EHIZTARI

Propuesta de esquema XML

Autor: Luis Javier Rodríguez Fuentes
Fecha: 20 de noviembre de 2006

1 Comentarios preliminares

Aunque las informaciones de nuestro índice no sean síncronas, es decir, aunque no se contengan jerárquicamente unas a otras, es posible definir un único esquema XML para todos los niveles de conocimiento, gracias a lo que se conoce como marcado "stand-off". El fichero resultante será largo y muy difícil de leer. Sin embargo, un recurso XML no se suele editar ni leer directamente, sino que es explorado mediante herramientas que permiten seleccionar ciertas informaciones (ciertas marcas) y no otras. Además, si fuera necesario, los distintos niveles de información contenidos en un recurso XML producido por Ehiztari podrían separarse y aislarse unos de otros.

Parto, pues, de la hipótesis de que toda la información que extraemos o generamos va a parar al mismo fichero XML. Hasta ahora, el esquema XML-Ehiztari incluía 6 elementos: *resource, source, cache, segments, segment y word*, cada uno con sus atributos, según la jerarquía que se muestra en la Figura 1.

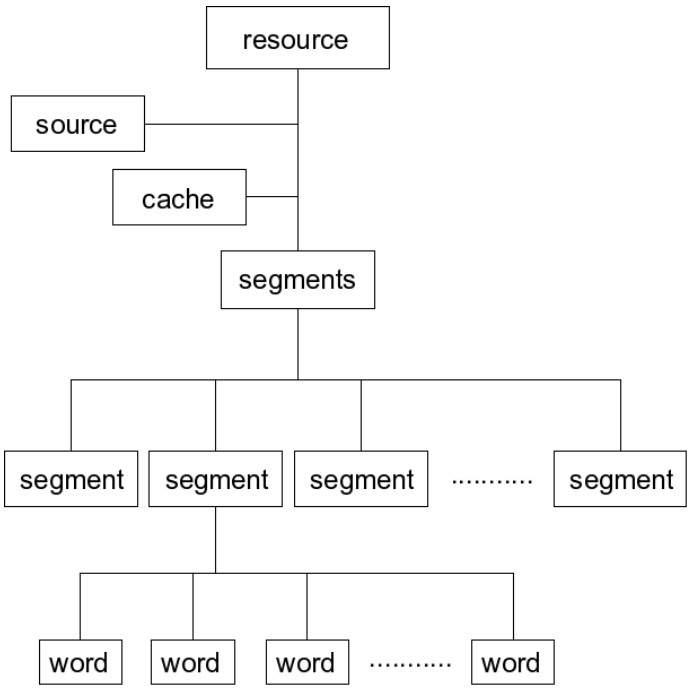

Figura 1. Primera versión de XML-Ehiztari.

A mi juicio, este esquema ha de incluir más información. Yo distinguiría tres bloques: meta-información, información que hace referencia al recurso completo e información relativa a la transcripción. Es en este último bloque donde se van a incluir las marcas y atributos para eventos no lingüísticos.

2 Meta-información

Lo que tratamos de conseguir en este nivel es suministrar información sobre qué persona y/o módulo de procesamiento modifica el recurso XML y cuándo lo hace. Precisamente, el elemento *source* dispone de un atributo, *date*, que permite asignar una fecha al recurso de audio original. Hay cierta ambigüedad sobre cómo debería interpretarse dicha fecha: fecha de creación, fecha del primer acceso, fecha del último acceso/modificación, etc. En cualquier caso, esa fecha no cumple el propósito enunciado más arriba.

Propongo añadir una secuencia de elementos *processor*, cada uno correspondiente a un procesamiento distinto del fichero XML, que incluya el nombre del agente (módulo o persona) que efectúa la modificación (atributo *agent*), la fecha (atributo *date*) y, por último, información colateral (atributo *info*) que podría ser interesante de cara a interpretar los resultados del procesamiento, como, por

ejemplo, el comando aplicado y sus opciones. Esta información nos permitirá disponer de la historia de los datos en el propio fichero XML, y mantener distintas variantes que hayan seguido distintas historias. Además, podría utilizarse para evitar que un cierto módulo se aplique a un fichero XML que ya contiene el nivel de información correspondiente. El esquema XML-Ehiztari quedaría tal como se muestra en la Figura 2.

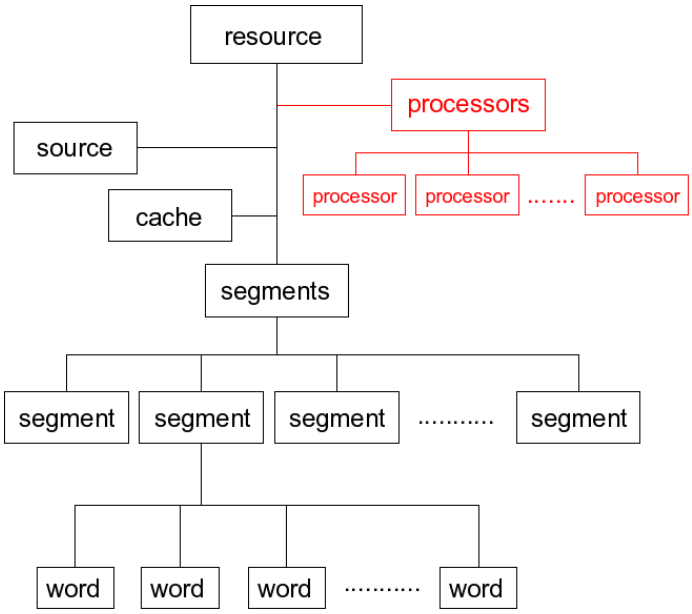

Figura 2. XML-Ehiztari tras añadir meta-información con la historia de procesamientos.

La especificación formal del esquema XML contendría dos nuevos tipos: *processorsType* y *processorType*, y el elemento *resource* habría sido redefinido:

```
<xs:element name="resource">
	<xs:complexType>
		<xs:sequence>
			<xs:element name="processors" type="processorsType"/>
			<xs:element name="source" type="sourceType"/>
			<xs:element name="cache" type="cacheType"/>
			<xs:element name="segments" type="segmentsType"/>
		</xs:sequence>
	</xs:complexType>
</xs:element>
<xs:complexType name="processorType">
	<xs:attribute name="agent" type="xs:string" use="required"/>
	<xs:attribute name="date" type="xs:string" use="required"/>
	<xs:attribute name="info" type="xs:string"/>
</xs:complexType>
<xs:complexType name="processorsType">
	<xs:sequence maxOccurs="unbounded">
		<xs:element name="processor" type="xs:processorType"/>
	</xs:sequence>
</xs:complexType>
```

3 Información general sobre el recurso

En este caso se trata de recoger aquella información que no se refiere a un evento o segmento localizados en el tiempo, sino que afecta a todo el recurso. En esta categoría se incluyen dos de los elementos ya definidos: *source* y *cache*. En mi opinión, *cache* funciona mejor como atributo de *source* que como elemento independiente (Figura 3). Además, será necesario incluir un nuevo atributo *pcm_cache* para señalar la ubicación del audio en formato PCM que se toma como entrada por parte de los distintos módulos de procesamiento.

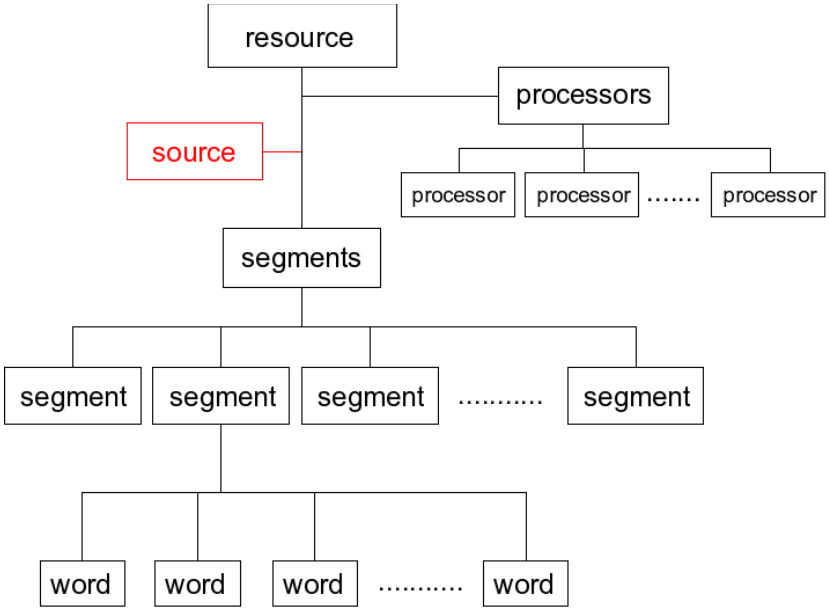

Figura 3. XML-Ehiztari tras la reconversión del elemento *cache* como atributo del elemento *source*.

Por otra parte, el atributo *size* (tamaño en bytes del fichero de audio), que pretende dar una medida de la duración del fichero de audio, es difícilmente interpretable si, por ejemplo, hay varios canales y el audio va en formato *mp3* con una resolución de 48 bits por muestra. Sin eliminar la información sobre el tamaño del fichero, yo añadiría un atributo (*length*) para la duración de la señal en segundos. El mismo problema se planteará más adelante para especificar el inicio (*offset*) y la duración (*length*) de un segmento. En mi opinión, las medidas de tiempo deberíamos especificarlas en segundos. De esta manera, las hacemos independientes de la frecuencia de muestreo. Por último, el elemento *source* debería incluir algunos otros atributos, no sé si obligatorios, para detallar las características de la fuente. En suma, propongo los siguientes atributos:

- ✓ *cache* (copia local del recurso original)
- ✓ *pcm_cache* (fichero de audio en formato PCM extraído del recurso original)
- ✓ *length* (duración de la señal, en segundos)
- ✓ *mode* (modo de interacción): conversation, news, podcast, lecture, unknown
- ✓ *code* (codificación): mulaw, alaw, linear
- ✓ *resolution* (bits por muestra): 8, 16, 32, 48

La especificación formal del tipo *sourceType* requiere además de ciertos cambios para forzar tipos numéricos y enumerados. Quedaría como sigue:

```
<xs:simpleType name="modeType">
      <xs:restriction base="xs:string">
            <xs:pattern value="conversation|news|podcast|lecture|unknown"/>
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="codeType">
      <xs:restriction base="xs:string">
            <xs:pattern value="mulaw|alaw|linear"/>
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="resolutionType">
      <xs:restriction base="xs:positiveInteger">
            <xs:pattern value="8|16|32|48"/>
      </xs:restriction>
</xs:simpleType>
```

```
<xs:complexType name="sourceType">
      <xs:attribute name="url" type="xs:string" use="required"/>
      <xs:attribute name="cache" type="xs:string"/>
      <xs:attribute name="pcm_cache" type="xs:string"/>
      <xs:attribute name="date" type="xs:string" use="required"/>
      <xs:attribute name="size" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="length" type="xs:double" use="required"/>
      <xs:attribute name="format" type="xs:string" use="required"/>
      <xs:attribute name="sampling_rate" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="channels" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="code" type="xs:codeType" use="required"/>
      <xs:attribute name="resolution" type="xs:resolutionType" use="required"/>
      <xs:attribute name="mode" type="xs:modeType" use="required"/>
</xs:complexType>
```

4 Información relativa a la transcripción

4.1 Segmentos

Todos estaremos de acuerdo en que el elemento más pequeño que podemos manejar en el indexado es el ***segmento***, definido como el *fragmento de señal de tamaño máximo producido sin interrupciones apreciables (salvo ruidos aislados) por la misma fuente, en condiciones ambientales y de canal homogéneas*. Esta definición no coincide con la del protocolo de anotación, en el que un segmento es un *fragmento manejable (de 1 a 15 segundos) entre pausas*. Nuestra definición de segmento tampoco coincide con el concepto de *turno* que maneja dicho protocolo (*fragmento de señal de tamaño máximo producido por un mismo hablante*). Aquí tendremos en cuenta las características acústicas (hablante, condiciones ambientales y de canal, etc.) para separar unos segmentos de otros, mientras que el protocolo de anotación sólo tiene en cuenta el hablante. Para especificar las condiciones ambientales y de canal, el protocolo de anotación define un nivel de anotación independiente, que no es síncrono con secciones, turnos y segmentos.

En nuestro esquema cada segmento constará de una secuencia no vacía de ***palabras*** y ***eventos*** (Figura 4). En el caso más simple, un segmento constaría de un único evento de silencio. Los eventos vienen a completar la descripción acústica de los segmentos, aunque en general, desde el punto de vista del indexado, no tendrán demasiada utilidad. Más adelante se describe una propuesta de atributos para palabras y eventos, que tiene en cuenta por un lado la información que proporcionan las anotaciones manuales y por otro los requerimientos de la aplicación de indexado y búsqueda. Los atributos más importantes de un segmento son el inicio (*offset*) y la duración (*length*). También es necesario definir un atributo (*id*) para identificar cada segmento de manera única, porque, como veremos más adelante, otros niveles de conocimiento harán referencia a grupos de segmentos. Precisamente, no vamos a reservar atributos para el contenido acústico del segmento (silencio, voz, música, ruido, etc.), la identidad del hablante o la lengua, porque se trata de informaciones que no se obtienen en primera instancia al segmentar la señal, sino que es necesario postprocesar y categorizar la secuencia de segmentos. En lugar de eso, se establecerá un mecanismo flexible para efectuar agrupamientos de segmentos de acuerdo a distintos criterios, como el contenido acústico, el hablante, la lengua o el tópico, en niveles más altos del esquema, que llamaremos *secciones*. La especificación formal del tipo *segmentType* queda como sigue:

```
<xs:complexType name="segmentType">
      <xs:choice maxOccurs="unbounded">
            <xs:element name="word" type="wordType"/>
            <xs:element name="event" type="eventType"/>
      </xs:choice>
      <xs:attribute name="id" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="offset" type="xs:double" use="required"/>
      <xs:attribute name="length" type="xs:double" use="required"/>
</xs:complexType>
```

Por último, me parece adecuado cambiar el nombre del elemento *segments*, para denominarlo *segmentation*, de modo que la especificación formal del esquema XML quedaría como sigue:

```
<xs:element name="resource">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="processors" type="processorsType"/>
                  <xs:element name="source" type="sourceType"/>
                  <xs:element name="segmentation" type="segmentationType"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:complexType name="segmentationType">
      <xs:sequence maxOccurs="unbounded">
            <xs:element name="segment" type="xs:segmentType"/>
      </xs:sequence>
</xs:complexType>
```

4.2 Secciones

El protocolo de anotación introduce el concepto de ***sección***, que en mi opinión debemos adoptar para distinguir distintos temas, asuntos o tópicos dentro de un recurso, tarea incluida también dentro de Ehiztari. Esto permitirá indexar con un mayor grado de precisión y efectuar búsquedas no solamente por palabras clave sino también por tópicos. Si se plantea una estructura jerárquica estricta, en el caso más sencillo habría una única sección que agruparía a todos los segmentos, sin asignación de tópico o identificando el tópico como desconocido. Esa estructura jerárquica permitiría definir tan sólo un cierto agrupamiento de los segmentos. Sin embargo, podría ser interesante disponer de varios criterios de agrupamiento. Es decir, la idea de estructurar los recursos por secciones de acuerdo a un criterio temático podría generalizarse a cualquier otro criterio, como el contenido acústico, la lengua o el hablante. En este último caso, la agrupación de segmentos consecutivos del mismo hablante equivaldría a identificar turnos (cada sección correspondería a un turno). Para hacer posibles varios seccionamientos, el mismo fichero XML debe contener la especificación de distintos agrupamientos asíncronos, lo que impide una representación jerárquica y nos lleva a una propuesta de tipo "stand-off".

El marcado de tipo "stand-off" utiliza una capa básica como referencia, de modo que varias capas distintas de más alto nivel, independientes entre sí, pueden relacionarse a través de su sincronización con la capa de referencia. En anotación de habla, esta capa suele ser o bien la propia señal, o bien una capa de nivel un poco más alto (la transcripción fonética o la transcripción ortográfica a nivel de palabra) que a su vez estaría sincronizada con la señal. En el marcado de tipo "stand-off" cada capa suele ir físicamente en un fichero XML distinto, aunque también es posible especificar independientemente todas las capas dentro de un único fichero XML. En nuestra propuesta seguiremos esta última aproximación, incluyendo también en el mismo fichero la propia capa de referencia: los segmentos. Por tanto, todas las capas de nivel superior (las secciones) estarán sincronizadas explícitamente con la segmentación, y a su vez, cada segmento estará sincronizado con la señal de audio (que, como es lógico, irá en un fichero aparte).

Todo lo anterior se concreta en tres nuevos elementos: *sectioning*, *section* y *class* (Figura 4). El elemento *sectioning* permite definir un nuevo agrupamiento de segmentos, de acuerdo a un cierto criterio que se describe a través del atributo *criterion*. Cada nuevo agrupamiento deberá constar de al menos una sección. Precisamente, el elemento *section* sirve para agrupar dentro de una misma sección un cierto número de segmentos consecutivos, empezando en *seg_start* y terminando en *seg_end*. El identificador de la sección se suministra a través del atributo *id*. Opcionalmente cada sección se puede sincronizar con la señal de audio a través de los atributos *offset* y *length* (ambos en segundos). Cada sección agrupa segmentos que tienen en común pertenecer a una o varias clases, en cada caso con diferente grado de pertenencia. Para especificar a qué clases y en qué grado pertenecen a ellas los segmentos agrupados en una sección, se ha definido el elemento *class*, con los atributos *descriptor* (nombre descriptivo de la clase) y *confidence* (grado de pertenencia; un valor de coma flotante comprendido entre 0 y 1, que por defecto vale 1). Cada sección ha de incorporar al menos un elemento de tipo *class* que defina la clase a la cual pertenecen sus segmentos. La especificación formal de estos elementos quedaría como sigue:

```
<xs:element name="resource">
<xs:complexType>
<xs:sequence>
      <xs:element name="processors" type="processorsType"/>
      <xs:element name="source" type="sourceType"/>
      <xs:element name="sectioning" type="sectioningType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="segmentation" type="segmentationType"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:complexType name="sectioningType">
      <xs:sequence maxOccurs="unbounded">
            <xs:element name="section" type="xs:sectionType"/>
      </xs:sequence>
      <xs:attribute name="criterion" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="sectionType">
      <xs:sequence maxOccurs="unbounded">
            <xs:element name="class" type="xs:classType"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="seg_start" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="seg_end" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="offset" type="xs:double"/>
      <xs:attribute name="length" type="xs:double"/>
</xs:complexType>
<xs:complexType name="classType">
      <xs:attribute name="descriptor" type="xs:string" use="required"/>
      <xs:attribute name="confidence" type="xs:double" use="default" value="1.0"/>
</xs:complexType>
```

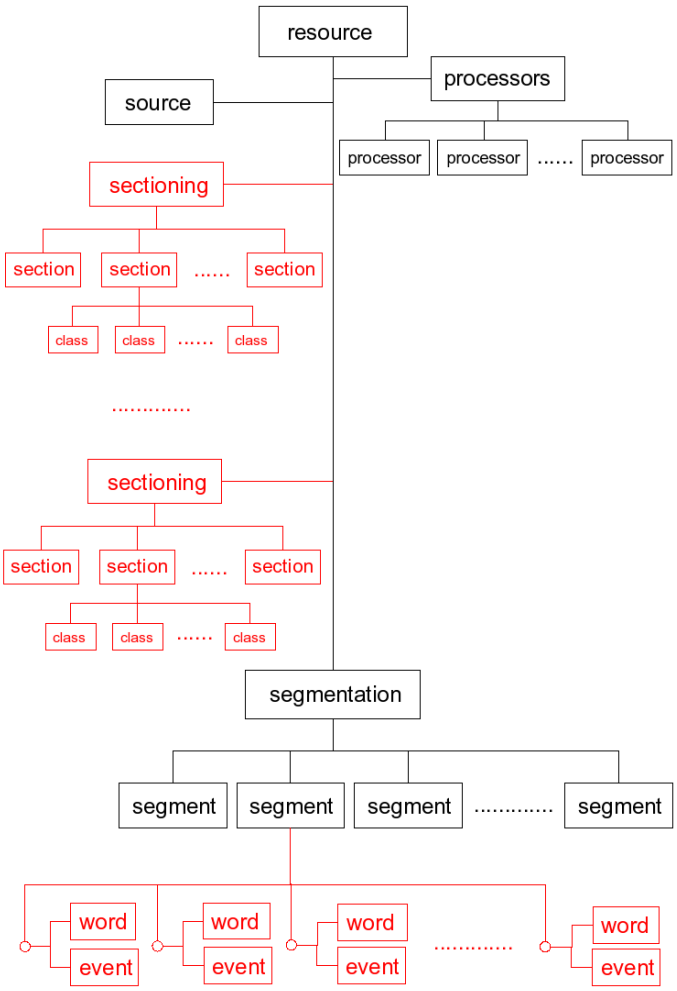

Figura 4. XML-Ehiztari tras la inclusión de los elementos *sectioning*, *section* y *class*, que permiten definir distintos agrupamientos de segmentos. Se muestra también la estructura interna de los segmentos como secuencia de palabras y eventos.

4.3 Palabras

Las palabras surgen o bien como resultado de una transcripción manual o bien como producto de un reconocedor. En este último caso, cada palabra habrá sido reconocida con una cierta confianza, por lo que será necesario reservar un atributo específico (*confidence*: valor de coma flotante comprendido entre 0 y 1). En todo caso, la confianza no es un atributo obligatorio. Su ausencia se interpreta como confianza unidad.

Las transcripciones ortográfica y fonética de la palabra se suministran también a través de sendos atributos (*transcription* y *pronunciation*). Es necesario incluir la pronunciación observada porque podría no coincidir con la pronunciación canónica. En todo caso, así como la transcripción ortográfica es obligatoria, la pronunciación será opcional, ya que por defecto se supone una pronunciación canónica. La pronunciación se especifica mediante una cadena de caracteres en la que los símbolos de los fonemas van separados por blancos. El inventario de fonemas podría incluir unidades especiales para sonidos alargados. Así, por ejemplo, la vocal *'a'* alargada podría representarse como *'a+'* o *'a_long'*. Esto permitiría representar los alargamientos de forma compacta. Igualmente, si el reconocedor fuera capaz de identificar palabras cortadas, el atributo *pronunciation* permitiría especificar la pronunciación parcial observada. Precisamente, para señalar la existencia de problemas en la pronunciación se ha definido el atributo opcional *comment*, que puede tomar 4 valores: *regular* (valor por defecto), *mispronounced*, *cutoff* y *oov* (out-of-vocabulary).

Para representar la información generada por un lematizador, ya se habían previsto dos atributos: *lemma* y *case*, ambos opcionales. Con respecto al atributo *case*, sería interesante que pudiera codificar información sobre la lengua y el tipo de palabra, ya que facilitaría la representación de extranjerismos, acrónimos, nombres propios, etc. que podrían requerir un tratamiento específico.

Por último, al igual que otros elementos, las palabras pueden ir opcionalmente sincronizadas con la señal de audio, por medio de los atributos *offset* y *length*. La especificación formal del elemento *word* en el esquema XML quedaría como sigue:

```
<xs:simpleType name="commentType">
      <xs:restriction base="xs:string">
            <xs:pattern value="regular|mispronounced|cutoff|oov"/>
      </xs:restriction>
</xs:simpleType>
<xs:complexType name="wordType">
      <xs:attribute name="transcription" type="xs:string" use="required"/>
      <xs:attribute name="pronunciation" type="xs:string"/>
      <xs:attribute name="comment" type="xs:commentType" use="default" value="regular"/>
      <xs:attribute name="confidence" type="xs:double" use="default" value="1.0"/>
      <xs:attribute name="lemma" type="xs:string"/>
      <xs:attribute name="case" type="xs:string"/>
      <xs:attribute name="offset" type="xs:double"/>
      <xs:attribute name="length" type="xs:double"/>
</xs:complexType>
```

4.4 Eventos

Como ya se ha dicho, los eventos completan la descripción acústica de los segmentos. A través del atributo *category* distinguiremos tres categorías: *noise*, *filled_pause* y *silent_pause*. Así como el silencio no es matizable, sí lo son los ruidos y las pausas habladas, por lo que a través del atributo *subcategory* se distinguirán varias subcategorías. En el caso de los ruidos, se proponen las siguientes: *breath*, *puff*, *cough*, *laugh*, *click*, *saturation* y *unknown*. En el caso de las pausas habladas, serán: *e_long*, *a_long*, *m_long* y *unknown*. Evidentemente, la experiencia nos dirá si con estas categorías y subcategorías es suficiente o si es necesario definir algunas otras. Además de los atributos obligatorios *category* y *subcategory*, se han definido otros tres opcionales: *confidence*, que permite especificar la confianza de un evento; *offset* y *length*, que permiten sincronizar un evento con la señal de audio. La especificación formal del elemento *event* quedaría como sigue:

```
<xs:simpleType name="categoryType">
<xs:restriction base="xs:string">
      <xs:pattern value="noise|filled_pause|silent_pause"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="subcategoryType">
<xs:restriction base="xs:string">
      <xs:pattern value="breath|puff|cough|laugh|click|saturation|e_long|a_long|m_long|unknown"/>
</xs:restriction>
</xs:simpleType>
<xs:complexType name="eventType">
      <xs:attribute name="category" type="xs:categoryType" use="required"/>
      <xs:attribute name="subcategory" type="xs:subcategoryType" use="required"/>
      <xs:attribute name="confidence" type="xs:double" use="default" value="1.0"/>
      <xs:attribute name="offset" type="xs:double"/>
      <xs:attribute name="length" type="xs:double"/>
</xs:complexType>
```

5 El esquema XML-Ehiztari completo

La propuesta, que gráficamente ha quedado resumida en la Figura 4, se concreta en el siguiente esquema:

```
<xs:element name="resource">
<xs:complexType>
<xs:sequence>
      <xs:element name="processors" type="processorsType"/>
      <xs:element name="source" type="sourceType"/>
      <xs:element name="sectioning" type="sectioningType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="segmentation" type="segmentationType"/>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:complexType name="processorType">
      <xs:attribute name="agent" type="xs:string" use="required"/>
      <xs:attribute name="date" type="xs:string" use="required"/>
      <xs:attribute name="info" type="xs:string"/>
</xs:complexType>
<xs:complexType name="processorsType">
      <xs:sequence maxOccurs="unbounded">
            <xs:element name="processor" type="xs:processorType"/>
      </xs:sequence>
</xs:complexType>

<xs:simpleType name="modeType">
      <xs:restriction base="xs:string">
            <xs:pattern value="conversation|news|podcast|lecture|unknown"/>
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="codeType">
      <xs:restriction base="xs:string">
            <xs:pattern value="mulaw|alaw|linear"/>
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="resolutionType">
      <xs:restriction base="xs:positiveInteger">
            <xs:pattern value="8|16|32|48"/>
      </xs:restriction>
</xs:simpleType>
<xs:complexType name="sourceType">
      <xs:attribute name="url" type="xs:string" use="required"/>
      <xs:attribute name="cache" type="xs:string"/>
      <xs:attribute name="pcm_cache" type="xs:string"/>
      <xs:attribute name="date" type="xs:string" use="required"/>
      <xs:attribute name="size" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="length" type="xs:double" use="required"/>
      <xs:attribute name="format" type="xs:string" use="required"/>
      <xs:attribute name="sampling_rate" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="channels" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="code" type="xs:codeType" use="required"/>
      <xs:attribute name="resolution" type="xs:resolutionType" use="required"/>
      <xs:attribute name="mode" type="xs:modeType" use="required"/>
</xs:complexType>
```

```
<xs:complexType name="sectioningType">
	<xs:sequence maxOccurs="unbounded">
		<xs:element name="section" type="xs:sectionType"/>
	</xs:sequence>
	<xs:attribute name="criterion" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="sectionType">
	<xs:sequence maxOccurs="unbounded">
		<xs:element name="class" type="xs:classType"/>
	</xs:sequence>
	<xs:attribute name="id" type="xs:positiveInteger" use="required"/>
	<xs:attribute name="seg_start" type="xs:positiveInteger" use="required"/>
	<xs:attribute name="seg_end" type="xs:positiveInteger" use="required"/>
	<xs:attribute name="offset" type="xs:double"/>
	<xs:attribute name="length" type="xs:double"/>
</xs:complexType>
<xs:complexType name="classType">
	<xs:attribute name="descriptor" type="xs:string" use="required"/>
	<xs:attribute name="confidence" type="xs:double" use="default" value="1.0"/>
</xs:complexType>

<xs:complexType name="segmentationType">
	<xs:sequence maxOccurs="unbounded">
		<xs:element name="segment" type="xs:segmentType"/>
	</xs:sequence>
</xs:complexType>
<xs:complexType name="segmentType">
	<xs:choice maxOccurs="unbounded">
		<xs:element name="word" type="wordType"/>
		<xs:element name="event" type="eventType"/>
	</xs:choice>
	<xs:attribute name="id" type="xs:positiveInteger" use="required"/>
	<xs:attribute name="offset" type="xs:double" use="required"/>
	<xs:attribute name="length" type="xs:double" use="required"/>
</xs:complexType>

<xs:simpleType name="commentType">
	<xs:restriction base="xs:string">
		<xs:pattern value="regular|mispronounced|cutoff|oov"/>
	</xs:restriction>
</xs:simpleType>
<xs:complexType name="wordType">
	<xs:attribute name="transcription" type="xs:string" use="required"/>
	<xs:attribute name="pronunciation" type="xs:string"/>
	<xs:attribute name="comment" type="xs:commentType" use="default" value="regular"/>
	<xs:attribute name="confidence" type="xs:double" use="default" value="1.0"/>
	<xs:attribute name="lemma" type="xs:string"/>
	<xs:attribute name="case" type="xs:string"/>
	<xs:attribute name="offset" type="xs:double"/>
	<xs:attribute name="length" type="xs:double"/>
</xs:complexType>

<xs:simpleType name="categoryType">
<xs:restriction base="xs:string">
	<xs:pattern value="noise|filled_pause|silent_pause"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="subcategoryType">
<xs:restriction base="xs:string">
	<xs:pattern value="breath|puff|cough|laugh|click|saturation|e_long|a_long|m_long|unknown"/>
</xs:restriction>
</xs:simpleType>
<xs:complexType name="eventType">
	<xs:attribute name="category" type="xs:categoryType" use="required"/>
	<xs:attribute name="subcategory" type="xs:subcategoryType" use="required"/>
	<xs:attribute name="confidence" type="xs:double" use="default" value="1.0"/>
	<xs:attribute name="offset" type="xs:double"/>
	<xs:attribute name="length" type="xs:double"/>
</xs:compleType>
```