Aprendizaje de la programación en el Citilab

Jordi Delgado
Dpto. Lenguajes y Sistemas Informáticos (UPC)
jdelgado@lsi.upc.edu

Joan Güell, José García, Marina Conde, Víctor Casado
Citilab-Cornellà
jgarcia@citilab.eu
jguell@e-citilab.eu
victorct@e-citilab.eu

Abril de 2013

Resumen
Este artículo pretende ser un resumen de las experiencias en el Citilab acercando 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 de 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.

Abstract
This article wants to be a compilation of our experiences at Citilab bringing computer programming to the people of the street. Without any prerequisite, anyone can enroll in any of our teaching offer in programming courses. This has been possible mainly thanks to the existence of suitable software, all developed with the Smalltalk programming language: Scratch for kids (and not that kids), Botsinc as introduction to Smalltalk and Squeak, a Smalltalk open source implementation used in the creation of the above mentioned software, for experienced programmers.

PALABRAS CLAVE
Scratch, S4A, Smalltalk, Botsinc, Innovación, Sociedad, TIC

KEY WORDS
Scratch, S4A, Smalltalk, Innovation, Botsinc, Society, TIC

1 Introducción

La enseñanza de la programación a la gente joven (niños y adolescentes), fuera de los itinerarios académicos estándar es una cuestión la viabilidad de la cual no está nada clara. Proyectos para enseñar a programar a todo el mundo han existido desde hace más de 30 años (niños[i], adolescentes[ii], estudiantes universitarios en general[iii]) y aun así no parece haber una manera clara y consensuada de hacerlo. Uno de los proyectos básicos del Citilab, centro de innovación tecnológica creado en Cornellá, es la difusión y divulgación de las posibilidades que las tecnologías de la información ponen

en manos de la gente de la calle, de todos. En este marco, el Citilab ha puesto en marcha una propuesta de formación tecnológica que va desde los rudimentos del 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 enseñando a programar. En la primera sección hacemos una pequeña y forzosamente incompleta reflexión sobre el sentido de nuestro proyecto y después explicamos los tres bloques en que hemos dividido nuestra propuesta: Scratch (`http://scratch.mit.edu`) para niños pequeños, el entorno BotsInc (`http://rmod.lille.inria.fr/botsinc`) como iniciación, partiendo de cero, en la programación y Squeak (`http://www.squeak.org`), implementación *open source* del lenguaje de programación Smalltalk, para enseñar a programadores experimentados una nueva (aunque existe desde hace más de 30 años) manera de desarrollar *software*. Tras nuestras primeras experiencias decidimos que la utilización de *hardware* sería un elemento motivador adicional y creamos S4A (*Scratch for Arduino*, `http://seaside.citilab.eu/scratch/arduino`), que es el entorno usado actualmente en el Citilab. Esta nueva propuesta se detalla más adelante.

2 ¿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 la Sociedad del Conocimiento.

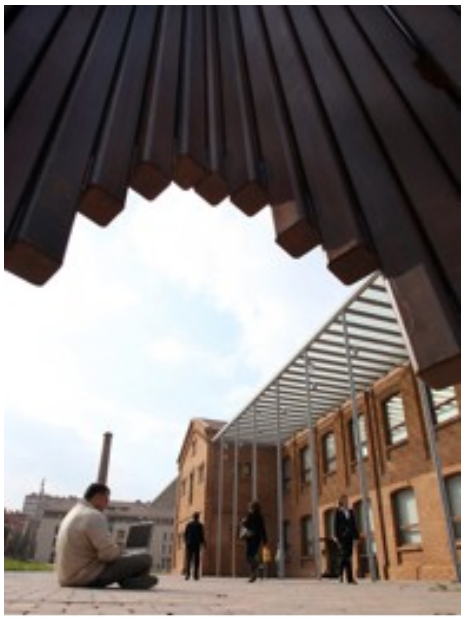

Figura 1: El Citilab: laboratorio de innovación tecnológica y social

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 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.

3 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[iv]

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 ellas son aplicables para poder justificar cursos de introducción la programación de ordenadores: crear y / o incentivar la disciplina en el razonamiento, sistematizar el hábito de trabajo, fomentar la perseverancia y hacernos un poco mas humildes (esta 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 un poco mas 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".*
> H.Abelson & G.J.Sussman[iv]

No solo lo proponen, sino que este nuevo tipo de epistemología es puesta en práctica en la enseñanza, por ejemplo, de la Mecánica Clásica[v] en el MIT. Sea como sea, en estos tiempos de crecimiento de supersticiones, fanatismos e irracionalidad, pretender que la gente piense mas 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 este 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.

4 Scratch

Scratch es un entorno de aprendizaje desarrollado por *Lifelong Kindergarten Group* el *MIT Media Lab* in Smalltalk, utilizando Squeak. Es 100% software libre, y pretende acercar la programación a todos a partir de 8 años. Actualmente se está utilizando en todo el mundo, y las experiencias con Scratch se encuentran en todas partes, en particular dentro del congreso mas 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 [vi] [vii] [viii]) hasta el punto de que el año pasado dedicaron una sesión especial al Scratch[ix].

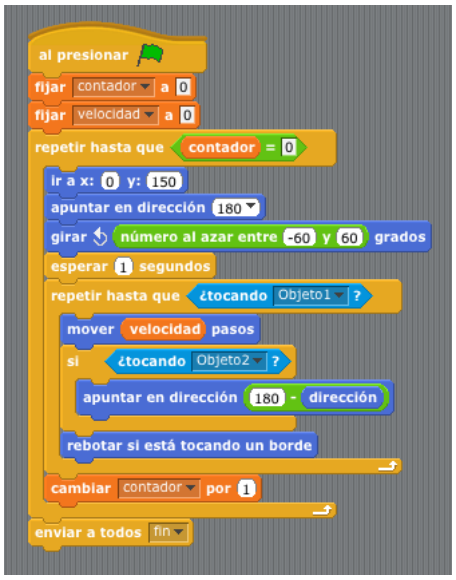

Figura 2: Ejemplo de programa en Scratch. Se ve claramente la estructura de construcción del programa

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 mas arriba). Lo que es más original en Scratch es la metáfora que hay detrás, que es ni mas 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* ,etc.) Y la introducción a la geometría bidimensional del espacio donde haremos vivir nuestros muñequitos y dibujos. Una vez tienen mas 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 (y Botsinc, y Squeak) en *Linkat* (`http://linkat.xtec.cat`), la distribución de Linux promovida por la Generalitat de Catalunya. 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 catalan (`http://www.xtec.cat/formaciotic/dvdformacio/materials/td209`). Hasta hoy día en el Citilab se ha co-organizado 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 pasado por el congreso para compartir experiencias y propuestas alrededor de la enseñanza de la programación. También, el Citilab ha estado presente en la *Festa de la Ciència* desde 2009 hasta hoy. Esta fiesta, organizada por el Ayto. 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 incompletas programadas con Scratch, que completaban los asistentes.

5 BotsInc

Uno de los entornos o lenguajes mas innovadores para enseñar a programar niños y adolescentes ha sido Logo[x], donde se aprenden los conceptos fundamentales de la programación haciendo dibujos geométricos, dando órdenes a lo que se llamaba una "tortuga". El entorno BotsInc[xi] 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, llamada Bot, etc.

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[xii]. Esta traducción fue necesaria, ya que en este país el nivel de ingles es insuficiente entre la población general como para pretender usar un libro de texto que no esté escrito en catalán o en castellano.

6 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[xiii]

Squeak[xiv] es una implementación del lenguaje de programación Smalltalk-80, creado en Xerox PARC durante los años setenta[xv]. Esta implementación se creó a principios de los años 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/o 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 y una sintaxis muy sencilla, 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,* etc. El entorno Squeak es ideal para desarrollar en Smalltalk profesionalmente, desde aplicaciones *web* (`http://seaside.st`) a entornos colaborativos 3D (`http://opencroquet.org`).

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 (años 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* (`http://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, considerando que la edición de 2010 fue organizada por el grupo de Smalltalk del Citilab, en el mismo Citilab (`http://esug.org/wiki/pier/Conferences/2010`). Hoy día, uno de los miembros de *Smalltalk.cat* es parte del *board* del ESUG (*European Smalltalk Users Group*).

7 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* (`http://arduino.cc/`), una plataforma de *hardware* libre perfecta para los propósitos del Citilab. 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 el código fuente de Scratch 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 de 2010, en Septiembre, aunque se dió a conocer en el congreso local *Programa* en Mayo de 2010. Desde entonces una gran comunidad internacional usa y conoce S4A, desde Alemania hasta EEUU pasando por la India[xvi]. 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.

8 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[iv]

Creemos que, esencialmente, la enseñanza de la programación a todo el público interesado (independientemente de su formación de base u otros aspectos académicos) es un 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 reciben ningún tıtulo al final, es todo otro tema que merecería una discusión parte .

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, cómo 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 mas 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 esta 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.

9 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. Rodriguez por habernos enseñado Arduino cuando muchos ni siquiera sabiamos que existía.

10 Licencia

Referencias

i Seymour Papert. *Mindstorms — Children, Computers and Powerful Ideas*. Basic Books, 2nd edition, 1993.

ii R. M. Aiken. Experiences and observations on teaching computer programming and simulation concepts to high school students. In *SIGCSE'72: Proceedings of the second SIGCSE technical symposium on Education in computer science*, pages 67–71, New York, NY, USA, 1972. ACM.

iii Leila de Campo. Introducing the computer at a small liberal arts college. In *SIGCSE'70: Proceedings of the first SIGCSE technical symposium on Education in computer science*, pages 113–117, New York, NY, USA, 1970. ACM.

iv Harold Abelson and Gerald Jay Sussman, with Julie Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, 2nd ed. 1996.

v Harold Abelson and Jack Wisdom with Meinhard Mayer . *Structure and Interpretation of Classical Mechanics*. MIT Press, 2001.

vi John Maloney, Kylie Peppler, Yasmin B. Kafai, Mitchel Resnick, and Natalie Rusk. Programming by choice: Urban youth learning programming with Scratch. In *39th SIGCSE technical symposium on Computer science education*, pages 367–371, Portland, Oregon, USA, March 2008.

vii Paolo A. G. Sivilotti and Stacey A. Laugel. Scratching the surface of advanced topics in software engineering: A workshop module for middle school students. In *39th SIGCSE technical symposium on Computer science education*, pages 291–295, Portland, Oregon, USA, March 2008. ACM, ACM.

viii David J. Malan and Henry H. Leitner. Scratch for budding computer scientists. In *38th SIGCSE technical symposium on Computer science*

ix Ursula Wolz, John Maloney, and Sarah Monisha Pulimood. 'Scratch' your way to introductory CS. In *39th SIGCSE technical symposium on Computer science education*, pages 298–299, Portland, Oregon, USA, March 2008. ACM, ACM.

x Harold Abelson and Andrea diSessa. *Turtle Geometry, The Computer as a Medium for Exploring Mathematics*. MIT Press, 1986.

xi Stéphane Ducasse. *Squeak – Learn Programming with Robots*. Apress, 2005.

xii Stéphane Ducasse. *Squeak, Aprèn a Programar amb Robots*, disponible on-line: `https://gforge.inria.fr/frs/download.php/12387/2008-12-13-BotsincCatala.pdf`.

xiii Stuart Feldman. A conversation with Alan Kay. *Queue*, 2(9):20–30, 2005.

[xiv] Oscar Nierstrasz Andrew P. Black, Stéphane Ducasse and Damien Pollet. *Squeak By Example*. Square Bracket Associates, 2nd edition, 2008.

[xv] Adele Goldberg and David Robson. *Smalltalk-80 — The Language and its Implementation*. Addison-Wesley, Reading, MA, 1983.

[xvi] N. Gupta, N. Tejovanth and P. Murthy. *Learning by creating: Interactive programming for Indian high schools.* 2012 IEEE Intl. Conference on Technology Enhanced Education (ICTEE) `http://dx.doi.org/10.1109/ICTEE.2012.6208643`