Generando presentaciones
Una aproximación personal*

Pablo Rodríguez

Índice

0. Advertencia

Estas páginas son una respuesta a una pregunta sobre una mejora técnica sobre presentaciones. Esta respuesta es el relato de mi experiencia, que me permite formular una propuesta que entiendo que puede ser muy útil para la mejora de la publicación de presentaciones en internet.

Estas páginas están dedicadas a Lawrence Lessig por su inspiración constante y por mostrarnos, entre otras con sus presentaciones geniales, la acuciante necesidad que tenemos de recuperar el equilibrio en el campo de los derechos de autor. En cierta medida, tratan de ser una pequeña muestra de agradecimiento a su excelente trabajo (ya que surgen de una pregunta planteada por él). Agradezco a Sergio Costas su disponibilidad para desarrollar *SuperShow*, aceptar mis sugerencias y por explicarme pacientemente todo aquello que no entendía. Y también agradezco a Burkhard Plaum sus explicaiones sobre tasas variables de fotogramas en vídeo.

Las afirmaciones sobre características o capacidades de programas pretenden ser precisas en la fecha de redacción de estas páginas (16 de mayo de 2008), pero pueden estar desfasadas después.

1. El origen de la historia

El 9 de enero de 2006, Lawrence Lessig escribió en su bitácora una entrada con el título *Experiments in presentation technology*[1]. Ahí plantea la pregunta de cómo lograr de modo sencillo generar presentaciones en las que voz y diapositivas correctamente sincronizadas.

Lessig es conocido, entre otros muchos motivos, por sus presentaciones relacionadas con

*Versión 0.8.

[1] *http://lessig.org/blog/2006/01/experiments_in_presentation_te.html*.

los derechos de autor, campo en el que es un experto. Algunas de estas presentaciones están grabadas en vídeo y pueden descargarse de la red[2]. Como él mismo apunta, lo peculiar de su estilo son muchas diapositivas en poco tiempo: algunas duran uno o dos segundos, cuando no menos, y cada diapositiva puede contener una única palabra.

2. La clave del problema

Para la generación de la presentación hacen falta tres elementos: las diapositivas, el sonido y la secuencia temporal de sincronización. Las diapositivas y el sonido son fáciles de conseguir, el problema surge con la secuencia temporal. En palabras de Lessig:

> *The only difficult part about this was listening to myself again (and again) as I built this.*

Generar una secuencia temporal de sincronización entre el sonido y las diapositivas puede ser más difícil cuanto mayor sea el número de diapositivas y menor sea la duración total del sonido. La clave para conseguir una sincronización adecuada es obtener una lista en la que cada tiempo marca la transición a la siguiente diapositiva.

La sincronización es mucho más fácil de hacer en el momento en que se graba el sonido. La generación simultánea del sonido y de la lista de tiempos es una tarea sencilla con el programa adecuado, independientemente del número de diapositivas y de la duración total de la grabación sonora. La sincronización se genera porque el sonido y la lista de tiempos son correlativas y se generan simultáneamente. Y por eso es imposible que la sincronización entre sonido y diapositivas generada simultáneamente sea errónea.

Sin embargo, la tarea es muchísimo más ardua si la generación de la lista de tiempos es posterior a la generación del sonido. Reconstruir una lista de tiempos es más complicada cuanto más precisa tenga que ser la sincronización. Tanto si lo hace el autor de la grabación sonora como una tercera persona con la ayuda de imagen, reconstruir la lista de transiciones puede ser algo realmente tedioso. Y la sincronización reconstruida puede ser errónea en una o varias transiciones.

3. Aumentando la eficiencia

Un problema añadido del ejemplo de Lessig es que con el programa *iMovie*, el resultado sólo puede ser una película. Eso tiene dos inconvenientes: el archivo resultante es excesivamente grande y además la resolución es relativamente baja (son dos factores directamente proporcionales).

El mayor tamaño del archivo hace más lenta su descarga y además genera mayor tráfico en la red. Existe un compromiso entre una resolución razonable y un archivo de vídeo lo menor posible. A mayor resolución el tamaño de archivo se disparará y un tamaño de archivo muy pequeño se hace apenas visible. El uso de formatos de vídeo que consigan una mayor comprensión, disminuyen esta tensión problemática entre tamaño del archivo y resolución, pero realmente no solucionan el problema.

Un modo de conseguir una resolución alta con archivos de vídeo convencional es usar una tasa variable de fotogramas[3]. De este modo, cada fotograma puede tener la duración que sea necesaria y se reduce el número total de fotogramas comparado con una tasa constante de fotogramas por segundo. De todas formas, el tamaño resultante, si bien la resolución es alta, es también relativamente alto.

En realidad, ambas soluciones son variaciones mejoradas de la grabación en vídeo de una presentación en directo. En este caso, la calidad de la presentación es todavía peor, sencillamente porque la presentación se muestra en una parte del vídeo.

La solución a la tensión entre tamaño y resolución no se puede lograr con vídeo convencional. Dado que se trata de presentaciones, que son fundamentalmente de texto con algunos gráficos, por lo que el número de imágenes y vídeos

[2] Un buen ejemplo puede ser *Against the current "Orphan Works Proposals"*, disponible en *http://video.google.com/videoplay?docid=-6359343497152317 6*.

[3] Como sucede en los vídeos de *http://lessig.blip.tv/*.

por puntos se pueden reducir al mínimo imprescindible. Una mayor eficiencia en el archivo resultante puede conseguirse usando vídeo vectorial.

4. Un ejemplo real

Para poder mostrar un ejemplo que funcionase y que fuese fácil de conseguir, decidí tomar la presentación *Against the current "Orphan Works Proposals"*[4]. Al disponer del vídeo[5], tenía los dos elementos fundamentales: el sonido y la sincronización temporal, y las diapositivas las podía reconstruir con facilidad.

Mi intención era generar una presentación en formato *Flash*, usando *SWFTools*[6], que contiene las utilidades *pdf2swf* (que convierte un documento PDF a formato SWF) y *swfc* (que permite escribir el código para la generación de archivos *Flash*). Por tanto, los elementos necesarios eran las diapositivas, el sonido y la lista de tiempos.

4.1. Diapositivas

Mi elección para la generación de la presentación en formato PDF es XeLaTeX[7]. XeTeX es, en palabras de su creador, «TeX para el resto del mundo». Aunque todavía se encuentra en desarrollo, proporciona una versión de TeX suficientemente estable con soporte nativo para Unicode y además de las propias del sistema TeX, puede trabajar con cualquier tipografía OpenType o TrueType instalada en el sistema operativo[8].

Pero TeX no es una buena alternativa para quienes carezcan de experiencia con él. *OpenOffice.org*[9] genera por sí mismo documentos PDF[10] y *PDFCreator*[11] es una buena utilidad para generar documentos PDF desde cualquier aplicación *Windows*. El sistema *MacOS X* incorpora la posibilidad de generar un documento PDF desde las opciones de impresión de los programas.

En el proceso de edición de las diapositivas he procurado convertir a formato vectorial todos los gráficos de puntos que lo permitiesen (casi todos, menos las fotografías digitales)[12]. Así no sólo se consigue un tamaño menor, sino que se consigue que esa imagen no esté fijada a una resolución dada. Si se aumenta o disminuye el tamaño, no se pierde calidad.

4.2. Sonido

El sonido ya estaba contenido en la película, por lo que sólo había que extraerlo. Para eso lo exporté sin modificarlo, usando el editor de vídeo *avidemux*[13]. El archivo del sonido sin modificación ocupaba 33 MB, pero dado que se trata de una conferencia que contiene voz hablada y no música, podía reducirse mucho más. Usando *lame*[14], convertí el archivo de una tasa original de 128 kb/s a 32 kb/s y de estéreo a mono. Con todo esto, el archivo de sonido final ocupa 8,3 MB[15].

Es posible que con una tasa de sonido de 16 kb/s, el archivo de sonido habría ocupado la mitad del tamaño (sin distorsión del sonido). Y es posible, aunque no lo he probado, que usando compresión de sonido con *Speex*[16], incluso se

[4] *http://lessig.org/blog/2007/02/copyright_policy_orphan_works.html*.

[5] *http://video.google.com/videoplay?docid=-6359343497152317 6*.

[6] *http://www.swftools.org*.

[7] *http://scripts.sil.org/xetex*.

[8] La versión 0.996 de XeTeX está incluída en *TeX Live 2007* (*http://tug.org/texlive/*).

[9] *http://www.openoffice.org*.

[10] Aunque tiene problemas con tipografías OpenType o PostScript: no es capaz de mostrar tipografías OpenType en Unix (*http://qa.openoffice.org/issues/show_bug.cgi?id=78858*) y no es capaz de generar documentos PDF con inserción parcial de tipografías con perfiles PostScript (*http://qa.openoffice.org/issues/show_bug.cgi?id=43029*). Ambos problemas esperan estar resueltos en la versión 3.0.

[11] *http://www.pdfforge.org/products/pdfcreator*.

[12] Una buena aplicación que permite vectorizar imágenes de puntos puede ser *VectorMagic* (*http://vectormagic.stanford.edu/*).

[13] *http://avidemux.org*.

[14] *http://lame.sourceforge.net*.

[15] Gracias a una implementación introducida en *swftools-0.8.2*, el sonido que se introduce en formato WAV se convierte internamente a MP3 con una tasa de 32 kb/s, y consigue una comprensión del archivo *Flash* es todavía mayor. Con el presente archivo, el tamaño final es de 5,9 MB.

[16] *http://www.speex.org*.

podría lograr un sonido aceptable a una tasa de 8 kb/s. Pero *Flash* sólo reproduce sonido en formato WAV o MP3, y no reproduce correctamente archivos MP3 con una tasa inferior a 32 kb/s en versiones anteriores a la 9.

4.3. Sincronización temporal

La tarea más ardua ha sido escribir la lista de tiempos. En un principio usé el reproductor *mplayer*[17]. Invocándolo desde la línea de comandos muestra los el tiempo transcurrido (en segundos y con una precisión de hasta décimas). Se puede copiar en un archivo de texto el tiempo de cada transición (en un esquema hh:mm:ss).

Al comprobar con los dos primeros minutos que la sincronización temporal era errónea, decidí hacerlo fotograma a fotograma. Usando *avidemux*, con el que ya había extraído el sonido, podía avanzar fotograma a fotograma y apuntar el tiempo cada vez que la imagen cambiaba. Este era el único modo de conseguir una reconstrucción precisa de la sincronización entre sonido y diapositivas. Así, esta tarea fue larga y tediosa, pero era la única manera de mostrar un ejemplo que funcione.

4.4. El resultado final

Una vez conseguida la lista de quinientas transiciones, había que escribir un guión que permitiese la generación del archivo *Flash* y que éste se integrase de manera automática en un programa que usase la lista de tiempos. *raccoonshow*[18] era una posible solución, tiene el inconveniente de que no convierte el documento PDF en una presentación vectorial, sino en imágenes de puntos.

La opción que tenía era adaptar *raccoonshow* con un guión que permitiese la generación de una presentación vectorial[19]. El resultado del archivo final es de 5,9 MB (sin fotografías digitales), lo que es una considerable reducción respecto al archivo original, que ocupa 64 MB con una resolución de 480×360 puntos. Esto supone una reducción del tamaño del archivo final en una décima parte[20].

Pero al reproducirlo con el módulo *Flash* de *Adobe*, estaba claro en los primeros ciento veinte segundos que la transición no era correcta. A pesar de que la lista de tiempos estaba escrita contrastando fotograma a fotograma, el sonido avanzaba más rápido que las diapositivas. Con gran sorpresa por mi parte, probé por casualidad a verla con *Gnash*[21]. Sorprendentemente la sincronización funcionaba correctamente.

Como posteriormente descubrí, gracias a la explicación de Sergio Costas, el problema del guión que generaba la presentación *Flash* es que tiene especificados la duración de cada diapositiva en fotogramas (y están definidos los fotogramas por segundo para la presentación). Ese modo *Adobe Flash* no lo interpreta de modo preciso, sino más o menos aproximado. Además de este problema, las transiciones se marcaban en modo relativo (en la duración en décimas de segundo de cada diapositiva) y no en modo absoluto (en la posición de la diapositiva en la reproducción de sonido). Eso produce que cualquier error en la duración de una diapositiva es incorrecta, afecta a todas las posteriores.

Ambas limitaciones están solucionadas en *SuperShow*[22], de Sergio Costas.

4.5. Un nuevo intento

Dado que la generación de la lista de tiempos a mano no era totalmente satisfactoria y además tenía una cierta duda de si además el reproductor de *Adobe Flash* no era totalmente preciso, me decidí a buscar algo una presentación de la que tuviese la seguridad de que la sucesión temporal era correcta. Y el ejemplo era *Free Culture* de Lawrence Lessig, editada con *Macromedia Flash* por Leonard Lin[23].

[17] *http://www.mplayerhq.hu*.

[18] El programa ya no están en desarrollo por su autor. La última versión del programa se encuentra en *http://www.jonobacon.org/files/raccoonshow-0.6.tgz*.

[19] El ejemplo se encuentra contenido en el mensaje de la lista de correo *http://lists.gnu.org/archive/html/swftools-common/2004-05/msg00022.html*.

[20] Suponiendo que el archivo completo alcanzase un tamaño de 8 MB, la reducción sería una octava parte del archivo original.

[21] *http://www.gnu.org/software/gnash/*.

[22] *http://www.rastersoft.com/programas/supershow.html*.

[23] *http://randomfoo.net/oscon/2002/lessig/*.

Lin ofrece el archivo fuente de la presentación[24], que traté de editar con una versión de demostración de *Macromedia Flash Professional 8*. Todo lo que quería era poder copiar la línea temporal de transiciones. Después de un par de horas buscando las transiciones de las diapositivas, abandoné la tarea por imposible, dado que tampoco había usado antes el programa.

Decidí usar un descompilador con la presentación en formato *Flash*[25]. *flasm*[26] mostró, entre toda la información, la duración de cada una de las diapositivas en segundos. Reconstruir la presentación de modo casi automático ha sido fácil, exceptuando los dos pequeños vídeos que incluye la presentación[27].

4.6. Una presentación nueva

Inspirado en la versión japonesa de la presentación *Free Culture*[28] es posible realizar una traducción española de la presentación con un sencillo método.

Para conseguir esto que editar la presentación (que Leonard Lin ofrece) y traducir todo el texto de las diapositivas al español (estas serán las diapositivas de la presentación final).

La presentación, tanto en la versión original como en la japonesa, ofrece un triple botón de pausa, diapositiva siguiente y anterior[29]. Estos tres botones permiten añadir con gran fidelidad el texto de cada diapositiva como subtítulo.

Una vez traducido el texto puede generarse una versión subtitulada con el sonido original o una versión traducida con el sonido también traducido. Esta última sería la mejor solución, pero requiere generar una lista nueva de transiciones.

[24] *http://randomfoo.net/oscon/2002/lessig/free.zip*.
[25] *http://randomfoo.net/oscon/2002/lessig/free_culture.swf*.
[26] *http://www.nowrap.de/flasm.html*.
[27] La presentación en *PowerPoint* (*http://randomfoo.net/oscon/2002/lessig/os_timed.zip*) parece contener los archivos *steamboa.mov* y *aibo01.mov*, pero exportando a formato HTML, no se generan los archivos, aunque el código se refiere a estos archivos.
[28] *http://ittousai.org/lessig/lessig_free_culture_japanese_1.1.swf*.
[29] Esta capacidad también se ha añadido a las presentaciones que genera *SuperShow*.

4.7. Una reconstrucción distinta

Tomando la versión original del vídeo de la presentación de *Change Congress*[30], es posible realizar una reconstrucción distinta de la sincronización. Este vídeo tiene una tasa variable de fotogramas, con lo que es más fácil reconstruir la duración real de cada fotograma.

El programa *qtdump*, que es parte de la biblioteca *libquicktime*[31], permite conseguir la tabla en la que se especifica la duración de (*stts*). Pero con esa tabla, no se consigue automáticamente una lista de transiciones de las diapositivas. Es necesario adaptar y borrar todos los fotogramas que no estén reflejados en las fotografías.

5. ¿Qué hace falta (todavía)?

La generación de presentaciones con sonido sincronizado puede hacerse de tres modos: en directo, editando el vídeo o editando el sonido ya grabado.

La generación de presentaciones con el sonido ya grabado son un intermedio entre la generación de una sincronización temporal nueva y la reconstrucción de la sincronización ya existente. Se trata de rehacer una lista de tiempos según una sincronización, a diferencia del vídeo, que no está fijada en ningún sitio. Pero las necesidades que plantea la generación de presentaciones con el sonido ya grabado están perfectamente cubiertas ya desde la versión 2 de *SuperShow*.

5.1. Presentaciones en directo

La generación de presentaciones en directo implica que se genera la lista de tiempos simultáneamente con la grabación del sonido. Para poder realizar eso, la solución es un contador de transiciones que opcionalmente pueda grabar el sonido. El inicio de la grabación marca el inicio de la cuenta temporal y el fin de la grabación supondría el fin de la cuenta.

En la próxima 3.0, todavía no publicada, *SuperShow* podrá mostrar los archivos PDF sin tener que convertir las páginas a imágenes de pun-

[30] *http://blip.tv/file/get/ChangeCongress-changecongresslaunch701.mov*.
[31] *http://libquicktime.sourceforge.net/*.

tos[32]. Esto permite que pueda mostrarse el PDF en modo de pantalla completa, lo que permite usarlo como programa para presentaciones en directo. Como todavía no tiene implementada la grabación de sonido, un rodeo para lograrlo es reproducir un archivo de sonido vacío lo suficientemente largo para que no se acabe antes de que acabe la presentación. Esto es necesario, porque *SuperShow* cuenta el tiempo por la reproducción de sonido. Y el sonido se puede grabar, del propio ordenador (usando *gnome-sound-recorder*) o con un grabador externo (como un reproductor de audio digital).

En *SuperShow* es necesario todavía implementar la grabación directa de sonido. El siguiente paso sería pasarlo a *Windows* y *MacOS X*, pero por la biblioteca que usa *SuperShow*, tanto para la creación del entorno gráfico (*GTK+*[33]), su traducción a *MacOS X* sería innecesariamente complicada.

Por eso, es posible que sea más fácil escribir una utilidad parecida que permita reproducir y grabar sonido, contar tiempo, mostrar diapositivas (también en modo de pantalla completa), crear una lista de transiciones y generar la presentación final en *Flash*. *wxPython*[34] o *wxWidgets*[35] pueden generar un entorno gráfico nativo para cada sistema operativo. Sin embargo, tanto la biblioteca que permite muestra los archivos PDF (*poppler*), como el conjunto de utilidades que permiten generar los archivos *Flash* (*SWFTools*), no están disponibles para *MacOS X*. No soy un programador experto, pero me temo que será difícil poder plantear un programa similar al que propongo en *MacOS X*, mientras ambas utilidades estén disponbiles en este sistema operativo.

La complejidad de la tarea de edición de vídeo exige un enfoque y un programa diferente que se trata en el siguiente apartado.

[32] Gracias a la biblioteca de enlaces de *poppler* (*http://poppler.freedesktop.org*) para *Python* realizada por Ali Asfar y Gian Mario Tagliarelli (*https://launchpad.net/poppler-python*).

[33] *http://www.gtk.org/*

[34] *http://www.wxpython.org/*

[35] *http://www.wxwidgets.org/*

5.2. Edición de vídeo

La edición del vídeo supone reconstruir una lista de tiempos con una sincronización que ya ofrece la propia película. *SuperShow* ofrece el modo de cargar el vídeo y las diapositivas en una misma ventana para ir marcando la sincronización que se ve en el vídeo y avanzando las diapositivas.

SuperShow es totalmente preciso en la escritura de tiempos, pero esa escritura de tiempos se hace según la entrada de la persona que usa el programa. Y aquí es fácil que la imprecisión venga de la parte humana, que será mayor cuantas más numerosas sean las diapositivas en una cantidad dada de tiempo.

Un modo de solucionar este problema es, como ya he descrito, examinar fotograma a fotograma y escribir una lista de tiempos en aquellos fotogramas que sean distintos al anterior. Hacerlo a mano es una tarea lenta y pesada, la alternativa es automatizar esta tarea por ordenador.

La automatización sería una utilidad (o incluso mejor un añadido a *avidemux*[36]) que permitiese seleccionar una zona del vídeo[37], comparando esa zona del primer fotograma con la del siguiente. Si es igual pasa a comparar la zona del último fotograma con la del fotograma siguiente y si no lo es, escribe el tiempo en una lista y realiza la siguiente comparación.

Un obstáculo para esta automatización, como Sergio Costas hizo notar, es que la mayoría de formatos de vídeo usan una compresión de imagen que hace que imágenes con colores aparentemente iguales realmente no los tengan.

La solución pasa por definir un nivel de tolerancia para la comparación de colores, basada en la que establecen los propios métodos de compresión de vídeo. También se puede definir una mayor tolerancia comparativa según un porcentaje en valores RGB, para detectar cambio cuando para un espectador el color cambia. Una tercera alternativa, y probablemente la mejor, sería dejar que fuese el usuario quien, del

[36] Este programa es un editor de vídeo y además es multiplataforma (disponible para *GNU/Linux*, *MacOS X* y *Windows*).

[37] Así el proceso sería más rápido y fácil, y consumiría menos recursos.

modo más sencillo posible, pudiese definir ese nivel de tolerancia según cada proyecto.

5.3. Edición de vídeo con tasa variable de fotogramas

La edición de vídeo con tasa variable de fotogramas permite una edición manual más fácil. En este caso la lista de transciones está medio hecha.

La mejora necesaria sería, tomando como ejemplo *SuperShow*, consistiría en permitir la edición de vídeo sin reproducirlo, pudiendo avanzar (y retroceder) de fotograma en fotograma, sin una cuenta temporal como en el modo de presentación, con la posibilidad de marcar el tiempo de los fotogramas que supongan una transición a una nueva diapositiva.

Esta implementación no sólo serviría para generar la lista de transiciones, sino que sería útil para crear la presentación en formato PDF. Y no sólo para los vídeos con tasa variable de fotogramas, sino también para los de tasa constante.

6. Ventajas y límites de la propuesta actual

La propuesta de estas páginas tiene ventajas e inconvenientes. Entiendo que en tiene más ventajas que inconvenientes. Pero conviene analizar ambos, para poder juzgar si la propuesta puede aplicarse a un caso concreto.

Las ventajas son las siguientes:

- Que el documento para las diapositivas sea PDF hace que el formato de entrada sea mucho más amplio que si sólo dependiese de un único programa.
- Es algo que funciona sin especiales complicaciones. Se trata de un proceso sencillo que permite generar presentaciones a un usuario medio, sin especiales conocimientos.
- La gran ventaja es que se trata de presentaciones vectoriales, o al menos lo son en su grandísima mayoría. Esto hace que, como ya he repetido, no tenga una resolución fija y su tamaño de salida sea menor.
- La clave de la generación de la presentación está en la sincronización temporal. Si la generación de la lista de tiempos está garantizada, nada impide que la presentación pueda llegar a tener otro formato.

Por el propio diseño tanto de los formatos de entrada y de salida de las presentaciones, el proceso presenta los siguientes límites:

- Hablando con precisión, no todos los documentos PDF pueden convertirse a formato *Flash*, sino sólo aquellos que *xpdf* muestre correctamente, ya que la utilidad que los convierte (*pdf2swf*) usa el código de *xpdf* para realizar la conversión.
- La conversión que *pdf2swf* hace de documentos PDF es casi de punto por punto y el formato *Flash* tiene una limitación de 65536 objetos en un mismo archivo. Esto puede llevar a que la conversión de un documento sea imposible porque se haya excedido el número de objetos que un archivo puede contener[38].
- El formato PDF tiene límites en la generación de efectos especiales con textos o con imágenes. E incluso la conversión de estos efectos a formato *Flash* puede ser en algunos casos imposible.

7. *SlideShare* y *slidecast*

SlideShare es un portal de internet que ofrece alojamiento para presentaciones, esto es, para que cualquiera pueda publicar su presentación en internet. Una de las últimas posibilidades técnicas de este sitio es lo que su desarrollador, Jonathan Boutelle, llama *slidecast*, que es precisamente la posibilidad de añadir sonido a las presentaciones y sincronizar de modo preciso sonido y diapositivas[39].

El *slidecast* permite la reproducción de diapositivas y sonido sincronizado, usando un pequeño programa en la propia página de internet en la que se edita la sincronización. La limitación de ese planteamiento es que al tener que

[38] Matthias Kramm está trabajando en la optimización de la conversión para evitar este tipo de problemas.

[39] Una presentación de esta innovación por su autor puede verse en *http://www.jonathanboutelle.com/mt/archives/2007/07/slidecasting_th.html*.

existir ya el archivo de sonido, impide generar la sincronización al mismo tiempo que se graba el sonido. Y la edición de la sincronización posterior es más trabajosa.

Los problemas del *slidecast* son en mi opinión dos. El primero es que la lista de transiciones se puede copiar a mano, pero no se puede exportar a un archivo como el que propongo (este problema sería fácilmente solucionable). El segundo problema es bastante más grave, ya que *SlideShare* matiene separados las diapositivas, el sonido (que ha de estar en un servidor externo) y la sincronización. Se puede ver en la página el archivo sincronizado, pero no se puede descargar la presentación con sonido sincronizado, sino sólo las diapositivas.

Que los archivos sean externos a *SlideShare* plantean un problema añadido, que es que generan un tráfico adicional al servidor que los aloja, cuando esto reporta un beneficio claro a *SlideShare*. Evidentemente, alojar archivos de sonido implica un mayor gasto, pero no puede subsidiarizarse en otros servidores. Incluso los derechos de autor pueden incluso impedir legalmente el enlace directo a archivos alojados en un servidor, como es este caso.

8. Conclusión

Es evidente que la sugerencia de estas páginas no es una propuesta novedosa. En el fondo sólo es la unión de posibilidades y capacidades técnicas ya existentes para un fin muy concreto.

La clave de todo el asunto gira en torno a una lista de transiciones. Un ejemplo de uso es generar con ella un archivo *Flash*, pero sólo es un formato. Nada impide realizarlo con otros tipos de documentos, como pueden ser PDF, SVG o *Silverlight*.

El resultado es de aplicación sencilla en la creación y la publicación de presentaciones. Y supone una mejora importante para su distribución en internet.