10.52712/issn.1850-0013-645

Aprendizaje de la programación en el Citilab

Learning to program in Citilab

Jordi Delgado, Joan Güell, José García, Marina Conde y Víctor Casado *

Este artículo pretende ser un resumen de las experiencias llevadas a cabo en el Citilab para acercar la programación de los ordenadores a la gente de la calle. Sin ningún requerimiento previo, cualquiera puede inscribirse en alguno de los cursos de nuestra oferta docente en programación. Esto ha sido posible principalmente gracias a la existencia del *software* adecuado, todo realizado en entornos *Smalltalk*: *Scratch* para niños pequeños (y no tan pequeños), *BotsInc* como iniciación a *Smalltalk* y el mismo *Squeak*, implementación open source de *Smalltalk* utilizada en la creación del *software* mencionado, para programadores experimentados.

Palabras clave: *Scratch*, S4A, *Smalltalk*, *Botsinc*, Innovación, Sociedad, TIC

This article aims at compiling the experiences gathered by Citilab in bringing computer programming to the people. Without any prerequisite, anyone can enroll in any of our courses. This is possible mainly thanks to the existence of a very suitable piece of software which was developed with the Smalltalk *programming language:* Scratch *for kids (but not only for them), BotsInc as an introduction to* Smalltalk*, and also* Squeak*, a* Smalltalk *open source implementation that was used in the creation of the software mentioned above and is directed to experienced programmers.*

Key words: Scratch*, S4A,* Smalltalk*, Innovation,* Botsinc*, Society, ICT*

* Departamento de Lenguajes y Sistemas Informáticos (UPC), Citilab-Cornellà. Correo electrónico: jdelgado@lsi.upc.edu, jgarcia@citilab.eu, jguell@e-citilab.eu, victorct@e-citilab.eu

Introducción

En la enseñanza de la programación a la gente joven (niños y adolescentes) fuera de los itinerarios académicos estándar radica una cuestión de viabilidad que hoy por hoy no está nada clara. Proyectos para enseñar a programar a todo el mundo han existido desde hace más de 30 años (niños, adolescentes y estudiantes universitarios en general), pero aún no parece haber una manera clara y consensuada de hacerlo. Uno de los proyectos básicos del Citilab, centro de innovación tecnológica creado en Cornellà, es la difusión y divulgación de las posibilidades que las tecnologías de la información ponen en manos de la gente común, de todos nosotros. En este marco, el Citilab ha puesto en marcha una propuesta de formación tecnológica que va desde los rudimentos de la utilización de Internet hasta la creación de redes sociales con tecnología web 2.0, pasando por el aprendizaje de la programación de ordenadores.

Este artículo pretende resumir la experiencia del Citilab en la enseñanza de programación. En la primera sección, haremos una pequeña y forzosamente incompleta reflexión sobre el sentido de nuestro proyecto y después explicaremos los tres bloques en que hemos dividido nuestra propuesta: *Scratch* para niños pequeños,[1] el entorno *BotsInc* como iniciación desde cero en la programación,[2] y *Squeak*, implementación open source del lenguaje de programación *Smalltalk* que se utiliza para enseñar a programadores experimentados una nueva manera de desarrollar software (aunque en realidad existe desde hace más de 30 años).[3] Tras nuestras primeras experiencias, decidimos que la utilización de hardware sería un elemento motivador adicional y creamos S4A (*Scratch for Arduino*), que es el entorno usado actualmente en el Citilab.[4] Esta nueva propuesta se detalla más adelante.

1. ¿Qué es el Citilab?

El Citilab (**Figura 1**) es un centro experimental de convergencia entre la nueva generación de Internet y la nueva generación de proyectos de la Sociedad del Conocimiento. Un espacio orientado a activar, impulsar y extender la capacidad creativa e innovadora en tecnología de emprendedores, empresas y ciudadanos en nuestras modernas sociedades.

1. Disponible en: http://scratch.mit.edu.
2. Disponible en: http://rmod.lille.inria.fr/botsinc.
3. Disponible en: http://www.squeak.org.
4. Disponible en: http://seaside.citilab.eu/scratch/arduino.

Figura 1. El Citilab: laboratorio de innovación tecnológica y social

La hipótesis que hay detrás del Citilab es que, gracias a la tecnología y en particular a Internet, en la Sociedad del Conocimiento cualquier persona puede acceder a los instrumentos para crear conocimiento en un contexto de innovación. Las ideas centrales del Citilab son:

- Los ciudadanos tienen capacidad para ser innovadores.
- La colaboración es la base de la innovación social.
- Las nuevas tecnologías crean una nueva cultura de innovación diferente a la de la época industrial.

2. ¿Para qué queremos enseñar a programar a todo el mundo?

"Educators, generals, dieticians, psychologists, and parents program.
Armies, students and some societies are programmed"
Alan Perlis

Las virtudes de aprender a programar como complemento de la formación general de cualquier persona son, por ahora, una cuestión a debatir. Hay detractores, promotores y, por supuesto, indiferentes. Todos los que tenemos cierta edad tuvimos que oír muchas razones para aprender latín en segundo de bachillerato; curiosamente todas esas razones son aplicables para poder justificar cursos de introducción a la programación de ordenadores: crear y incentivar la disciplina en el razonamiento, sistematizar el hábito de trabajo, fomentar la perseverancia y hacernos un poco más humildes (esta razón no se mencionaba en relación al latín, pero aquí es pertinente: ¿quién de nosotros no ha pensado alguna vez que era el compilador el que no funcionaba bien?).

Algunos proponen razones más profundas. Nosotros no lo podríamos expresar mejor que Abelson y Sussman:

> *"Underlying our approach to this subject is our conviction that "computer science" is not a science and that its significance has little to do with computers. The computer revolution is a revolution in the way we think and in the way we express what we think. The essence of this change is the emergence of what might best be called procedural epistemology - the study of the structure of knowledge from an imperative point of view, as opposed to the more declarative point of view taken by classical mathematical subjects. Mathematics provides a framework for dealing precisely with notions of "what is". Computation provides a framework for dealing precisely with notions of "how to"."*

No sólo lo proponen, sino que este nuevo tipo de epistemología es puesta en práctica en la enseñanza, por ejemplo, de la Mecánica Clásica en el MIT. Sea como sea, en estos tiempos de crecimiento de supersticiones, fanatismos e irracionalidad, pretender que la gente piense más en el "*how to*" puede ayudar a fomentar este escepticismo crítico que tanta falta hace.

Finalmente, parece claro que el debate continuará y que nosotros no lo resolveremos aquí, por falta de espacio y porque no es éste el propósito de este artículo. Por esto, una vez dadas algunas razones, más o menos convincentes, que justifican nuestra tarea, vamos a detallar cómo la hemos llevado a cabo.

3. *Scratch*

Scratch es un entorno de aprendizaje desarrollado por *Lifelong Kindergarten Group* el MIT Media Lab en *Smalltalk*, utilizando *Squeak*. Es 100 por ciento *software* libre y pretende acercar la programación a todos aquellas personas que ya tengan ocho o más años. Actualmente se está utilizando en todo el mundo, y las experiencias con *Scratch* se encuentran en todas partes, en particular dentro del congreso más importante del mundo de computer science education vinculado al grupo educación en informática de la ACM (estos últimos dos años, por ejemplo, podemos encontrar diversas referencias, todas ellas citadas en la bibliografía) hasta el punto de que el año pasado se dedicó una sesión especial a *Scratch*.

Figura 2. Ejemplo de programa en *Scratch*: estructura de construcción del programa

El lenguaje de programación vinculado al entorno *Scratch* es un lenguaje imperativo con las construcciones fundamentales (variables, asignación, bucles, condicionales y llamadas a funciones y acciones) ampliadas con un grupo muy numeroso de instrucciones para poder trabajar en proyectos multimedia. Un proyecto *Scratch* será típicamente un juego o una animación sofisticada con dibujos, iconos, sonido y otros gráficos en movimiento (es fácil hacerse una idea visitando los proyectos que seguidores de *Scratch* comparten en la página web mencionada más arriba). Lo que es más original en *Scratch* es la metáfora que hay detrás, que es ni más ni menos que la de los juegos de bloques, tipo *Lego* o el antiguo *Tente*. Cada instrucción o conjunto de instrucciones empotradas se relaciona con otras enganchándose al lado si las sucede o precede, o dentro si pasa a formar parte de una estructura superior, como un bucle. Así iremos construyendo nuestros programas, pieza a pieza, tal como si estuviésemos haciendo un castillo.

El entorno de trabajo que tenemos en el Citilab consiste en ordenadores portátiles con el *Scratch* instalado en cada uno de ellos, para que cada niño haga un seguimiento individual de la clase.

Figura 3. Niños y niñas haciendo sus primeros programas con *Scratch*

El curso ofrecido consiste en empezar a familiarizar al niño con el entorno de trabajo para acabar con un proyecto sencillo hecho por el niño. Los primeros proyectos que les presentamos tienen que ver con la utilización del programa (guardar y recuperar proyectos, añadir, quitar y modificar sprites) y la introducción a la geometría bidimensional del espacio donde haremos vivir nuestros muñequitos y dibujos. Una vez que tienen más o menos claros estos conceptos empezamos a introducir las diversas nociones básicas: repeticiones, condicionales, variables (puede sorprender la introducción de las variables al final, pero *Scratch* de alguna manera lo promueve). Este curso se empezó a ofrecer a principios de 2008, y se sigue ofreciendo en la actualidad, complementado con otro basado en una ampliación de *Scratch* llamada S4A, de la que hablaremos en detalle más adelante. Además, la oferta educativa del Citilab con *Scratch* se extiende también a talleres introductorios durante el período de vacaciones escolares (Tecnoestiu).

En el Citilab la apuesta por *Scratch* ha sido una apuesta firme. No sólo se decidió enseñar programación utilizando *Scratch*, sino que su promoción al exterior desde el Citlab ha sido una de las tareas en las que el Citilab ha tenido más éxito. Desde el Citilab se preparó el paquete que contenía *Scratch* (*BotsInc* y *Squeak* también) en *Linkat*, la distribución de Linux promovida por la *Generalitat de Catalunya*.[5] También en el Citilab se prepararon los contenidos del curso oficial de la Generalitat de Catalunya sobre programación en *Scratch*, dirigido al cuerpo de docentes del gobierno catalán.[6] Hasta hoy día en el Citilab se han realizado las cuatro ediciones del congreso *Programa*, co-organizado con la *Generalitat* desde 2009. Cada año, alrededor de 100 maestros y profesores de primaria, secundaria y bachillerato han

5. Disponible en: http://linkat.xtec.cat.
6. Disponible en: http://www.xtec.cat/formaciotic/dvdformacio/materials/td209.

pasado por el congreso para compartir experiencias y propuestas alrededor de la enseñanza de la programación.

El Citilab también ha estado presente en la *Festa de la Ciència* desde 2009 hasta hoy. Esta fiesta, organizada por el Ayuntamiento de Barcelona, tiene como protagonista la divulgación de la ciencia y la tecnología. Los talleres del Citilab en la *Festa de la Ciència* para introducir a la programación han procurado siempre motivar a los asistentes hacia la programación como una vía para acercarse a la ciencia. Por ejemplo, el primer taller, que se hizo en 2009, tuvo como tema la simulación de un sistema biológico simple de reproducción de bacterias y agotamiento de recursos, y en 2010 se explicaron las órbitas planetarias y el movimiento aparente de retrogradación con simulaciones programadas con *Scratch*, que los propios asistentes completaban en tiempo real.

4. *BotsInc*

Uno de los entornos o lenguajes más innovadores para enseñar a programar a niños y adolescentes ha sido *Logo*, con el que se aprenden los conceptos fundamentales de la programación haciendo dibujos geométricos, dando órdenes a lo que se llamaba una "tortuga". El entorno *BotsInc*, programado en *Smalltalk/Squeak*, por el profesor Stéphane Ducasse (actualmente en el Inria francés) sirve para iniciar la programación con *Smalltalk*, en un entorno muy similar, aunque más completo que el *Logo* original. Originalmente hecho para su mujer, profesora de informática de instituto, *BotsInc* es un entorno que permite aprender a programar en un lenguaje esencialmente imperativo, subconjunto de *Smalltalk*, pero que ya deja entrever algunos conceptos de orientación a objetos inherentes a *Smalltalk*. Todo es un objeto, nada se hace si no es a través del paso de mensajes, construcción de objetos a partir de clases, utilización de browsers de código para añadir métodos a la clase principal del entorno.

Un curso de programación basado en *BotsInc* se impartió en el Citilab durante 2008 y 2009. Iba dirigido a un público mayoritariamente de enseñanza secundaria y bachillerato. Para realizar el curso se tradujo al catalán el libro del profesor Ducasse. Esta traducción fue necesaria, ya que en este país el nivel de inglés es insuficiente entre la población general como para pretender usar un libro de texto que no esté escrito en catalán o en castellano.

5. Squeak/Smalltalk

"One could actually argue -as I sometimes do- that the success of commercial personal computing and operating systems has actually led to a considerable retrogression in many, many respects. You could think of it as putting a low -pass filter on some of the good ideas from the '60s and '70s, as computing spread out much, much faster than educating unsophisticated people can happen."

Alan Kay

Squeak es una implementación del lenguaje de programación *Smalltalk-80*, creado en Xerox PARC durante los años setenta. Esta implementación se creó a principios de

los 90 en Apple, de la mano de Alan Kay, Dan Ingalls, Ted Kaehler y otros, que son los mismos que inventaron *Smalltalk* en Xerox. Squeak hoy por hoy es considerado software libre, aunque su licencia es actualmente motivo de discusión debido a la gran cantidad de colaboraciones que lo han hecho ser lo que es ahora (la versión estándar actual es la 4.4).

Squeak es la implementación de *Smalltalk-80* que hemos escogido para enseñar *Smalltalk*, el auténtico eje vertebrador de nuestro proyecto. Hemos elegido *Squeak*, ya que es software libre y porque es la implementación de *Smalltalk* en la que se han desarrollado proyectos como *Sophie*, *Croquet* o *Seaside*, demostrando sobradamente su capacidad para crear software profesional a la altura de lo que se puede hacer con cualquier otro lenguaje y entorno hoy día. Es más, con *Squeak* se ha desarrollado tanto *Scratch* como *BotsInc*, por lo tanto era la culminación natural en esta secuencia de tres cursos.

¿Por qué razón *Smalltalk*? *Smalltalk* es un lenguaje orientado a objetos puro, tipado dinámico, con una sintaxis muy sencilla y con un número muy pequeño de reglas fáciles de aprender. Todo en *Smalltalk* sucede por paso de mensajes entre objetos y todo es un objeto (incluso las clases). Prácticamente todas las implementaciones *Smalltalk* vienen con un entorno integrado de desarrollo, con exploradores de código (clases y métodos), inspectores de objetos, depurador, gestor de versiones, herramientas de *refactoring*. El entorno *Squeak* es ideal para desarrollar en *Smalltalk* profesionalmente, desde aplicaciones web a entornos colaborativos 3D.[7] [8]

Además, *Smalltalk* es un lenguaje donde el código es un ciudadano de primera clase; es decir, puede ser usado como argumento, retornado, asignado a variables, evaluado cuando se requiera. En este sentido, es una gran mejora sobre sus sucesores, aunque muchos de ellos no esconden su deuda con *Smalltalk*, por ejemplo *Objective-C* o *Ruby*.

El curso que impartimos en el Citilab (2008 y 2009) fue un curso para gente con conocimientos previos de programación (no era necesario que fueran de *Smalltalk*). A partir de los asistentes al curso nació la asociación *Smalltalk.cat* que a día de hoy organiza reuniones mensuales para discutir temas relacionados con *Smalltalk*. Se ha participado en los congresos International *Smalltalk Joint Conference* desde 2009 hasta 2012, con el agregado de que la edición de 2010 fue organizada por el grupo de *Smalltalk* del Citilab, en el mismo Citilab.[9] Hoy día, uno de los miembros de *Smalltalk.cat* es parte del *board* del ESUG (*European Smalltalk Users Group*).

7. Disponible en: http://seaside.st.
8. Disponible en: http://opencroquet.org.
9. Disponible en: http://esug.org/wiki/pier/Conferences/2010.

6. *Scratch for Arduino* (S4A)

A raíz de los cursos de *Smalltalk* impartidos en el Citilab y en la Facultad de Informática de Barcelona (UPC) se dirigieron diversos proyectos fin de carrera relacionados con *Smalltalk*. Uno en particular fue la base del proyecto más relevante del Citilab en lo que respecta a la docencia de la programación: *Scratch for Arduino* (S4A).

Es sobradamente conocido que la posibilidad de cacharrear multiplica considerablemente la motivación de un estudiante que está aprendiendo a programar. El hecho de poder interactuar con el mundo físico añade a la programación el componente del control sobre objetos creados por el estudiante. Estaba claro que a nuestro juicio el mejor entorno para aprender a programar era, y sigue siendo, *Scratch*. La siguiente pregunta fue: ¿existe un equivalente a *Scratch* en el mundo del hardware? Una condición inamovible era que las licencias fueran lo más abiertas posible. Rápidamente llegamos a Arduino, una plataforma de hardware libre perfecta para los propósitos del Citilab.[10] En el Citilab nos propusimos crear un entorno de desarrollo que fuera lo más ideal posible, y, sobre todo, open source, y no hay que ser un genio para llegar a la conclusión que *Scratch* y Arduino formaban la combinación perfecta. Por otra parte, el énfasis que en el Citilab se puso en *Smalltalk* nos colocaba en la tesitura perfecta para poder modificar su código fuente y así conseguir poder controlar placas Arduino desde *Scratch*. El resultado es lo que hoy conocemos como S4A (*Scratch for Arduino*).

La presentación oficial de S4A fue durante el congreso internacional ESUG en septiembre de 2010, aunque se dio a conocer en el congreso local Programa en mayo de 2010. Desde entonces una gran comunidad internacional usa y conoce S4A, desde Alemania hasta EE.UU. pasando por la India. Hoy día Citilab usa intensivamente S4A para enseñar a programar, centrando su oferta educativa alrededor de *Scratch* y S4A, con Arduino. Hasta hoy podemos decir que esta oferta ha sido considerablemente exitosa, dado que los cursos ofertados por el Citilab siempre se imparten con la matrícula completa.

Conclusiones

"Every reader should ask himself periodically "Toward what end, toward what end?", but do not ask it too often lest you pass up the fun of programming for the constipation of bittersweet philosophy".
Alan Perlis

Creemos que la enseñanza de la programación a todo el público interesado (independientemente de su formación de base u otros aspectos académicos) es un

10. Disponible en: http://arduino.cc/.

proyecto a medio-largo plazo que necesita perseverancia por parte de los profesores y de Citilab y fidelidad por parte de la gente que quiere formar parte de este experimento. Por cierto, la cuestión de la fidelidad de los estudiantes, cuando el asistencia es completamente voluntaria y no recibe ningún título al final, es otro tema que merecería una discusión aparte.

Resumiendo, sobre la mesa tenemos un par de cuestiones que se han revisado superficialmente en este artículo, pero que creemos importantes:

1. ¿Hay que enseñar a programar a todo el mundo? ¿Es un objetivo deseable? ¿Qué aporta exactamente la programación de ordenadores en la formación intelectual y cultural de una persona? Nosotros creemos que la programación de ordenadores aporta algo positivo y diferente a la formación de una persona: hábitos y conocimientos que tienen un cierto valor práctico en el día a día de, como mínimo, cualquier persona que viva en un entorno urbano del primer mundo.

2. Si la pregunta anterior tiene una respuesta afirmativa, ¿cómo debe hacerse esto? Ya sabemos que las diversas formas de programar (orientada a objetos, funcional, lógica) implican visiones del mundo muy diferentes, lo que hace que los debates para responder a esta pregunta a menudo parezcan debates de tipo religioso más que debates sensatos y serios. Nosotros hemos tomado partido por *Smalltalk*, pero claramente no es la única solución posible.

Las decisiones ya han sido tomadas y el proyecto ya está en marcha. Los resultados han sido resumidos en este artículo. Como ya hemos dicho, es demasiado temprano como para concluir nada, aunque las expectativas, a la luz de la experiencia de estos cinco años, son desde luego positivas.

Agradecimientos

Queremos agradecer al Citilab y a su personal la confianza que ha permitido poner en marcha el proyecto aquí descrito. A Jorge Gómez por haber colaborado desinteresadamente en el proyecto S4A, y a Marco A. Rodríguez por habernos enseñado Arduino cuando muchos ni siquiera sabíamos que existía.

Licencia

11. Ver: http://creativecommons.org/licenses/by-nd/2.5/es/deed.ca.

Bibliografía

ABELSON, H. (2001): *Structure and Interpretation of Classical Mechanics*, MIT Press.

ABELSON, H. y DI SESSA, A. (1986): *Turtle Geometry, The Computer as a Medium for Exploring Mathematics,* MIT Press.

ABELSON, H. (1996): *Structure and Interpretation of Computer Programs,* MIT Press.

AIKEN, R. M. (1972): "Experiences and observations on teaching computer programming and simulation concepts to high school students", SIGCSE '72: Proceedings of the second SIGCSE technical symposium on Education in computer science, pp. 67–71.

DE CAMPO, L. (1970): "Introducing the computer at a small liberal arts college". SIGCSE'70: Proceedings of the first SIGCSE technical symposium on Education in computer science, pp. 113–117.

DUCASSE, S. (2008): "Squeak, Aprèn a Programar amb Robots". Disponible en: https://gforge.inria.fr/frs/download.php/12387/2008-12-13-BotsincCatala.pdf.

DUCASSE, S. (2005): *Squeak – Learn Programming with Robots,* Apress.

FELDMAN, S. (2005): "A conversation with Alan Kay", *Queue,* vol. 2, nº 9, pp. 20–30.

GOLDBERG, A. y ROBSON, D. (1983): *Smalltalk-80 — The Language and its Implementation,* Reading, MA, Addison-Wesley.

GUPTA, N. (2012): "Learning by creating: Interactive programming for Indian high schools", IEEE Intl. Conference on Technology Enhanced Education (ICTEE). Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6208643&tag=1.

MALAN, D. y LEITNER, H. (s/f): "Scratch for budding computer scientists", 38th SIGCSE technical symposium on Computer science.

MALONEY, J. (2008): "Programming by choice: Urban youth learning programming with Scratch", 39th SIGCSE technical symposium on Computer science education, pp. 367–371.

NIERSTRASZ, O. (2008): *Squeak By Example,* Square Bracket Associates.

PAPERT, S. (1993): *Mindstorms — Children, Computers and Powerful Ideas,* Basic Books.

SIVILOTTI, P. LAUGEL, S. (2008): "Scratching the surface of advanced topics in software engineering: A workshop module for middle school students", 39th SIGCSE technical symposium on Computer science education, pp. 291–295.

WOLZ, U. (2008): "'Scratch' your way to introductory CS", 39th SIGCSE technical symposium on Computer science education, pp. 298–299.