práctica, un formato de fuente incorpora aspectos de la codificación de un carácter.

3.35
formateador NCL
componente de *software* responsable por recibir la especificación de un documento NCL y controlar su presentación, intentando garantizar que se respeten las relaciones entre los objetos de media, especificados por el autor

NOTA Renderizador (*renderer*) de documentos, agente del usuario (*user agent*) y exhibidor son otros nombres que se utilizan con el mismo significado del formateador de documentos.

3.36
flujo de transporte
se refiere a la sintaxis del flujo de transporte MPEG-2 para empaquetado y multiplexación de video, audio y señales de datos en sistemas de radiodifusión digital

3.37
flujo elemental
elementary stream
ES
flujo básico que contiene datos de video, audio, o datos privados

NOTA Un único flujo elemental se transporta en una secuencia de paquetes PES con un, y sólo un, identificador (*stream_id*).

3.38
gestor de aplicaciones
entidad responsable por la administración del ciclo de vida de las aplicaciones y que administra las aplicaciones, funcionando tanto en la máquina de presentación como en la máquina de ejecución

3.39
identificador de paquete
PID
valor entero único, utilizado para asociar los flujos elementales de un programa, tanto en un flujo de transporte único como multiprograma

3.40
información de servicio
SI
datos que describen programas y servicios

3.41
informaciones específicas del programa
program specific information
PSI
datos normativos necesarios para demultiplexar los flujos de transporte y regenerar los programas

3.42
Interfaz de programación de la aplicación
API
bibliotecas de *software* que ofrecen acceso uniforme a los servicios del sistema

3.43
lenguaje de marcación
formalismo que describe una clase de documentos que emplean marcación para delinear la estructura, apariencia u otros aspectos del documento

3.44
lenguaje de *script*
lenguaje utilizado para describir un contenido de objeto activo incorporado en documentos NCL y en documentos HTML

3.45
localizador
identificador que suministra una referencia a una aplicación o recurso

3.46
máquina de presentación
subsistema en un receptor que analiza y presenta aplicaciones declarativas, con contenidos como audio, video, gráficos y texto, con base en reglas definidas en la máquina de presentación

NOTA Una máquina de presentación es responsable por el control del comportamiento de la presentación y por iniciar otros procesos en respuesta a entradas del usuario y otros eventos.

EJEMPLO Navegador HTML y formateador NCL.

3.47
máquina de ejecución
subsistema en un receptor que evalúa y ejecuta aplicaciones imperativas, compuestas por instrucciones en lenguaje de computadora, contenido de media asociados y otros datos

NOTA Una máquina de ejecución se puede implementar con un sistema operativo, compiladores de lenguaje de computadora, interpretadores e interfaces de programación de aplicaciones (API), que una aplicación imperativa puede utilizar para presentar contenido audiovisual, interactuar con el usuario o ejecutar otras tareas que no sean evidentes para el usuario.

EJEMPLO Ambiente de *software* JavaTV, utilizando lenguaje de programación Java e interpretador *bytecode*, API JavaTV y máquina virtual Java para ejecución del programa.

3.48
método
función asociada a un objeto autorizado para manejar los datos del objeto

3.49
nudo NCL
elemento <media>, <context>, <body> o <switch> de NCL

3.50
normal play time
NPT
coordenada temporal absoluta que representa la posición en un flujo

3.51
objeto de media
colección de pedazos de datos identificados por nombre que puede representar un contenido de media o un programa escrito en lenguaje específico

3.52
perfil
especificación de una clase de capacidades, ofreciendo diferentes niveles de funcionalidades en un receptor

3.53
perfil *one-seg*
caracteriza el servicio que puede ser recibido por un sintonizador de banda estrecha (430 KHz) y por lo tanto con ahorro en el consumo de batería

NOTA El perfil *one-seg* también es conocido como perfil portátil.

3.54
perfil *full-seg*
caracteriza el servicio que necesita necesariamente un demodulador de banda ancha (5,7 MHz) para ser recibido

NOTA Dependiendo de las configuraciones de transmisión y de funcionalidad específicas del receptor, se puede recibir en movimiento o sólo por receptores fijos, aunque sin el beneficio del ahorro de energía. La resolución del video transmitido puede ser o no de alta definición.

3.55
plug-in
conjunto de funcionalidades que se puede agregar a una plataforma genérica para suministrar funcionalidad adicional

3.56
plataforma receptora
plataforma
hardware, sistema operativo y bibliotecas de *software* nativas del receptor, elegidos por el fabricante

3.57
recurso
objeto de datos o un servicio de la red que es identificado unívocamente

3.58
sistema de archivos local
sistema de archivos suministrado por la plataforma receptora local

3.59
tiempo de vida de una aplicación
período de tiempo entre el momento en que una aplicación se carga y el momento en que se destruye

3.60
uniform resource identifier
URI
método de encaminamiento que permite el acceso a objetos en una red

3.61
user agent
agente del usuario
cualquier programa que interpreta un documento escrito en el lenguaje NCL

NOTA Un *user agent* puede exhibir un documento, intentando garantizar que se respeten las relaciones especificadas por el autor entre objetos de media, pronunciarlo en audio sintetizado, convertirlo en otro formato etc.

3.62
usuario
persona que interactúa con un formateador para visualizar, oír o utilizar de otra forma un documento NCL

3.63
usuario final
individuo que opera o interactúa con un receptor

4 Abreviaturas

Para los efectos de esta parte de la ABNT NBR 15606, se aplican las siguientes abreviaturas:

API	*Application Programming Interface*
BML	*Broadcast Markup Language*

CLUT	*Color Look-up Table*
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*
DSM-CC	*Digital Storage Media Command and Control*
DTD	*Document Type Definition*
DTV	*Digital Television*
DVB	*Digital Video Broadcasting*
GIF	*Graphics Interchange Format*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
JPEG	*Joint Photographic Expert Group*
MIME	*Multipurpose Internet Mail Extensions*
MNG	*Multiple Network Graphics*
MPEG	*Moving Picture Expert Group*
NCL	*Nested Context Language*
NCM	*Nested Context Model*
NPT	*Normal Play Time*
OS	*Operating System*
PAT	*Program Association Table*
PES	*Packetized Elementary Stream*
PID	*Packet Identifier*
PMT	*Program Map Table*
PNG	*Portable Network Graphics*
PSI	*Program Specific Information*
SBTVD	*Sistema Brasileño de Televisión Digital Terrestre*
SMIL	*Synchronized Multimedia Integration Language*
TS	*Transport Stream*
UCS	*Universal (Coded) Character Set*
URI	*Universal Resource Identifier*
URL	*Universal Resource Locator*
XHTML	*eXtensible HTML*
XML	*Extensible Markup Language*
W3C	*World-Wide Web Consortium*

5 Arquitectura Ginga

5.1 Ginga *main modules*

El universo de las aplicaciones Ginga se puede dividir en un conjunto de aplicaciones declarativas y un conjunto de aplicaciones imperativas. Una aplicación declarativa es aquella donde el tipo del contenido de la entidad inicial es declarativo. Por otro lado, una aplicación imperativa es aquella cuyo tipo del contenido de la entidad inicial es imperativa. Una aplicación declarativa pura es aquella en la cual el contenido de todas las entidades es del tipo declarativo, y una aplicación imperativa pura es aquella en la cual el contenido de todas las entidades es del tipo imperativa. Una aplicación híbrida es aquella cuyo conjunto de entidades posee tanto contenido del tipo declarativo como imperativa. Una aplicación Ginga no necesita ser puramente declarativa o imperativa.

En particular, las aplicaciones declarativas frecuentemente utilizan scripts, cuyo contenido es de modalidad imperativa. Además de ello, una aplicación declarativa puede hacer referencia a un código Java TV Xlet incorporado. Del mismo modo, una aplicación imperativa puede hacer referencia a una aplicación declarativa, conteniendo, por ejemplo, contenido gráfico, o puede construir e iniciar la presentación de aplicaciones con contenido declarativo. Por lo tanto, ambos tipos de aplicación Ginga pueden utilizar las facilidades de los ambientes de aplicación: declarativo y imperativa.

Ginga-NCL, ambiente obligatorio para receptores portátiles, es un subsistema lógico del sistema Ginga responsable por el procesamiento de documentos NCL. Un componente clave del Ginga-NCL es la máquina de interpretación del contenido declarativo (formateador NCL). Otros módulos importantes son el exhibidor (user agent) XHTML y la máquina de presentación Lua, que es responsable por la interpretación de los scripts Lua (ver la ABNT NBR 15606-2:2007, Anexo B).

NOTA NCL es marca registrada y su especificación es propiedad intelectual de la PUC-Rio (INPI Departamento de Transferencia Tecnológica - No. 0007162-5; 20/12/2005).

Ginga-J, ambiente obligatorio para receptores portátiles, es un subsistema lógico del sistema Ginga, responsable por el procesamiento de contenidos activos. Un componente clave del ambiente de aplicación imperativa es la máquina de ejecución del contenido imperativa, compuesta por una máquina virtual Java.

Decodificadores de contenido comunes sirven tanto para las aplicaciones imperativas con respecto a las declarativas que necesitan decodificar y presentar tipos comunes de contenido como PNG, JPEG, MPEG y otros formatos. El núcleo común Ginga (Ginga *Common Core*) está compuesto por los decodificadores de contenido comunes y por procedimientos para obtener contenidos transportados en flujos de transporte (transport streams) MPEG-2 y a través del canal de interactividad. El Núcleo Común Ginga también debe soportar obligatoriamente el modelo conceptual de exhibición, tal como se describe en la ABNT NBR 15606-1.

La arquitectura (ver Figura 1) y facilidades Ginga fueron proyectadas para ser aplicadas en sistemas de transmisión y recepción terrestres de radiodifusión. Adicionalmente, la misma arquitectura y facilidades se pueden aplicar a sistemas que utilizan otros mecanismos de transporte de datos (como sistemas de televisión vía satélite o por cable).

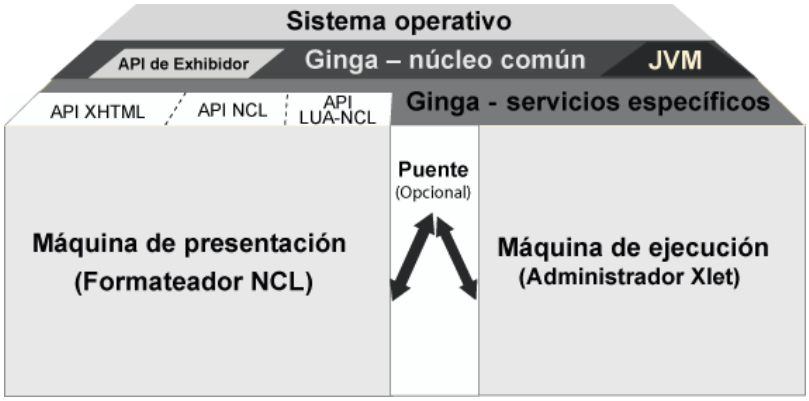

Figura 1 — Arquitectura Ginga

5.2 Interacción con el ambiente nativo

En general, Ginga es independiente de cualquier aplicación nativa que pueda también optar por utilizar el plano gráfico. Eso incluye, pero no se limita a aplicaciones como: *closed caption,* mensajes del sistema de acceso condicional (CA), menús del receptor y guías de programación nativos.

Las aplicaciones nativas pueden tener prioridad sobre las aplicaciones Ginga. El *closed caption* y los mensajes de emergencia deben tener obligatoriamente prioridad sobre el sistema Ginga.

Algunas aplicaciones nativas, como el *closed caption*, representan un caso especial en el cual la aplicación nativa puede estar activa por largos períodos en conjunto con las aplicaciones Ginga.

6 Objetos XHTML incorporados en presentaciones NCL

6.1 NCL como lenguaje cola

Diferentemente de XHTML o HTML, NCL define una separación bien delimitada entre el contenido y la estructura de un documento (o aplicación), probando un control no invasivo del enlace entre el contenido y su presentación y layout.

El foco del lenguaje declarativo NCL es más amplio que el ofrecido por la XHTML. La sincronización espacio-temporal, definida genéricamente por los enlaces NCL; adaptabilidad, definida por los elementos switch y descriptor switch de NCL; y soporte a múltiples dispositivos de exhibición, definidos por regiones NCL, es el foco de ese lenguaje declarativo. La interacción del usuario se trata sólo como un caso particular de sincronización temporal.

Como la NCL tiene una separación más exacta entre el contenido y la estructura, la misma no define ninguna media en sí. Al contrario, define la cola que sujeta la media en presentaciones multimedia.

Un documento NCL sólo define cómo los objetos de media son estructurados y relacionados en el tiempo y espacio. Como un lenguaje de cola, la misma no restringe o prescribe los tipos de contenido de los objetos de media. En ese sentido, es posible tener objetos de imagen (GIF, JPEG etc.), de video (MPEG, MOV etc.), de audio (MP3, WMA etc.), de texto (TXT, PDF etc.), de ejecución (Xlet, Lua etc.), entre otros, como objetos de media NCL. Cuáles objetos de media son soportados depende de los exhibidores de media que están acoplados al formateador NCL (exhibidor NCL). Uno de esos exhibidores es el decodificador/exhibidor MPEG-4, normalmente implementado en hardware en el receptor de televisión digital. De esa forma, el video y el audio MPEG-4 principal se tratan como todos los demás objetos de media que pueden estar relacionados utilizando NCL.

Otro objeto de media NCL que debe ser obligatoriamente soportado es el objeto de media basado en XHTML. La NCL no reemplaza, pero incorpora documentos (u objetos) basados en XHTML. Como ocurre con otros objetos de media, qué lenguaje basado en XHTML tiene soporte en un formateador NCL es una elección de implementación y, por lo tanto, depende de qué navegador XHTML, incorporado en el formateador NCL, actúa como exhibidor de esa media.

Aunque un navegador XHTML deba ser obligatoriamente soportado, se recomienda evitar la utilización de elementos XHTML para definir relaciones (incluso enlaces XHTML) en la autoría de documentos NCL. Se recomienda que la autoría con base en la estructura sea priorizada por razones conocidas y ampliamente divulgadas en la literatura.

Durante la exhibición del contenido de objetos de media se generan varios eventos. Algunos ejemplos son la presentación de parte del contenido de un objeto de media, la selección de parte del contenido de un objeto etc. Los eventos pueden generar acciones sobre otros objetos de media, como iniciar o terminar sus presentaciones. Por lo tanto, los eventos deben ser obligatoriamente relatados por los exhibidores de media al formateador NCL que, a su vez, puede generar acciones a ser aplicadas a ésos u otros exhibidores. Ginga-NCL define la API (ver Sección 8) de un adaptador con el objetivo de estandarizar la interfaz entre el formateador Ginga-NCL y cada exhibidor específico.

Para que cualquier exhibidor de media, en particular un navegador XHTML, sea acoplado al formateador Ginga-NCL, debe brindar soporte obligatoriamente a la API de los adaptadores. Así, para algunos exhibidores de media, incluso navegadores XHTML, puede ser necesario un módulo adaptador para alcanzar la integración.

Para edición en vivo, el Ginga-NCL también define eventos de flujo NCL para ofrecer soporte a los eventos generados en vivo sobre flujos de media, en particular sobre el flujo de video del programa principal. Esos eventos son una generalización del mismo concepto encontrado en otras normas, como, por ejemplo, los b-events de BML Aunque un navegador XHTML deba ser obligatoriamente soportado, se recomienda evitar la utilización de elementos XHTML para definir relaciones (incluso eventos de flujo) durante la creación de documentos NCL, por la misma razón, es decir, se recomienda que la autoría con base en la estructura sea priorizada por razones conocidas y ampliamente divulgadas en la literatura.

6.2 Formato de contenido XHTML

Formatos comunes de contenido deben ser obligatoriamente adoptados para la producción e intercambio de contenido multimedia, como definido en la ABNT NBR 15606-1. Además de ello, en el ambiente de aplicación declarativa también se exige la especificación de formatos comunes de contenidos XHTML para las aplicaciones de televisión interactiva.

De este modo, esta Norma también especifica los elementos obligatorios de los objetos de media XHTML incorporados en aplicaciones NCL, así como las propiedades de hojas de estilo obligatorias.

6.3 XHTML para el perfil portátil

6.3.1 Marcaciones XML

NOTA Objetos de media NCL basados en XHTML siguen la recomendación W3C "*Modularization of XHTML*".

Las colecciones de atributo XHTML se definen de acuerdo con la Tabla 1. Las marcaciones XML que deben ser soportadas obligatoriamente por cualquier implementación, se listan en la Tabla 2.

Tabla 1 — Colecciones de atributos

Nombre de la colección	Atributos en la colección	Condición del atributo
Core	class (NMTOKENS)	Requerido
	Id (ID),	Requerido
	title (CDATA)	–
I18N	xml:lang (CDATA)	Requerido (default)
Events	onclick (Script)	-
	ondblclick (Script)	–
	onmousedown (Script)	–
	onmouseup (Script)	–
	onmouseover (Script)	–
	onmousemove (Script)	–
	onmouseout (Script)	–
	onkeypress (Script)	–
	onkeydown (Script)	-
	onkeyup (Script)	-
Style	style (CDATA)	Requerido
Common	Core + Events + I18N + Style	

Tabla 2 — Elementos de marcación XML obligatorios

Módulo		Elemento	Condición del elemento	Atributo	Condición del atributo
Core	Structure	body	Requerido	%Common.attrib	
				%Core.attrib	Requerido
				%I18n.attrib	Requerido
				%Events.attrib	–
		head	Requerido	%I18n.attrib	Requerido
				profile	–
		html	Requerido		
		title	Requerido	%I18n.attrib	Requerido
	Text	abbr	–		
		acronym	–		
		address	–		
		blockquote	–		
		br	Requerido	%Core.attrib	Requerido
		cite	–		
		code	–		
		dfn	–		
		div	Requerido	%Common.attrib	Requerido
		em	–		
		h1	Requerido	%Common.attrib	Requerido
		h2	Requerido	%Common.attrib	Requerido
		h3	Requerido	%Common.attrib	Requerido
		h4	Requerido	%Common.attrib	Requerido
		h5	Requerido	%Common.attrib	Requerido
		h6	Requerido	%Common.attrib	Requerido
		kbd	–		
		p	Requerido	%Common.attrib	Requerido
		pre	–		
		q	–		
		samp	–		
		span	Requerido	%Common.attrib	Requerido
		strong	–		
		var	–		
	Hypertext	a	Requerido	%Common.attrib	Requerido
				accesskey	Requerido
				charset	Requerido
				href	Requerido
				hreflang	–
				rel	–
				rev	–
				tabindex	–
				type	–
	List	dl	–		
		dt	–		
		dd	–		
		ol	–		
		ul	–		
		li	–		

Tabla 2 (continuación)

Módulo		Elemento	Condición del elemento	Atributo	Condición del atributo
Applet		applet	–		
		param	–		
Text extension	Presentation	b	–		
		big	–		
		hr	–		
		i	–		
		small	–		
		sub	–		
		sup	–		
		tt	–		
	Edit	del	–		
		ins	–		
	Bi-directional text	bdo	–		
Forms	Basic forms	form	–		
		input	–		
		label	–		
		select	–		
		option	–		
		textarea	–		
	Forms	form	Requerido	%Common.attrib	Requerido
				action	Requerido
				method	Requerido
				enctype	Requerido
				accept-charset	Requerido
				accept	Requerido
				name	Requerido
		input	Requerido	%Common.attrib	Requerido
				accesskey	Requerido
				checked	–
				disabled	Requerido
				readonly	Requerido
				maxlength	Requerido
				alt	–
				name	–
				size	Requerido
				src	–
				tabindex	–
				accept	–
				type	Requerido
				value	Requerido
		select	Requerido		
		option	Requerido		
		textarea	Requerido		
		button	–		
		fieldset	–		
		label	–		
		legend	–		
		optgroup	–		

Tabla 2 (continuación)

Módulo		Elemento	Condición del elemento	Atributo	Condición del atributo
Table	Basic tables	caption	–		
		table	Requerido		
		td	Requerido		
		th	–		
		tr	Requerido		
	Tables	caption	–		
		table	–		
		td	–		
		th	–		
		tr	–		
		col	–		
		colgroup	–		
		tbody	–		
		thead	–		
		tfoot	–		
Image		img	–		
Client side map		a&	–		
		area	–		
		img&	–		
		input&	–		
		map	–		
		object&	–		
Server side image map		img&	–		
		Input&	–		
Object		object	Requerido	%Common.attrib	Requerido
				archive	–
				classid	–
				codebase	–
				codetype	–
				data	Requerido
				declare	–
				height	Requerido
				name	–
				standby	–
				tabindex	–
				type	Requerido
				width	Requerido
		param	–		
Frames		frameset	–		
		frame	–		
		noframe	–		
Target		a&	–		
		area&	–		
		base&	–		
		link&	–		
		form&	–		
IFrame		iframe	–		

Tabla 2 (continuación)

Módulo	Elemento	Condición del elemento	Atributo	Condición del atributo
Intrinsic events	a&	–		
	area&	–		
	frameset&	–		
	form&	–		
	body&	–		
	label&	–		
	input&	–		
	select&	–		
	textarea&	–		
	button&	–		
Metainformation	meta	Requerido	%I18n.attrib	–
			http-equiv	–
			name	Requerido
			content	Requerido
			scheme	–
Scripting	noscript			
	script	–	charset	
			type	
			src	
			defer	
Stylesheet	style	Requerido	%I18n.attrib	Requerido
			id	–
			type	Requerido
			media	Requerido
			title	–
Style attribute		Requerido		
Link	link	Requerido		
Base	base	–		

6.3.2 Hojas de estilo

Las propiedades de hojas de estilo que deben ser soportadas obligatoriamente por cualquier implementación están listadas en la Tabla 3.

Tabla 3 — Propiedades de hojas de estilo CSS 2 obligatorias

Propiedad	Condición de la propiedad
Value assignment/Inheritance	
@import	–
!important	–
Media type	
@media	Requerido
box model	
margin-top	–
margin-right	–
margin-bottom	–
margin-left	–
margin	Requerido
padding-top	Requerido
padding-right	Requerido
padding-bottom	Requerido
padding-left	Requerido
padding	Requerido
border-top-width	–
border-right-width	–
border-bottom-width	–
border-left-width	–
border-width	Requerido
border-top-color	–
border-right-color	–
border-bottom-color	–
border-left-color	–
border-color	Requerido
border-top-style	–
border-right-style	–
border-bottom-style	–
border-left-style	–
border-style	Requerido
border-top	–
border-right	–
border-bottom	–
border-left	–
border	Requerido
Visual formatting model	
position	Requerido
left	Requerido
top	Requerido
width	Requerido
height	Requerido
z-index	Requerido
line-height	Requerido
vertical-align	–
display	Requerido
bottom	–
right	–
float	–
clear	–
direction	–

Tabla 3 (continuación)

Propiedad	Condición de la propiedad
unicode-bidi	–
min-width	–
max-width	–
min-height	–
max-height	–
Other visual effects	
visibility	Requerido
overflow	Requerido
clip	–
Generated content/Auto numbering/List	
content	–
quotes	–
counter-reset	–
counter-increment	–
marker-offset	–
list-style-type	–
list-style-image	–
list-style-position	–
list-style	–
Page media	
"@page"	–
size	–
marks	–
page-break-before	–
page-break-after	–
page-break-inside	–
page	–
orphans	–
widows	–
Background	
background	–
background-color	–
background-image	Requerido
background-repeat	Requerido
background-position	–
background-attachment	–
Font	
color	Requerido
font-family	Requerido
font-style	Requerido
font-size	Requerido
font-variant	Requerido
font-weight	Requerido
font	Requerido
font-stretch	–
font-adjust	–
Text	
text-indent	–
text-align	Requerido
text-decoration	–

Tabla 3 (continuación)

Propiedad	Condición de la propiedad
text-shadow	–
letter-spacing	Requerido
word-spacing	–
text-transform	–
white-space	Requerido
Pseudo class/ Pseudo element	
:link	–
:visited	–
:active	Requerido
:hover	–
:focus	Requerido
:lang	–
:first-child	–
:first-line	–
:first-letter	–
:before	–
:after	–
Table	
caption-side	–
border-collapse	–
border-spacing	–
table-layout	–
empty-cells	–
speak-header	–
User interface	
outline-color	–
outline-width	–
outline-style	–
outline	–
cursor	–
Voice style sheet	
volume	–
speak	–
pause-before	–
pause-after	–
pause	–
cue-before	–
cue-after	–
cue	–
play-during	–
azimuth	–
elevation	–
speech-rate	–
voice-family	–
pitch	–
pitch-range	–
stress	–
richness	–
speak-punctuation	–
peak-numeral	–

Tabla 3 (continuación)

Propiedad	Condición de la propiedad
Extended property	
clut	–
color-index	–
background-color-index	–
border-color-index	–
border-top-color-index	–
border-right-color-index	–
border-bottom-color-index	–
border-left-color-index	–
outline-color-index	–
resolution	–
display-aspect-ratio	–
grayscale-color-index	–
nav-index	–
nav-up	–
nav-down	–
nav-left	–
nav-right	–
used-key-list	–

Las siguientes restricciones se deberán aplicar obligatoriamente a las propiedades de exhibición:

- Solamente elementos de bloque se pueden aplicar para <p>, , <body>, <input> y <object>;
- solamente valores definidos en el mismo elemento HTML se pueden aplicar para
, <a> y .

Además de ello, las siguientes restricciones se deberán aplicar obligatoriamente a las propiedades de posición:

- solamente valores absolutos se pueden aplicar para <p>, , <input> y <object>;
- solamente valores estáticos se pueden aplicar para
, y <a>.

Los selectores CSS que deben ser soportados obligatoriamente por cualquier implementación son los siguientes:

- *universal*;
- *type*;
- *class*;
- *id*.

7 NCL - Lenguaje declarativo XML para especificación de presentaciones multimedia interactivas

7.1 Lenguajes modulares y perfiles de lenguajes

7.1.1 Módulos NCL

El abordaje modular ha sido utilizado en varios lenguajes recomendados por el W3C.

Módulos son colecciones de elementos, atributos y valores de atributos XML semánticamente relacionados que representan una unidad de funcionalidad. Módulos se definen en conjuntos coherentes. Esa coherencia se expresa por medio de la asociación de un mismo *namespace* a los elementos de esos módulos.

NOTA *Namespaces* se discuten en Namespaces in XML:1999.

Un perfil de lenguaje es una combinación de módulos. Los módulos son atómicos, es decir, no se pueden subdividir cuando incluidos en un perfil de lenguaje. Además de ello, la especificación de un módulo puede incluir un conjunto de requisitos para integración, con el cual los perfiles de lenguaje, que incluyen el módulo, deben ser compatibles obligatoriamente.

NCL fue especificada de forma modular, permitiendo la combinación de sus módulos en perfiles de lenguaje. Cada perfil puede agrupar un subconjunto de módulos NCL, permitiendo la creación de lenguajes orientados hacia las necesidades específicas de los usuarios. Además de ello, los módulos y perfiles NCL se pueden combinar con módulos definidos en otros lenguajes, permitiendo la incorporación de características de la NCL en esos lenguajes y viceversa.

Normalmente, hay un perfil de lenguaje que incorpora casi todos los módulos asociados a un único *namespace*. Ése es el caso del perfil *Lenguaje NCL*.

Otros perfiles de lenguaje se pueden especificar como subconjuntos de un perfil mayor o incorporar una combinación de módulos asociados a diferentes *namespaces.* Ejemplos del primer caso son los perfiles TVD Básico (perfil BDTV) y TVD Avanzado (perfil EDTV) de la NCL.

Subconjuntos de los módulos del perfil Lenguaje NCL utilizados en la definición de los perfiles TVD Básico y TVD Avanzado se definen para ajustar el lenguaje a las características del ambiente de radiodifusión de televisión, con sus varios dispositivos de presentación: Aparato de televisión, dispositivos móviles etc.

NOTA Un abordaje análogo también se encuentra en otros lenguajes (SMIL 2.1 Specification:2005 y XHTML 1.0:2002).

El principal objetivo de la conformidad con perfiles de lenguaje es aumentar la interoperabilidad. Los módulos obligatorios se definen de tal forma que cualquier documento, especificado de conformidad con un perfil de lenguaje, da como resultado una presentación razonable cuando es presentado en un perfil distinto de aquél para el cual fue especificado. El formateador de documentos, soportando el conjunto de módulos obligatorios, ignoraría todos los otros elementos y atributos desconocidos.

NOTA Renderizador de documentos, agente del usuario y exhibidor son otros nombres atribuidos al formateador de documentos.

El perfil BDTV es el perfil mínimo exigido para dispositivos portátiles. En forma alternativa, se puede usar el perfil EDTV.

La versión NCL 3.0 revisa las funcionalidades contenidas en la NCL 2.3 (NCL Main Profile: 2005) y se divide en 15 áreas funcionales, que se subdividen nuevamente en módulos. A partir de las 15 áreas funcionales, 14 se utilizan para definir los perfiles TVD Avanzado y TVD Básico. Dos áreas funcionales tienen módulos con la misma semántica definida por SMIL 2.0. Las 14 áreas funcionales utilizadas y sus módulos correspondientes son:

1) *Structure*

Módulo *Structure*

2) *Layout*

Módulo *Layout*

3) *Components*

Módulo *Media*

Módulo *Context*

4*)* Interfaces

Módulo *MediaContentAnchor*

Módulo *CompositeNodeInterface*

Módulo *PropertyAnchor*

Módulo *SwitchInterface*

5) *Presentation Specification*

Módulo *Descriptor*

6) *Linking*

Módulo *Linking*

7) *Connectors*

Módulo *ConnectorCommonPart*

Módulo *ConnectorAssessmentExpression*

Módulo *ConnectorCausalExpression*

Módulo *CausalConnector*

Módulo *CausalConnectorFunctionality*

Módulo *ConnectorBase*

8) *Presentation Control*

Módulo *TestRule*

Módulo *TestRuleUse*

Módulo *ContentControl*

Módulo *DescriptorControl*

9) *Timing*

Módulo *Timing*

10) *Reuse*

Módulo *Import*

Módulo *EntityReuse*

Módulo *ExtendedEntityReuse*

11) *Navigational Key*

Módulo *KeyNavigation*

12) *Animation*

Módulo *Animation*

13) *Transition Effects*

Módulo *TransitionBase*

Módulo *Transition*

14) *Meta-Information*

Módulo *Metainformation*

7.1.2 Identificadores para módulos y perfiles de lenguaje de la NCL 3.0

Se recomienda que cada perfil NCL declare explícitamente el URI del *namespace* que será usado para identificarlo.

Los documentos creados en perfiles de lenguaje que incluyen el módulo *Structure* de NCL se pueden asociar con el tipo *MIME* “application/x-ncl+xml”. Los documentos que utilizan el tipo *MIME* “application/x-ncl+xml” deben obligatoriamente estar de conformidad con el lenguaje hospedero.

Los identificadores de *namespace* XML para el conjunto completo de módulos, elementos y atributos NCL 3.0 están contenidos en el siguiente *namespace*: http://www.ncl.org.br/NCL3.0/

Cada módulo NCL tiene un identificador único asociado a él. Los identificadores de los módulos NCL 3.0 deben estar de acuerdo obligatoriamente con la Tabla 4.

Los módulos también pueden ser identificados colectivamente. Las siguientes colecciones de módulos se definen:

— módulos utilizados por el perfil Lenguaje NCL 3.0: http://www.ncl.org.br/NCL3.0/LanguageProfile

— módulos utilizados por el perfil Conector Causal NCL 3.0: http://www.ncl.org.br/NCL3.0/CausalConnectorProfile

— módulos utilizados por el perfil DTV Avanzado NCL 3.0: http://www.ncl.org.br/NCL3.0/EDTVProfile

— módulos utilizados por el perfil DTV Básico NCL 3.0: http://www.ncl.org.br/NCL3.0/BDTVProfile

Tabla 4 — Identificadores de los módulos de NCL 3.0

Módulos	Identificadores
Animation	http://www.ncl.org.br/NCL3.0/Animation
CompositeNodeInterface	http://www.ncl.org.br/NCL3.0/CompositeNodeInterface
CausalConnector	http://www.ncl.org.br/NCL3.0/CausalConnector
CausalConnectorFunctionality	http://www.ncl.org.br/NCL3.0/CausalConnectorFunctionality
ConnectorCausalExpression	http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression
ConnectorAssessmentExpression	http://www.ncl.org.br/NCL3.0/ConnectorAssessmentExpression
ConnectorBase	http://www.ncl.org.br/NCL3.0/ConnectorBase
ConnectorCommonPart	http://www.ncl.org.br/NCL3.0/ConnectorCommonPart
ContentControl	http://www.ncl.org.br/NCL3.0/ContentControl
Context	http://www.ncl.org.br/NCL3.0/Context
Descriptor	http://www.ncl.org.br/NCL3.0/Descriptor
DescriptorControl	http://www.ncl.org.br/NCL3.0/DescriptorControl
EntityReuse	http://www.ncl.org.br/NCL3.0/EntityReuse
ExtendedEntityReuse	http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse
Import	http://www.ncl.org.br/NCL3.0/Import
Layout	http://www.ncl.org.br/NCL3.0/Layout
Linking	http://www.ncl.org.br/NCL3.0/Linking
Media	http://www.ncl.org.br/NCL3.0/Media
MediaContentAnchor	http://www.ncl.org.br/NCL3.0/MediaContentAnchor
KeyNavigation	http://www.ncl.org.br/NCL3.0/KeyNavigation
PropertyAnchor	http://www.ncl.org.br/NCL3.0/PropertyAnchor
Structure	http://www.ncl.org.br/NCL3.0/Structure
SwitchInterface	http://www.ncl.org.br/NCL3.0/SwitchInterface
TestRule	http://www.ncl.org.br/NCL3.0/TestRule
TestRuleUse	http://www.ncl.org.br/NCL3.0/TestRuleUse
Timing	http://www.ncl.org.br/NCL3.0/Timing
TransitionBase	http://www.ncl.org.br/NCL3.0/TransitionBase
Transition	http://www.ncl.org.br/NCL3.0/Transition
Metainformation	http://www.ncl.org.br/NCL3.0/MetaInformation

Tres modulos SMIL [SMIL 2.1 Specification, 2005] se usaron como base para la definición de los módulos NCL Transition y Metainformation. Los identificadores de estos módulos SMIL 2.0 están representados en la Tabla 5.

Tabla 5 – Identificadores de los módulos SMIL 2.0

Módulos	Identificadores
BasicTransitions	http://www.w3.org/2001/SMIL20/BasicTransitions
TransitionModifiers	http://www.w3.org/2001/SMIL20/TransitionsModifiers
Metainformation	http://www.w3.org/2001/SMIL20/Metainformation

7.1.3 Informaciones sobre versiones de NCL

Las siguientes instrucciones de procesamiento se deben incluir obligatoriamente en un documento NCL (identifican documentos NCL que contengan sólo los elementos definidos en esta Norma, y la versión NCL con la cual el documento está de acuerdo):

<?xml version="1.0" encoding="ISO-8859-1 "?>
<ncl id="cualquier string" xmlns="http://www.ncl.org.br/NCL3.0/profileName">

El atributo *id* del elemento <ncl> puede recibir como valor cualquier cadena de caracteres que sea compatible con el tipo NCName [*Namespaces* in XML: 1999]. O sea, puede tener como valor cualquier cadena de caracteres que comience con una letra o subrayado ("_") y que contenga sólo letras, dígitos, "." y "_".

El número de versión de una especificación NCL consiste en un número principal y otro secundario, separados por un punto. Los números son representados como una cadena de caracteres formada por números decimales, en la cual los ceros a la izquierda se suprimen. El número de versión inicial del estándar es 3.0.

Las nuevas versiones de la NCL deben ser obligatoriamente publicadas de acuerdo con la siguiente política de versión:

— si los receptores compatibles con versiones más antiguas aún pueden recibir un documento con base en la especificación revisada, con relación a correcciones de error o por motivos operativos, la nueva versión de la NCL debe ser obligatoriamente publicada con el número secundario actualizado;

— si los receptores compatibles con versiones más antiguas no pueden recibir un documento basado en las especificaciones revisadas, el número principal debe ser obligatoriamente actualizado.

Una versión específica está definida en URI http://www.ncl.org.br/NCLx.y/profileName, donde el número de la versión “x.y” se escribe inmediatamente después de la sigla “NCL”.

El nombre del perfil (*profileName*) en URI debe obligatoriamente ser *EDTVProfile* (Perfil TVD Avanzado), *BDTVProfile* (Perfil TVD Básico), o CausalConnectorProfile (Perfil Conector causal).

7.2 Módulos NCL

Los módulos NCL deben ser acordes con la ABNT NBR 15606-2:2007, Subsección 7.2.

7.3 Perfiles del lenguaje NCL para el SBTVD

7.3.1 Módulos de perfiles

Los módulos de perfiles deben ser acordes con la ABNT NBR 15606-2:2007, Subsección 7.3.1.

7.3.2 Esquema del perfil NCL 3.0 DTV avanzado

NCL30EDTV.xsd

```
<!--
XML Schema for the NCL Language

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/profiles/NCL30EDTV.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006
-->

<schema xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:animation="http://www.ncl.org.br/NCL3.0/Animation"
 xmlns:compositeInterface="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"
 xmlns:causalConnectorFunctionality="http://www.ncl.org.br/NCL3.0/CausalConnectorFunctionality"
 xmlns:connectorBase="http://www.ncl.org.br/NCL3.0/ConnectorBase"
 xmlns:connectorCausalExpression="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression"
 xmlns:contentControl="http://www.ncl.org.br/NCL3.0/ContentControl"
 xmlns:context="http://www.ncl.org.br/NCL3.0/Context"
 xmlns:descriptor="http://www.ncl.org.br/NCL3.0/Descriptor"
 xmlns:entityReuse="http://www.ncl.org.br/NCL3.0/EntityReuse"
 xmlns:extendedEntityReuse="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"
 xmlns:descriptorControl="http://www.ncl.org.br/NCL3.0/DescriptorControl"
 xmlns:import="http://www.ncl.org.br/NCL3.0/Import"
 xmlns:keyNavigation="http://www.ncl.org.br/NCL3.0/KeyNavigation"
 xmlns:layout="http://www.ncl.org.br/NCL3.0/Layout"
 xmlns:linking="http://www.ncl.org.br/NCL3.0/Linking"
 xmlns:media="http://www.ncl.org.br/NCL3.0/Media"
 xmlns:mediaAnchor="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"
 xmlns:propertyAnchor="http://www.ncl.org.br/NCL3.0/PropertyAnchor"
 xmlns:structure="http://www.ncl.org.br/NCL3.0/Structure"
 xmlns:switchInterface="http://www.ncl.org.br/NCL3.0/SwitchInterface"
 xmlns:testRule="http://www.ncl.org.br/NCL3.0/TestRule"
 xmlns:testRuleUse="http://www.ncl.org.br/NCL3.0/TestRuleUse"
 xmlns:timing="http://www.ncl.org.br/NCL3.0/Timing"
 xmlns:transitionBase="http://www.ncl.org.br/NCL3.0/TransitionBase"
 xmlns:metainformation="http://www.ncl.org.br/NCL3.0/Metainformation"
 xmlns:transition="http://www.ncl.org.br/NCL3.0/Transition"
 xmlns:profile="http://www.ncl.org.br/NCL3.0/EDTVProfile"
 targetNamespace="http://www.ncl.org.br/NCL3.0/EDTVProfile"
 elementFormDefault="qualified" attributeFormDefault="unqualified" >

 <!-- import the definitions in the modules namespaces -->
 <import namespace="http://www.ncl.org.br/NCL3.0/Animation"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Animation.xsd"/>
 <import namespace="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30CompositeNodeInterface.xsd"/>
 <import namespace="http://www.ncl.org.br/NCL3.0/CausalConnectorFunctionality"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30CausalConnectorFunctionality.xsd"/>
 <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorBase"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorBase.xsd"/>
 <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorCausalExpression.xsd"/>
 <import namespace="http://www.ncl.org.br/NCL3.0/ContentControl"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ContentControl.xsd"/>
 <import namespace="http://www.ncl.org.br/NCL3.0/Context"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Context.xsd"/>
 <import namespace="http://www.ncl.org.br/NCL3.0/Descriptor"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Descriptor.xsd"/>
```

```
<import namespace="http://www.ncl.org.br/NCL3.0/DescriptorControl"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30DescriptorControl.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/EntityReuse"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30EntityReuse.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ExtendedEntityReuse.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Import"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Import.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/KeyNavigation"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30KeyNavigation.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Layout"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Layout.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Linking"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Linking.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Media"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Media.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"

  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30MediaContentAnchor.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/PropertyAnchor"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30PropertyAnchor.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Structure"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Structure.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/SwitchInterface"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30SwitchInterface.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/TestRule"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30TestRule.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/TestRuleUse"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30TestRuleUse.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Timing"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Timing.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/TransitionBase"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30TransitionBase.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/Metainformation"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Metainformation.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Transition"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Transition.xsd"/>


<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Structure -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends ncl element -->

<element name="ncl" substitutionGroup="structure:ncl"/>

<!-- extends head element -->

<complexType name="headType">
  <complexContent>
    <extension base="structure:headPrototype">
      <sequence>
        <element ref="profile:importedDocumentBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:ruleBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:transitionBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:regionBase" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="profile:descriptorBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:connectorBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:meta" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="profile:metadata" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```
<element name="head" type="profile:headType" substitutionGroup="structure:head"/>

<!-- extends body element -->

<complexType name="bodyType">
  <complexContent>
    <extension base="structure:bodyPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:contextInterfaceElementGroup"/>
        <element ref="profile:media"/>
        <element ref="profile:context"/>
        <element ref="profile:switch"/>
        <element ref="profile:link"/>
        <element ref="profile:meta"/>
        <element ref="profile:metadata"/>

      </choice>
    </extension>
  </complexContent>
</complexType>

<element name="body" type="profile:bodyType" substitutionGroup="structure:body"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Layout -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends regionBase element -->

<complexType name="regionBaseType">
  <complexContent>
    <extension base="layout:regionBasePrototype">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="profile:importBase"/>
        <element ref="profile:region"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<complexType name="regionType">
  <complexContent>
    <extension base="layout:regionPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="regionBase" type="profile:regionBaseType" substitutionGroup="layout:regionBase"/>
<element name="region" type="profile:regionType" substitutionGroup="layout:region"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Media -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends Media elements -->

<!-- media interface element groups -->
<group name="mediaInterfaceElementGroup">
  <choice>
    <element ref="profile:area"/>
    <element ref="profile:property"/>
  </choice>
</group>

<complexType name="mediaType">
  <complexContent>
```



```
    <extension base="media:mediaPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:mediaInterfaceElementGroup"/>
      </choice>
      <attributeGroup ref="descriptor:descriptorAttrs"/>
      <attributeGroup ref="entityReuse:entityReuseAttrs"/>
      <attributeGroup ref="extendedEntityReuse:extendedEntityReuseAttrs"/>
    </extension>

  </complexContent>
</complexType>

<element name="media" type="profile:mediaType" substitutionGroup="media:media"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Context -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends context element -->

<!-- composite node interface element groups -->
<group name="contextInterfaceElementGroup">
  <choice>
    <element ref="profile:port"/>
    <element ref="profile:property"/>
  </choice>
</group>

<complexType name="contextType">
  <complexContent>
    <extension base="context:contextPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:contextInterfaceElementGroup"/>
        <element ref="profile:media"/>
        <element ref="profile:context"/>
        <element ref="profile:link"/>
        <element ref="profile:link"/>
        <element ref="profile:SWWITCHlink"/>
        <element ref="profile:meta"/>
        <element ref="profile:metadata"/>
      </choice>
      <attributeGroup ref="entityReuse:entityReuseAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="context" type="profile:contextType" substitutionGroup="context:context"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- MediaContentAnchor -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends area element -->

<complexType name="componentAnchorType">
  <complexContent>
    <extension base="mediaAnchor:componentAnchorPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="area" type="profile:componentAnchorType" substitutionGroup="mediaAnchor:area"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- CompositeNodeInterface -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends port element -->
```



```xml
<complexType name="compositeNodePortType">
  <complexContent>
    <extension base="compositeInterface:compositeNodePortPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="port" type="profile:compositeNodePortType" substitutionGroup="compositeInterface:port"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- PropertyAnchor -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends property element -->

<complexType name="propertyAnchorType">
  <complexContent>
    <extension base="propertyAnchor:propertyAnchorPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="property" type="profile:propertyAnchorType" substitutionGroup="propertyAnchor:property"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- SwitchInterface -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends switchPort element -->

<complexType name="switchPortType">
  <complexContent>
    <extension base="switchInterface:switchPortPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="mapping" substitutionGroup="switchInterface:mapping"/>
<element name="switchPort" type="profile:switchPortType" substitutionGroup="switchInterface:switchPort"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Descriptor -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- substitutes descriptorParam element -->

<element name="descriptorParam" substitutionGroup="descriptor:descriptorParam"/>


<!-- extends descriptor element -->

<complexType name="descriptorType">
  <complexContent>
    <extension base="descriptor:descriptorPrototype">
      <attributeGroup ref="layout:regionAttrs"/>
      <attributeGroup ref="timing:explicitDurAttrs"/>
      <attributeGroup ref="timing:freezeAttrs"/>
      <attributeGroup ref="keyNavigation:keyNavigationAttrs"/>
      <attributeGroup ref="transition:transAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="descriptor" type="profile:descriptorType" substitutionGroup="descriptor:descriptor"/>

<!-- extends descriptorBase element -->
<complexType name="descriptorBaseType">
```

```
  <complexContent>
    <extension base="descriptor:descriptorBasePrototype">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="profile:importBase"/>
        <element ref="profile:descriptor"/>
        <element ref="profile:descriptorSwitch"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<element name="descriptorBase" type="profile:descriptorBaseType" substitutionGroup="descriptor:descriptorBase"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Linking -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->

<!-- substitutes linkParam and bindParam elements -->
<element name="linkParam" substitutionGroup="linking:linkParam"/>
<element name="bindParam" substitutionGroup="linking:bindParam"/>

<!-- extends bind element and link element, as a consequence-->

<complexType name="bindType">
  <complexContent>
    <extension base="linking:bindPrototype">
      <attributeGroup ref="descriptor:descriptorAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="bind" type="profile:bindType" substitutionGroup="linking:bind"/>

<!-- extends link element -->
<complexType name="linkType">
  <complexContent>
    <extension base="linking:linkPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="link" type="profile:linkType" substitutionGroup="linking:link"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Connector -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends connectorBase element -->

  <complexType name="connectorBaseType">
      <complexContent>
          <extension base="connectorBase:connectorBasePrototype">
            <choice minOccurs="0" maxOccurs="unbounded">
               <element ref="profile:importBase"/>
                       <element ref="profile:causalConnector" />
            </choice>
          </extension>
      </complexContent>
  </complexType>

  <complexType name="simpleActionType">
    <complexContent>
      <extension base="connectorCausalExpression:simpleActionPrototype">
         <attributeGroup ref="animation:animationAttrs"/>
      </extension>
    </complexContent>
```



```
    </complexType>

  <element name="connectorBase" type="profile:connectorBaseType" substitutionGroup="connectorBase:connectorBase"/>

  <element name="causalConnector" substitutionGroup="causalConnectorFunctionality:causalConnector"/>

  <element name="connectorParam" substitutionGroup="causalConnectorFunctionality:connectorParam"/>

  <element name="simpleCondition" substitutionGroup="causalConnectorFunctionality:simpleCondition"/>

  <element name="compoundCondition" substitutionGroup="causalConnectorFunctionality:compoundCondition"/>

  <element name="simpleAction" type="profile:simpleActionType"
substitutionGroup="causalConnectorFunctionality:simpleAction"/>

  <element name="compoundAction" substitutionGroup="causalConnectorFunctionality:compoundAction"/>

  <element name="assessmentStatement" substitutionGroup="causalConnectorFunctionality:assessmentStatement"/>

  <element name="attributeAssessment" substitutionGroup="causalConnectorFunctionality:attributeAssessment"/>

  <element name="valueAssessment" substitutionGroup="causalConnectorFunctionality:valueAssessment"/>

  <element name="compoundStatement" substitutionGroup="causalConnectorFunctionality:compoundStatement"/>

  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- TestRule -->
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- extends rule element -->
  <complexType name="ruleType">
    <complexContent>
      <extension base="testRule:rulePrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="rule" type="profile:ruleType" substitutionGroup="testRule:rule"/>

  <!-- extends compositeRule element -->
  <complexType name="compositeRuleType">
    <complexContent>
      <extension base="testRule:compositeRulePrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="compositeRule" type="profile:compositeRuleType" substitutionGroup="testRule:compositeRule"/>

  <!-- extends ruleBase element -->
  <complexType name="ruleBaseType">
    <complexContent>
      <extension base="testRule:ruleBasePrototype">
        <choice minOccurs="1" maxOccurs="unbounded">
          <element ref="profile:importBase"/>
          <element ref="profile:rule"/>
          <element ref="profile:compositeRule"/>
        </choice>
      </extension>
    </complexContent>
  </complexType>

  <element name="ruleBase" type="profile:ruleBaseType" substitutionGroup="testRule:ruleBase"/>

   <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- TestRuleUse -->
```

```
<!-- ================================================================= -->
<!-- extends bindRule element -->
<complexType name="bindRuleType">
  <complexContent>
    <extension base="testRuleUse:bindRulePrototype">
    </extension>
  </complexContent>
</complexType>

<element name="bindRule" type="profile:bindRuleType" substitutionGroup="testRuleUse:bindRule"/>

<!-- ================================================================= -->
<!-- ContentControl -->
<!-- ================================================================= -->
<!-- extends switch element -->

<!-- switch interface element groups -->
<group name="switchInterfaceElementGroup">
  <choice>
    <element ref="profile:switchPort"/>
  </choice>
</group>

<!-- extends defaultComponent element -->
<complexType name="defaultComponentType">
  <complexContent>
    <extension base="contentControl:defaultComponentPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="defaultComponent" type="profile:defaultComponentType"
substitutionGroup="contentControl:defaultComponent"/>

<complexType name="switchType">
  <complexContent>
    <extension base="contentControl:switchPrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <group ref="profile:switchInterfaceElementGroup"/>
          <element ref="profile:bindRule"/>
          <element ref="profile:switch"/>
          <element ref="profile:media"/>
          <element ref="profile:context"/>
        </choice>
      <attributeGroup ref="entityReuse:entityReuseAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="switch" type="profile:switchType" substitutionGroup="contentControl:switch"/>

<!-- ================================================================= -->
<!-- DescriptorControl -->
<!-- ================================================================= -->
<!-- extends defaultDescriptor element -->
<complexType name="defaultDescriptorType">
  <complexContent>
    <extension base="descriptorControl:defaultDescriptorPrototype">
    </extension>
  </complexContent>
</complexType>
```



```
  <element name="defaultDescriptor" type="profile:defaultDescriptorType"
substitutionGroup="descriptorControl:defaultDescriptor"/>

  <!-- extends descriptorSwitch element -->

  <complexType name="descriptorSwitchType">
    <complexContent>
      <extension base="descriptorControl:descriptorSwitchPrototype">
          <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="profile:descriptor"/>
            <element ref="profile:bindRule"/>
          </choice>
      </extension>
    </complexContent>
  </complexType>

  <element name="descriptorSwitch" type="profile:descriptorSwitchType"
substitutionGroup="descriptorControl:descriptorSwitch"/>

  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- Timing -->
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->


  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- Import -->
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <complexType name="importBaseType">
    <complexContent>
      <extension base="import:importBasePrototype">
      </extension>
    </complexContent>
  </complexType>

  <complexType name="importNCLType">
    <complexContent>
      <extension base="import:importNCLPrototype">
      </extension>
    </complexContent>
  </complexType>

  <complexType name="importedDocumentBaseType">
    <complexContent>
      <extension base="import:importedDocumentBasePrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="importBase" type="profile:importBaseType" substitutionGroup="import:importBase"/>

  <element name="importNCL" type="profile:importNCLType" substitutionGroup="import:importNCL"/>
  <element name="importedDocumentBase" type="profile:importedDocumentBaseType"
substitutionGroup="import:importedDocumentBase"/>

  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- EntityReuse -->
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->

  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- ExtendedEntityReuse -->
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->

  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- KeyNavigation -->
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
```



```
<!-- ================================================================= -->
<!-- TransitionBase -->
<!-- ================================================================= -->
<!-- extends transitionBase element -->

<complexType name="transitionBaseType">
  <complexContent>
    <extension base="transitionBase:transitionBasePrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <element ref="profile:transition"/>
        <element ref="profile:importBase"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<element name="transitionBase" type="profile:transitionBaseType" substitutionGroup="transitionBase:transitionBase"/>

<!-- ================================================================= -->
<!-- Transition                                    -->
<!-- ================================================================= -->

<element name="transition" substitutionGroup="transition:transition"/>

<!-- ================================================================= -->
<!-- Metainformation -->
<!-- ================================================================= -->

<element name="meta" substitutionGroup="metainformation:meta"/>

<element name="metadata" substitutionGroup="metainformation:metadata"/>

</schema>
```

7.3.3 Esquema do perfil NCL 3.0 CausalConnector

El esquema del perfil NCL 3.0 CausalConnector debe ser acorde con la ABNT NBR 15606-2:2007, Subsección 7.3.3.

7.3.4 Atributos y elementos del perfil NCL 3.0 DTV básico

Los elementos y sus atributos, utilizados en el perfil NCL 3.0 DTV Básico, se presentan en las Tablas 6 a 22. Se destaca que los atributos y contenidos (elementos hijos) de elementos se pueden definir en el módulo en sí o en el perfil NCL DTV Básico que agrupa los módulos. Los atributos obligatorios están subrayados. En las Tablas 6 a 22, se emplean los siguientes símbolos: (?) opcional (cero o una ocurrencia), (|) o (*) cero o más ocurrencias, (+) una o más ocurrencias.

Tabla 6 — Elementos y atributos del módulo Structure extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
ncl	*id, title, xmlns*	(head?, body?)
head		(importedDocumentBase? ruleBase?, regionBase*, descriptorBase?, connectorBase?),
body	*Id*	(port| property| media|context|switch|link)*

Tabla 7 — Elementos y atributos del módulo *Layout* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
regionBase	*id, device, region*	(importBase|region)+
Region	*id, title, left, right, top, bottom, height, width, zIndex*	(region)*

Tabla 8 — Elementos y atributos del Módulo *Media* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
Media	*id, src, refer, instance, type, descriptor*	(area|property)*

Tabla 9 — Elementos y atributos del módulo *Context* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
context	*id, refer*	(port|property|media|context|link|switch)*

Tabla 10 — Elementos y atributos del módulo *MediaContentAnchor* extendido que se utilizan en el perfil DTV Básico

Elementos	Atributos	Contenido
area	*id, coords, begin, end, beginText, beginPosition endText, endPosition, first, last, label, clip*	vacío

Tabla 11 — Elementos y atributos del módulo *CompositeNodeInterface* extendido que se utilizan en el perfil DTV Básico

Elementos	Atributos	Contenido
port	*id, component, interface*	vacío

Tabla 12 — Elementos y atributos del módulo *PropertyAnchor* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
property	*name, value*	vacío

Tabla 13 — Elementos y atributos del módulo *SwitchInterface* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
switchPort	*Id*	mapping+
mapping	*component, interfaz*	vacío

Tabla 14 — Elementos y atributos del módulo *Descriptor* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
descriptor	*id, player, explicitDur, region, freeze, moveLeft, moveRight, moveUp; moveDown, focusIndex, focusBorderColor; focusBorderWidth; focusBorderTransparency, focusSrc,focusSelSrc, selBorderColor*	(descriptorParam)*
descriptorParam	*name, value*	vacío
descriptorBase	*Id*	(importBase | descriptor | descriptorSwitch)+

Tabla 15 — Elementos y atributos del módulo *Linking* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
bind	*role, component, interface, descriptor*	(bindParam)*
bindParam	*name, value*	vacío
linkParam	*name, value*	vacío
link	*id, xconnector*	(linkParam*, bind+)

Tabla 16 — Elementos y atributos del módulo *CausalConnectorFunctionality* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
causalConnector	*Id*	(connectorParam*, (simpleCondition | compoundCondition), (simpleAction | compoundAction))
connectorParam	*name, type*	vacío
simpleCondition	*role, delay, eventType, key, transition, min, max, qualifier*	vacío
compoundCondition	*operator, delay*	((simpleCondition | compoundCondition)+, (assessmentStatement | compoundStatement)*)
simpleAction	*role, delay, eventType, actionType, value, min, max, qualifier, repeat, repeatDelay*	vacío
compoundAction	*operator, delay*	(simpleAction | compoundAction)+
assessmentStatement	*comparator*	(attributeAssessment, (attributeAssessment | valueAssessment))
attributeAssessment	*role, eventType, key, attributeType, offset*	vacío
valueAssessment	*Value*	vacío
compoundStatement	*operator, isNegated*	(assessmentStatement | compoundStatement)+

Tabla 17 — Elementos y atributos del módulo *ConnectorBase* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
ConnectorBase	*Id*	(importBase|causalConnector)*

Tabla 18 — Elementos y atributos del módulo *TestRule* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
ruleBase	*Id*	(importBase|rule|compositeRule)+
rule	*id, var, comparator, value*	vacío
compositeRule	*id, operator*	(rule | compositeRule)+

Tabla 19 — Elementos y atributos del módulo *TestRuleUse* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
bindRule	*constituent, rule*	vacío

Tabla 20 — Elementos y atributos del módulo *ContentControl* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
switch	*id, refer*	(defaultComponent?,(switchPort| bindRule|media| context | switch)*)
defaultComponent	*Component*	vacío

Tabla 21 — Elementos y atributos del módulo *DescriptorControl* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
descriptorSwitch	*Id*	(defaultDescriptor?, (bindRule | descriptor)*)
defaultDescriptor	*Descriptor*	vacío

Tabla 22 — Elementos y atributos del módulo *Import* extendido utilizados en el perfil DTV Básico

Elementos	Atributos	Contenido
importBase	*alias, documentURI, region*	vacío
importedDocumentBase	*Id*	(importNCL)+
importNCL	*alias, documentURI,*	vacío

7.3.5 Esquema del perfil NCL 3.0 DTV Básico

NCL30BDTV.xsd

```
<!--
XML Schema for the NCL Language

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/profiles/NCL30BDTV.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006
-->
```

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:compositeInterface="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"
  xmlns:causalConnectorFunctionality="http://www.ncl.org.br/NCL3.0/CausalConnectorFunctionality"
  xmlns:connectorBase="http://www.ncl.org.br/NCL3.0/ConnectorBase"
  xmlns:contentControl="http://www.ncl.org.br/NCL3.0/ContentControl"
  xmlns:context="http://www.ncl.org.br/NCL3.0/Context"
  xmlns:descriptor="http://www.ncl.org.br/NCL3.0/Descriptor"
  xmlns:entityReuse="http://www.ncl.org.br/NCL3.0/EntityReuse"
  xmlns:extendedEntityReuse="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"
  xmlns:descriptorControl="http://www.ncl.org.br/NCL3.0/DescriptorControl"
  xmlns:import="http://www.ncl.org.br/NCL3.0/Import"
  xmlns:keyNavigation="http://www.ncl.org.br/NCL3.0/KeyNavigation"
  xmlns:layout="http://www.ncl.org.br/NCL3.0/Layout"
  xmlns:linking="http://www.ncl.org.br/NCL3.0/Linking"
  xmlns:media="http://www.ncl.org.br/NCL3.0/Media"
  xmlns:mediaAnchor="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"
  xmlns:propertyAnchor="http://www.ncl.org.br/NCL3.0/PropertyAnchor"
  xmlns:structure="http://www.ncl.org.br/NCL3.0/Structure"
  xmlns:switchInterface="http://www.ncl.org.br/NCL3.0/SwitchInterface"
  xmlns:testRule="http://www.ncl.org.br/NCL3.0/TestRule"
  xmlns:testRuleUse="http://www.ncl.org.br/NCL3.0/TestRuleUse"
  xmlns:timing="http://www.ncl.org.br/NCL3.0/Timing"
  xmlns:profile="http://www.ncl.org.br/NCL3.0/BDTVProfile"
  targetNamespace="http://www.ncl.org.br/NCL3.0/BDTVProfile"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- import the definitions in the modules namespaces -->
  <import namespace="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30CompositeNodeInterface.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/CausalConnectorFunctionality"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30CausalConnectorFunctionality.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorBase"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorBase.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/ContentControl"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ContentControl.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/Context"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Context.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/Descriptor"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Descriptor.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/DescriptorControl"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30DescriptorControl.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/EntityReuse"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30EntityReuse.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ExtendedEntityReuse.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/Import"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Import.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/KeyNavigation"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30KeyNavigation.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/Layout"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Layout.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/Linking"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Linking.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/Media"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Media.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30MediaContentAnchor.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/PropertyAnchor"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30PropertyAnchor.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/Structure"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Structure.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/SwitchInterface"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30SwitchInterface.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/TestRule"
```

```
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30TestRule.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/TestRuleUse"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30TestRuleUse.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Timing"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Timing.xsd"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Structure -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends ncl element -->

<element name="ncl" substitutionGroup="structure:ncl"/>

<!-- extends head element -->

<complexType name="headType">
  <complexContent>
    <extension base="structure:headPrototype">
      <sequence>
        <element ref="profile:importedDocumentBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:ruleBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:regionBase" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="profile:descriptorBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:connectorBase" minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="head" type="profile:headType" substitutionGroup="structure:head"/>

<!-- extends body element -->

<complexType name="bodyType">
  <complexContent>
    <extension base="structure:bodyPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:contextInterfaceElementGroup"/>
        <element ref="profile:media"/>
        <element ref="profile:context"/>
        <element ref="profile:switch"/>
        <element ref="profile:link"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<element name="body" type="profile:bodyType" substitutionGroup="structure:body"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Layout -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends regionBase element -->

<complexType name="regionBaseType">
  <complexContent>
    <extension base="layout:regionBasePrototype">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="profile:region"/>
        <element ref="profile:importBase"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

```
<complexType name="regionType">
  <complexContent>
    <extension base="layout:regionPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="regionBase" type="profile:regionBaseType" substitutionGroup="layout:regionBase"/>
<element name="region" type="profile:regionType" substitutionGroup="layout:region"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Media -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends Media elements -->

<!-- media interface element groups -->
<group name="mediaInterfaceElementGroup">
  <choice>
    <element ref="profile:area"/>
    <element ref="profile:property"/>
  </choice>
</group>

<complexType name="mediaType">
  <complexContent>
    <extension base="media:mediaPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:mediaInterfaceElementGroup"/>
      </choice>
      <attributeGroup ref="descriptor:descriptorAttrs"/>
      <attributeGroup ref="entityReuse:entityReuseAttrs"/>
      <attributeGroup ref="extendedEntityReuse:extendedEntityReuseAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="media" type="profile:mediaType" substitutionGroup="media:media"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Context -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends context element -->

<!-- composite node interface element groups -->
<group name="contextInterfaceElementGroup">
  <choice>
    <element ref="profile:port"/>
    <element ref="profile:property"/>
  </choice>
</group>

<complexType name="contextType">
  <complexContent>
    <extension base="context:contextPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:contextInterfaceElementGroup"/>
        <element ref="profile:media"/>
        <element ref="profile:context"/>
        <element ref="profile:link"/>
        <element ref="profile:switch"/>
      </choice>
      <attributeGroup ref="entityReuse:entityReuseAttrs"/>
    </extension>
  </complexContent>
</complexType>
```

```
<element name="context" type="profile:contextType" substitutionGroup="context:context"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- MediaContentAnchor -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends area element -->

<complexType name="componentAnchorType">
  <complexContent>
    <extension base="mediaAnchor:componentAnchorPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="area" type="profile:componentAnchorType" substitutionGroup="mediaAnchor:area"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- CompositeNodeInterface -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends port element -->

<complexType name="compositeNodePortType">
  <complexContent>
    <extension base="compositeInterface:compositeNodePortPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="port" type="profile:compositeNodePortType" substitutionGroup="compositeInterface:port"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- PropertyAnchor -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends property element -->

<complexType name="propertyAnchorType">
  <complexContent>
    <extension base="propertyAnchor:propertyAnchorPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="property" type="profile:propertyAnchorType" substitutionGroup="propertyAnchor:property"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- SwitchInterface -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends switchPort element -->

<complexType name="switchPortType">
  <complexContent>
    <extension base="switchInterface:switchPortPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="mapping" substitutionGroup="switchInterface:mapping"/>
<element name="switchPort" type="profile:switchPortType" substitutionGroup="switchInterface:switchPort"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Descriptor -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->

<!-- substitutes descriptorParam element -->
```



```
<element name="descriptorParam" substitutionGroup="descriptor:descriptorParam"/>


<!-- extends descriptor element -->

<complexType name="descriptorType">
  <complexContent>
    <extension base="descriptor:descriptorPrototype">
      <attributeGroup ref="layout:regionAttrs"/>
      <attributeGroup ref="timing:explicitDurAttrs"/>
      <attributeGroup ref="timing:freezeAttrs"/>
      <attributeGroup ref="keyNavigation:keyNavigationAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="descriptor" type="profile:descriptorType" substitutionGroup="descriptor:descriptor"/>

<!-- extends descriptorBase element -->
<complexType name="descriptorBaseType">
  <complexContent>
    <extension base="descriptor:descriptorBasePrototype">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="profile:importBase"/>
        <element ref="profile:descriptor"/>
        <element ref="profile:descriptorSwitch"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<element name="descriptorBase" type="profile:descriptorBaseType" substitutionGroup="descriptor:descriptorBase"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Linking -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- substitutes linkParam and bindParam elements -->
<element name="linkParam" substitutionGroup="linking:linkParam"/>
<element name="bindParam" substitutionGroup="linking:bindParam"/>

<!-- extends bind element and link element, as a consequence-->

<complexType name="bindType">
  <complexContent>
    <extension base="linking:bindPrototype">
      <attributeGroup ref="descriptor:descriptorAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="bind" type="profile:bindType" substitutionGroup="linking:bind"/>

<!-- extends link element -->
<complexType name="linkType">
  <complexContent>
    <extension base="linking:linkPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="link" type="profile:linkType" substitutionGroup="linking:link"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- Connector -->
```

```
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends connectorBase element -->

  <complexType name="connectorBaseType">
    <complexContent>
      <extension base="connectorBase:connectorBasePrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <element ref="profile:importBase"/>
              <element ref="profile:causalConnector" />
        </choice>
      </extension>
    </complexContent>
  </complexType>

<element name="connectorBase" type="profile:connectorBaseType" substitutionGroup="connectorBase:connectorBase"/>

<element name="causalConnector" substitutionGroup="causalConnectorFunctionality:causalConnector"/>

<element name="connectorParam" substitutionGroup="causalConnectorFunctionality:connectorParam"/>

<element name="simpleCondition" substitutionGroup="causalConnectorFunctionality:simpleCondition"/>

<element name="compoundCondition" substitutionGroup="causalConnectorFunctionality:compoundCondition"/>

<element name="simpleAction" substitutionGroup="causalConnectorFunctionality:simpleAction"/>

<element name="compoundAction" substitutionGroup="causalConnectorFunctionality:compoundAction"/>

<element name="assessmentStatement" substitutionGroup="causalConnectorFunctionality:assessmentStatement"/>

<element name="attributeAssessment" substitutionGroup="causalConnectorFunctionality:attributeAssessment"/>

<element name="valueAssessment" substitutionGroup="causalConnectorFunctionality:valueAssessment"/>

<element name="compoundStatement" substitutionGroup="causalConnectorFunctionality:compoundStatement"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- TestRule -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends rule element -->
<complexType name="ruleType">
  <complexContent>
    <extension base="testRule:rulePrototype">
    </extension>
  </complexContent>
</complexType>

<element name="rule" type="profile:ruleType" substitutionGroup="testRule:rule"/>

<!-- extends compositeRule element -->
<complexType name="compositeRuleType">
  <complexContent>
    <extension base="testRule:compositeRulePrototype">
    </extension>
  </complexContent>
</complexType>

<element name="compositeRule" type="profile:compositeRuleType" substitutionGroup="testRule:compositeRule"/>

<!-- extends ruleBase element -->
<complexType name="ruleBaseType">
  <complexContent>
    <extension base="testRule:ruleBasePrototype">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="profile:importBase"/>
```

```
        <element ref="profile:rule"/>
        <element ref="profile:compositeRule"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<element name="ruleBase" type="profile:ruleBaseType" substitutionGroup="testRule:ruleBase"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- TestRuleUse -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends bindRule element -->
<complexType name="bindRuleType">
  <complexContent>
    <extension base="testRuleUse:bindRulePrototype">
    </extension>
  </complexContent>
</complexType>

<element name="bindRule" type="profile:bindRuleType" substitutionGroup="testRuleUse:bindRule"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- ContentControl -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- extends switch element -->

<!-- switch interface element groups -->
<group name="switchInterfaceElementGroup">
  <choice>
    <element ref="profile:switchPort"/>
  </choice>
</group>

<!-- extends defaultComponent element -->
<complexType name="defaultComponentType">
  <complexContent>
    <extension base="contentControl:defaultComponentPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="defaultComponent" type="profile:defaultComponentType"
substitutionGroup="contentControl:defaultComponent"/>

<complexType name="switchType">
  <complexContent>
    <extension base="contentControl:switchPrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <group ref="profile:switchInterfaceElementGroup"/>
          <element ref="profile:bindRule"/>
          <element ref="profile:switch"/>
          <element ref="profile:media"/>
          <element ref="profile:context"/>
        </choice>
      <attributeGroup ref="entityReuse:entityReuseAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="switch" type="profile:switchType" substitutionGroup="contentControl:switch"/>

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- DescriptorControl -->
```



```
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- extends defaultDescriptor element -->
  <complexType name="defaultDescriptorType">
    <complexContent>
      <extension base="descriptorControl:defaultDescriptorPrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="defaultDescriptor" type="profile:defaultDescriptorType"
substitutionGroup="descriptorControl:defaultDescriptor"/>

  <!-- extends descriptorSwitch element -->

  <complexType name="descriptorSwitchType">
    <complexContent>
      <extension base="descriptorControl:descriptorSwitchPrototype">
          <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="profile:descriptor"/>
            <element ref="profile:bindRule"/>
          </choice>
      </extension>
    </complexContent>
  </complexType>

  <element name="descriptorSwitch" type="profile:descriptorSwitchType"
substitutionGroup="descriptorControl:descriptorSwitch"/>

  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- Timing -->
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->


  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- Import -->
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <complexType name="importBaseType">
    <complexContent>
      <extension base="import:importBasePrototype">
      </extension>
    </complexContent>
  </complexType>

  <complexType name="importNCLType">
    <complexContent>
      <extension base="import:importNCLPrototype">
      </extension>
    </complexContent>
  </complexType>

  <complexType name="importedDocumentBaseType">
    <complexContent>
      <extension base="import:importedDocumentBasePrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="importBase" type="profile:importBaseType" substitutionGroup="import:importBase"/>

  <element name="importNCL" type="profile:importNCLType" substitutionGroup="import:importNCL"/>
  <element name="importedDocumentBase" type="profile:importedDocumentBaseType"
substitutionGroup="import:importedDocumentBase"/>

  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- EntityReuse -->
```

```
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- ExtendedEntityReuse -->
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->

<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
<!-- KeyNavigation -->
<!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->

</schema>
```

8 Objetos de media en presentaciones NCL

8.1 Implementación modular de Ginga-NCL

La presentación de un documento NCL requiere el control de la sincronización de varios objetos de media (especificados a través del elemento <media>). Para cada objeto de media, un player (exhibidor de media) puede ser cargado para el control del objeto y de sus eventos NCL. Un exhibidor de media (o su adaptador) debe ser capaz obligatoriamente de recibir los comandos de presentación, controlar las máquinas de estado de los eventos del objeto de media controlado y responder a las exigencias del formateador.

Para favorecer la incorporación de *players* de terceras partes dentro de la implementación de la arquitectura Ginga, se recomienda un proyecto modular del Ginga-NCL, para separar los *players* de la máquina de presentación (formateador NCL).

La Figura 2 sugiere una organización modular para la implementación Ginga-NCL. Los exhibidores son módulos *plug-in* de la máquina de presentación. Por ser interesante utilizar los *players* existentes, que puedan tener interfaces dueñas que no sean compatibles con aquellas exigidas por la máquina de presentación, será necesario desarrollar módulos para realizar las adaptaciones necesarias. En ese caso, el exhibidor será constituido por un adaptador además del exhibidor no conforme en sí.

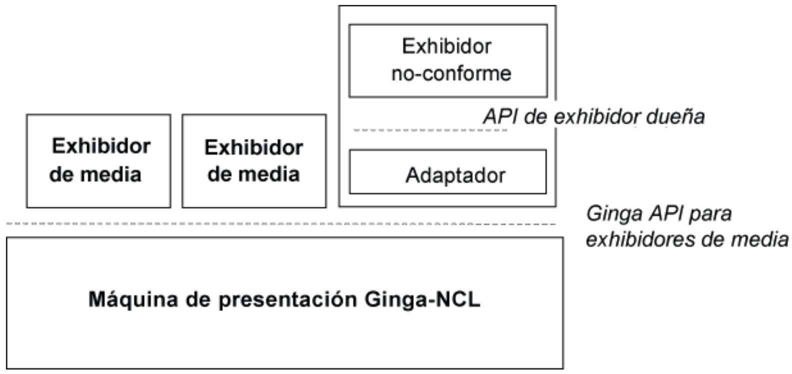

Figura 2 — API para integrar los *players* a la implementación de la máquina de presentación NCL

NOTA Como la arquitectura e implementación Ginga-NCL es una elección de cada receptor, esta Norma no tiene la intención de estandarizar la sintaxis de la API de la máquina de presentación y sí solamente definir el comportamiento esperado del exhibidor de media cuando se están controlando objetos que forman parte de un documento NCL.

8.2 Comportamiento esperado de los exhibidores básicos de media

El comportamiento esperado de los exhibidores de media debe ser acorde con la ABNT NBR 15606-2:2007, Subsección 8.2.

8.3 Comportamiento esperado de los exhibidores de media después de instrucciones aplicadas a los objetos de composición

El comportamiento esperado de los exhibidores de media después de las instrucciones aplicadas a los objetos de composición debe ser acorde con la ABNT NBR 15606-2:2007, Subsección 8.3.

8.4 Relación entre la máquina de estado de eventos de presentación de un nudo y la máquina de estado del evento de presentación de su nudo de composición padre

La relación entre la máquina de estado de eventos de presentación de un nudo y la máquina de estado del evento de presentación de su nudo de composición padre deben ser acordes con la ABNT NBR 15606-2:2007, Subsección 8.4.

8.5 Comportamiento esperado de los exhibidores de media imperativa en aplicativos NCL

Los objetos imperativos en código Lua se pueden insertar en documentos NCL, trayendo capacidades computacionales adicionales a los documentos declarativos. La forma de agregar objetos imperativos en documentos NCL es definir un elemento <media> cuyo contenido (localizado por el atributo src) es el código imperativo a ser ejecutado. Los perfiles EDTV y BDTV de NCL 3.0 deben obligatoriamente dar soporte a elementos <media> del tipo “application/x-Ginga-NCLua” (extensión de archivo .lua).

Los autores pueden definir eslabones NCL para iniciar, parar, pausar, retomar o abortar la ejecución de un código Lua. Un exhibidor imperativo (máquina de ejecución del lenguaje) debe proveer obligatoriamente la interfaz del ambiente de ejecución imperativo con el formateador NCL.

En forma similar a lo realizado por los exhibidores de contenidos de media convencionales, los exhibidores imperativos deben obligatoriamente controlar las máquinas de estado de los eventos asociados con el nudo imperativo Lua. Como ejemplo, si un código termina su ejecución, el exhibidor debe obligatoriamente generar la transición stops en la máquina de estado de presentación del evento correspondiente a la ejecución imperativo.

NCL permite que la ejecución del código imperativo sea sincronizada con otros objetos NCL (imperativos o no). Un elemento <media> del tipo “application/x-ginga-NCLua” también puede definir anclas (a través de elementos <area>) y propiedades (a través de elementos <property>).

El código Lua puede ser asociado a elementos <area> (a través del atributo *label*). Si eslabones externos empiezan, paran, pausan, reinician o abortan la presentación del ancla, *callbacks* en el código imperativo deben ser obligatoriamente disparados. La forma como esos *callbacks* se definen es responsabilidad de cada código imperativo asociado con el objeto NCLua.

Por otro lado, el código Lua también puede comandar el inicio, parada, pausa, reinicio o aborto de esas anclas, a través de una API ofrecida por el lenguaje. Esas transiciones se pueden utilizar como condiciones de eslabones NCL para disparar acciones en otros objetos NCL del mismo documento. Así, se puede establecer una sincronización de dos vías entre el código Lua y el resto del documento NCL.

La otra forma que un código Lua se puede sincronizar con otros objetos NCL es a través de elementos <property>. Un elemento <property> definido como hijo de un elemento <media> del tipo “application/x-ginga-NCLua” puede ser mapeado para un trecho del código o para un atributo del código Lua. Cuando es mapeado para un trecho de código, una acción de eslabón “set” aplicada a la propiedad debe obligatoriamente causar la ejecución del código con los valores atribuidos interpretados como parámetros de entrada. El atributo *name* del elemento <property> se debe usar obligatoriamente para identificar el trecho del código Lua. Cuando el elemento <property> es mapeado para un atributo del código imperativo, la acción “set” debe obligatoriamente atribuir el valor al atributo. Como de costumbre, la máquina de estado de evento asociada a la propiedad debe ser controlada por el exhibidor Lua.

Un elemento <property> definido como hijo de un elemento <media> del tipo “application/x-ginga-NCLua”, también puede estar asociado a un *assessment role* de un eslabón NCL. En ese caso, el formateador NCL debe obligatoriamente cuestionar el valor de la propiedad para evaluar la expresión del eslabón. Si el elemento <property> es mapeado para un atributo de código, su valor debe ser retornado obligatoriamente por el exhibidor imperativo al formateador NCL. Si el elemento <property> es mapeado para una función de código Lua, debe ser llamado obligatoriamente y el valor de su resultado debe ser retornado obligatoriamente por el exhibidor imperativo al formateador NCL.

La instrucción *start* emitida por un formatador debe obligatoriamente informar al exhibidor imperativo los siguien (definidos por los elementos <area> y <property>, hijos del elemento <media> que define el objeto imperativo) que necesitan ser monitoreados por el exhibidor Lua, el identificador (*id*) del elemento <area> asociado al código Lua a ser ejecutado, y un tiempo de retardo, opcional. A partir del atributo *src*, el exhibidor Lua debe intentar localizar el código Lua e iniciar su ejecución. Si el contenido no se puede localizar, el exhibidor Lua debe obligatoriamente cancelar la operación de inicialización sin realizar ninguna acción.

Se aconseja que la lista de eventos a ser monitoreados por un exhibidor de media también sea computada por el formateador, teniendo en cuenta la especificación del documento NCL. El formateador debe obligatoriamente chequear todos los eslabones de los cuales participa el objeto de media NCLua y el descriptor resultante. Al computar los eventos a ser monitoreados, el formateador debe considerar obligatoriamente la perspectiva del objeto de media NCLua, es decir, el camino de los varios elementos <body> y <context> para alcanzar en profundidad el elemento <media> correspondiente. Conviene que solamente eslabones contenidos en esos elementos <body> y <context> sean considerados en la computación de los eventos monitoreados.

Como con todos los tipos de elementos <media>, el tiempo de retardo es un parámetro opcional, y su valor default es “cero”. Si es mayor que cero, ese parámetro contiene un tiempo a ser esperado por el exhibidor imperativo antes de iniciar la ejecución.

A diferencia de los procedimientos realizados para otros tipos de elementos <media>, si un exhibidor Lua recibe una instrucción *start* para un evento asociado a un elemento <area> y ese evento esté en estado *sleeping*, el mismo debe dar inicio a la ejecución del código Lua asociado al elemento, incluso si otra parte del código imperativo del objeto de media está siendo ejecutado (pausado o no). Sin embargo, si el evento asociado al elemento <area> meta está en el estado *occurring* o *paused*, la instrucción *start* debe ser ignorada por el exhibidor Lua que seguirá controlando la ejecución anteriormente iniciada. Como consecuencia de ello, a diferencia de lo que ocurre para los demás elementos <media>, una acción <simpleAction> con el atributo *actionType* igual a “stop”, “pause”, “resume” o “abort” debe conectarse, a través de un eslabón, a una interfaz del nudo Lua, que no debe ser ignorada cuando se aplica la acción.

La instrucción start emitida por un formateador para un evento asociado a un elemento <property> puede ser aplicada a un objeto imperativo, independiente del hecho de que esté siendo ejecutado o no (en ese último caso, aunque el objeto no esté siendo ejecutado, su exhibidor de procedencia debe obligatoriamente haber sido instanciado). La instrucción start deberá identificar el objeto de procedencia en ejecución, un evento de atribución monitoreado y un valor que será pasado al código de procedencia asociado al evento.

Para cada evento de atribución monitoreado, si el exhibidor Lua cambia por sí mismo el valor del atributo, el mismo debe proceder obligatoriamente como si hubiese recibido una instrucción externa de *start*.”

El lenguaje Lua también debe ofrecer obligatoriamente una API que permita que los códigos imperativos cuestionen cualquier valor de propiedad predefinida o dinámica del nudo *settings* NCL (elemento <media> del tipo “application/x-ginga-settings”). Sin embargo, se debe observar que no se permite atribuir valores a esas propiedades directamente. Las propiedades de los nudos del tipo application/x-ginga-settings sólo pueden ser modificadas a través del uso de eslabones NCL.

Las API que suministren un conjunto de métodos para dar soporte a los comandos de edición del NCL y a los comandos del Administrador de la Base Privada también deben obligatoriamente ser ofrecidas, tal como se presenta en la Sección 10.

9 Transmisión de contenido y eventos de flujo NCL

La transmisión de contenido y eventos de flujo NCL debe ser acorde con la ABNT NBR 15606-2:2007, Sección 9.

10 Objetos imperativos Lua en presentaciones NCL

10.1 Lenguaje Lua - Funciones retiradas de la biblioteca Lua

El lenguaje de *script* adoptado por Ginga-NCL es Lua (elementos <media> del tipo application/x-ginga-NCLua o application/x-ncl-NCLua). La definición completa de Lua es presentada en ABNT NBR 15606-2:2007, Anexo B.

Las funciones a continuación son dependientes de plataforma y fueron retiradas:

1) en el módulo *package: loadlib*;

2) en el módulo *os*: *clock, execute, exit, getenv, remove, rename, tmpname* y *setlocale*;

3) en el módulo *debug:* todas las funciones.

10.2 Modelo de ejecución

El ciclo de vida de un objeto NCLua es controlado por el formateador NCL. El formateador es responsable por iniciar la ejecución de un objeto NCLua y por mediar la comunicación entre ese objeto y otros objetos en un documento NCL, como definido en 8.5.

Como con todos los exhibidores de objetos de media, un exhibidor Lua, una vez instado, debe ejecutar los procedimientos de iniciación del objeto NCL que controlará. Sin embargo, diferentemente de los demás exhibidores de media, el código de iniciación también debe ser especificado por el autor del objeto NCLua. Los procedimientos de iniciación se ejecutan una sola vez, para cada instancia, y crean funciones y objetos que se pueden usar durante la ejecución del objeto NCLua y, en particular, registra uno o más tratadores de eventos para la comunicación con el formateador NCL.

Después de la iniciación, la ejecución del objeto NCLua se torna orientada a evento, en ambas direcciones. Es decir, cualquier acción comandada por el formateador NCL es dirigida a los tratadores de evento registrados, y cualquier notificación de cambio de estado de eventos NCL es enviada como un evento al formateador NCL (como por ejemplo, el fin de la ejecución del código imperativo). El exhibidor Lua estará entonces listo para ejecutar cualquier intrusión de *start* o *set* (ver 8.5).

10.3 Módulos adicionales

10.3.1 Módulos obligatorios

Además de la biblioteca estándar de Lua, los siguientes módulos deben ser obligatoriamente ofrecidos y cargados automáticamente:

1) módulo *canvas*: ofrece una API para dibujar primitivas gráficas y manejar imágenes;

2) 2) módulo *event*: permite que aplicaciones NCLua se comuniquen con el *middleware* a través de eventos (eventos NCL, pointer y de teclas);

3) módulo *settings*: exporta una tabla con variables definidas por el autor del documento NCL y variables de ambiente reservadas en un nudo "application/x-ginga-settings";

4) módulo *persistent*: exporta una tabla con variables persistentes, que están disponibles para manipulación solamente por objetos imperativos.

La definición de cada función en los módulos mencionados respeta la siguiente nomenclatura:

funcname (parnamel: partypel [; optnamel: opttypel]) -> retname: rettype

10.3.2 Módulo *canvas*

10.3.2.1 Objeto *canvas*

Cuando se inicia un objeto de media NCLua, la región del elemento <media> correspondiente (del tipo application/x-ginga-NCLua) queda disponible como la variable global *canvas* para el *script* Lua. Si el elemento de <media> no tiene ninguna región especificada (propiedades *left, right, top and bottom),* entonces el valor para *canvas* debe ser establecido como “nil”.

Ejemplo: para una región definida en un documento NCL como:

```
<Region iD="luaRegion" width="300" height="100" Top="200" left="20"/>
```

La variable '`canvas`' en un objeto de media asociado a la región “luaRegion” se conecta a un objeto canvas de tamaño 300x100, asociada con la región (20,200).

Un canvas ofrece una API gráfica para ser usada por aplicaciones NCLua. A través de ella es posible dibujar líneas, rectángulos, letras, imágenes etc.

Un canvas guarda en su estado atributos bajo los cuales operan las primitivas de dibujo, por ejemplo, si su atributo de color es azul, una llamada a `canvas:drawLine()` dibuja una línea azul em el canvas.

Las coordenadas pasadas son siempre relativas al punto más a la izquierda y a la parte superior del canvas (0,0).

10.3.2.2 Constructores

A través de cualquier canvas es posible crear nuevos canvas y combinarlos a través de operaciones de composición.

---

canvas:new (image_path: String) -> canvas: object

Argumentos

image_path	Camino de la imagen

Valor de retorno

canvas	Canvas representa la imagen

Descripción

Retorna un nuevo canvas cuyo contenido es la imagen recibida como parámetro.

El nuevo canvas debe obligatoriamente mantener los aspectos de transparencia de la imagen original.

---

canvas:new (width, height: Number) -> canvas: object

Argumentos

width	Anchura del canvas
height	Altura del canvas

Valores de retorno

canvas	Nuevo canvas

Descripción

Retorna un nuevo canvas con el tamaño recibido.

Inicialmente todos los pixels deben ser transparentes, obligatoriamente.

10.3.2.3 Atributos

Todos los métodos de atributos poseen el código attr y sirven tanto para leer como para alterar un atributo (con algunas excepciones).

Cuando el método es llamado sin parámetros de entrada, el valor corriente del atributo es retornado, en contrapartida, cuando es llamado con parámetros, ésos deben ser los nuevos valores del atributo.

---

canvas:attrSize () -> width, height: number

Argumentos - sin argumentos

Valores de retorno

width	Anchura del canvas
height	Altura del canvas

Descripción

Retorna las dimensiones del canvas.

Es importante observar que no se permite alterar las dimensiones de un canvas.

---

canvas:attrColor (R, G, B, A: Number)

Argumentos

R	Componente rojo del color
G	Componente verde del color
B	Componente azul del color
A	Componente Alpha del color

Descripción

Altera el color del canvas.

Los colores se pasan en RGBA, donde A varia de 0 (totalmente transparente) a 255 (totalmente opaco).

Las primitivas (ver 10.3.3.4) se dibujan con el color de ese atributo del canvas.

El valor inicial es ‘0,0,0,255’ (negro).

---

canvas:attrColor (clr_name: string)

Argumentos

clr_name	Nombre del color

Altera el color del canvas.

Los colores pasan a través de una string correspondiente a uno de los 16 colores NCL predefinidos:

'white', 'aqua', 'lime', 'yellow', 'red', 'fuchsia', 'purple', 'maroon',

'blue', 'navy', 'teal', 'green', 'olive', 'silver', 'gray', 'black'

Para valores en string, el Alpha es opaco (correspondiendo a “A = 255”).

Las primitivas (ver 10.3.3.4) se dibujan con el color de ese atributo del canvas.

El valor inicial es 'black'.

---

canvas:attrColor () -> R, G, B, A: number

Valores de retorno

R	Componente rojo del color
G	Componente verde del color
B	Componente azul del color
A	Componente Alpha del color

Descripción

Retorna el color del canvas.

---

canvas:attrFont (face: string; size: number; style: string)

Argumentos

face	Nombre de la fuente
size	Tamaño de la fuente
style	Estilo de la fuente

Descripción

Altera la fuente del canvas.

Las siguientes fuentes deben estar disponibles obligatoriamente: 'Tiresias', 'Verdana'.

El tamaño es en pixels y representa la altura máxima de una línea escrita con la fuente elegida.

Los estilos posibles son: 'bold', 'italic' o 'bold-italic'. El valor nil asume que ninguno de los estilos será usado.

Cualquier valor pasado no soportado debe obligatoriamente generar un error.

El valor inicial de la fuente es indeterminado.

---

canvas:attrFont () -> face: string; size: number; style: string

Valores de retorno

face	Nombre de la fuente
size	Tamaño de la fuente
style	Estilo de la fuente

Descripción

Retorna la fuente del canvas.

---

canvas:attrClip (x, y, width, height: number)

Argumentos

x	Coordenada del área de *clipping*
y	Coordenada del área de *clipping*
width	Anchura del área de *clipping*
height	Altura del área de *clipping*

Descripción

Altera el área de *clipping* del canvas.

Las primitivas de dibujo (ver 10.3.3.4) y el método `canvas:compose()` solo operan dentro de esta región de clipping.

El valor inicial es el canvas entero.

canvas:attrCrop (x, y, w, h: number)

Argumentos

x	Coordenada de la región de crop
y	Coordenada de la región de crop
w	Anchura de la región de crop
h	Altura de la región de crop

Descripción

Altera el área de crop del canvas.

Solamente la región configurada pasa a ser usada en operaciones de composición.

El valor inicial es el canvas entero.

El canvas principal no puede tener su valor alterado pues es controlado por el formateador NCL.

canvas:attrCrop () -> x, y, w, h: number

Valores de retorno

x	Coordenada de la región de crop
y	Coordenada de la región de crop
w	Anchura de la región de crop
h	Altura de la región de crop

Descripción

Retorna la región de *crop* del canvas.

canvas:attrFlip (horiz, vert: boolean)

Argumentos

horiz	Si el espejado es horizontal
vert	Si el espejado es vertical

Descripción

Configura el espejado del canvas usado en funciones de composición.

El canvas principal no puede ser alterado pues es controlado por el formateador NCL

canvas:attrFlip () -> horiz, vert: boolean

Valores de retorno

horiz	Si el espejado es horizontal
vert	Si el espejado es vertical

Descripción

Retorna la configuración de espejado del canvas.

canvas:attrOpacity (opacity: number)

Argumento

opacity	Nuevo valor de opacidad del canvas

Descripción

Altera la opacidad del canvas.

El valor de la opacidad varía entre 0 (transparente) y 255 (opaco).

El canvas principal no puede tener su valor alterado pues es controlado por el formateador NCL.

canvas:attrOpacity () -> opacity: number

Valor de retorno

opacity	Opacidad del canvas

Descripción

Retorna el valor de la opacidad del canvas.

canvas:attrRotation (degrees: number)

Argumento

degrees	Rotación del canvas en grados

Descripción

Configura el atributo de rotación del canvas, que debe ser múltiplo de 90 grados.

El canvas principal no puede tener su valor alterado pues es controlado por el formateador NCL

canvas:attrRotation () -> degrees: number

Valor de retorno

degrees	Rotación del canvas en grados

Descripción

Retorna el atributo de rotación del canvas.

canvas:attrScale () -> w, h: number|boolean

Argumentos

w	Anchura de escalonamiento del canvas
h	Altura de escalonamiento del canvas

Descripción

Escalona el canvas con nueva anchura y altura.

Uno de los valores puede ser *true*, indicando que la proporción del canvas debe mantenerse.

El atributo de escalonamiento es independiente del atributo de tamaño, o sea, el tamaño del canvas se mantiene.

El canvas principal no puede tener su valor alterado pues es controlado por el formateador NCL.

canvas:attrScale () -> w, h: number

Valores de retorno

x	Anchura de escalonamiento del canvas
y	Altura de escalonamiento del canvas

Descripción

Retorna los valores de escalonamiento del canvas.

10.3.2.4 Primitivas

Todos los métodos siguientes tienen en cuenta los atributos del canvas.

---

canvas:drawLine (x1, y1, x2, y2: number)

Argumentos

x1	Extremidad 1 de la línea
y1	Extremidad 1 de la línea
x2	Extremidad 2 de la línea
y2	Extremidad 2 de la línea

Descripción

Dibuja una línea con sus extremidades en (x1,y1) y (x2,y2).

---

canvas:drawRect (mode: string; x, y, width, height: Number)

Argumentos

mode	Modo de dibujo
x	Coordenada del rectángulo
y	Coordenada del rectángulo
width	Anchura del rectángulo
height	Altura del rectángulo

Descripción

Función para dibujo y rellenado de rectángulos.

El parámetro mode puede recibir 'frame' para dibujar apenas el marco del rectángulo o 'fill' para rellenarlo.

---

canvas:drawRoundRect (mode: string; x, y, width, height, arcWidth, arc Height: number)

Argumentos

mode	Modo de dibujo
x	Coordenada del rectángulo
y	Coordenada del rectángulo
width	Anchora del rectángulo
height	Altura del rectángulo

arcWidth	Anchura del arco de la esquina redondeada
arcHeight	Altura del arco de la esquina redondeada

Descripción

Función para dibujo y rellenado de rectángulos redondeados.

El parámetro mode puede recibir ‘frame’ para dibujar sólo el contorno del rectángulo o ‘fill’ para rellenarlo.

---

canvas:drawPolygon (mode: string) -> drawer: function

Argumentos

mode	Modo de dibujo

Valores de retorno

f	Función de dibujo

Descripción

Método para dibujo y rellenado de polígonos.

El parámetro mode recibe el valor 'open', para dibujar el polígono sin conectar el último punto al primero; 'close', para dibujar el polígono conectando el último punto al primero; or 'fill', para dibujar el polígono enlazando el último punto al primero y pintar la región interior.

La función canvas:drawPolygon retorna una función anónima "drawer" con la firma:

```
function (x, y) end
```

La función retornada recibe las coordenadas del próximo vértice del polígono y retorna a sí mismo con resultado. Ese procedimiento recurrente facilita la composición:

```
canvas:drawPolygon('fill')(1,1)(10,1)(10,10)(1,10)()
```

Al recibir nil la función "drawer" efectúa la operación encadenada. Cualquier llamada subsiguiente debe obligatoriamente generar un error.

---

canvas:drawEllipse (mode: string; xc, yc, width, height, ang_start, ang_end: number)

Argumentos

mode	Modo de dibujo
xc	Centro de la elipsis
yc	Centro de la elipsis
width	Anchura de la elipsis
height	Altura de la elipsis
ang_start	Ángulo de inicio
ang_end	Ángulo de fin

Descripción

Dibuja elipsis y otras primitivas análogas tales como círculos, arcos y sectores.

Parámetro mode puede recibir 'arc' para dibujar sólo la circunferencia o 'fill' para rellenado interno.

---

canvas:drawText (x, y: number; text: string)

Argumentos

x	Coordenada del texto
y	Coordenada del texto
text	Texto a ser dibujado

Descripción

Dibuja el texto pasado en la posición (x,y) del canvas utilizando la fuente configurada en canvas:attrFont().

10.3.2.5 Miscelánea

---

canvas:clear ([x, y, w, h: number])

Argumentos

x	Coordenada del área de clear
y	Coordenada del área de clear
w	Anchura del área de clear
h	Altura del área de clear

Descripción

Limpia el canvas con el color configurado en *attrColor*.

Caso que los parámetros de área no hayan sido pasados, se asume que se limpiará el canvas entero.

---

canvas:flush ()

Descripción

Actualiza el canvas después de operaciones de dibujo y composición.

Es suficiente llamarlo una sola vez después de una secuencia de operaciones.

---

canvas:compose (x, y: number; src: canvas; [src_x, src_y, src_width, src_height: number])

Argumentos

x	Posición de la composición

y	Posición de la composición
src	Canvas a ser compuesto
src_x	Posición de la composición en el canvas src
src_y	Posición de la composición en el canvas src
src_width	Anchura de la composición en el canvas src
src_height	Altura de la composición en el canvas src

Descripción

Hace sobre el canvas (canvas de destino), en su posición (x,y), la composición pixel a pixel con src (canvas de origen).

Los otros parámetros son opcionales e indican qué parte del canvas src componer. Cuando están ausentes, el canvas entero está compuesto.

Esa operación llama la src:flush() automáticamente antes de la composición.

La composición satisface la ecuación:

Cd = Cs*As + Cd*(255 - As)/255

Ad = As*As + Ad*(255 - As)/255

donde:

Cd = color del canvas de destino (canvas)

Ad = alfa del canvas de destino (canvas)

Cs = color del canvas de origen (src)

As = alfa del canvas de origen (src)

Después de la operación el canvas de destino posee el resultado de la composición y src no sufre cualquier alteración.

---

canvas:pixel (x, y, R, G, B, A: number)

Argumentos

x	Posición del *pixel*
y	Posición del *pixel*
R	Componente rojo del color
G	Componente verde del color
B	Componente azul del color

A	Componente Alpha del color

Descripción

Altera el color de un *pixel* del canvas.

---

canvas:pixel (x, y: Number) -> R, G, B, A: number

Argumentos

x	Posición del *pixel*
y	Posición del *pixel*

Valores de retorno

R	Componente rojo del color
G	Componente verde del color
B	Componente azul del color
A	Componente alpha del color

Descripción

Retorna el color de un *pixel* del canvas.

---

canvas:measureText (text: string) -> dx, dy: number

Argumentos

texto	Texto a ser medido

Valores de retorno

dx	Anchura del texto
dy	Altura del texto

Descripción

Retorna las coordenadas limítrofes para el texto pasado, en el caso que el mismo hubiese sido proyectado en la posición (x,y) del canvas con la fuente configurada en canvas:attrFont().

10.3.3 Módulo *event*

10.3.3.1 Visión general

Este módulo ofrece una API para tratamiento de eventos. A través del mismo el formateador NCL y una aplicación NCLua pueden comunicarse de manera asíncrona.

Una aplicación también puede disfrutar de ese mecanismo internamente, a través de eventos de la clase "user".

Probablemente el uso más común de aplicaciones NCLua será tratando eventos: Ya sean eventos NCL (ver ABNT NBR 15606-2:2007, Sección 7.2.8) o de interacción con el usuario (por el mando a distancia, por ejemplo).

Durante su iniciación, antes de tornarse orientado a eventos, un *script* Lua debe obligatoriamente registrar una función de tratamiento de eventos. Después de la iniciación, cualquier acción tomada por la aplicación es solamente en respuesta a un evento enviado por el formateador NCL a la función "handler".

=== example.lua ===

```
...                 -- Código de iniciación

function handler (evt)

  ...               -- Código tratador

end

event.register(handler) -- registro del tratador en el middleware

=== fin ===
```

Entre los tipos de eventos que pueden llegar al handler están todos los eventos generados por el formateador NCL. Un *script* Lua también es capaz de generar eventos, denominados "espontáneos", con una llamada a la función `event.post(evt)`.

10.3.3.2 Funciones

---

event.post ([dst: string]; evt: event) -> sent: boolean; err_msg: string

Argumentos

dst	Destinatario del evento
evt	Evento a ser enviado

Valores de retorno

sent	Si el evento fue enviado con éxito
err_msg	Mensaje de error en caso de fallo

Descripción

Envía el evento pasado.

El parámetro "dst" es el destinatario del evento y puede asumir los valores "in" (envío para la propia aplicación) y "Oct" (envío para el formateador NCL). El valor default es ‘out’.

---

event.timer (time: number, f: function) -> cancel: function

Argumentos

time	Tiempo en milisegundos
f	Función de *callback*

Valor de retorno

Unreg	Función para cancelar el timer

Descripción

Crea un timer que expira después del time (en milisegundos) y entonces llama a la función f.

La firma de f es simple, sin recibimiento de parámetros:

```
function f () end
```

Una vez creado, no hay como cancelar el timer.

El valor de 0 milisegundos es válido. En ese caso, event.timer() debe, obligatoriamente, retornar inmediatamente y f debe ser llamada en cuanto sea posible.

---

event.register ([pos: number]; f: function; [class: string]; […: any])

Argumentos

pos	Posición del registro (opcional)
f	Función de *callback*.
class	Filtro de clase (opcional)
…	Filtro dependiente de la clase (opcional)

Descripción

Registra la función pasada como un *listener* de eventos, es decir, siempre que ocurra un evento, f será llamada. La función f es, así, la función de tratamiento de eventos (function “handler”).

El parámetro *pos* es opcional e indica la posición en que *f* es registrada. Caso no haya sido pasado, la función es registrada en último lugar.

El parámetro *class* también es opcional y cuando se pasa indica qué clase de eventos debe recibir la función. Si es especificado el parámetro, pueden ser definidos otros filtros dependientes de la clase. El valor *nil* en cualquier posición indica que el parámetro no debe ser filtrado.

La firma de `f` es:

```
function f (evt) end -> handled: boolean
```

Donde `evt` es el evento que, al ocurrir, activa la función.

La función puede retornar “true”, para señalizar que el evento fue tratado y, por lo tanto, no debe ser enviado a otros tratadores.

Se recomienda que la función, definida por la aplicación, retorne rápidamente, ya que, mientras esté siendo ejecutada, ningún otro evento será procesado.

El formateador NCL debe garantizar obligatoriamente que las funciones reciban los eventos en el orden en que se registraron, y si ninguna de las mismas retorna el valor “true”, el formateador NCL debe notificar los otros tratadores registrados.

---

event.unregister (f: function)

Argumentos

f	Función de *callback*

Descripción

Quita del registro la función pasada como un *listener,* es decir, nuevos eventos dejarán de ser pasados a `f`.

---

event.uptime () -> ms: number

Valores de retorno

ms	Tiempo en milisegundos

Descripción

Retorna el número de milisegundos transcurridos desde el inicio de la aplicación.

10.3.3.3 Clases de eventos

La función `event.post()` y el “handler” registrado en `event.register()` reciben eventos como parámetros.

Un evento se describe por una tabla Lua normal, donde el campo `class` es obligatorio e identifica la clase del evento.

Las siguientes clases de eventos se definen:

Clase `key`:

evt = { class='key', type: string, key: string}

* type puede ser 'press' o 'release'.

* key es el valor de la tecla correspondiente.

EJEMPLO evt = { class='key', type='press', key=’0’ }

NOTA En la clase key, el filtro dependiente de la clase puede ser *type* y *key*, en ese orden.

Classe `pointer`:

evt = { class=’pointer’, type: string, x=number, y=number }

* el tipo puede ser "press", "release" o "move"

* X e Y se refieren a las coordinadas de la ocurrencia del evento puntero

NOTA En el tipo pointer, el filtro que depende del tipo sólo puede ser *type*.

EJEMPLO evt = { class=’pointer’, type=’press’, x=20, y=50 }

Clase `ncl`:

Las relaciones entre los nudos de media NCL se basan en eventos. Lua tiene acceso a esos eventos a través de la `Clase ncl`.

Los eventos pueden actuar en las dos direcciones, es decir, el formateador puede enviar eventos de acción para alterar el estado del exhibidor Lua, y éste, a su vez, puede disparar eventos de transición para indicar cambios de estado.

En los eventos, el campo type debe asumir obligatoriamente uno de los tres valores: 'presentation', 'selection' o 'attribution'

Eventos pueden ser orientados a anclas específicas o al nodo en su totalidad, esto es identificado en el campo label, que asume totalmente el nodo, cuando esté ausente.

En el caso de un evento generado por el formateador, el campo action debe tener obligatoriamente uno de los valores:

'start', 'stop', 'abort', 'pause' o 'resume'

Tipo ‘presentation’:

---

evt = { class='ncl', type='presentation', label='?', action=’?’}

Tipo ‘attribution’:

evt = { class='ncl', type='attribution', name='?', action=’?', value='?' }

Para eventos generados por el exhibidor Lua, el campo "action" debe obligatoriamente asumir uno de los valores

'start', 'stop', 'abort', 'pause' o 'resume'

Tipo ‘presentation’:

evt = { class='ncl', type='presentation', label='?', action=’start’/‘stop’/‘abort’/‘pause’/‘resume’**}**}

Tipo ‘selection’:

evt = { class='ncl', type='selection', label='?', action='stop' }

Tipo ‘attribution’:

evt = { class='ncl', type='attribution', name='?', action='start'/'stop'/'abort'/'pause'/'resume', value='?' }

por:

NOTA En la clase ncl, lo filtro dependiente de la clase puede ser *type, area* y *action*, en esa orden.

Classe `edit`:

Esta clase espeja los comandos de edición para el Gerenciador de Bases Privadas (ver Sección 9). Sin embargo, hay una importante diferencia entre los comandos de edición procedentes de los eventos de flujo DSM-CC y los comandos de edición realizados por los scripts Lua (objetos NCLua). Los primeros alteran no sólo la presentación de un documento NCL, sino también la especificación de un documento NCL. O sea, al final del proceso es generado un nuevo documento NCL, incorporando todos los resultados de la edición. Por otro lado, los comandos de edición procedentes de los objetos de media NCLua alteran solamente la presentación del documento NCL. El documento original es preservado durante todo el proceso de edición.

Así como en las otras clases de evento, un comando de edición es representado por una tabla Lua. Todo evento debe obligatoriamente cargar el campo *command*: un string com el nombre del comando de edición. Los otros campos dependen del tipo de comando, conforme Tabla 56. La única diferencia es con relación al campo que define los pares de referencia {uri,id}, denominado *data* en la clase edit. El valor del campo acepta no sólo los pares de referencia {uri,id} mencionados en la Tabla 56, sino también *strings* XML con el contenido a ser adicionado.

EJEMPLO

```
evt = {
    command = ‘addNode’,
    compositeId = ‘someId’,
    data = ‘<media>...’,
}
```

Los campos *baseId* y *documentId* (cuando son aplicables) son opcionales y asumen por *default* el identificador de la base y del documento en el cual el NCLua está ejecutando.

El evento describiendo el comando de edición también puede recibir una referencia de tiempo como parámetro opcional *time*. Ese parámetro opcional puede usarse para especificar el exacto momento en que el comando de edición debe ser ejecutado. Si este parámetro no há sido suministrado em la llamada de función, el comando de edición debe ser ejecutado inmediatamente. Cuando es suministrado, el parámetro puede tener dos tipos distintos de valores, con diferentes significados. Si es un valor numérico, define la cantidad de tiempo, en segundos, que la ejecución del comando debe ser postergada. No obstante, ese parámetro también puede especificar el exacto momento para ejecución del comando, en valores absolutos. Para ello, el parámetro debe obligatoriamente ser una tabla con los siguientes campos: *year* (cuatro dígitos), *month* (1 a 12), *day* (1 a 31), *hour* (0 a 23), *minute* (0 a 59), *second* (0 a 61) e *isdst* (señalizador de horario de verano, un booleano).”

Clase tcp:

O uso do canal de interatividade é realizado por meio desta classe de eventos.

Para el envío o recibimiento de datos tcp, es obligatorio que se establezca inicialmente una conexión, mandando un evento en la forma:

evt = { class='tcp', type='connect', host=addr, port=number, [timeout=number] }

El resultado de la conexión retorna en un tratador de eventos prerregistrado para la clase. El evento retornado tiene la siguiente forma:

evt = { class='tcp', type='connect', host=addr, port=number, connection=identifier, error=<err_msg>}

Los campos *error* y *connection* son mutuamente exclusivos. Cuando hay un error de comunicación, una mensaje debe retornar en el campo *error*. Cuando hay éxito en la comunicación, el identificador de la conexión es retornado en el campo *connection.*

Una aplicación NCLua envía datos por un canal de interactividad, mandando eventos en la forma:

evt = { class=’tcp’, type='data', connection=identifier, value=string, [timeout=number] }

De forma análoga, una aplicación NCLua recibe datos transportados por un canal de interactividad haciendo uso de eventos en la forma:

evt = { class=’tcp’, type='data', connection=identifier, value=string, error=msg }

Los campos *error* y *value* son mutuamente exclusivos. Cuando hay un error en la comunicación, retorna un mensaje por el campo *error.* Cuando la comunicación es exitosa, el mensaje transportado pasa en el campo *value.*

Para cerrar una conexión, debe mandarse el siguiente evento obligatoriamente:

evt = { class='tcp', type='disconnect', connection=identifier }

NOTA 1 Se recomienda que cuestiones tales como autenticación, se traten en una implementación específica del *middleware*.

NOTA 2 En la clase tcp, el filtro dependiente de la clase sólo puede ser *connection*.

Classe SMS:

Una aplicación NCLua envía datos, utilizando SMS, por medio de publicación de eventos en el siguiente formato:

evt = { class=’sms’, type=’send’, to=’string’, value=string [, id:string]}

El campo `to` contiene el número de destino (número del teléfono o número de la cuenta). Se no estuvieran

especificados, la región y los prefijos de código del país recibirán la región y los códigos del país desde donde el mensaje está siendo enviado.

El campo `value` guarda el contenido del mensaje.

El campo `id` puede ser usado para identificar el SMS que será enviado. La aplicación es responsable por definir el valor de `id` y garantizar su unicidad.

Un evento de confirmación debe ser enviado obligatoriamente de vuelta a la aplicación NCLua, manteniendo el formato:

evt = { class='sms', type='send', to:string, sent:Boolean [,error:string] [, id:string] }

En el mensaje de confirmación, el campo `to` debe obligatoriamente tener el mismo valor que en el evento original publicado por la aplicación NCLua. El campo `sent` notifica si el SMS fue despachado por el dispositivo (verdadero) o no. El campo `error` es opcional. Si el valor del campo `sent` fuera falso, puede contener un mensaje de error detallado. Si el SMS original fuera publicado con el campo `id` definido, el evento de confirmación debe llegar obligatoriamente con el mismo valor de id. De ese modo, la aplicación NCLua puede hacer una asociación entre ambos eventos y tratar varios mensajes SMS enviados simultáneamente.

De forma similar, una aplicación NCLua se registra para recibir mensajes SMS, publicando eventos en el formato:

evt = { class='sms', type ='register', port:number }

El campo `port` debe recibir obligatoriamente un número válido de puerta TCP. Para el cumplimiento de las normas GSM (3GPP TS 23.040 V6.8.1, de 2006-10), ese valor debe obligatoriamente estar en el intervalo [16000,16999].

Eventos recibidos por el tratador poseen el siguiente formato:

evt = { class='sms', type='receive', from:string, port:number, value:string }

El campo `port` es definido como en el tipo = 'register'. El campo `from` contiene el número del mensaje de origen (número del teléfono o número de la cuenta). El prefijo de la región y el código del país pueden ser omitidos si fueran iguales a los del receptor. El campo `value` tiene el contenido del mensaje.

En cualquier momento, la aplicación puede solicitar para dejar de recibir mensajes SMS en determinada puerta, registrando el evento:

evt = { class='sms', type='unregister', port:number }

El campo puerta es definido como en el tipo = 'register'.

En el momento que la presentación de la media NCLua se detenga, la implementación de *middleware* debe obligatoriamente asegurar que todas las puertas sean dadas de baja.

NOTA 1 Se recomienda que una implementación específica de *middleware* trate cuestiones como autenticación etc.

NOTA 2 En el tipo sms, el filtro dependiente del tipo sólo puede ser *from* y *port*, en ese orden.

NOTA 3 La finalidad del número de puerta es evitar conflictos entre los mensajes comunes SMS recibidos por un usuario y los mensajes SMS que deben obligatoriamente ser tratados solamente por la aplicación.

Una implementación de Ginga-NCL debe obligatoriamente devolver *false* inmediatamente en cada llamada a event.post () que usa un tipo de evento no soportado. Se recomienda que la aplicación NCLua capture esa condición de error para verificar si el envío de SMS falló.

Clase si:

La clase de eventos 'si' permite el acceso a un conjunto de informaciones multiplexadas en un flujo de transporte y transmitidas periódicamente por difusión.

El proceso de adquisición de las informaciones debe obligatoriamente realizarse en dos pasos:

1) una requisición realizada por una llamada a `event.post()`, que no bloquea la ejecución del script;

2) un evento, posteriormente repasado a los tratadores de eventos registrados del script NCLua, cuyo campo data contiene un conjunto de subcampos que depende del tipo de la información requisitada, y que es representado por una tabla lua.

NOTA En la clase si, el filtro dependiente de la clase sólo puede ser *type*.

Cuatro tipos de eventos son definidos por los siguientes tipos de tablas:

---

type = 'services'

La tabla del tipo 'services' consiste en un conjunto de vectores. Cada vector posee informaciones relativas a un servicio multiplexado del flujo de transporte sintonizado.

Cada requisición para una tabla del tipo de evento 'services' se debe obligatoriamente realizar a través de la siguiente llamada:

event.post('out', { class='si', type='services'[, index=N][, fields={field_1, field_2,…, field_j}]}),

donde:

a) el campo index, si es especificado, indica el índice del servicio; caso no sea especificado, todos los servicios deben obligatoriamente ser retornados;

b) el campo fields puede tener como valor cualquier subconjunto de los subcampos definidos para la tabla retornada en el campo data del evento de respuesta (así, field_i representa uno de los subcampos de la tabla data. Caso el campo fields no sea especificado, todos los subcampos de la tabla retornada en el campo data deben ser rellenados obligatoriamente.

El evento de respuesta es generado después que todas las informaciones requisitadas sean procesadas por el middleware (las informaciones no transmitidas por difusión dentro de un intervalo máximo de tiempo especificado en la ABNT NBR 15603-2:2007, Tabla 6, son retornadas como nulas). La tabla del campo data es retornada en el evento, como sigue:

```
evt = {
   class = 'si',
   type = 'services',
   data = {
      [i] = {  -- cada servicio está en una posición
         id                          = <number>,
         isAvailable                  = <boolean>,
         isPartialReception           = <boolean>,
         parentalControlRating        = <number>,
         runningStatus                = <number>,
         serviceType                          = <number>,
         providerName                 = <string>,
         serviceName                  = <string>,
         stream = {
            [j] = {
               pid = <number>,
               componentTag = <number>,
               type = <number>,
               regionSpecType = <number>,
               regionSpec     = <string>,
```

```
        }
    }
  }
}
```

Para la obtención de las respuestas al envío de eventos del tipo ‘services’, se recomienda que los subcampos de la tabla data sean calculados con base en tablas SI y descriptores asociados al servicio [i].

Se recomienda que los valores de los subcampos id y runningStatus de la tabla data se computen conforme los valores de los campos service_id y running_status, respectivamente, de la tabla SDT (ver ABNT NBR 15603-2:2007, Tabla 13) que describe el servicio [i].

Se recomienda que se establezca la misma relación entre los subcampos providerName y serviceName de la tabla data y los campos service_name y service_provider_name, respectivamente, del descriptor service_descriptor (ver ABNT NBR 15603-2:2007).

Se recomienda que el subcampo parentalControlRating de la tabla data sea calculado con el valor del campo rating del descriptor parentalControlRating, en el cual el campo country_code presente el mismo país referente al valor de la variable de ambiente (nudo Settings) user.location.

Se recomienda que el subcampo isAvailable de la tabla data se calcule basado en el valor del campo country_code (con el grupo de países disponibles) del descriptor country_availability_descriptor (ver ABNT NBR 15603-2:2007, Sección 8.3.6) relativo al servicio [i]. El valor “true” es atribuido solamente si el campo country_code contiene el mismo país referente al valor de la variable de ambiente (nudo Settings) user.location.

Se recomienda que el subcampo isPartialReception de la tabla data sea calculado basado en el valor del campo service_id del descriptor partial_reception_descriptor (ver ABNT NBR 15603-2:2007, Sección 8.3.32).

Se recomienda que la semántica del subcampo serviceType de la tabla data sea definida por la ABNT NBR 15603-2:2007 ABNT NBR 15603-2:2007, Tabla H.2.

Se recomienda que la semántica del subcampo runningStatus sea definida de acuerdo con la ABNT NBR 15603-2:2007, Tabla 14.

Se recomienda que el subcampo pid de la tabla stream tenga el valor del campo pid del encabezamiento del paquete del flujo elemental [i] (ver ISO/IEC 13818-1).

Se recomienda que el valor del subcampo componentTag de la tabla stream se obtenga a través del campo component_tag del descriptor stream_identifier_descriptor (ver ABNT NBR 15603-2:2007, Sección 8.3.16) relacionado al flujo elemental [i].

Se recomienda que el subcampo type de la tabla stream tenga la semántica definida en la ISO/IEC 13818-1:2008, Tabla 2-34, relacionada al flujo elemental [i].

Se recomienda que el subcampo regionSpecType de la tabla stream defina el método de codificación del campo regionSpec de la misma tabla, conforme semántica definida en la ABNT NBR 15603-2:2007, Tabla 53.

Se recomienda que el campo regionSpec de la tabla stream defina la región para la cual es designado el flujo elemental [i].

También se recomienda que los campos regionSpec y regionSpecType se obtengan a través del descriptor target_region_descriptor especificado en la ABNT NBR 15603-2:2007.

---

type = ‘mosaic’

La tabla del tipo ‘mosaic’ consiste en un subconjunto de informaciones para el mosaico que es abastecido como una matriz. La tabla es, por lo tanto, opcional.

NOTA Todo servicio presente en el mosaico (identificado por un subconjunto del conjunto "bouquetId, networkId, tsId, serviceId, eventId") es del tipo one-seg (aspect ratio, bitrate etc.).

Cuando la tabla del tipo ‘mosaic’ es abastecida, la requisición de la tabla debe obligatoriamente realizarse a través de la siguiente llamada:

```
event.post('out', { class='si', type='mosaic'[, fields={field_1, field_2,…, field_j}]}),
```

donde el campo fields del evento de requisición puede tener como valor cualquier subconjunto de los subcampos definidos para la tabla retornada en el campo data del evento de respuesta (así, field_i representa uno de los subcampos de la tabla data). Caso el campo fields no sea especificado, todos los subcampos de la tabla retornada en el campo data deben ser rellenados obligatoriamente.

El evento de respuesta es generado después que todas las informaciones requisitadas sean procesadas por el middleware (informaciones no transmitidas por difusión dentro de un intervalo máximo de tiempo especificado en la ABNT NBR 15603-2:2007, Tabla 6, son retornadas como nulas). La tabla del campo data retorna en el evento, como sigue:

```
evt = {
   class = 'si',
   type = 'mosaic',
   data = {
      [i] = {
         [j] = {
            logicalId       = <number>,
            presentationInfo = <number>,
            id              = <number>,
            linkageInfo     = <number>,
            bouquetId       = <number>,
            networkId       = <number>,
            tsId            = <number>,
            serviceId       = <number>,
            eventId         = <number>,
         }
      }
   }
}
```

NOTA Para la obtención de las respuestas a envíos de eventos del tipo ‘mosaic’, se recomienda que los subcampos de la tabla data sean calculados con base en tablas SI y descriptores asociados al mosaico. Se recomienda que los valores máximos de [i] y [j], así como los valores de los subcampos logicalId, presentationInfo, id, linkageInfo, bouquetId, networkId, tsId, serviceId y eventId de la tabla data se obtengan a través de los campos number_of_horizontal_elementary_cells, number_of_vertical_elementary_cells, logical_cell_id, logical_cell_presentation_info, id, cell_linkage_info, bouquet_id, original_network_id, transport_stream_id, service_id y event_id del descriptor mosaic_descriptor (ver ABNT NBR 15603-2:2007, Sección 8.3.9), respectivamente.

type = ‘epg’

La tabla del tipo ‘epg’ consiste en un conjunto de vectores. Cada vector posee informaciones relativas a un evento del contenido transmitido.

NOTA Todo servicio presente en el epg es del tipo one-seg (aspect ratio, bitrate etc.).

Una requisición de la tabla del tipo ‘epg’ se debe realizar obligatoriamente a través de una de las posibles llamadas a seguir:

1) event.post('out', { class='si', type='epg', stage=’current’[, fields={field_1, field_2,…, field_j}]})

donde el campo fields del evento de requisición puede tener como valor cualquier subconjunto de los subcampos definidos para la tabla retornada en el campo data del evento de respuesta (así, field_i representa

uno de los subcampos de la tabla data). Caso el campo fields no sea especificado, todos los subcampos de la tabla retornada en el campo data deben ser rellenados obligatoriamente.

Descripción: obtiene las informaciones del evento corriente de la programación

2) event.post('out', {class='si', type='epg', stage='next'[, eventId=<number>][, fields={field_1, field_2,…, field_j}]})

donde:

a) el campo eventId, cuando es especificado, identifica el evento inmediatamente anterior al evento del que se quieren las informaciones. Cuando no es especificado indica que las informaciones deseadas son las del evento siguiente al evento corriente de la programación;

b) el campo fields del evento de requisición puede tener como valor cualquier subconjunto de los subcampos definidos para la tabla retornada en el campo data del evento de respuesta (así, field_i representa uno de los subcampos de la tabla data, como se especifica a continuación). Caso el campo fields no se especifique, todos los subcampos de la tabla retornada en el campo data se deben rellenar obligatoriamente.

Descripción: obtiene las informaciones del evento siguiente al evento identificado en *eventId* o, en el caso de eventId no ser especificado, del evento siguiente al evento corriente de la programación.

3) event.post('out', {class='si', type='epg', stage='schedule', startTime=<date>, endTime=<date>[, fields={field_1, field_2,…, field_j}]})

donde el campo fields del evento de requisición puede tener como valor cualquier subconjunto de los subcampos definidos para la tabla retornada en el campo data del evento de respuesta (así, field_i representa uno de los subcampos de la tabla data). Caso el campo fields no sea especificado, todos los subcampos de la tabla retornada en el campo data se deben rellenar obligatoriamente.

Descripción: obtiene las informaciones de los eventos comprendidos en la banda de tiempo pasada en los campos startTime y endTime que tienen como valor tablas en el formato de <date>.

El evento de respuesta es generado después que todas las informaciones requisitadas sean procesadas por el middleware (informaciones no transmitidas por difusión dentro de un intervalo máximo de tiempo especificado en la ABNT NBR 15603-2:2007, Tabla 6 son retornadas como nulas). La tabla del campo data es retornada en el evento, como sigue:

```
evt = {
   class = 'si',
   type = 'epg',
   data = {
      [i] – {
         startTime              = <date>,
         endTime                = <date>,
         runningStatus          = <number>,
         name                   = <string>,
         originalNetworkId      = <number>,
         shortDescription       = <string>,
         extendedDescription    = <string>,
         copyrightId            = <number>,
         copyrightInfo          = <string>,
         parentalRating         = <number>,
         parentalRatingDescription = <string>,
         audioLanguageCode      = <string>,
```

```
audioLanguageCode2       = <string>,
dataContentLanguageCode  = <string>,
dataContentText          = <string>,
hasInteractivity         = <boolean>,
logoURI                  = <string>,
contentDescription       = {
   [1] = <content_nibble_1>,
   [2] = <content_nibble_2>,
   [3] = <user_nibble_1>,
   [4] = <user_nibble_2> }
},
linkage = {
   tsId     = <number>,
   networkId = <number>,
   serviceId = <number>,
   type     = <number>, (table 30, norma 3 vol 2)
   data     = <string>,
},
hyperlink = {
   type           = <number>,
   destinationType = <number>,
   tsId           = <number>,
   networkId      = <number>,
   eventId        = <number>,
   componentTag   = <number>,
   moduleId       = <number>,
   serviceId      = <number>,
   contentId      = <number>,
   url            = <string>,
},
series = {
   id             = <number>,
   repeatLabel    = <number>,
   programPattern  = <number>,
   episodeNumber   = <number>,
   lastEpisodeNumber = <number>,
   name = <string>,
},
eventGroup = {
   type = <number>,
   [j] = {
      id       = <number>,
      tsId     = <number>,
```

```
            networkId = <number>,
            serviceId = <number>,
         }
      },
      componentGroup = {
         type = <number>,
         [j] = {
            id          = <number>,
            totalBitRate = <number>,
            description  = <string>,
            caUnit = {
               id       = <number>,   -- (table 80, norma 3 vol 2)
               component = {
                  [k] = tag (<number>)
               }
            },
         }
      }
   }
}
```

Para la obtención de las respuestas a envíos de eventos del tipo ‘epg’, se recomienda que los campos de la tabla data se calculen con base en tablas SI y descriptores asociados al evento [i].

Se recomienda que los valores de los campos startTime, endTime, runningStatus y originalNetworkId de la tabla data sean obtenidos a través de los campos start_time, (duration + start_time), running_status y original_network_id de la tabla SI event_information_section (ver ABNT NBR 15603-2:2007, Tabla 15), respectivamente.

Se recomienda que los valores de los campos name y shortDescription se obtengan a través de los campos event_name_char y text_char del descriptor short_event_descriptor (ver ABNT NBR 15603-2:2007, Sección 8.3.15), respectivamente.

Se recomienda que el valor del campo extendedDescription se obtenga a través del campo text_char del descriptor extended_event_descriptor (ver ABNT NBR 15603-2:2007, Sección 8.3.7)

Se recomienda que los valores de los campos copyrightId y copyrightInfo sean obtenidos a través de los campos copyright_identifier y additional_copyright_info del descriptor copyright_descriptor (conforme la ISO/IEC 13818-1:2008, Tabla 2-63, respectivamente.

El campo parentalRating de la tabla data tiene la semántica definida en la ABNT NBR 15603-2:2007, Tabla 32 y se recomienda que su valor se calcule con base en el campo country_code del descriptor parental_rating_descriptor y en la variable de ambiente (Settings node) user.location.

El campo parentalRatingDescription de la tabla data tiene la semántica definida en la ABNT NBR 15603-2:2007, Tabla 33 y se recomienda que su valor sea calculado con base en el campo country_code del descriptor parental_rating_descriptor y en la variable de ambiente (Settings node) user.location.

Se recomienda que los valores de los campos audioLanguageCode y audioLanguageCode2 sean obtenidos a través de los campos ISO_639_language_code e ISO_639_language_code2 del descriptor audio_component_descriptor (ver ABNT NBR 15603-2:2007, Tabla 48), respectivamente.

Se recomienda que los valores de los campos dataContentLanguageCode y dataContextText se obtengan a través de los campos ISO_639_language_code y text_char del descriptor data_content_descriptor (ABNT NBR 15603-2:2007, Tabla 54), respectivamente.

El campo hasInteractivity de la tabla data debe tener obligatoriamente el valor true cuando el evento [i] tenga una aplicación interactiva disponible.

Se recomienda que el valor del campo logoURI de la tabla data posea la localización del logotipo, transmitido por una tabla CDT (ver ABNT NBR 15603-2:2007, Sección 8.3.44).

Se recomienda que los valores de los campos de la tabla contentDescription se obtengan a través de los campos del mismo nombre del descriptor content_descriptor (ver ABNT NBR 15603-2:2007, Sección 8.3.5).

Se recomienda que los valores de los campos tsId, networkId, serviceId, type y data de la tabla linkage se obtengan a través de los campos transport_stream_id, original_network_id, original_service_id, description_type y user_defined del descriptor linkage_descriptor (ver ABNT NBR 15603-2:2007, Sección 8.3.40), respectivamente.

Se recomienda que los valores de los campos type, destinationType, tsId, networkId, eventId, componentTag, moduleId, contentId y url de la tabla hyperlink sean obtenidos a través de los campos hyper_linkage_type, link_destination_type, transport_stream_id, original_network_id, event_id, component_tag, moduleId, content_id y url_char del descritor hyperlink_descriptor respectivamente (ver ABNT NBR 15603-2:2007, Sección 8.3.29).

Se recomienda que los valores de los campos id, repeatLabel, programPattern, episodeNumber, lastEpisodeNumber y name de la tabla series sean obtenidos a través de los campos series_id, repeat_label, program_pattern, episode_number, last_episode_number y series_name_char del descriptor series_descriptor (ver ABNT NBR 15603-2:2007, Sección 8.3.33), respectivamente.

Se recomienda que los valores de los campos type, id, tsId, networkId y serviceId de la tabla eventGroup se obtengan a través de los campos group_type, event_id, transport_stream_id, original_network_id y service_id del descriptor event_group_descriptor respectivamente (ver ABNT NBR 15603-2:2007, Sección 8.3.34).

Se recomienda que los valores de los campos type, id, totalBitRate, description, caUnit.id, caUnit.component[k].tag, tsId, networkId y serviceId de la tabla componentGroup sean obtenidos a través de los campos component_group_type, component_group_id, total_bit_rate, text_char, CA_unit_id y component_tag del descriptor component_group_descriptor respectivamente (ver ABNT NBR 15603-2:2007, Sección 8.3.37).

type=’time’

La tabla del tipo ‘time’ consiste en informaciones sobre la fecha y hora corrientes basadas en la UTC, pero en el horario oficial del país en que se encuentra el receptor.

La requisición de la tabla se debe efectuar obligatoriamente a través de la siguiente llamada:

event.post('out', { class='si', type=’time’}),

El evento de respuesta es generado después que la información sobre fecha y hora corrientes requisitada sea procesada por el middleware (informaciones no transmitidas por difusión dentro de un intervalo máximo de tiempo especificado en la ABNT NBR 15603-2:2007, Tabla 6, son retornadas como nulas). La tabla del campo data es retornada en el evento, como sigue:

```
evt = {
   class = 'si',
   type = 'time',
   data = {
            year          = <number>,
            month         = <number>,
            day           = <number>,
            hours          = <number>,
            minutes       = <number>,
            seconds       = <number>
   }
```

NOTA Para la obtención de las respuestas a envíos de eventos del tipo 'time', se recomienda que los campos de la tabla data sean calculados con base en la tabla TOT y en el descriptor local_time_offset_descriptor (ver ABNT NBR 15603-2:2007, Sección 7.2.9).

Clase `user`:

Utilizando Clase user, aplicaciones pueden extender sus funcionalidades, creando sus propios eventos.

En esa clase, no está definido ningún campo (aparte, obviamente, del campo class).

NOTA En la clase user, el filtro dependiente de la clase puede ser *type*, si el filtro es definido.

10.3.4 Módulo *settings*

Exporta la tabla *settings* con variables definidas por el autor del documento NCL y variables de ambiente reservadas, contenidas en el nudo application/x-ginga-settings.

No se permite atribuir valores a los campos que representan variables en los nudos *settings.* En el caso que ocurra un intento de atribución debe ser generado un error. Las propiedades de un nudo *settings* solo se pueden modificar por medio de eslabones NCL.

La tabla *settings* subdivide sus grupos en varias subtablas, correspondiendo a cada grupo del nudo application/x-ginga-settings. Por ejemplo, en un objeto NCLua, la variable del nudo *settings* "system.CPU" es referida como settings.system.CPU.

Ejemplos de uso:

lang = settings.system.language

age = settings.user.age

val = settings.default.selBorderColor

settings.service.myVar = 10

settings.user.age = 18 --> ERROR!

10.3.5 Módulo *persistent*

Aplicaciones NCLua pueden grabar datos en un área restringida del middleware y recuperarlos entre ejecuciones. El exhibidor Lua permite a una aplicación NCLua persistir un valor para ser posteriormente usado por la misma o por otro objeto imperativo. Para ello, el exhibidor define un área reservada, inaccesible a objetos NCL que no sean imperativos. Este área se divide entre los grupos *"service", "channel"* y *"shared",* con la misma semántica de los grupos homónimos del nudo NCL *settings.* No existe ninguna variable predefinida o reservada en esos grupos, y objetos imperativos pueden atribuir valores a esas variables directamente. Se recomienda que otros lenguajes imperativas, Java en particular para los objetos NCLets (<media> elements of type application/x-ginga-NCLet), ofrezcan una API dando acceso al mismo área.

En éste modulo *persistent,* Lua ofrece una API para exportar la tabla *persistent* con las variables definidas en el área reservada.

El uso de la tabla *persistent* es semejante al uso de la tabla *settings* excepto por el hecho de que, en este caso, el código imperativo puede alterar los valores de los campos.

Ejemplos de uso:

persistent.service.total = 10

color = persistent.shared.color

11 Objetos imperativos Java en documentos NCL

El soporte a objetos imperativos en código Java es opcional en documentos NCL para dispositivos *one-seg*. La forma de agregar objetos imperativos en documentos NCL es definir un elemento <media> cuyo contenido (localizado por el atributo *src)* es el código imperativo a ser ejecutado. Los perfiles EDTV y BDTV da NCL 3.0 pueden, opcionalmente, dar soporte a elementos <media> del tipo "application/x-Ginga-NCLet" (extensión de archivo .class o .jar).

Los autores pueden definir eslabones NCL para iniciar, parar, pausar, retomar o abortar la ejecución de un código Java. Un exhibidor imperativo (máquina de ejecución del lenguaje) debe proveer obligatoriamente la interfaz del ambiente de ejecución imperativo con el formateador NCL.

En forma similar a lo realizado por los exhibidores de contenidos de media convencional, los exhibidores Java deben obligatoriamente controlar las máquinas de estado de los eventos asociados con el nudo imperativo. Como ejemplo, si un código termina su ejecución, el exhibidor debe obligatoriamente generar la transición *stops* en la máquina de estado de presentación del evento correspondiente a la ejecución imperativa.

NCL permite que la ejecución del código imperativo sea sincronizada con otros objetos NCL (imperativos o no). Un elemento <media> del tipo "application/x-ginga-NCLet" también puede definir anclas (a través de elementos <area>) y propiedades (a través de elementos <property>).

El código Lua puede ser asociado a elementos <area> (a través del atributo *label).* Si eslabones externos empiezan, paran, pausan, reinician o abortan la presentación del ancla, deben ser obligatoriamente disparados *callbacks* en el código imperativo. La forma como se definen esos *callbacks* es responsabilidad de cada código Lua asociado con el objeto imperativo NCLet.

Por otro lado, el código imperativo también puede comandar el inicio, parada, pausa, reinicio o aborto de eses anclas, a través de una API ofrecida por el lenguaje. Esas transiciones se pueden utilizar como condiciones de eslabones NCL para disparar acciones en otros objetos NCL del mismo documento. De ese modo, una sincronización (puente) de dos vías puede ser establecida entre el código Java y el resto del documento NCL.

La otra forma como el código Java puede ser sincronizado con otros objetos NCL es a través de elementos <property>. Un elemento <property> definido como hijo del elemento <media> del tipo application/x-ginga-NCLet puede ser mapeado para un trecho del código o para un atributo del código java. Cuando se mapea para un trecho de código, una acción de eslabón "set" aplicada a la propiedad debe obligatoriamente ejecutar la función con los valores atribuidos interpretados como parámetros de entrada. El atributo *name* del elemento <property> se debe usar obligatoriamente para identificar el trecho de código Java. Cuando el elemento <property> es mapeado para un atributo del código imperativo, la acción "set" debe obligatoriamente atribuir el valor al atributo. La máquina de estado de evento asociada a la propiedad debe ser controlada por el exhibidor Java.

Un elemento <property> definido como hijo de un elemento <media> do tipo "application/x-ginga-NCLet" también puede estar asociado a un *assessment role* de un eslabón NCL. En ese caso, el formateador NCL debe obligatoriamente cuestionar el valor de la propiedad para evaluar la expresión del eslabón. Si el elemento <property> es mapeado para un atributo de código, su valor debe ser retornado obligatoriamente por el exhibidor imperativo al formateador NCL. Si el elemento <property> se mapea para un trecho de código Java, el mismo debe ser obligatoriamente llamado y el valor de su resultado debe ser retornado obligatoriamente por el exhibidor Java al formateador NCL.

La instrucción *start* emitida por un formateador debe informar obligatoriamente al exhibidor Java los siguientes parámetros: El objeto NCLet a ser controlado, su descriptor asociado, una lista de eventos (definidos por los elementos <area> y <property>, hijos del elemento <media> que define el objeto imperativo) que necesitan ser monitoreados por el exhibidor imperativo, el identificador *(id)* del elemento <area> asociado al código imperativo a

ser ejecutado, y un tiempo de retardo, opcional. Desde el atributo *src,* el exhibidor Java debe intentar localizar el código Java e iniciar su ejecución. Si el contenido no se puede localizar, el exhibidor Java debe obligatoriamente cancelar la operación de inicialización sin realizar ninguna acción.

Se aconseja que la lista de eventos a ser monitoreados por un exhibidor Java también sea computada por el formateador, teniendo en cuenta la especificación del documento NCL. El mismo debe obligatoriamente chequear todos los eslabones de los cuales participa el objeto de media y el descriptor resultante. Al computar los eventos a ser monitoreados, el formateador debe considerar obligatoriamente la perspectiva del objeto de media NCLet, es decir, el camino de los varios elementos <body> y <context> para alcanzar en profundidad el elemento <media> correspondiente. Conviene que apenas eslabones contenidos en esos elementos <body> y <context> sean considerados en la computación de los eventos monitoreados.

Como con todos los tipos de elementos <media>, el tiempo de retardo es un parámetro opcional, y su valor *default* es “cero”. Si es mayor que cero, ese parámetro contiene un tiempo a ser esperado por el exhibidor imperativo antes de iniciar la ejecución.

En forma idéntica a los procedimientos realizados para elementos <media> con código imperativo Lua, si un exhibidor imperativo recibe una instrucción *start* para un evento asociado a un elemento <area> y ese evento se encuentra en el estado *sleeping,* podrá dar inicio a la ejecución del código Java asociado al elemento, incluso si otra parte del código imperativo del objeto de media está en ejecución (pausado o no). Sin embargo, si el evento asociado al elemento <area> meta está en el estado *occurring* o *paused,* la instrucción *start* debe ser ignorada por el exhibidor imperativo que seguirá controlando la ejecución anteriormente iniciada. Como consecuencia, de forma similar a lo que ocurre para los otros elementos <media>, conteniendo código Lua, una acción <simpleAction> con el atributo *actionType* igual a “stop”, “pause”, “resume” o “abort” debe ser conectada, a través de un eslabón, a una interfaz del nudo NCLet, que no debe ser ignorada cuando se aplica la acción.

La instrucción *set* emitida por un formateador puede ser aplicada a un objeto NCLet, al margen del hecho de estar siendo ejecutado o no (en ese último caso, aunque el objeto no esté siendo ejecutado, su exhibidor imperativo deberá obligatoriamente ya haber sido instado). En el primer caso, la instrucción *set* necesita identificar el objeto NCLet, un evento de atribución monitoreado y un valor a ser pasado al código Java asociado al evento. En el segundo caso, también debe identificar obligatoriamente el elemento <descriptor> a ser usado durante la ejecución del objeto (análogo a lo que se hace para la instrucción *start)*.

Para cada evento de atribución monitoreado, si el exhibidor imperativo cambia por sí mismo el valor del atributo, debe proceder obligatoriamente como si hubiese recibido una instrucción externa de *set.*

Se recomienda también que se ofrezca una API Java que permita que los códigos imperativoes cuestionen cualquier valor de propiedades predefinidas o dinámicas del nudo *settings* NCL (elemento <media> del tipo “application/x-ginga-settings”). Sin embargo, se debe observar que no se permite atribuir valores a esas propiedades directamente. Las propiedades de los nudos del tipo application/x-ginga-settings sólo pueden ser modificadas a través del uso de eslabones NCL. También se recomienda ofrecer una API Java que permita manejar variables persistentes definidas en el área reservada de persistencia de Lua.

También se recomiendan API que suministren un conjunto de métodos para dar soporte a los comandos de edición del NCL y comandos del Administrador de la Base Privada, en el caso que se ofrezca soporte a Java.

12 Requisitos de codificación de media y métodos de transmisión instados en documentos NCL

La codificación de media y métodos de transmisión instados en documentos NCL deben ser acordes con la ABNT NBR 15606-2:2007, Sección 12.

13 Seguridad

La seguridad en la ejecución de las aplicaciones debe estar de acuerdo con la ABNT NBR 15606-2:2007, Sección 13.

Anexo A
(normativo)

Esquemas de los módulos NCL 3.0 que se utilizan en los perfiles TVD Básico y TVD Avanzado

A.1 Módulo *Structure*: NCL30Structure.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Structure.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the Structure module namespace,
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:structure="http://www.ncl.org.br/NCL3.0/Structure"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Structure"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->
  <!-- define the top-down structure of an NCL language document.          -->
  <!-- = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = -->

  <complexType name="nclPrototype">
    <sequence>
      <element ref="structure:head" minOccurs="0" maxOccurs="1"/>
      <element ref="structure:body" minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="id" type="ID" use="required"/>
    <attribute name="title" type="string" use="optional"/>
  </complexType>

  <complexType name="headPrototype">
  </complexType>

  <complexType name="bodyPrototype">
    <attribute name="id" type="ID" use="optional"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="ncl" type="structure:nclPrototype"/>
  <element name="head" type="structure:headPrototype"/>
  <element name="body" type="structure:bodyPrototype"/>

</schema>
```

A.2 Módulo *Layout*: NCL30Layout.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Layout.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Layout module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:layout="http://www.ncl.org.br/NCL3.0/Layout"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Layout"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <complexType name="regionBasePrototype">
    <attribute name="id" type="ID" use="optional"/>
    <attribute name="device" type="string" use="optional"/>
  </complexType>

  <complexType name="regionPrototype">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="layout:region" />
    </sequence>
    <attribute name="id" type="ID" use="required"/>
    <attribute name="title" type="string" use="optional"/>
    <attribute name="height" type="string" use="optional"/>
    <attribute name="left" type="string" use="optional"/>
    <attribute name="right" type="string" use="optional"/>
    <attribute name="top" type="string" use="optional"/>
    <attribute name="bottom" type="string" use="optional"/>
    <attribute name="width" type="string" use="optional"/>
    <attribute name="zIndex" type="integer" use="optional"/>
  </complexType>


  <!-- declare global attributes in this module -->

  <!-- define the region attributeGroup -->
  <attributeGroup name="regionAttrs">
    <attribute name="region" type="string" use="optional"/>
  </attributeGroup>

  <!-- declare global elements in this module -->
  <element name="regionBase" type="layout:regionBasePrototype"/>
  <element name="region" type="layout:regionPrototype"/>

</schema>
```

A.3 Módulo *Media*: NCL30Media.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Media.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Media module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:media="http://www.ncl.org.br/NCL3.0/Media"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Media"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <complexType name="mediaPrototype">
    <attribute name="id" type="ID" use="required"/>
    <attribute name="type" type="string" use="optional"/>
    <attribute name="src" type="anyURI" use="optional"/>
  </complexType>


  <!-- declare global elements in this module -->
  <element name="media" type="media:mediaPrototype"/>

</schema>
```

A.4 Módulo *Context*: NCL30Context.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Context.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Context module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:context="http://www.ncl.org.br/NCL3.0/Context"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Context"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <!-- define the compositeNode element prototype -->
  <complexType name="contextPrototype">
    <attribute name="id" type="ID" use="required"/>
  </complexType>


  <!-- declare global elements in this module -->
  <element name="context" type="context:contextPrototype"/>

</schema>
```

A.5 Módulo *MediaContentAnchor*: NCL30MediaContentAnchor.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30MediaContentAnchor.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Media Content Anchor module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:mediaAnchor="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"
  targetNamespace="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- define the temporalAnchorAttrs attribute group -->
  <attributeGroup name="temporalAnchorAttrs">
    <attribute name="begin" type="string" use="optional"/>
    <attribute name="end" type="string" use="optional"/>
  </attributeGroup>

  <!-- define the textAnchorAttrs attribute group -->
  <attributeGroup name="textAnchorAttrs">
<attribute name="beginText" type="string" use="optional"/>
<attribute name="beginPosition" type="unsignedLong" use="optional"/>
<attribute name="endText" type="string" use="optional"/>
<attribute name="endPosition" type="unsignedLong" use="optional"/>
</attributeGroup>

<!-- define the sampleAnchorAttrs attribute group -->
<attributeGroup name="sampleAnchorAttrs">
<attribute name="first" type="string" use="optional"/>
<attribute name="last" type="string" use="optional"/>
  </attributeGroup>

  <!-- define the coordsAnchorAttrs attribute group -->
  <attributeGroup name="coordsAnchorAttrs">
    <attribute name="coords" type="string" use="optional"/>
  </attributeGroup>

  <!-- define the labelAttrs attribute group -->
  <attributeGroup name="labelAttrs">
    <attribute name="label" type="string" use="optional"/>
  </attributeGroup>

</attributeGroup>

<!-- define the clipAttrs attribute group -->
<attributeGroup name="clipAttrs">
<attribute name="clip" type="string" use="optional"/>
```

```
</attributeGroup>

<complexType name="componentAnchorPrototype">
<attribute name="id" type="ID" use="required"/>
<attributeGroup ref="mediaAnchor:coordsAnchorAttrs" />
<attributeGroup ref="mediaAnchor:temporalAnchorAttrs" />
<attributeGroup ref="mediaAnchor:textAnchorAttrs" />
<attributeGroup ref="mediaAnchor:sampleAnchorAttrs" />
<attributeGroup ref="mediaAnchor:labelAttrs" />
<attributeGroup ref="mediaAnchor:clipAttrs" />
</complexType>

  <!-- declare global elements in this module -->
  <element name="area" type="mediaAnchor:componentAnchorPrototype"/>

</schema>
```

A.6 Módulo *CompositeNodeInterface*: NC30CompositeNodeInterface.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30CompositeNodeInterface.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Composite Node Interface module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:compositeInterface="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"
  targetNamespace="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <complexType name="compositeNodePortPrototype">
    <attribute name="id" type="ID" use="required"/>
    <attribute name="component" type="IDREF" use="required"/>
    <attribute name="interface" type="string" use="optional" />
  </complexType>


  <!-- declare global elements in this module -->
  <element name="port" type="compositeInterface:compositeNodePortPrototype" />

</schema>
```

A.7 Módulo *PropertyAnchor*: NCL30PropertyAnchor.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30PropertyAnchor.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Property Anchor module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:propertyAnchor="http://www.ncl.org.br/NCL3.0/PropertyAnchor"
  targetNamespace="http://www.ncl.org.br/NCL3.0/PropertyAnchor"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <complexType name="propertyAnchorPrototype">
    <attribute name="name" type="string" use="required" />

    <attribute name="value" type="string" use="optional" />
    <attribute name="externable" type="boolean" use="optional" />
  </complexType>

<!--
The following reserved words are used for properties' names.
* For audio media-obejcts: soundLevel; balanceLevel; trebleLevel; bassLevel.
* For text media-objects: style, which refers to a style sheet with information for text presentation; textAlign;
fontColor; fontFamily; fontStyle; fontSize; fontVariant; fontWeight.
* For visual media-objects: background, specifying the background color used to fill the area of a region displaying
media; scroll, which allows the specification of how an author would like to configure the scroll in a region; fit,
indicating how an object will be presented (hidden, fill, meet, meetBest, slice); transparency, indicating the degree
of transparency of an object presentation (the value shall be between 0 and 1, or a real value in the range [0,100]
ending with the character "%" (e.g. 30%)); visible, indicating if the presentation is to be seen or hidden;
rgbChromakey; the object positioning parameters: top, left, bottom, right, width, height, zIndex, plan, location, size
and bounds; the focus movement parameters: moveLeft, moveRight, moveUp, moveDown, focusIndex; the other
related focus parameters: focusBorderColor, selBorderColor, focusBorderWidth, focusBorderTransparency,
focusSrc, and focusSelSrc; the transition parameters: transIn and transOut; the timing parameters: explicitDur and
freeze; and the multiple device parameters: baseDeviceRegion and deviceClass.
* For media-objects in general: player; reusePlayer, which determines if a new player shall be instantiated or if a
player already instantiated shall be used; and playerLife, which specifies what will happen to the player instance at
the end of the presentation.
-->

  <!-- declare global elements in this module -->
  <element name="property" type="propertyAnchor:propertyAnchorPrototype"/>

</schema>
```

A.8 Módulo *SwitchInterface*: NCL30SwitchInterface.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30SwitchInterface.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Switch Interface module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:switchInterface="http://www.ncl.org.br/NCL3.0/SwitchInterface"
  targetNamespace="http://www.ncl.org.br/NCL3.0/SwitchInterface"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <complexType name="mappingPrototype">
    <attribute name="component" type="IDREF" use="required"/>
    <attribute name="interface" type="string" use="optional" />
  </complexType>

  <complexType name="switchPortPrototype">
    <sequence>
      <element ref="switchInterface:mapping" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="id" type="ID" use="required"/>
  </complexType>


  <!-- declare global elements in this module -->
  <element name="mapping" type="switchInterface:mappingPrototype"/>
  <element name="switchPort" type="switchInterface:switchPortPrototype" />

</schema>
```

A.9 Módulo *Descriptor*: NCL30Descriptor.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Descriptor.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Descriptor module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:descriptor="http://www.ncl.org.br/NCL3.0/Descriptor"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Descriptor"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="descriptorParamPrototype">
    <attribute name="name" type="string" use="required" />
    <attribute name="value" type="string" use="required"/>
  </complexType>

  <complexType name="descriptorPrototype">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="descriptor:descriptorParam"/>
    </sequence>
    <attribute name="id" type="ID" use="required"/>
    <attribute name="player" type="string" use="optional"/>
  </complexType>

<!--
Formatters should support the following descriptorParam names.
* For audio players: soundLevel; balanceLevel; trebleLevel; bassLevel.
* For text players: style, which refers to a style sheet with information for text presentation; textAlign; fontColor;
FontFamily; fontStyle; fontSize; fontVariant; fontWeight.
* For visual media players: background, specifying the background color used to fill the area of a region displaying
media; scroll, which allows the specification of how an author would like to configure the scroll in a region; fit,
indicating how an object will be presented (hidden, fill, meet, meetBest, slice); transparency, indicating the degree
of transparency of an object presentation (the value shall be between 0 and 1, or a real value in the range [0,100]
ending with the character “%” (e.g. 30%)); visible, indicating if the presentation is to be seen or hidden;
rgbChromakey; the object positioning parameters: top, left, bottom, right, width, height, zIndex, plan, location, size
and bounds; the focus movement parameters: moveLeft, moveRight, moveUp, moveDown, focusIndex; the other
related focus parameters: focusBorderColor, selBorderColor, focusBorderWidth, focusBorderTransparency,
focusSrc, and focusSelSrc; the transition parameters: transIn and transOut; the timing parameters: explicitDur and
freeze; and the multiple device parameters: baseDeviceRegion and deviceClass.
* For players in general: player; reusePlayer, which determines if a new player shall be instantiated or if a player
already instantiated shall be used; and playerLife, which specifies what will happen to the player instance at the
end of the presentation.
```

```
  </complexType>
  <complexType name="descriptorBasePrototype">
    <attribute name="id" type="ID" use="optional"/>
  <!-- declare global elements in this module -->
  <element name="descriptorParam" type="descriptor:descriptorParamPrototype"/>
  <element name="descriptor" type="descriptor:descriptorPrototype"/>
  <element name="descriptorBase" type="descriptor:descriptorBasePrototype"/>

  <!-- declare global attributes in this module -->
  <attributeGroup name="descriptorAttrs">
    <attribute name="descriptor" type="string" use="optional"/>
  </attributeGroup>

</schema>
```

A.10 Módulo *Linking*: NCL30Linking.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Linking.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Linking module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:linking="http://www.ncl.org.br/NCL3.0/Linking"
 targetNamespace="http://www.ncl.org.br/NCL3.0/Linking"
 elementFormDefault="qualified" attributeFormDefault="unqualified" >


 <complexType name="paramPrototype">
  <attribute name="name" type="string" use="required" />
  <attribute name="value" type="anySimpleType" use="required"/>
 </complexType>

 <complexType name="bindPrototype">
  <sequence minOccurs="0" maxOccurs="unbounded">
   <element ref="linking:bindParam"/>
  </sequence>
  <attribute name="role" type="string" use="required"/>
  <attribute name="component" type="IDREF" use="required"/>
  <attribute name="interface" type="string" use="optional" />
 </complexType>

 <complexType name="linkPrototype">
  <sequence>
   <element ref="linking:linkParam" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="linking:bind" minOccurs="2" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="id" type="ID" use="optional"/>
  <attribute name="xconnector" type="string" use="required"/>
 </complexType>


 <!-- declare global elements in this module -->
 <element name="linkParam" type="linking:paramPrototype"/>
 <element name="bindParam" type="linking:paramPrototype"/>
 <element name="bind" type="linking:bindPrototype" />
 <element name="link" type="linking:linkPrototype" />

</schema>
```

A.11 Módulo *ConnectorCommonPart*: NCL30ConnectorCommonPart.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorCommonPart.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Connector Common Part module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:connectorCommonPart="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


<complexType name="parameterPrototype">
  <attribute name="name" type="string" use="required" />
  <attribute name="type" type="string" use="optional"/>
</complexType>

<simpleType name="eventPrototype">
  <restriction base="string">
    <enumeration value="presentation" />
    <enumeration value="selection" />
    <enumeration value="attribution" />
    <enumeration value="composition" />
  </restriction>
</simpleType>

<simpleType name="logicalOperatorPrototype">
  <restriction base="string">
    <enumeration value="and" />
    <enumeration value="or" />
  </restriction>
</simpleType>

<simpleType name="transitionPrototype">
  <restriction base="string">
    <enumeration value="starts" />
    <enumeration value="stops" />
    <enumeration value="pauses" />
    <enumeration value="resumes" />
    <enumeration value="aborts" />
  </restriction>
</simpleType>

</schema>
```

A.12 Módulo *ConnectorAssessmentExpression*: NCL30ConnectorAssessmentExpression.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorAssessmentExpression.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Connector Assessment Expression module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:connectorAssessmentExpression="http://www.ncl.org.br/NCL3.0/ConnectorAssessmentExpression"
  xmlns:connectorCommonPart="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ConnectorAssessmentExpression"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<!-- import the definitions in the modules namespaces -->
<import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorCommonPart.xsd"/>

<simpleType name="comparatorPrototype">
  <restriction base="string">
    <enumeration value="eq" />
    <enumeration value="ne" />
    <enumeration value="gt" />
    <enumeration value="lt" />
    <enumeration value="gte" />
    <enumeration value="lte" />
  </restriction>
</simpleType>

<simpleType name="attributePrototype">
  <restriction base="string">
    <enumeration value="repeat" />
    <enumeration value="occurrences" />
    <enumeration value="state" />
    <enumeration value="nodeProperty" />
  </restriction>
</simpleType>

<simpleType name="statePrototype">
  <restriction base="string">
    <enumeration value="sleeping" />
    <enumeration value="occurring" />
    <enumeration value="paused" />
  </restriction>
</simpleType>
```

```
<simpleType name="valueUnion">
  <union memberTypes="string connectorAssessmentExpression:stateProtoype"/>
</simpleType>

<complexType name="assessmentStatementPrototype" >
  <sequence>
    <element ref="connectorAssessmentExpression:attributeAssessment"/>
    <choice>
      <element ref="connectorAssessmentExpression:attributeAssessment"/>
      <element ref="connectorAssessmentExpression:valueAssessment"/>
    </choice>
  </sequence>
  <attribute name="comparator" type="connectorAssessmentExpression:comparatorPrototype" use="required"/>
</complexType>

<complexType name="attributeAssessmentPrototype">
  <attribute name="role" type="string" use="required"/>
  <attribute name="eventType" type="connectorCommonPart:eventPrototype" use="required"/>
  <attribute name="key" type="string" use="optional"/>
  <attribute name="attributeType" type="connectorAssessmentExpression:attributePrototype" use="optional"/>
  <attribute name="offset" type="string" use="optional"/>
</complexType>

<complexType name="valueAssessmentPrototype">
  <attribute name="value" type="connectorAssessmentExpression:valueUnion" use="required"/>
</complexType>

<complexType name="compoundStatementPrototype">
  <choice minOccurs="1" maxOccurs="unbounded">
    <element ref="connectorAssessmentExpression:assessmentStatement" />
    <element ref="connectorAssessmentExpression:compoundStatement" />
  </choice>
  <attribute name="operator" type="connectorCommonPart:logicalOperatorPrototype" use="required"/>
  <attribute name="isNegated" type="boolean" use="optional"/>
</complexType>

  <!-- declare global elements in this module -->
<element name="assessmentStatement" type="connectorAssessmentExpression:assessmentStatementPrototype"
/>
<element name="attributeAssessment" type="connectorAssessmentExpression:attributeAssessmentPrototype" />
<element name="valueAssessment" type="connectorAssessmentExpression:valueAssessmentPrototype" />
<element name="compoundStatement" type="connectorAssessmentExpression:compoundStatementPrototype" />

</schema>
```

A.13 Módulo *ConnectorCausalExpression*: NCL30ConnectorCausalExpression.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorCausalExpression.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Connector Causal Expression module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:connectorCausalExpression="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression"
    xmlns:connectorCommonPart="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<!-- import the definitions in the modules namespaces -->
<import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorCommonPart.xsd"/>

<simpleType name="conditionRoleUnion">
  <union memberTypes="string connectorCausalExpression:conditionRolePrototype"/>
</simpleType>

<simpleType name="conditionRolePrototype">
  <restriction base="string">
    <enumeration value="onBegin" />
    <enumeration value="onEnd" />
    <enumeration value="onPause" />
    <enumeration value="onResume" />
    <enumeration value="onAbort" />
  </restriction>
</simpleType>

<simpleType name="maxUnion">
  <union memberTypes="positiveInteger connectorCausalExpression:unboundedString"/>
</simpleType>

<simpleType name="unboundedString">
  <restriction base="string">
    <pattern value="unbounded"/>
  </restriction>
</simpleType>

<complexType name="simpleConditionPrototype">
  <attribute name="role" type="connectorCausalExpression:conditionRoleUnion" use="required"/>
  <attribute name="eventType" type="connectorCommonPart:eventPrototype" use="optional"/>
  <attribute name="key" type="string" use="optional"/>
  <attribute name="transition" type="connectorCommonPart:transitionPrototype" use="optional"/>
  <attribute name="delay" type="string" use="optional"/>
```

```
  <attribute name="min" type="positiveInteger" use="optional"/>
  <attribute name="max" type="connectorCausalExpression:maxUnion" use="optional"/>
  <attribute name="qualifier" type="connectorCommonPart:logicalOperatorPrototype" use="optional"/>
</complexType>

<complexType name="compoundConditionPrototype">
  <attribute name="operator" type="connectorCommonPart:logicalOperatorPrototype" use="required"/>
  <attribute name="delay" type="string" use="optional"/>
</complexType>

<simpleType name="actionRoleUnion">
  <union memberTypes="string connectorCausalExpression:actionNamePrototype"/>
</simpleType>

<simpleType name="actionNamePrototype">
  <restriction base="string">
    <enumeration value="start" />
    <enumeration value="stop" />
    <enumeration value="pause" />
    <enumeration value="resume" />
    <enumeration value="abort" />
    <enumeration value="set" />
  </restriction>
</simpleType>

<simpleType name="actionOperatorPrototype">
  <restriction base="string">
    <enumeration value="par" />
    <enumeration value="seq" />
  </restriction>
</simpleType>

<complexType name="simpleActionPrototype">
  <attribute name="role" type="connectorCausalExpression:actionRoleUnion" use="required"/>
  <attribute name="eventType" type="connectorCommonPart:eventPrototype" use="optional"/>
  <attribute name="actionType" type="connectorCausalExpression:actionNamePrototype" use="optional"/>
  <attribute name="delay" type="string" use="optional"/>
  <attribute name="value" type="string" use="optional" />
  <attribute name="repeat" type="positiveInteger" use="optional"/>
  <attribute name="repeatDelay" type="string" use="optional"/>
  <attribute name="min" type="positiveInteger" use="optional"/>
  <attribute name="max" type="connectorCausalExpression:maxUnion" use="optional"/>
  <attribute name="qualifier" type="connectorCausalExpression:actionOperatorPrototype" use="optional"/>
</complexType>

<complexType name="compoundActionPrototype">
  <choice minOccurs="2" maxOccurs="unbounded">
    <element ref="connectorCausalExpression:simpleAction" />
    <element ref="connectorCausalExpression:compoundAction" />
  </choice>
  <attribute name="operator" type="connectorCausalExpression:actionOperatorPrototype" use="required"/>
  <attribute name="delay" type="string" use="optional"/>
</complexType>

  <!-- declare global elements in this module -->
<element name="simpleCondition" type="connectorCausalExpression:simpleConditionPrototype" />
```

```
<element name="compoundCondition" type="connectorCausalExpression:compoundConditionPrototype" />
<element name="simpleAction" type="connectorCausalExpression:simpleActionPrototype" />
<element name="compoundAction" type="connectorCausalExpression:compoundActionPrototype" />

</schema>
```

A.14 Módulo *CausalConnector*: NCL30CausalConnector.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30CausalConnector.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Causal Connector module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:causalConnector="http://www.ncl.org.br/NCL3.0/CausalConnector"
  targetNamespace="http://www.ncl.org.br/NCL3.0/CausalConnector"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<complexType name="causalConnectorPrototype">
  <attribute name="id" type="ID" use="required"/>
</complexType>

  <!-- declare global elements in this module -->
<element name="causalConnector" type="causalConnector:causalConnectorPrototype"/>
</schema>
```

A.15 Módulo *ConnectorBase*: NCL30ConnectorBase.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorBase.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Connector Base module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:connectorBase="http://www.ncl.org.br/NCL3.0/ConnectorBase"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ConnectorBase"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<complexType name="connectorBasePrototype">
  <attribute name="id" type="ID" use="optional"/>
</complexType>

  <!-- declare global elements in this module -->
<element name="connectorBase" type="connectorBase:connectorBasePrototype"/>
</schema>
```

A.16 NCL30CausalConnectorFunctionality.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/
NCL30CausalConnectorFunctionality.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL CausalConnectorFunctionality module namespace.
-->

<schema xmlns="http://www.w3.org/2001/XMLSchema"
       xmlns:connectorCommonPart="http://www.ncl.org.br/NCL3.0/
ConnectorCommonPart"
       xmlns:connectorAssessmentExpression="http://www.ncl.org.br/NCL3.0/
ConnectorAssessmentExpression"
       xmlns:connectorCausalExpression="http://www.ncl.org.br/NCL3.0/
ConnectorCausalExpression"
       xmlns:causalConnector="http://www.ncl.org.br/NCL3.0/
CausalConnector"
       xmlns:causalConnectorFunctionality="http://www.ncl.org.br/NCL3.0/
CausalConnectorFunctionality"
       targetNamespace="http://www.ncl.org.br/NCL3.0/
CausalConnectorFunctionality"
       elementFormDefault="qualified" attributeFormDefault="unqualified" >

   <!-- import the definitions in the modules namespaces -->

   <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
   schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
NCL30ConnectorCommonPart.xsd"/>
   <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorAssessmentExpression"
   schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
NCL30ConnectorAssessmentExpression.xsd"/>
   <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression"
   schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
NCL30ConnectorCausalExpression.xsd"/>
   <import namespace="http://www.ncl.org.br/NCL3.0/CausalConnector"
   schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
NCL30CausalConnector.xsd"/>

  <!-- ========================================================== -->
  <!-- CausalConnectorFunctionality -->
  <!-- ========================================================== -->
   <element name="connectorParam" type="connectorCommonPart:parameterPrototype"/>

    <!-- extends causalConnector element -->

    <complexType name="causalConnectorType">
```

```
    <complexContent>
     <extension base="causalConnector:causalConnectorPrototype">
      <sequence>
       <element ref="causalConnectorFunctionality:connectorParam" minOccurs="0" maxOccurs="unbounded"/>
       <choice>
        <element ref="causalConnectorFunctionality:simpleCondition" />
        <element ref="causalConnectorFunctionality:compoundCondition" />
       </choice>
       <choice>
        <element ref="causalConnectorFunctionality:simpleAction" />
        <element ref="causalConnectorFunctionality:compoundAction" />
       </choice>
      </sequence>
     </extension>
    </complexContent>
   </complexType>

   <!-- extends compoundCondition element -->

   <complexType name="compoundConditionType">
    <complexContent>
     <extension base="connectorCausalExpression:compoundConditionPrototype">
      <sequence>
       <choice>
        <element ref="causalConnectorFunctionality:simpleCondition" />
        <element ref="causalConnectorFunctionality:compoundCondition" />
       </choice>
       <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="causalConnectorFunctionality:simpleCondition" />
        <element ref="causalConnectorFunctionality:compoundCondition" />
        <element ref="causalConnectorFunctionality:assessmentStatement" />
        <element ref="causalConnectorFunctionality:compoundStatement" />
       </choice>
      </sequence>
     </extension>
    </complexContent>
   </complexType>

  <element name="causalConnector" type="causalConnectorFunctionality:causalConnectorType"
substitutionGroup="causalConnector:causalConnector"/>

  <element name="simpleCondition" substitutionGroup="connectorCausalExpression:simpleCondition"/>

  <element name="compoundCondition" type="causalConnectorFunctionality:compoundConditionType"
substitutionGroup="connectorCausalExpression:compoundCondition"/>

  <element name="simpleAction" substitutionGroup="connectorCausalExpression:simpleAction"/>

  <element name="compoundAction" substitutionGroup="connectorCausalExpression:compoundAction"/>

  <element name="assessmentStatement"
substitutionGroup="connectorAssessmentExpression:assessmentStatement"/>

  <element name="attributeAssessment"
substitutionGroup="connectorAssessmentExpression:attributeAssessment"/>
```

```
  <element name="valueAssessment" substitutionGroup="connectorAssessmentExpression:valueAssessment"/>

  <element name="compoundStatement"
substitutionGroup="connectorAssessmentExpression:compoundStatement"/>

</schema>
```

A.17 Módulo *TestRule*: NCL30TestRule.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30TestRule.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL TestRule module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:testRule="http://www.ncl.org.br/NCL3.0/TestRule"
  targetNamespace="http://www.ncl.org.br/NCL3.0/TestRule"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <complexType name="rulePrototype">
<attribute name="id" type="ID" use="optional"/>
    <attribute name="var" type="string" use="required"/>
    <attribute name="value" type="string" use="required"/>
    <attribute name="comparator" use="required">
      <simpleType>
        <restriction base="string">
          <enumeration value="eq" />
          <enumeration value="ne" />
          <enumeration value="gt" />
          <enumeration value="gte" />
          <enumeration value="lt" />
          <enumeration value="lte" />
        </restriction>
      </simpleType>
    </attribute>
  </complexType>

  <complexType name="compositeRulePrototype">
    <choice minOccurs="2" maxOccurs="unbounded">
      <element ref="testRule:rule"/>
      <element ref="testRule:compositeRule"/>
    </choice>
    <attribute name="id" type="ID" use="required"/>
    <attribute name="operator" use="required">
      <simpleType>
        <restriction base="string">
          <enumeration value="and" />
          <enumeration value="or" />
        </restriction>
      </simpleType>
    </attribute>
  </complexType>
```

```
  <complexType name="ruleBasePrototype">
    <attribute name="id" type="ID" use="optional"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="rule" type="testRule:rulePrototype"/>
  <element name="compositeRule" type="testRule:compositeRulePrototype"/>
  <element name="ruleBase" type="testRule:ruleBasePrototype"/>

</schema>
```

A.18 Módulo *TestRuleUse*: NCL30TestRuleUse.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30TestRuleUse.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL TestRuleUse module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:testRule="http://www.ncl.org.br/NCL3.0/TestRuleUse"
  targetNamespace="http://www.ncl.org.br/NCL3.0/TestRuleUse"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <complexType name="bindRulePrototype">
    <attribute name="constituent" type="IDREF" use="required" />
    <attribute name="rule" type="string" use="required" />
  </complexType>


  <!-- declare global elements in this module -->
  <element name="bindRule" type="testRule:bindRulePrototype"/>

</schema>
```

A.19 Módulo *ContentControl*: NCL30ContentControl.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ContentControl.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL ContentControl module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:contentControl="http://www.ncl.org.br/NCL3.0/ContentControl"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ContentControl"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <complexType name="defaultComponentPrototype">
    <attribute name="component" type="IDREF" use="required"/>
  </complexType>

  <!-- define the switch element prototype -->

  <complexType name="switchPrototype">
    <choice>
    <element ref="contentControl:defaultComponent" minOccurs="0" maxOccurs="1"/>
    </choice>
    <attribute name="id" type="ID" use="required"/>
  </complexType>


  <!-- declare global elements in this module -->
  <element name="defaultComponent" type="contentControl:defaultComponentPrototype"/>
  <element name="switch" type="contentControl:switchPrototype"/>

</schema>
```

A.20 Módulo *DescriptorControl*: NCL30DescriptorControl.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30DescriptorControl.xsd
Author: TeleMidia Laboratory
Revision: 19/06/2006

Schema for the NCL DescriptorControl module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:descriptorControl="http://www.ncl.org.br/NCL3.0/DescriptorControl"
  targetNamespace="http://www.ncl.org.br/NCL3.0/DescriptorControl"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <complexType name="defaultDescriptorPrototype">
    <attribute name="descriptor" type="IDREF" use="required" />
  </complexType>

  <!-- define the descriptor switch element prototype -->
  <complexType name="descriptorSwitchPrototype">
    <choice>
    <element ref="descriptorControl:defaultDescriptor" minOccurs="0" maxOccurs="1"/>
    </choice>
    <attribute name="id" type="ID" use="required”/>
  </complexType>


  <!-- declare global elements in this module -->
  <element name="defaultDescriptor" type="descriptorControl:defaultDescriptorPrototype"/>
  <element name="descriptorSwitch" type="descriptorControl:descriptorSwitchPrototype"/>

</schema>
```

A.21 Módulo *Timing*: NCL30Timing.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Timing.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Timing module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:timing="http://www.ncl.org.br/NCL3.0/Timing"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Timing"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <!-- declare global attributes in this module -->

  <!-- define the explicitDur attribute group -->
  <attributeGroup name="explicitDurAttrs">
    <attribute name="explicitDur" type="string" use="optional"/>
  </attributeGroup>

  <!-- define the freeze attribute group -->
  <attributeGroup name="freezeAttrs">
    <attribute name="freeze" type="boolean" use="optional"/>
  </attributeGroup>

</schema>
```

A.22 Módulo *Import*: NCL30Import.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Import.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Import module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:import="http://www.ncl.org.br/NCL3.0/Import"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Import"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<complexType name="importBasePrototype">
<attribute name="alias" type="ID" use="required"/>
<attribute name="region" type="IDREF" use="optional"/>
<attribute name="documentURI" type="anyURI" use="required"/>
<attribute name="baseId" type="IDREF" use="optional"/>
</complexType>

  <complexType name="importNCLPrototype">
     <attribute name="alias" type="ID" use="required"/>
     <attribute name="documentURI" type="anyURI" use="required"/>
  </complexType>

  <complexType name="importedDocumentBasePrototype">
    <sequence minOccurs="1" maxOccurs="unbounded">
      <element ref="import:importNCL" />
    </sequence>
    <attribute name="id" type="ID" use="optional"/>
  </complexType>


  <!-- declare global elements in this module -->
  <element name="importBase" type="import:importBasePrototype"/>
  <element name="importNCL" type="import:importNCLPrototype"/>
  <element name="importedDocumentBase" type="import:importedDocumentBasePrototype"/>

</schema>
```

A.23 Módulo *EntityReuse*: NCL30EntityReuse.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30EntityReuse.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL EntityReuse module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:entityReuse="http://www.ncl.org.br/NCL3.0/EntityReuse"
  targetNamespace="http://www.ncl.org.br/NCL3.0/EntityReuse"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <attributeGroup name="entityReuseAttrs">
    <attribute name="refer" type="string" use="optional"/>
  </attributeGroup>

</schema>
```

A.24 Módulo *ExtendedEntityReuse*: NCL30ExtendedEntityReuse.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ExtendedEntityReuse.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL ExtendedEntityReuse module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:extendedEntityReuse="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <attributeGroup name="extendedEntityReuseAttrs">
    <attribute name="instance" type="boolean" use="optional"/>
  </attributeGroup>

</schema>
```

A.25 Módulo *KeyNavigation*: NCL30KeyNavigation.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30KeyNavigation.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL KeyNavigation module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:keyNavigation="http://www.ncl.org.br/NCL3.0/KeyNavigation"
  targetNamespace="http://www.ncl.org.br/NCL3.0/KeyNavigation"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<simpleType name="colorPrototype">
  <restriction base="string">
    <enumeration value="white" />
    <enumeration value="black" />
    <enumeration value="silver" />
    <enumeration value="gray" />
    <enumeration value="red" />
    <enumeration value="maroon" />
    <enumeration value="fuchsia" />
    <enumeration value="purple" />
    <enumeration value="lime" />
    <enumeration value="green" />
    <enumeration value="yellow" />
    <enumeration value="olive" />
    <enumeration value="blue" />
    <enumeration value="navy" />
    <enumeration value="aqua" />
    <enumeration value="teal" />
  </restriction>
</simpleType>

  <!-- declare global attributes in this module -->

  <!-- define the keyNavigation attribute group -->
  <attributeGroup name="keyNavigationAttrs">
    <attribute name="moveLeft" type="IDREF" use="optional"/>
    <attribute name="moveRight" type="IDREF" use="optional"/>
    <attribute name="moveUp" type="IDREF" use="optional"/>
    <attribute name="moveDown" type="IDREF" use="optional"/>
    <attribute name="focusIndex" type="IDREF" use="optional"/>
    <attribute name="focusBorderColor" type="keyNavigation:colorPrototype" use="optional"/>
    <attribute name="focusBorderWidth" type="string" use="optional"/>
    <attribute name="focusBorderTransparency" type="string" use="optional"/>
    <attribute name="focusScr" type="string" use="optional"/>
    <attribute name="focusSelScr" type="string" use="optional"/>
    <attribute name="selBorderColor" type="keyNavigation:colorPrototype" use="optional"/>
  </attributeGroup>

</schema>
```

A.26 Módulo *TransitionBase*: NCL30TransitionBase.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30TransitionBase.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Transition Base module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:transitionBase="http://www.ncl.org.br/NCL3.0/TransitionBase"
  targetNamespace="http://www.ncl.org.br/NCL3.0/TransitionBase"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<complexType name="transitionBasePrototype">
  <attribute name="id" type="ID" use="optional"/>
</complexType>

  <!-- declare global elements in this module -->
<element name="transitionBase" type="transitionBase:transitionBasePrototype"/>
</schema>
```

A.27 Módulo *Animation*: NCL30Animation.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Animation.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Timing module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:animation="http://www.ncl.org.br/NCL3.0/Animation"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Animation"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <!-- declare global attributes in this module -->

  <!-- define the animation attribute group -->
  <attributeGroup name="animationAttrs">
    <attribute name="duration" type="string" use="optional"/>
    <attribute name="by" type="string" use="optional"/>
  </attributeGroup>

</schema>
```

A.28 Transition module: NCL30Transition.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Transition.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Transition module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:transition="http://www.ncl.org.br/NCL3.0/Transition"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Transition"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >


  <!-- declare global attributes in this module -->

  <!-- define the type attribute prototype -->
  <simpleType name="typePrototype">
    <restriction base="string">
      <enumeration value="in"/>
      <enumeration value="barWipe"/>
      <enumeration value="boxWipe"/>
      <enumeration value="fourBoxWipe"/>
      <enumeration value="barnDoorWipe"/>
      <enumeration value="diagonalWipe"/>
      <enumeration value="bowTieWipe"/>
      <enumeration value="miscDiagonalWipe"/>
      <enumeration value="veeWipe"/>
      <enumeration value="barnVeeWipe"/>
      <enumeration value="zigZagWipe"/>
      <enumeration value="barnZigZagWipe"/>
      <enumeration value="irisWipe"/>
      <enumeration value="triangleWipe"/>
      <enumeration value="arrowHeadWipe"/>
      <enumeration value="pentagonWipe"/>
      <enumeration value="hexagonWipe"/>
      <enumeration value="ellipseWipe"/>
      <enumeration value="eyeWipe"/>
      <enumeration value="roundRectWipe"/>
      <enumeration value="starWipe"/>
      <enumeration value="miscShapeWipe"/>
      <enumeration value="clockWipe"/>
      <enumeration value="pinWheelWipe"/>
      <enumeration value="singleSweepWipe"/>
      <enumeration value="fanWipe"/>
      <enumeration value="doubleFanWipe"/>
      <enumeration value="doubleSweepWipe"/>
      <enumeration value="saloonDoorWipe"/>
      <enumeration value="windshieldWipe"/>
      <enumeration value="snakeWipe"/>
      <enumeration value="spiralWipe"/>
      <enumeration value="parallelSnakesWipe"/>
      <enumeration value="boxSnakesWipe"/>
      <enumeration value="waterfallWipe"/>
      <enumeration value="pushWipe"/>
      <enumeration value="slideWipe"/>
      <enumeration value="fade"/>
```

```
      <enumeration value="audioFade"/>
      <enumeration value="audioVisualFade"/>
    </restriction>
  </simpleType>

  <!-- define subType attribute prototype-->
  <simpleType name="subTypePrototype">
    <restriction base="string">
      <enumeration value="bottom"/>
      <enumeration value="bottomCenter"/>
      <enumeration value="bottomLeft"/>
      <enumeration value="bottomLeftClockwise"/>
      <enumeration value="bottomLeftCounterClockwise"/>
      <enumeration value="bottomLeftDiagonal"/>
      <enumeration value="bottomRight"/>
      <enumeration value="bottomRightClockwise"/>
      <enumeration value="bottomRightCounterClockwise"/>
      <enumeration value="bottomRightDiagonal"/>
      <enumeration value="centerRight"/>
      <enumeration value="centerTop"/>
      <enumeration value="circle"/>
      <enumeration value="clockwiseBottom"/>
      <enumeration value="clockwiseBottomRight"/>
      <enumeration value="clockwiseLeft"/>
      <enumeration value="clockwiseNine"/>
      <enumeration value="clockwiseRight"/>
      <enumeration value="clockwiseSix"/>
      <enumeration value="clockwiseThree"/>
      <enumeration value="clockwiseTop"/>
      <enumeration value="clockwiseTopLeft"/>
      <enumeration value="clockwiseTwelve"/>
      <enumeration value="cornersIn"/>
      <enumeration value="cornersOut"/>
      <enumeration value="counterClockwiseBottomLeft"/>
      <enumeration value="counterClockwiseTopRight"/>
      <enumeration value="crossfade"/>
      <enumeration value="diagonalBottomLeft"/>
      <enumeration value="diagonalBottomLeftOpposite"/>
      <enumeration value="diagonalTopLeft"/>
      <enumeration value="diagonalTopLeftOpposite"/>
      <enumeration value="diamond"/>
      <enumeration value="doubleBarnDoor"/>
      <enumeration value="doubleDiamond"/>
      <enumeration value="down"/>
      <enumeration value="fadeFromColor"/>
      <enumeration value="fadeToColor"/>
      <enumeration value="fanInHorizontal"/>
      <enumeration value="fanInVertical"/>
      <enumeration value="fanOutHorizontal"/>
      <enumeration value="fanOutVertical"/>
      <enumeration value="fivePoint"/>
      <enumeration value="fourBlade"/>
      <enumeration value="fourBoxHorizontal"/>
      <enumeration value="fourBoxVertical"/>
      <enumeration value="fourPoint"/>
      <enumeration value="fromBottom"/>
      <enumeration value="fromLeft"/>
      <enumeration value="fromRight"/>
      <enumeration value="fromTop"/>
      <enumeration value="heart"/>
      <enumeration value="horizontal"/>
      <enumeration value="horizontalLeft"/>
      <enumeration value="horizontalLeftSame"/>
      <enumeration value="horizontalRight"/>
      <enumeration value="horizontalRightSame"/>
```

```
      <enumeration value="horizontalTopLeftOpposite"/>
      <enumeration value="horizontalTopRightOpposite"/>
      <enumeration value="keyhole"/>
      <enumeration value="left"/>
      <enumeration value="leftCenter"/>
      <enumeration value="leftToRight"/>
      <enumeration value="oppositeHorizontal"/>
      <enumeration value="oppositeVertical"/>
      <enumeration value="parallelDiagonal"/>
      <enumeration value="parallelDiagonalBottomLeft"/>
      <enumeration value="parallelDiagonalTopLeft"/>
      <enumeration value="parallelVertical"/>
      <enumeration value="rectangle"/>
      <enumeration value="right"/>
      <enumeration value="rightCenter"/>
      <enumeration value="sixPoint"/>
      <enumeration value="top"/>
      <enumeration value="topCenter"/>
      <enumeration value="topLeft"/>
      <enumeration value="topLeftClockwise"/>
      <enumeration value="topLeftCounterClockwise"/>
      <enumeration value="topLeftDiagonal"/>
      <enumeration value="topLeftHorizontal"/>
      <enumeration value="topLeftVertical"/>
      <enumeration value="topRight"/>
      <enumeration value="topRightClockwise"/>
      <enumeration value="topRightCounterClockwise"/>
      <enumeration value="topRightDiagonal"/>
      <enumeration value="topToBottom"/>
      <enumeration value="twoBladeHorizontal"/>
      <enumeration value="twoBladeVertical"/>
      <enumeration value="twoBoxBottom"/>
      <enumeration value="twoBoxLeft"/>
      <enumeration value="twoBoxRight"/>
      <enumeration value="twoBoxTop"/>
      <enumeration value="up"/>
      <enumeration value="vertical"/>
      <enumeration value="verticalBottomLeftOpposite"/>
      <enumeration value="verticalBottomSame"/>
      <enumeration value="verticalLeft"/>
      <enumeration value="verticalRight"/>
      <enumeration value="verticalTopLeftOpposite"/>
      <enumeration value="verticalTopSame"/>
    </restriction>
  </simpleType>

  <attributeGroup name="transAttrs">
    <attribute name="transIn" type="string" use="optional"/>
    <attribute name="transOut" type="string" use="optional"/>
  </attributeGroup>

  <!-- define the transition attribute group -->
  <attributeGroup name="transitionAttrs">
    <attribute name="type" type="transition:typePrototype" use="required"/>
    <attribute name="subtype" type="transition:subTypePrototype" use="optional"/>
    <attribute name="fadecolor" type="string" use="optional" default="black"/>
    <attribute name="dur" type="string" use="optional"/>
    <attribute name="startProgress" use="optional" default="0.0">
      <simpleType>
        <restriction base="decimal">
          <minInclusive value="0.0"/>
          <maxInclusive value="1.0"/>
        </restriction>
      </simpleType>
    </attribute>
```

```
    <attribute name="endProgress" use="optional" default="1.0">
      <simpleType>
        <restriction base="decimal">
          <minInclusive value="0.0"/>
          <maxInclusive value="1.0"/>
        </restriction>
      </simpleType>
    </attribute>
    <attribute name="direction" use="optional" default="forward">
      <simpleType>
        <restriction base="string">
          <enumeration value="forward"/>
          <enumeration value="reverse"/>
        </restriction>
      </simpleType>
    </attribute>
  </attributeGroup>

  <!-- define the transition-modifier attribute group -->
  <attributeGroup name="transitionModifierAttrs">
    <attribute name="horzRepeat" type="decimal" use="optional"  default="1.0"/>
    <attribute name="vertRepeat" type="decimal" use="optional"  default="1.0"/>
    <attribute name="borderWidth" type="nonNegativeInteger" use="optional"  default="0"/>
    <attribute name="borderColor" type="string" use="optional"  default="black"/>
  </attributeGroup>

  <complexType name="transitionPrototype">
    <attributeGroup ref="transition:transitionAttrs"/>
    <attributeGroup ref="transition:transitionModifierAttrs"/>
  </complexType>

  <!-- declare global element in this module -->
  <element name="transition" type="transition:transitionPrototype"/>

</schema>
```

A.29 Metainformation module: NCL30Metainformation.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Metainformation.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Metainformation module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:metainformation="http://www.ncl.org.br/NCL3.0/Metainformation"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Metainformation"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="metaPrototype">
    <attribute name="name" type="string" use="required"/>
    <attribute name="content" type="string" use="required"/>
  </complexType>

  <complexType name="metadataPrototype">
     <sequence>
          <any minOccurs="0"/>
     </sequence>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="meta" type="metainformation:metaPrototype"/>

  <!-- declare global elements in this module -->
  <element name="metadata" type="metainformation:metadataPrototype"/>

</schema>
```

Bibliografía

[1] ISO 8859-1, *Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet N° 1*

[2] ISO/IEC 13818-6:2001, *Information technology - Generic coding of moving pictures and associated audio information - Part 6: Extensions for DSM-CC*

[3] ITU Recommendation J.201:2004, *Harmonization of declarative content format for interactive television applications*

[4] ARIB STD-B24:2004, *Data coding and transmission specifications for digital broadcasting*

[5] ACAP, Advanced Application Platform (ACAP), ATSC Standard: Document A/101. Agosto de 2005

[6] Cascading Style Sheets, Cascading Style Sheets, level 2, Bert Bos, Håkon Wium Lie, Chris Lilley, Ian Jacobs. W3C Recommendation 12. Mayo de 1998, disponible en <http://www.w3.org/TR/REC-CSS2>

[7] DVB-HTML, Perrot P. DVB-HTML - An Optional Declarative Language within MHP 1.1, EBU Technical Review. 2001

[8] Namespaces in XML, Namespaces in XML, W3C Recommendation. Janeiro de 1999

[9] NCM Core, Soares L.F.G; Rodrigues R.F. Nested Context Model 3.0: Part 1 – NCM Core, Technical Report, Departamento de Informática PUC-Rio. Maio de 2005, ISSN: 0103-9741. También disponible en <http://www.ncl.org.br>

[10] NCL Digital TV Profiles, Soares L.F.G; Rodrigues R.F. Part 8 – NCL (Nested Context Language) Digital TV Profiles, Technical Report, Departamento de Informática PUC-Rio, No. 35/06. Outubro de 2006, ISSN: 0103-9741. También disponible en <http://www.ncl.org.br>

[11] NCL Live Editing Commands, Soares L.F.G; Rodrigues R.F; Costa, R.R.; Moreno, M.F. Part 9 – NCL Live Editing Commands. Technical Report, Departamento de Informática PUC-Rio, No. 36/06. Dezembro de 2006, ISSN: 0103-9741. También disponible en <http://www.ncl.org.br>

[12] NCL-Lua, Cerqueira, R,; Sant'Anna, F. Nested Context Model 3.0: Part 11 – Lua Scripting Language for NCL, Technical Report, Departamento de Informática PUC-Rio. Maio de 2007, ISSN: 0103-9741.

[13] NCL Main Profile, Soares L.F.G; Rodrigues R.F; Costa, R.R. Nested Context Model 3.0: Part 6 – NCL (Nested Context Language) Main Profile, Technical Report, Departamento de Informática PUC-Rio. Maio de 2005, ISSN: 0103-9741. También disponible en <http://www.ncl.org.br>

[14] RDF, Resource Description Framework (RDF) Model and Syntax Specification, Ora Lassila and Ralph R. Swick. W3C Recommendation. 22 de fevereiro de 1999. Disponible en <http://www.w3.org/TR/REC-rdf-syntax/>

[15] SMIL 2.1 Specification, SMIL 2.1 - *Synchronized Multimedia Integration Language – SMIL 2.1 Specification,* W3C Recommendation. Dezembro de 2005

[16] XHTML 1.0, XHTML™ 1.0 2º Edition - Extensible HyperText Markup Language, W3C Recommendation, Agosto de 2002

[17] Programming in Lua, Segunda Edição, de Roberto Ierusalimschy et al. Março de 2006, ISBN 85-903798-2-5.

