

Squeak, la herramienta de los niños

¿UN JUICIO JUSTO?



En las anteriores entregas de esta miniserie hemos visto ejemplos de lo que otros han hecho con Squeak. Ahora bien, para no cerrar el estudio en falso debemos examinar si la creación de nuestros propios recursos didácticos está fuera del alcance de lo que se puede esperar de un profesor medio. Y nos faltan los testigos del *juicio* en el que nos habíamos embarcado: ¿qué dicen los desarrolladores?, ¿qué dicen los profesores, cómo lo usan? y más importante aún, ¿qué dicen los niños? **POR JUAN RAFAEL FERNÁNDEZ GARCÍA.**

Recordemos brevemente lo aprendido en las entregas anteriores acerca de cómo trabajar con Squeak. Habíamos instalado los paquetes *.deb* creados por José Luis Redrejo para GNU Linux por estar traducidos al castellano y ser los más actualizados (la imagen está en el paquete **educarex-squeak**). Para nuestro estudio también habíamos instalado las imágenes creadas por el grupo francés de desarrolladores de Squeak y la de Squeakland en inglés, así como las imágenes especiales en torno a proyectos específicos que se han hecho en Extremadura (*tusitala*, *algebra-squeak*). No hay problema con disponer de varias imágenes, el ejecutable *squeak* nos preguntará con qué

imagen queremos trabajar. Posteriormente habíamos actualizado la imagen (menú *Configuración - Actualizar desde un servidor*) y habíamos creado una copia local propia de la imagen (menú *Squeak - guardar como, jr.image*), para que nuestras modificaciones y personalizaciones, experimentos y errores no repercutieran en la imagen estándar.

Ahora, al pulsar en el icono que la instalación ha creado en el escritorio, ha aparecido una ventana que nos pregunta qué imagen queremos ejecutar (era la séptima figura de la primera entrega) y hemos seleccionado nuestra versión. Empezamos a trabajar con el entorno como lo habíamos dejado (figura 1: *el mundo*), con un conjunto

de proyectos descargados de servidores de todo el mundo.

Nuestro primer proyecto

A continuación nos ponemos la gorra de programadores; vamos a ver cómo se programa un proyecto. Pequeño consejo de aprendiz de casi todo y maestro de nada: sólo muy adelante en el proceso de aprendizaje se debe empezar un proyecto desde cero. Antes hay que ver cómo lo han hecho otros, aprender añadiendo pequeñas modificaciones, reutilizando código, copiando y adaptando ideas... Y una precisión: usamos la expresión «*programación en Squeak*» en el sentido laxo; por supuesto que programar Squeak es profundizar en el código Smalltalk, modificar y ampliar las clases, crear nuevos objetos para el catálogo, etc., etc., nosotros los profesores nos vamos a limitar a programar guiones utilizando la capa de más alto nivel de Squeak, los *eToys*, los *ensayos activos*... Squeak es como una cebolla, y el acceso a cada una de las capas está abierto, pero no es el tema de estos artículos (ni yo el autor adecuado). Aclaración hecha.



Figura 1: Entorno inicial (el mundo).

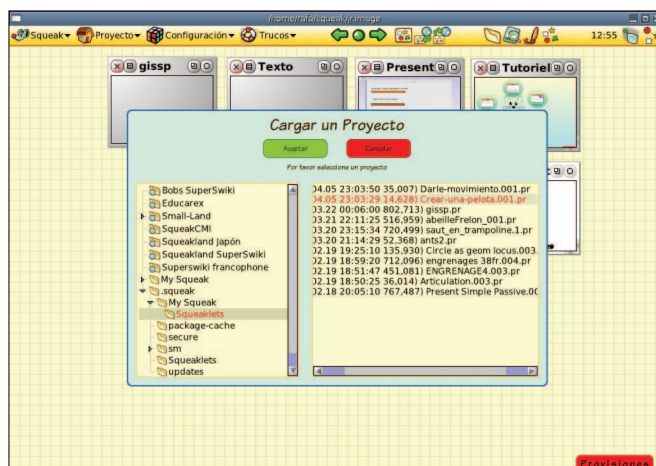


Figura 2: Carga de un proyecto.

Empecemos repasando: ¿cómo se instalaba un proyecto? Aunque podíamos haber utilizado el mismo Squeak para buscar proyectos en internet, en este caso hemos utilizado el navegador para descargar dos ficheros *.pr* (dos proyectos, «Crear una pelota» y «Darle movimiento») de Squeakpolis y los hemos copiado en `~/squeak/My Squeak/Squeaklets`. Ahora, desde nuestra imagen, vamos a abrirlos (menú *Proyecto - buscar un proyecto* o directamente pulsando el icono correspondiente, lanzará la figura 2).

El resultado es el de la figura 3: el proyecto cargado ocupa todo el escritorio. Recordemos que el menú superior nos permite navegar por el listado de proyectos instalados: hemos pulsado en el botón redondo verde (no obstante esto no es estándar de Squeak, sino una característica de las imágenes de Educarex que estamos usando). La flecha atrás nos devuelve al mundo.

¿Qué hay en el proyecto «Darle movimiento» (a una pelota)? Tenemos la posibilidad de averiguarlo: el icono «Jerarquía de objetos» nos devuelve la figura 4, con el listado de objetos en ejecución. Vemos un objeto *Dibujo1*, la pelota, y un guión, que regula su movimiento. Nos ponemos a jugar con el proyecto y sus objetos (¿hay alguna otra forma de aprender?). Averiguamos que la pelota está creada con la herramienta de dibujo de Squeak (si pulsamos el botón central sobre la pelota nos aparece el halo, y en él un pequeño icono *Volver a pintar*; también podríamos lanzar la herramienta de dibujo desde el menú, pulsando en el icono de acceso

directo). La herramienta de dibujo es muy completa, pero para reproducir una pelota no nos hacía falta: vamos a usar el **catálogo de objetos**. Tomamos nota también de las órdenes que se utilizan en el guión. Es el momento de reproducir el proyecto por nosotros mismos.

Desde la pestaña *Provisiones* o el icono del panel superior (entre otras posibilidades) podemos abrir el catálogo de objetos Squeak creados. Después de examinarlo brevemente (es cada vez más completo), vemos que bajo las categorías *Básico* y *Gráficos* nos aparece un objeto *Círculo1*. Sólo tenemos que arrastrarlo sobre el escritorio para incorporarlo al proyecto. ¿Cómo hacer que se mueva? Vinculándole un guión. Volvemos a desplegar el halo (botón central sobre el círculo) y pulsamos en el icono del ojo (*Abrir un Visor*). Nos aparecerá el editor visual de guiones, con distintos métodos clasificados por categorías. A este nivel básico nos interesa la categoría *Movimiento*: arrastramos las órdenes *Avanza*, *Rebota* y *Gira* y jugamos con los valores. El resultado es el de la figura 5: la pelota ya no avanza en línea recta sino en círculos y rebota cuando choca con las paredes de la ventana.

Y ahora la reflexión pedagógica. ¿Qué ocurre si los valores de *Avanza* son mayores, o negativos, o aleatorios? ¿Qué pasa variando el giro? ¿Cómo lograr que la pelota trace un triángulo? Este es el momento crucial de la justificación del esfuerzo de programación que implica el desarrollo de Squeak: en un entorno vivo y con

toda la apariencia agradable de una plataforma de juegos, una actividad lleva a los alumnos a plantearse problemas matemáticos (de aritmética y de geometría en este caso). Es el proceso inverso al tradicional: normalmente se parte de una explicación teórica (hoy toca explicar los triángulos) y después se piensa en los ejercicios que ayuden a que se asimile lo aprendido. Pero estamos cansados de ver que, aunque se aprendan terminología y fórmulas, no se interiorizan los procedimientos que permiten reconocer un triángulo y sus propiedades cuando se cambian los planteamientos y las circunstancias.

Lo que nos lleva a retomar el hilo de la primera entrega, la pregunta de Papert: ¿cómo un alumno que fracasa en nuestra enseñanza reglada es capaz de aprender sin ningún esfuerzo las alineaciones de todos los clubes de primera división, atajos, personajes y trucos de juegos, reglas pragmáticas de cálculo o complejas estrategias para hacer trampa en la vida de la calle? ¿es sólo un problema de interés y motivación? ¿nuestros métodos de enseñanza se corresponden con la forma real en que se aprende? ¿cómo es posible que un alumno que repite perfectamente las propiedades de un triángulo en sus exámenes, y aplica correctamente los distintos teoremas, no sea capaz de ver los triángulos cuando ha salido de clase de matemáticas? Squeak (no sólo Squeak) quiere romper esa *ceguera*: quiere darle cuerpo (intuitividad) a las matemáticas y a la ciencia, quiere convertir al ordenador en la máquina que produce ideas[1].

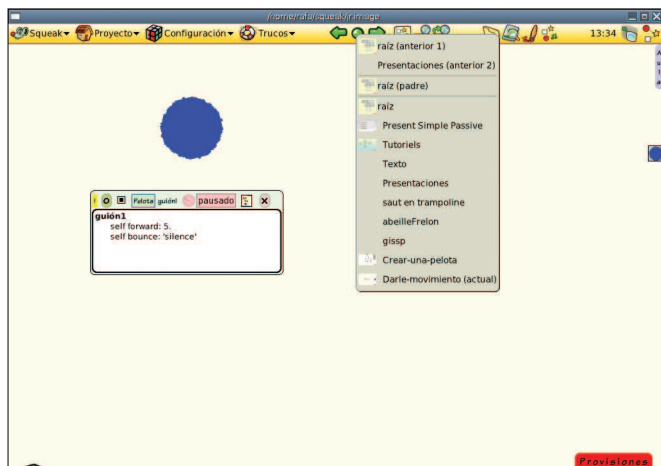


Figura 3: Lista de proyectos instalados.

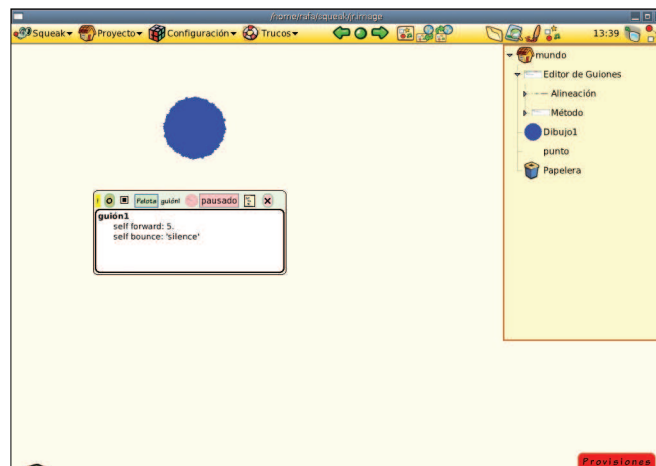


Figura 4: Jerarquía de objetos en ejecución.

Una presentación

Ahora planteémoslo de otra manera: es habitual que los profesores que se deciden a utilizar las nuevas tecnologías en sus aulas utilicen algún programa de presentaciones. ¿Qué diríamos si tuviéramos una herramienta que permitiera crear presentaciones arrastrando y soltando objetos, fotos, vídeos? ¿Qué pasaría si además de esas al parecer tan importantes transiciones entre diapositivas nuestra presentación integrara guiones, se ejecutaran pequeños programas y animaciones? ¿No estaría bien? ¿Es muy caro? ¿muy difícil? ¿hace daño cambiar el chip?

El capítulo noveno del estupendo *Squeak: un mundo para aprender* nos conduce de la mano, paso a paso, en la creación de nuestra primera

presentación con Squeak. Hemos jugado un poco, hasta crear nuestro primer *libro* (el tipo de objeto que nos permitirá hacer presentaciones). Para empezar sencillamente hemos arrastrado el objeto *libro* desde el catálogo. Comprobamos ahora que los menús son contextuales: el menú de un libro (figura 6) presenta entradas para crear una plantilla, configurar las transiciones, etc. El fondo de nuestra presentación consiste en un fichero de imagen en formato *png* que hemos salvado como plantilla para todas las páginas, y hemos añadido objetos varios a nuestro capricho. El resultado es el de la figura 7, donde podemos ver varias modalidades de cuadros de texto, una imagen (simplemente arrastrada desde el navegador de ficheros y a la que se le ha ajustado el

tamaño), un video y una lupa circular.

Ahora estamos en situación de comprender en qué sentido hablamos de programar en Squeak: nos referimos a utilizar objetos, a configurar sus propiedades, y a manejarlos mediante órdenes reunidas en guiones. Incluso en un momento más avanzado podemos plantearnos la interacción entre distintos objetos (las relaciones de choque, contaminación, etc.), y guiones en los que intervenga una lógica un tanto más compleja (si ocurre tal cosa, el objeto se moverá o cambiará de color..., de lo contrario el objeto rebotará, o hará un ruido, o se ocultará). A esto nos limitamos cuando hablamos de programar en Squeak, a utilizar objetos y órdenes predefinidos. No parece complicado,

Cuadro 1: La educación como viaje

Imaginemos la educación como un viaje (qué original, Itaca, etc.). El objetivo consiste en ayudar a que los alumnos lleguen al destino, las herramientas que utilizemos son el vehículo. No estamos inventando nada nuevo: profesores-conductores, ¿no es esa la metáfora fundacional de la pedagogía?

Se repite con frecuencia como un gran descubrimiento: las herramientas son sólo medios, lo importante es no perder de vista el fin. Y es verdad, pero también es verdad que los medios influyen de manera poderosa; será de perogrullo pero hay que decirlo: las condiciones condicionan. ¿De qué estamos hablando? De varios hechos que tenemos delante como montañas (y por tanto son nuestros problemas): estamos tratando de convencer al profesorado de que cambie de vehículo, por dos motivos fundamentales, porque los medios tradicionales servían para una minoría muy reducida de viajeros y porque nuestros medios nuevos son los únicos adecuados para nuestro tiempo. Los alumnos tienen que aprender a resolver problemas y situaciones cuyas soluciones tampoco conocen los profesores.

El medio influye a dos niveles: en primer lugar, nuestro autobús -buscamos una solución generalizable a la totalidad de la población escolar- tiene que funcionar correctamente (no llegaríamos a ningún sitio si las herramientas fallan o son inadecuadas, la tendencia habitual es que lo nuevo se abandone para refugiarse en los procedimientos conocidos); pero en segundo lugar, y dados por solucionados los problemas iniciales, el medio condiciona y modifica el uso. Las herramientas dependerán del tipo de práctica educativa que queremos establecer. ¿Pero nos hacemos estas preguntas cuando descargamos un ejercicio o enlazamos a una página de la web? Nos hace falta aprender a preguntarnos por el valor pedagógico de las actividades que planteamos, valorar el momento de la repetición, de la asimilación, del análisis del problema, de la creación de estrategias... y por otro lado establecer criterios que nos permitan identificar cuándo se están produciendo descubrimientos significativos y cuándo se está construyendo conocimiento.

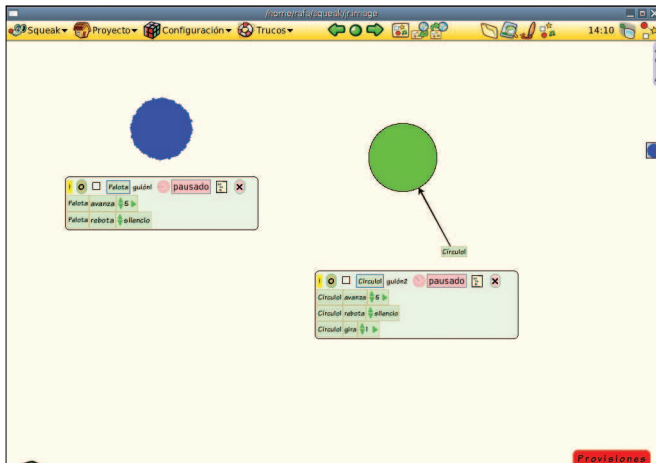


Figura 5: Nuestro primer guión.



Figura 6: Menú para crear una presentación.

¿verdad? Y con un poco de formación y de práctica, está a nuestro alcance, si lo consideramos necesario. ¿Lo es?

Los desarrolladores opinan

Habíamos hecho una pequeña encuesta por correo electrónico, pero o bien tenemos suerte o bien el tema es actual, porque en los medios saltó exactamente el punto crucial del debate.

Sigamos el orden cronológico. Por un lado teníamos a los programadores en Smalltalk, destacando que es el mejor entorno de **simulaciones** disponible actualmente, el vehículo más potente para la **creación de recursos educativos no conductistas** al alcance de los profesores, el **vehículo para introducir a los alumnos en el mundo de la programación**, y el **gran número de proyectos de gran calidad** que se están generando; por otro a las comunidades de desarrolladores de otros lenguajes (por ejemplo de python, en torno a la lista de distribución *Edu-sig*, por lo demás muy interesante y útil) criticando a los primeros como una comunidad elitista, y a Smalltalk como un lenguaje arcaico.

Y de pronto la revolución. Shuttleworth, el hombre del momento, el alma de Ubuntu, reúne en Londres los días 13

y 14 de abril a un conjunto de desarrolladores y pone sobre la mesa la siguiente pregunta concreta y directa[2]: *¿Qué puede aportar el software educativo para luchar contra el fracaso del aprendizaje de las matemáticas entre los alumnos más desfavorecidos de Sudáfrica?* ¿Quiénes están invitados a la reunión? Desarrolladores de LAMS (mencionado en nuestra serie de artículos sobre moodle), de python, y algunos de los principales programadores de Squeak, incluido Alan Kay.

La pregunta no es mala, ¿verdad?, incluso redireccionada a los niños de otros lugares y contextos desfavorecidos. De hecho es la pregunta decisoria: si la informática educativa no puede ayudar a los alumnos, a todos los alumnos, de una manera significativa, ¿para qué el esfuerzo y la inversión? La respuesta no puede ser sólo que los ordenadores ayudan a **motivar** a los alumnos, o que su presencia en las aulas contribuye a **romper la brecha digital**, la respuesta tiene que ser que estas herramientas nuevas facilitan el aprendizaje a un nivel y en unos aspectos que las herramientas antiguas no podían. De la reunión salió la idea de trabajar para crear un currículum con actividades y recursos informáticos

que, independientemente del lenguaje y de los programas utilizados, desarrollen el razonamiento analítico, y la necesidad a formar a los profesores. De la llamada *cumbre de Londres* y de las discusiones posteriores en distintos foros surgió la necesidad de incidir en la creación de **entornos de aprendizaje** que faciliten el aprendizaje de ideas poderosas. Podríamos seguir la serie de artículos (quizás dentro de un tiempo volvamos sobre el tema, en concreto Croquet se merece un estudio en profundidad) examinando desarrollos tan interesantes con PySqueak, Croquet... pero por el momento creo que se entienden las implicaciones, llevamos tres artículos hablando de ellas.

¿Qué dicen en las escuelas?

Los profesores que reciben formación inicial en Squeak ven una herramienta de grandes posibilidades, pero muy distinta a las que están acostumbrados, una herramienta que pide actitudes y destrezas que no piden las otras. Debemos tener en cuenta un aspecto esencial: la transformación que implica la presencia de la informática en las aulas es radical, pero en tanto que radical despierta resistencias; del mismo modo que en un

NOTAS

[1] Recordemos este concepto que presentamos en la primera entrega, y que en inglés suena más rico en connotaciones: Kay hablaba mediante un paralelismo con el piano: si el piano es un instrumento que produce (*plays*, con el múltiple sentido de jugar, tocar y reproducir) música, el ordenador puede ser la máquina que genera y pone a prueba ideas (y permite jugar con ellas).

[2] Hay reseña oficial de la reunión, en inglés, en http://wiki.tsf.org.za/shuttleworthfoundationwiki/Day_one y http://wiki.tsf.org.za/shuttleworthfoundationwiki/Day_two. Y varias bitácoras la comentan y desarrollan sus discusiones.

[3] <http://squeak.blog.com/519834/>, <http://squeak.blog.com/233581/> y <http://squeak.blog.com/342280/>, <http://squeak.blog.com/729061/>. Es en el entorno de la Universidad de Santiago donde se han hecho las descripciones más detalladas de lo que ocurría en la escuela.

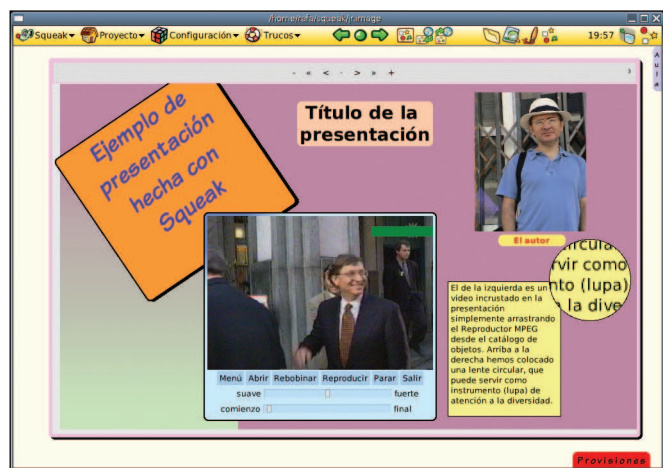


Figura 7: Una presentación hecha con Squeak.

primer momento la imprenta no se usó más que para repetir de manera más fácil y ágil la copia de los textos que se venía haciendo a mano, en la actualidad la informática educativa tiende a usarse como otro auxiliar del libro, la pizarra y el aula cerrada. Los ejercicios que crean los profesores o que prefieren son los que reproducen las fichas y cuadernos de ejercicios a los que estamos acostumbrados, y la presión de la industria para que no se salga de ese camino va a ir creciendo.

¿Y qué dicen los niños? En varias entradas de la bitácora de Squeak de Fraga y Gewerc se lo han preguntado[3], y también en las escuelas extremeñas. La primera impresión de los niños, que por supuesto depende del acercamiento que hagan los profesores, es que se trata de un programa de dibujo. Después van descubriendo que pueden hacer cosas que no están en ningún programa de dibujo, hasta que llegan con asombro a los guiones, la posibilidad de controlar las figuras. Avanzan mediante ensayo y error, cambiando parámetros en las órdenes y disfrutando de sus efectos. Los alumnos asimilan los mecanismos de uso a mayor velocidad que los adultos, y se familiarizan fácilmente con el entorno de trabajo. ¿Hasta dónde llegan? Quizás sea pronto, quizás falte algo antes de poder hacernos esta pregunta.

¿Y usted, señor autor de artículos?

Empezamos el primer artículo de esta serie preguntándonos por el sentido de puzzles en los que reconstruimos una fotografía que ya teníamos, o de enlazar elementos que previamente hemos

desordenado. Y actividades de este tipo, fácilmente clasificables como conductistas, y que son las que se crean con las herramientas habituales, tienen su momento y su justificación. No podemos demonizar porque en la simplificación está la mentira, pero nos encontramos de repente ante un entorno en el que el aprendizaje constructivista no sólo es posible, una herramienta que favorece el descubrimiento y la creatividad. Y este es un aspecto fundamental: **las herramientas no determinan, pero sí condicionan el uso que se hace de ellas**; es posible encontrar usos imaginativos de herramientas de creación de ejercicios clásicos, en los que los propios alumnos crean los crucigramas o los textos a los que les faltan palabras... pero es más fácil lograr que los alumnos creen actividades que promuevan la reflexión con un programa con Squeak.

Partíamos también en aquel artículo de una duda: ¿es posible acercar el aprendizaje en el aula a la forma en que aprendemos en el resto de nuestra vida, en la calle o después de la escolarización? Creo que si hay hoy una herramienta que puede acercar estos procesos es Squeak.

Pero es el momento de mojarse. Estamos en el cuarto año de implantación del software libre en los institutos extremeños, tercer año en muchos centros de primaria y secundaria de Andalucía, etc., no estamos hablando de planes o de posibilidades futuras. Mi conclusión provisional es que **la sola presencia en los escritorios de ninguna herramienta va a transformar la práctica docente**. Intentaré explicarlo, porque la frase puede

fácilmente ser malinterpretada.

Partiré de mi experiencia directa, por supuesto que limitada: llevamos tres años con un icono Squeak en los escritorios de los ordenadores de los centros TIC andaluces. Se pulsa en el icono, se abren varias ventanas de colores poco habituales, y se cierran. Es necesario un proceso probablemente lento de formación y de reflexión sobre cómo utilizar los ordenadores en nuestra aulas, sobre cuántos son necesarios, y en qué agrupaciones, para el cambio de modelo (ver cuadro 1).

¿Que qué opino entonces de Squeak? Opino que, con todos sus aspectos por pulir (cierta incomodidad general en el uso, la impresión de que es un producto aún en construcción y cierta manía de reinventarlo todo, además del olvido del trascendental tema de la accesibilidad), es una gran herramienta para simular y para descubrir. Para que se convierta en una máquina que permita *tocar ideas* hacen falta profesores que sepan orientar a los niños hacia los problemas y hacia el placer de resolverlos.

En el próximo número

Un aspecto en el que uno, quizás ingenuamente, esperaba unos resultados que no se han producido, es en el de la contribución de los profesores de idiomas a la mejora de la traducción de los programas. Quiero creer que no se trata más que de un retraso, producido por la falta de conciencia de que se puede modificar el software que utilizamos y porque no se conocen las herramientas y mecanismos mediante los que es posible la cooperación. Exploraremos herramientas y mecanismos en una nueva serie de artículos.

EL AUTOR

Juan Rafael Fernández García es socio de OFSET, profesor de educación secundaria y tiene una larga experiencia en la traducción y documentación del software libre. Ha sido coordinador de uno de los Centros que participan en la experiencia andaluza de integrar las TIC en la educación y actualmente trabaja como asesor de formación del profesorado.