novática
CEPIS UPGRADE Revista de la Asociación de Técnicos de Informática

Novática, revista fundada en 1975 y decana de la prensa informática española, es el órgano oficial de expresión y formación continua de **ATI** (Asociación de Técnicos de Informática), organización que edita también la revista **REICIS** (Revista Española de Innovación, Calidad e Ingeniería del Software). ***Novática*** co-edita asimismo **UP**GRADE, revista digital de **CEPIS** (*Council of European Professional Informatics Societies*), en lengua inglesa, y es miembro fundador de **UP**ENET (**UP**GRAD ***E***uropean ***NET***work).

<http://www.ati.es/novatica/>
<http://www.ati.es/reicis/>
<http://www.upgrade-cepis.org/>

ATI es miembro fundador de **CEPIS** (*Council of European Professional Informatics Societies*) y es representante de España en **IFIP** (*International Federation for Information Processing*); tiene un acuerdo de colaboración con **ACM** (*Association for Computing Machinery*), así como acuerdos de vinculación o colaboración con **AdaSpain**, **AI2**, **ASTIC**, **RITSI** e **Hispalinux**, junto a la que participa en **ProInnova**.

Consejo Editorial
Ignacio Agulló Sousa, Guillem Alsina González, María José Escalona Cuaresma, Rafael Fernández Calvo (presidente del Consejo), Jaime Fernández Martínez, Luís Fernández Sanz, Didac Lopez Viñas, Celestino Martín Alonso, José Onofre Montesa Andrés, Francesc Noguera Puig, Ignacio Pérez Martínez, Andrés Pérez Payeras, Viktu Pons i Colomer, Juan Carlos Vigo López

Coordinación Editorial
Llorenç Pagés Casas <pages@ati.es>
Composición y autoedición
Jorge Llácer Gil de Ramales
Traducciones
Grupo de Lengua e Informática de ATI <http://www.ati.es/gt/lengua-informatica/>
Administración
Tomás Brunete, María José Fernández, Enric Camarero, Felicidad López

Secciones Técnicas - Coordinadores
Acceso y recuperación de la información
José María Gómez Hidalgo (Optenet), <jmgomezh@yahoo.es>
Manuel J. Maña López (Universidad de Huelva), <manuel.mana@diesia.uhu.es>
Administración Pública electrónica
Francisco López Crespo (MAE), <flc@ati.es>
Arquitecturas
Enrique F. Torres Moreno (Universidad de Zaragoza), <enrique.torres@unizar.es>
Jordi Tubella Morgadas (DAC-UPC), <jordit@ac.upc.es>
Auditoría SITIC
Marina Touriño Troitiño, <marinatourino@marinatourino.com>
Manuel Palao García-Suelto (ASIA), <manuel@palao.com>
Derecho y tecnologías
Isabel Hernando Collazos (Fac. Derecho de Donostia, UPV), <isabel.hernando@ehu.es>
Elena Davara Fernández de Marcos (Davara & Davara), <edavara@davara.com>
Enseñanza Universitaria de la Informática
Cristóbal Pareja Flores (DSIP-UCM), <cpareja@sip.ucm.es>
J. Angel Velázquez Iturbide (DLSI I, URJC), angel.velazquez@urjc.es>
Entorno digital personal
Andrés Marín López (Univ. Carlos III), <amarin@it.uc3m.es>
Diego Gachet Páez (Universidad Europea de Madrid), <gachet@uem.es>
Estándares Web
Encarna Quesada Ruiz (Virati), <encarna.quesada@virati.com>
José Carlos del Arco Prieto (TCP Sistemas e Ingeniería), <jcarco@gmail.com>
Gestión del Conocimiento
Joan Baiget Solé (Cap Gemini Ernst & Young), <joan.baiget@ati.es>
Informática y Filosofía
José Angel Olivas Varela (Escuela Superior de Informática, UCLM), <joseangel.olivas@uclm.es>
Karim Gherab Martín (Harvard University), <kgherab@gmail.com>
Informática Gráfica
Miguel Chover Sellés (Universitat Jaume I de Castellón), <chover@lsi.uji.es>
Roberto Vivó Hernando (Eurographics, sección española), <rvivo@dsic.upv.es>
Ingeniería del Software
Javier Dolado Cosín (DLSI-UPV), <dolado@si.ehu.es>
Daniel Rodríguez García (Universidad de Alcalá), <daniel.rodriguez@uah.es>
Inteligencia Artificial
Vicente Botti Navarro, Vicente Julián Inglada (DSIC-UPV), <{vbotti,vinglada}@dsic.upv.es>
Interacción Persona-Computador
Pedro M. Latorre Andrés (Universidad de Zaragoza, AIPO), <platorre@unizar.es>
Francisco L. Gutierrez Vela (Universidad de Granada, AIPO), <fgutierr@ugr.es>
Lengua e Informática
M. del Carmen Ugarte García (IBM), <cugarte@ati.es>
Lenguajes Informáticos
Óscar Belmonte Fernández (Univ. Jaime I de Castellón), <belfern@lsi.uji.es>
Inmaculada Coma Tatay (Univ. de Valencia), <Inmaculada.Coma@uv.es>
Lingüística computacional
Xavier Gómez Guinovart (Univ. de Vigo), <xgg@uvigo.es>
Manuel Palomar (Univ. de Alicante), <mpalomar@dlsi.ua.es>
Mundo estudiantil y jóvenes profesionales
Federico G. Mon Trotti (RITSI), <gnu.fede@gmail.com>
Mikel Salazar Peña (Area de Jovenes Profesionales, Junta de ATI Madrid), <mikeltxo_uni@yahoo.es>
Profesión informática
Rafael Fernández Calvo (ATI), <rfcalvo@ati.es>
Miquel Sárries Griñó (Ayto. de Barcelona), <msarries@ati.es>
Redes y servicios telemáticos
José Luis Marzo Lázaro (Univ. de Girona), <joseluis.marzo@udg.es>
Juan Carlos López López (UCLM), <juancarlos@uclm.es>
Robótica
José Cortés Arenas (Sopra Group), <joscorare@gmail.com>
Seguridad
Javier Areitio Bertolín (Univ. de Deusto), <jareitio@eside.deusto.es>
Javier López Muñoz (ETSI Informática-UMA), <jlm@lcc.uma.es>
Sistemas de Tiempo Real
Alejandro Alonso Muñoz, Juan Antonio de la Puente Alfaro (DIT-UPM),
<{aalonso,jpuente}@dit.upm.es>
Software Libre
Jesús M. González Barahona (GSYC-URJC), <jgb@gsyc.es>
Israel Herráiz Tabernero (UAX), <isra@herraiz.org>
Tecnología de Objetos
Jesus García Molina (DIS-UM), <jmolina@um.es>
Gustavo Rossi (LIFIA-UNLP, Argentina), <gustavo@sol.info.unlp.edu.ar>
Tecnologías para la Educación
Juan Manuel Dodero Beardo (UC3M), <dodero@inf.uc3m.es>
César Pablo Córcoles Briongo (UOC), <ccorcoles@uoc.edu>.
Tecnologías y Empresa
Didac López Viñas (Universitat de Girona), <didac.lopez@ati.es>
Francisco Javier Cantais Sánchez (Indra Sistemas), <fjcantais@gmail.com>
Tendencias tecnológicas
Alonso Alvarez García (TID), <aad@tid.es>
Gabriel Martí Fuentes (Interbits), <gabi@atinet.es>
TIC y Turismo
Andrés Aguayo Maldonado, Antonio Guevara Plaza (Univ. de Málaga), <{aguayo, guevara}@lcc.uma.es>

Las opiniones expresadas por los autores son responsabilidad exclusiva de losmismos. ***Novática*** permite la reproducción, sin ánimo de lucro, de todos los artículos, a menos que lo impida la modalidad de © o copyright elegida por el autor, debiéndose en todo caso citar su procedencia y enviar a ***Novática*** un ejemplar de la publicación.

Coordinación Editorial, Redacción Central y Redacción ATI Madrid
Padilla 66, 3º, dcha., 28006 Madrid
Tlfn.914029391; fax.913093685 <novatica@ati.es>
Composición, Edición y Redacción ATI Valencia
Av. del Reino de Valencia 23, 46005 Valencia
Tlfn./fax 963330392 <secreval@ati.es>
Administración y Redacción ATI Cataluña
Via Laietana 46, ppal. 1ª, 08003 Barcelona
Tlfn.934125235; fax 934127713 <secregen@ati.es>
Redacción ATI Aragón
Lagasca 9, 3-B, 50006 Zaragoza.
Tlfn./fax 976235181 <secreara@ati.es>
Redacción ATI Andalucía <secreand@ati.es>
Redacción ATI Galicia <secregal@ati.es>
Suscripción y Ventas <http://www.ati.es/novatica/linteres.html>, ATI Cataluña, ATI Madrid
Publicidad
Padilla 66, 3º, dcha., 28006 Madrid
Tlnf.914029391; fax.913093685 <novatica@ati.es>
Imprenta: Derra S.A., Juan de Austria 66, 08005 Barcelona.
Depósito legal: B 15.154-1975 -- ISSN: 0211-2124; CODEN NOVAEC
Portada: El devorador de fantasías - Concha Arias Pérez / © ATI
Diseño: Fernando Agresta / © ATI 2003

Nº 209, enero-febrero 2011, año XXXVII

sumario

editorial

El valor que aportan las asociaciones de profesionales de las TIC a la sociedad > 02
La Junta Directiva General de ATI

en resumen

Los próximos 20 años de Internet > 02
Llorenç Pagés Casas

Actividades de ATI

Nueva Junta Directiva General de ATI > 03

Noticias de IFIP

Reunión del TC6 (Communication Networks) > 03
Ramon Puigjaner Trepat

monografía

Internet de las cosas

(En colaboración con **UP**GRADE)
Editores invitados: Germán Montoro Manrique, Pablo Haya Coll y Dirk Schnelle-Walka

Presentación. Internet de las cosas: De los sistemas RFID a las aplicaciones inteligentes > 06
Pablo A. Haya Coll, Germán Montoro Manrique, Dirk Schnelle-Walka

Middleware semántico orientado a recursos para entornos ubicuos > 09
Aitor Gómez-Goiri, Mikel Emaldi Manrique, Diego López de Ipiña

El método Mundo - Un enfoque ascendente mejorado de ingeniería informática de sistemas ubicuos > 17
Daniel Schreiber, Erwin Aitenbichler, Marcus Ständer, Melanie Hartman, Syed Zahid Ali, Max Mühlhäuser

Desarrollo Dirigido por Modelos aplicado a la Internet de las cosas > 24
Vicente Pelechano Ferragud, Joan Josep Fons Cors, Pau Giner Blasco

Memorias digitales de objetos en la Internet de las cosas > 31
Michael Schneider, Alexander Kröner, Patrick Gebhard, Boris Brandherm

Explicaciones Ubicuas: Soporte al usuario en cualquier momento y en cualquier lugar > 37
Fernando Lyardet, Dirk Schnelle-Walka

secciones técnicas

Acceso y recuperación de la información

Medidas técnicas de protección del menor en Internet > 42
José María Gómez Hidalgo, Guillermo Cánovas Gaillemín, José Miguel Martín Abreu

Empresa y Tecnologías

La paradoja de la incertidumbre: ¿cuándo menos significa más? > 49
Darren Dalcher

Enseñanza Universitaria de la Informática

Uso de recursos *online* y rendimiento académico del alumnado > 55
José Miguel Blanco Arbe, Jesús Ibáñez Medrano, Ana Sánchez Ortega

Lenguajes informáticos

Historia de los algoritmos y de los lenguajes de programación > 60
Entrevista a Ricardo Peña Marí

Seguridad

La física cuántica en rescate de la seguridad y privacidad de la información en el siglo XXI > 64
Javier Areitio Bertolín

Referencias con firma > 68

sociedad de la información

La Forja

Creación de un Clúster de Alta Disponibilidad con software libre (enunciado) > 75
Miguel Vidal López, José Castro Luis

Programar es crear

Triangulo de Pascal y la Potencia Binomial > 76
(Competencia UTN-FRC 2010, problema E, enunciado)
Julio Javier Castillo, Diego Javier Serrano

asuntos interiores

Coordinación Editorial / Programación de Novática / Socios Institucionales > 77

Monografía del próximo número:
"Ingeniería del Software en proyectos de e-Learning"

Los coordinadores de las Secciones Técnicas de nuestra revista nos ofrecen nuevas visiones, acerca de una amplia variedad de temas relacionados con la Informática, refrendadas con su prestigiosa firma. Añadimos algunas contribuciones de destacados socios de ATI por invitación del Coordinador Editorial.

El peligro de los *blobs* en los *kernels* libres

En la pasada entrega de "Referencias con firma" (núm. 206 de ***Novática***) describimos la presencia de *blobs* en el *kernel* Linux, proyecto emblema del paradigma de desarrollo de Software Libre. **Estos *blobs* son fragmentos binarios que se incrustan dentro del código fuente** y, de este modo, se distribuyen con el resto de código fuente del *kernel*. Aunque se codifican en texto plano, no cumplen uno de los requisitos básicos del software libre, ya que no se pueden modificar. Veamos un ejemplo de *blob* presente en el fichero drivers/net/wireless/atmel.c [1], donde podemos encontrar este trozo de código entre las líneas 394 y 439:

```
/* A stub firmware image which reads the MAC address
from NVRAM on the card.
   For copyright information and source see the end
of this file. */
static
u8 mac_reader[] = {
        0x06, 0x00, 0x00, 0xea, 0x04, 0x00, 0x00,
0xea, 0x03, 0x00, 0x00,
       […] 0x00, 0x01, 0x00, 0x02
};
```

Hemos recortado la mayor parte por brevedad. En este caso, tenemos el *firmware* de una tarjeta inalámbrica empotrado dentro del código fuente. Es la representación en texto plano con caracteres hexadecimales de un fichero binario. Si modificáramos cualquiera de los caracteres hexadecimales, es probable que corrompiéramos el *firmware* y la tarjeta dejara de funcionar. En la práctica, no es posible modificarlo aunque podamos hacer un cambio en el fichero y volverlo a compilar. En la anterior entrega explicábamos que Debian, la distribución de software libre desarrollada por voluntarios incluía dentro de su *kernel* estos *blobs*. Señalábamos el caso de Debian porque en su contrato social se explica que Debian distribuye de manera exclusiva software libre. Es decir, es requisito imprescindible para que un programa forme parte de Debian que esté disponible como software libre. **La imposibilidad de modificar un *blob* implica que no se puede considerar software libre**. Esta situación desató un pequeño conflicto en el seno de Debian. La situación era sorprendente porque es relativamente sencillo identificar estos *blobs* y eliminarlos, y Debian, aun así, decidió mantenerlos como parte del sistema.

Pues bien, **Debian ha optado finalmente por eliminar los *blobs* y ofrecer un *kernel* completamente libre** [2]. En la noticia, Debian comenta que le "*enorgullece anunciar que, hasta donde hemos podido comprobar, hemos resuelto todos los problemas [con los blobs] y podemos distribuir un kernel Linux en Debian Squeeze [la próxima versión estable] que es completamente libre, de acuerdo con las Directrices de Software Libre de Debian. Por la presente, reafirmamos que el software libre es una de nuestras prioridades, tal y como detallamos en el Contrato Social de Debian.*" En la entrega anterior señalábamos el riesgo de tener *blobs* dentro de *kernels* libres. Desde el punto de vista de seguridad, instalar un *kernel* libre supone confiar en que el código fuente está disponible, y que se puede auditar para comprobar que carece de agujeros de seguridad. De hecho, existe (al menos) una distribución libre que realiza auditorías periódicas y que se ofrece como uno de los sistemas más seguros disponibles: OpenBSD.

Curiosamente, desde la publicación de ***Novática*** 206 se ha producido un incidente grave de seguridad dentro de OpenBSD [3]. El líder del proyecto, Theo de Raadt, publicaba un mensaje en una lista de correo del proyecto diciendo que le habían comunicado que **la implementación de IPSec del *kernel* de OpenBSD había sido comprometida hace años por el FBI**, que había pagado a dos desarrolladores que se habían ganado la confianza del equipo del proyecto para que introdujeran los cambios de manera inadvertida al resto de desarrolladores. El problema es incluso más grave, porque la implementación de IPSec se tomó prestada para otros *kernels* libres, como FreeBSD [4], que a su vez es la base del *kernel* de MacOS X [5]. Afortunadamente, **ninguna parte de la implementación de IPSec contenía *blobs*** y el código se podrá auditar para comprobar que está libre de "puertas traseras", o al menos eso parece porque todavía hoy no ha habido ningún comunicado por parte de OpenBSD acerca de si se ha resuelto este incidente.

La noticia es inquietante. OpenBSD, a pesar de la seguridad aparente de las auditorías, contiene *blobs*, al igual que muchos otros sistemas libres. Existen muchas instalaciones de OpenBSD en Internet que se han realizado confiando en la robustez y seguridad de este sistema. Si incluso un *kernel* con auditorías periódicas no está libre de estos problemas, ¿podemos confiar en el software libre para aplicaciones de máxima seguridad?

Uno de los mensajes de respuesta a Theo de Raadt [6] teoriza precisamente acerca de si es posible que el FBI haya introducido una puerta trasera, diciendo "*por supuesto, en estos tiempos en los que aceptamos blobs y drivers binarios [sin código fuente], no creo que el gobierno [de EE.UU.] necesite acudir a este tipo de tácticas [pagar a desarrolladores infiltrados]. Los drivers binarios que tan alegremente usamos en las tarjetas gráficas y wireless nos aseguran que hay montones de sitios dentro de un proyecto de software libre donde el gobierno puede meter mano y acceder a tu kernel completo. No necesitan ser sutiles.*"

Lo dijimos en la entrega anterior y lo volvemos a repetir. **¿Cuánto tiempo pasará hasta que el primer binario malicioso forme parte de un *kernel* libre y sea distribuido a millones de usuarios de manera inadvertida?** Los *blobs* son una nueva entrega del debate entre pragmatismo e idealismo en el software libre. **Los pragmáticos quieren aceptar los *blobs* porque incrementan la funcionalidad**. Los idealistas defienden las cuatro libertades del software libre desde un punto de vista filosófico, porque aceptarlos amenaza al software libre como ideal. Quizás hemos llegado al punto de encuentro en el debate: la presencia de *blobs* amenaza a millones de ordenadores con *kernels* basados en software libre (pensemos en todos los teléfonos libres con *kernel* Darwin o Linux, servidores, etc). **No estamos ante la amenaza de un ideal, esto es un problema práctico de escala planetaria. Los pragmáticos, como los idealistas, deberían también rechazar los *blobs*.**

[1] El fichero pertenece al kernel 2.6.37, última versión estable en el momento de escribir estas líneas, <http://git.kernel.org/?p=linux/kernel/git/stable/linux-2.6.37.y.git;a=blob;f=drivers/net/wireless/atmel.c>.
[2] <http://www.debian.org/News/2010/20101215>.
[3] <http://barrapunto.com/articles/10/12/15/1530231.shtml>.
[4] <http://marc.info/?l=freebsd-security&m=129241460111296&w=2>.
[5] <http://es.wikipedia.org/wiki/Darwin_BSD>.
[6] <http://marc.info/?l=openbsd-tech&m=129236730027908&w=2>.

Israel Herráiz Tabernero (coordinador de la sección técnica "*Software Libre*") y ***Antonio J. Reinoso Peinado*** (Universidad Alfonso X el Sabio).

* * * * * *

Insert coin... if language is not Spanish

Los videojuegos forman parte de nuestra cultura audiovisual desde

mediados de la pasada década de los 70. En aquella época, los juegos electrónicos de ping pong, de asteroides, de comecocos o de matar marcianitos se jugaban en los bares en unas máquinas recreativas que funcionaban con monedas (las máquinas *arcade*) y que competían en voracidad con las máquinas de jugar al millón, los famosos *pinballs* magistralmente elevados a la categoría de mito en la ópera rock *Tommy* musicada por los *Who*. A pesar de que estas entrañables maquinitas desaparecieron ya de los bares, dejando en su lugar el lastre de las insufribles máquinas tragaperras, el desarrollo de los videojuegos y de su industria siguió avanzando de manera imparable y paralelo a las innovaciones en el sector de la informática, produciendo resultados tan memorables como el *Tetris*, el *Super Mario Bros*, los *Sims* o, más recientemente, los juegos de deportes para la *Wii* o el *FIFA 11*, por mencionar sólo algunos pocos hitos de esta industria.

En los últimos 30 años, los videojuegos migraron paulatinamente de las maquinitas de los bares a las consolas, a los ordenadores personales y a las consolas portátiles. Hoy en día, el sector del videojuego está a la cabeza del mercado del ocio audiovisual, con un volumen de ventas superior al que registran conjuntamente el sector del cine y de la música, y cifras de consumo que en el mercado español superaron en 2009 los 1.200 millones de euros y en el mercado estadounidense ascendieron a 10.500 millones de dólares, según los informes de la Asociación Española de Distribuidores y Editores de Software de Entretenimiento (ADESE) y de la Entertainment Software Association (ESA). A nivel estatal, y de acuerdo con los datos ofrecidos por ADESE para 2009, un 22,5% de la población mayor de 15 años es usuaria de videojuegos o, dicho de otro modo, dos de cada nueve personas mayores de 15 años consumen estos productos.

En su origen, los videojuegos no contenían apenas palabras y las pocas que contenían estaban en inglés. Mi generación, que aún tenía el francés como lengua extranjera predominante, descubrió gracias a las maquinitas un puñado de expresiones inglesas fascinantes como *insert coin*, *player one*, *tilt*, *extra ball*, *high scores* y la siempre temida *game over*. Sin embargo, los videojuegos actuales, mucho más complejos que los marcianitos originales, incorporan abundante material textual oral y escrito, con descripciones de escenas, diálogos entre personajes, voces de fondo y textos diversos en función de la línea argumental y el diseño de cada juego, por lo que exigen para su disfrute completo un conocimiento lingüístico profundo o nativo de la lengua del juego.

Actualmente, los videojuegos más populares provienen de los ámbitos lingüísticos japonés e inglés estadounidense, y llegan a nosotros mayoritariamente traducidos al español, bien sea desde su inglés original, bien desde la traducción inglesa del original en lengua japonesa. En muchos casos, no sólo están traducidos al español los textos escritos del videojuego, sino que también están doblados en español los diálogos hablados y demás material textual sonoro del juego. De este modo, los productos de la industria del videojuego alcanzan una penetración en el mercado español que sería impensable si la versión comercializada no estuviera traducida.

No obstante, la presencia del gallego, del euskera o del catalán en los videojuegos es prácticamente nula. Resulta interesante recoger aquí las declaraciones que, preguntado a este respecto, realizó James Armstrong, Consejero Delegado para España y Portugal de la PlayStation de Sony: "*sólo vamos a hacer lo que nos exija el consumidor y al mismo tiempo nos dé rentabilidad. Si el consumidor nos exige videojuegos doblados o subtitulados en catalán, y ganamos dinero con la comercialización de este producto, nosotros estaríamos encantados de poderlo incorporar*" <http://www.directe.cat/noticia/19257/playstation-no-tanca-la-porta-a-doblar-videojocs-al-catala-si-el-consumidor-ho-exigeix-19257>. Aun así, y a pesar de las iniciativas populares para lograr la incorporación del catalán entre las lenguas de traducción de los videojuegos, el catalán sigue sin estar presente en los productos de ocio electrónico de Sony, siendo las causas fundamentalmente económicas. Localizar, traducir y doblar un videojuego es un esfuerzo económico considerable que las empresas del sector sólo están dispuestas a asumir cuando la lengua de destino disfruta de una demanda que justifica su inversión.

Por su implantación presente y por sus perspectivas de futuro, los videojuegos deberían ser considerados un producto cultural de interés prioritario para los objetivos de la política cultural y lingüística de cualquier lengua. La presencia del gallego, del euskera o del catalán en los videojuegos constituye sin ninguna duda un factor importante en su normalización, ya que permite incorporar la lengua en un ámbito tecnológico cultural audiovisual y de ocio ubicuo, de gran penetración en todas las capas sociales y en todas las edades. Lo que está en juego es la preservación de la diversidad cultural y lingüística.

Xavier Gómez Guinovart (coordinador de la sección técnica "*Lingüística computacional*")

* * * * * *

La caja de herramientas del desarrollo de software

Todos sabemos que el desarrollo de software es una tarea compleja. A la complejidad de la gestión de cualquier proyecto de desarrollo se suman las específicas del desarrollo del software. Y es que en desarrollo de software trabajamos con código escrito en algún lenguaje de programación, y de la aparente sencillez del trabajo con código se derivan la mayoría de quebraderos de cabeza en la gestión de proyectos software. En el desarrollo de proyectos software tan importante como la creación de código es su "gestión" entendiendo de manera global por "gestión" mediante qué herramientas se genera el código, cómo se comparte, cómo se garantiza en la medida de lo posible que está libre de errores y cómo se compila, en su caso, este código para finalmente crear una herramienta software.

En estas líneas se presentan una serie de herramientas que el autor considera "imprescindibles" en el desarrollo de software. No pretende ser una revisión exhaustiva ni completa de todas ellas, puede incluso que otros desarrolladores consideren que herramientas igual de importantes que las que aquí se comentan no están presentes.

Estas herramientas son independientes del lenguaje de programación que se utilice, aunque cuando tenga que poner ejemplos haré referencia al lenguaje de programación Java porque es con el que habitualmente trabajo.

Sistema operativo

Muchas veces el sistema operativo donde se desplegará el proyecto software es una de los requisitos del proyecto. Otras veces puede ocurrir que las herramientas de desarrollo que necesitemos sólo estén disponibles para un determinado sistema operativo, o incluso sólo para una determinada plataforma. La elección del sistema operativo es por lo tanto uno de los primeros pasos que debemos dar al iniciar un proyecto software. Aquí la norma puede ser acercarse lo más posible al sistema operativo sobre el que se desplegará nuestro proyecto.

Entorno integrado de desarrollo

No existe un entorno de desarrollo integrado que proporcione la misma potencia para cualquier lenguaje de programación. Hay entornos que se ajustan como un guante a un determinado lenguaje de programación y que son completamente inútiles al cambiar de lenguaje.

Los entornos de desarrollo actuales permiten la integración de un buen número de herramientas de desarrollo, como puede ser el sistema de control de versiones. Si puedes elegir entre varios entornos de desarrollo elije el que dé soporte a un mayor número de estas herramientas.

Otra buena norma a seguir es, familiarízate con el entorno de desarrollo como si fuese tu casa, esto te hará ser mucho más productivo.

Herramienta de control de versiones

"*Te envío por correo electrónico la última versión de mi código*" ¿Te acuerdas de cuando se trabajaba así? La pregunta está formulada en pasado porque ningún equipo de desarrollo debería trabajar así, tampoco de esta otra forma: "*Bájate la última versión del directorio compartido*". Estas frases deberían ponernos los pelos de punta. Las herramientas de control de versiones nos permiten mantener el versionado de nuestro código fuente. Cualquier modificación en el código queda registrada en el sistema para, en su caso, poder recuperarla. Además el sistema de control de versiones guarda información sobre quién hizo el último cambio, y en casos sencillos es capaz de fusionar código modificado por más de un desarrollador, o permitir que sean los propios desarrolladores los que deshagan el conflicto, además de ofrecer la posibilidad de añadir comentarios cada vez que se actualiza el código del repositorio. Muchos entornos de desarrollo permiten la conexión al sistema de control de versiones, enviar, actualizar o descargarse la totalidad del código del proyecto se consigue con muy pocos clicks de ratón.

CVS ha sido la herramienta de control de versiones más utilizada hasta la aparición de Subversion. Y hoy en día Git compite con Subversion en popularidad. No empieces nunca un proyecto sin haber creado previamente un repositorio para él.

Pruebas unitarias

Probablemente, los *frameworks* de pruebas unitarias son de las herramientas que más han ayudado a los desarrolladores de software. Gracias a ellos podemos probar, de manera aislada, el comportamiento nuestras clases. A las pruebas unitarias hay dedicarles tanto esfuerzo como al propio código de producción. Un desarrollo software que carezca o adolezca de pruebas unitarias no es un desarrollo maduro.

Existen excelentes *frameworks* de pruebas unitarias disponibles para muchos lenguajes de programación: SUnit para Smalltalk, JUnit y TestNG para Java, NUnit para .Net, y un largo etcétera. Las ventajas del uso de estos *frameworks* es tan grande que el mismo concepto se ha adaptado a la prueba de bases de datos (DBUnit), páginas web (HTMLUnit) y otros.

Estos frameworks se integran a la perfección de muchos entornos de desarrollo y existen excelentes extensiones para calcular, a partir de la ejecución de las pruebas unitarias, la cantidad de líneas de código que han cubierto nuestras pruebas (cobertura código). Como complemento a los *frameworks* de pruebas en los casos en que el código de nuestras clases esté acoplado. los *frameworks* de dobles de prueba *(mock objects y stub objects*) nos permiten "simular" el comportamiento de las clases que no estamos probando para desacoplarlas del código que sí estamos probando. La importancia del código de pruebas es tal que también debe estar alojado en nuestro sistema de control de versiones.

Herramienta de gestión de errores

Ligadas a las pruebas unitarias y el sistema de control de versiones se encuentran las herramientas de gestión de errores. Estas herramientas nos permiten dar de alta los errores que encontramos en nuestro código, asignar su resolución a algún desarrollador del equipo y hacer un seguimiento de su resolución. Existen excelentes herramientas tales como Bugzilla, Trac o Jira. Existe integración de todas ellas en muchos entornos de desarrollo, y lo que es de gran utilidad, se integran a su vez con el sistema de control de versiones de modo que, por ejemplo, al corregir un error y subirlo al sistema de control de versiones cambia el estado del error en la herramienta de gestión de errores.

Tan importante como la creación del repositorio del proyecto antes de empezar a generar código es la creación del "Producto" en el sistema de gestión de errores.

Herramienta de construcción de proyectos

Los entornos de desarrollo están centrados en el sujeto, y las tareas que el sujeto hace con ellos, pero, en general, tienen poco soporte para las tareas que se realizan de modo rutinario en el desarrollo del software. Una secuencia de tareas típica es: compilar el código fuente del proyecto, realizar la batería de pruebas unitarias, generar los informes de las pruebas unitarias, generar la documentación del código del proyecto, empaquetar la aplicación, desplegarla, en su caso, en un servidor de aplicaciones, etc.

Para este tipo de tareas rutinarias son especialmente útiles las herramientas de construcción de proyectos. La idea es tener las ventajas de herramientas tan conocidas como "*make*" para C/C++ con nuevas funcionalidades como las arriba enumeradas.

Una de las más conocidas en Java es "Ant" que permite especificar como tareas, dependientes o no, la enumeración arriba indicada. De una manera más o menos sencilla permite configurar el proyecto para compilarlo, probarlo, etc. Otra excelente herramienta de construcción de proyectos también para Java es "Maven" que, además de todo lo anterior, gestiona de modo automático la dependencia entre las bibliotecas que usamos en nuestro proyecto. En proyectos que necesitan de una gran cantidad de bibliotecas con dependencias unas de otras sin duda es una excelente opción.

Herramienta de integración continua

"*Mañana tenemos reunión con el cliente y quiere ver el estado del proyecto. Baja la última versión del repositorio en el portátil que nos llevamos a la demo y la compilas*". "¡Vaya!, no consigo compilar el proyecto". "*Pues en mi máquina sí que compila. No tengo ni idea de lo que falta*". Desgraciadamente este es otro escenario típico en el desarrollo de proyectos. Pensamos que teniendo un repositorio de código donde vamos haciendo las actualizaciones es suficiente garantía para poder compilar el proyecto a partir del código del repositorio, y nada más alejado de la realidad. En nuestras máquinas de desarrollo hemos descargado bibliotecas, añadido ficheros de configuración y un largo etcétera que nunca llegaron al repositorio.

Para minimizar el problema anterior podemos hacer uso de herramientas de integración continua. Estas herramientas, cada vez que se invocan, son capaces de descargarse el código fuente del repositorio, bibliotecas y ficheros de configuración incluidos, y compilar el proyecto, de tal manera que si se produce un error durante la compilación generan un aviso de manera automática. De nuevo existen estupendas herramientas de integración continua como Hudson o Coninuum.

Herramienta de documentación del proyecto

Para acabar me gustaría reseñar las herramientas de generación de la documentación del proyecto, documentación interna útil para el equipo de desarrollo. En particular sólo comentaré las herramientas de tipo "wiki" como Mediawiki, Trac o Confluence. Estas herramientas, todas ellas basadas en la web, permiten ir creando, a medida que el proyecto avanza, documentación útil para los desarrolladores. Así, cada nuevo desarrollador que llega al proyecto puede encontrar toda la información a él relativa si hemos tenido la precaución de mantener esta importante tarea activa por parte de todos los miembros del equipo.

Otras herramientas de utilidad

Existen otras herramientas que he dejado por comentar pero que son

igual de importantes que las que sí que he comentado. Algunas de ellas son herramientas de análisis estático del código, herramientas de depuración de código, herramientas de documentación del código y un largo etcétera. Conocerlas y saber cuando utilizarlas mejorará sin duda la calidad de nuestros proyectos.

Más allá de las herramientas... el equipo

Todas estas herramientas son inútiles si el equipo no las necesita. Esto es evidente. Pero las razones, o falta de ellas, que un equipo puede tener para no necesitar estas herramientas no lo son tanto. Si es por desconocimiento, una buena opción es dedicar esfuerzo a informarse sobre su existencia, los beneficios que se obtendrán están muy por encima del esfuerzo dedicado. Si es por negligencia o por falta de auténtico trabajo en grupo, con o sin herramientas el desarrollo de software, se convertirá en una penitencia antes de cometer el pecado.

La pieza más importante del desarrollo es el equipo, la cultura de ser y formar parte del mismo. Y dentro del equipo la pieza más importante son las personas, si no cuidamos de las personas difícilmente vamos a poder cuidar del equipo.

Óscar Belmonte Fernández (coordinador de la sección técnica "*Lenguajes informáticos*")

* * * * * *

La programación de móviles, ¡qué torre de Babel!

Aprovechando las vacaciones de Navidad me puse a programar un par de cosillas para el i-Phone, antes lo había hecho con cierto éxito para terminales con Windows Mobile o Android. Y (ahora viene lo bueno) no tengo un Mac para poder instalar el Xcode que es un IDE necesario para desarrollar las aplicaciones para i-Phone.

Pues bien, después de varios intentos fallidos, pude instalar una versión de MacOs como una máquina virtual en Windows, y luego instalé el Xcode. La verdad que está bien el entorno de desarrollo pero nada tiene que ver con lo que hacía a nivel de Windows Mobile o Android. Tanto las posibilidades de desarrollo de interfaces como la forma de programación son realmente distintas. Súmese a ello las distintas características de cada sistema operativo, y a uno no le queda mas que exclamar que esto de la programación de móviles… ¡Es una torre de Babel!

Diego Gachet Páez (coordinador de la sección técnica "*Entorno digital personal*")

* * * * * *

¿Quién regula la red?

Hablemos de *Internet;* en español Interred, de acuerdo con el glosario básico inglés-español <http://www.ati.es/spip.php?article97> editado por ATI. La Interred no se puede controlar; se construyó para que no pudiera ser controlada. Su estructura es descentralizada; lo más aproximado que hay a un gobierno centralizado es la Corporación de Internet para la Asignación de Nombres y Números <http://www.icann.org/>, que se encarga de hacer cambios en las direcciones *IP* y nombres de dominio. Pero incluso si esta corporación sin ánimo de lucro desapareciera, la Interred apenas notaría el efecto: tan sólo que no podrían incorporarse nuevos dominios y máquinas. Esencialmente, la Interred funciona de forma autónoma y automática. Lo único que necesita para continuar funcionando es suministro eléctrico.

Sin embargo, aunque las máquinas funcionen a la perfección, no sucede lo mismo con quienes las manejan. Mentes malintencionadas tratan constantemente de manipular las máquinas en su provecho. Robo y tráfico de datos personales. Contenidos ilícitos de todo tipo. Gusanos. Redes de bots. Spam. Denegación de servicio. Envenenamiento de memorias temporales *DNS* y *ARP*. Secuestro de sesión. Suplantación. Falsificación *web*, por enlace (*Phising*), por *DNS* (*Pharming*) y por superposición (*Cross site scripting*). Conforme la potencia y complejidad de la Interred aumenta, también lo hacen las amenazas.

Responder a esta clase de desafíos en una red de estructura descentralizada requiere un esfuerzo incesante. A menudo se buscan soluciones mediante técnicas simples como bloqueos y filtrados, pero siempre aparecen otras técnicas que eluden a las primeras, como atajos y tunelados. Esta clase de parches nunca aguantan mucho tiempo, porque chocan con las cualidades inherentes a la Interred: la red de redes se construyó para comunicar, no para incomunicar. Desde el comienzo de la Interred se intenta alcanzar la autorregulación; hoy sabemos que no se puede alcanzar del todo.

Sólo hay una manera cierta de eliminar servicios ilícitos de la Interred, y es desconectando las máquinas que prestan esos servicios. Y sólo hay una manera legal de desconectar esas máquinas, y es que lo hagan las autoridades del territorio correspondiente. Sin embargo, el alcance global de la Interred conlleva una absoluta diversidad en la acción de las diferentes autoridades. Unos persiguen la infracción de derechos de autor, otros persiguen solamente la infracción de derechos de autor con ánimo de lucro, y unos terceros no reconocen los derechos de autor. Unos persiguen los sitios de organizaciones terroristas, otros persiguen las negaciones del Holocausto, otros persiguen las representaciones de Mahoma, y casi todos persiguen la pornografía infantil.

Con 28 años cumplidos, la Interred funciona sin una normativa global, y no parece que eso vaya a cambiar pronto. La Organización de las Naciones Unidas creó en 2006 el Foro para el Gobierno de la Interred <http://www.intgovforum.org/cms/> que, a pesar de su rimbombante nombre, reduce su papel a foro de debate. Otra fuente de iniciativas es el G8, el grupo formado por Rusia más los siete países que fueron los más ricos hace 20 años. Durante la presidencia italiana del año 2009, Berlusconi intentó promover la regulación de la Interred, pero sin resultado; quizás la idea de prohibir pornografía de menores no convenciera a alguien. Para 2011, Sarkozy se propone idéntico objetivo durante la presidencia francesa; ya veremos en qué queda.

La ausencia de una normativa global no significa en absoluto que la Interred se halle en la anarquía. Lo que sucede es cada estado elabora sus propias normas para regularla, y actualiza sus códigos civiles y penales para incluir figuras jurídicas específicas. Algunos ejemplos: la Unión Europea aprobó la directiva de protección de datos, Francia aprobó las leyes Hadopi 1 y 2, Corea del Sur aprobó la ley del negocio de la comunicación electrónica, y Estados Unidos aprobó la ley del derechos de autor del milenio digital.

A la hora de regular, se producen puntos de concordancia entre los países. Algunos son no planificados: muchos países coinciden en censurar la Interred de una forma u otra. Otros son planificados, mediante iniciativas como los tratados, por ejemplo el Acuerdo Comercial Anti-Falsificación (*Anti-Counterfeiting Trade Agreement, ACTA*). Pero esta clase de iniciativas no pretende alcanzar una normativa global, sino tan sólo normas coincidentes. Cuestiones fundamentales como la jurisdicción quedan al margen.

Dado que Estados Unidos fue el inventor de la Interred, y el que ha prestado y continúa prestando los mejores servicios sobre la misma, en la práctica su legislación se ha convertido en la legislación *oficiosa* de la Interred. En particular, la legislación del estado de California, que

por algo es donde se alojan los principales servicios como *Google*, *Facebook*, *Yahoo!* y *Twitter*. Esto plantea algunos problemas a los que no son californianos ni estadounidenses. Por ejemplo, los europeos comprobamos que al usar estos servicios no se respetan nuestros derechos en materia de protección de datos personales. Si la empresa tiene sede en Europa, los tribunales europeos pueden responsabilizarla por sus infracciones; pero si no es así, acuda usted a un tribunal de California.

Esta es la situación actual, una situación estable que puede prolongarse indefinidamente. Puede cambiar, pero para ello se requiere algo difícil de conseguir: poner de acuerdo a los diferentes países.

Ignacio Agulló Sousa (socio sénior de ATI)

* * * * * *

¿Reinventamos la rueda?

Llevamos una temporada de reaparición de una serie de figuras que pretenden pasar por nuevas tecnologías, pero que en realidad representan la reinvención de la rueda. Los responsables de marketing y servicios comerciales de las múltiples empresas dedicadas a las tecnológías informáticas o como se denominan en el año 2011, las mal llamadas TI (yo prescindo de la C que poco tiene que hacer, o ¿mucho? en este mundillo nuestro), no hacen más que pensar en como intentar vender más con los mismos productos de siempre o con algunos que ya surgieron hace tiempo, pero que por razones variadas pasaron a mejor vida.

¿Y con qué nos encontramos? Pues tenemos delante a eso que se ha llamado "*Cloud computing*", o "computación en la nube", o también la "virtualización", la "gestión de los datos", o incluso la "movilidad". Pero, ¿es que tenemos algo nuevo realmente?

Si lo analizamos fríamente y desde un punto de vista puramente neutral se puede ver que la mal llamada "*Cloud computing*" o "computación en la nube" no es más que, con el uso de las actuales tecnologías de las comunicaciones, lo que en los años 50 del siglo pasado se denominaba "Oficina de Servicios Informáticos o "*Data Processing Bureaux*".

Nos encontramos 50 años después de que la Informática tomara consciencia de su existencia, no porque naciera entonces, allá por el inicio de la década de los años 60 del siglo pasado, sino porque fue cuando se comenzó a manejar con las denominadas calculadoras, luego llamadas ordenadores o computadores/as, los datos e informaciones de las organizaciones, de las empresas y de las administraciones publicas. En aquellos años iniciales, aquellas máquinas todavía no hacían más que lo que desde los años 20 se venía realizando con las máquinas de tarjetas perforadas, tabuladoras, clasificadoras, etc., que en el fondo no era otra cosa más que ordenar datos para poder realizar contabilidades y nóminas más deprisa y con mayor exactitud que cuando se hacían de manera manual y sobre papel. Luego, con la aparición de ordenadores más potentes a fines de los años 60, ya se pasó no solo a manejar los datos sino también a ajustar a los nuevos medios electromecánicos, los procedimientos de gestión de aquellos datos.

En aquellos tiempos reajustar los procedimientos de gestión se denominaba "Organización y Métodos", porque se organizaban los datos y las informaciones y se "racionalizaban" los procedimientos de gestión o métodos de trabajo. Pero a eso mismo, en los años 80 y 90 se le denominaba "reingeniería de sistemas" y en la actualidad hablamos de "inteligencia de los negocios", la *Business Intelligence*. Lo que indudablemente se ha modificado y ha cambiado es la tecnología utilizada, las herramientas, pero el fondo sigue siendo lo mismo, estamos reinventando la rueda modificando la terminología empleada porque así se vende mejor. Son instrumentos de marketing los que nos pueden despistar y hacer creer que estamos ante cosas nuevas, ante nuevos inventos, cuando en realidad, en el mejor de los casos, lo que se ha realizado es mejorar los procedimientos de análisis y organización, y los métodos de gestión de esos datos e informaciones para lograr un mejor conocimiento.

Ya no se habla prácticamente de procesos *batch*, ni de *batch* remoto, pero lo que se está desarrollando en la actualidad con el fenómeno denominado "*Cloud computing*", "computación en la nube", tiene mucho de proceso en *batch* y de proceso en *batch* remoto, y en algunos casos procesos en tiempo real. De hecho, el procesamiento en la nube no es más que la vuelta a las "Oficinas de servicios informáticos", en inglés "*Data Processing Service Bureaux*", de hace 50 años. Las facilidades de comunicación vía Internet y otras tecnologías similares, así como la necesidad de centralizar los datos de las organizaciones, hacen que los PC individuales y los servidores locales de las organizaciones, empresas y administraciones públicas se vean desfasados.

Dentro de este contexto nos encontramos también con el tema de la virtualización que tiene una gran semejanza con el procesamiento en la nube y con la simulación, pues en el fondo la virtualización no es más que una simulación, una apariencia de lo que en realidad no es y también la simulación lleva años de existencia en nuestro mundo informático. Pero hablar de "virtualización" queda mucho mejor a efectos de marketing.

Lo que sí puede resultar más privativo de los tiempos actuales es eso que se viene denominando "Informática móvil", que no es más que el soporte físico, pequeño y transportable, que se utiliza para ver y consultar datos e informaciones. Es decir, hemos regresado a los primitivos terminales no inteligentes, los predecesores de los PC, aunque eso sí pequeños y que requieren grandes habilidades manuales y visuales para que rindan algún beneficio. Y como la capacidad de datos en pantalla es bastante reducida, posiblemente favorezca la difusión de gráficos representativos de esos datos.

La informática móvil favorecerá la visualización mediante diagramas y gráficos que dan una imagen de la situación de cada actividad y función de la organización. Facilita la captura de información en forma de imágenes que posibilitan la toma de decisiones de manera inmediata. Y la inmediatez es lo que prevalece en estos momentos, al final de la primera década del siglo XXI. Ahora se habla de aplicaciones móviles, terminales móviles, etc. pero estos pequeños terminales, que ahora son tontos, pronto dejarán de serlo y pasarán a ser inteligentes con lo que se podrá volver al PC, pero al PC de bolsillo, no solo portátil. Busquen Vds. las gafas de aumento, yo me llevo el portátil en la cartera, tengo problemas de vista. Y luego viene el problema de incorporación del teclado en una pantalla táctil… "Veremos", dijo el ciego.

Fernando Piera Gómez (socio sénior de ATI)

* * * * * *

Robótica: Presentación de la nueva sección técnica de Novática

Nacimiento y expansión de la Robótica

En 1961, hace ya 50 años, en Estados Unidos Condec Corporation entregaba a General Motors el primer robot estático para líneas de producción. Su trabajo consistiría en extraer piezas metálicas a altas temperaturas de la máquina de moldeado y apilarlas. Un año antes,

Concec Corporation había comprado Unimation, que creó el primer prototipo basándose en las investigaciones de George Devol.

Desde entonces, la utilización de robots se ha extendido a otras industrias de fabricación y a otros sectores, como empaquetado, almacenamiento, investigación y armamento, entre muchos otros. Su utilización cruzó fronteras, siendo Japón a finales de los años 80 el líder mundial en fabricación y utilización de robots industriales. En 2001 la Unión Europea superaba por primera vez a Japón en número de robots industriales instalados.

La robótica ha seguido cruzando nuevas fronteras. Los robots ya no forman parte exclusivamente de la maquinaria industrial, progresivamente se están incorporando a nuestra vida diaria. Impulsados especialmente por los grandes avances en electrónica y software, están llegando tanto al ámbito doméstico (Aibo de Sony o Pleo para ocio, y Roomba de iRobot para tareas domésticas), como al ámbito educativo, donde creo que es especialmente interesante su incorporación en todos los niveles, desde primaria a universitaria, ya no solamente como asignatura especializada, sino como instrumento educativo.

Nuevos retos

La robótica presenta nuevos y apasionantes retos a sus pilares fundamentales: el software, la electrónica y la mecánica. No solamente por las nuevas necesidades tecnológicas en cada ámbito, sino por la necesaria coordinación e integración del trabajo que realizan personas de tan distintas especialidades.

El software realizado en robótica está fuertemente ligado al concepto de "*embedded software*", ya que en ambos casos el software controla o se integra con distintos dispositivos electrónicos. Los requisitos que ha de cubrir suelen requerir un profundo conocimiento de algunos de los aspectos más complejos del desarrollo de software, como concurrencia, tiempo real, e inteligencia artificial, por mencionar sólo algunos de los principales.

En microcontroladores o SOC (*System On a Chip*) se necesita más potencia en menor espacio y con menor consumo eléctrico. Sensores más precisos, actuadores (principalmente servomotores) con la mayor potencia, precisión y rapidez, con el menor consumo y peso posible.

Para los elementos utilizados en la estructura se requiere el menor peso posible pero con la resistencia adecuada para soportar el resto de elementos que componen el robot. Todos los elementos físicos se han de acomodar a la estructura diseñada del robot para realizar las funciones planteadas, con una correcta distribución del peso, que en algunos casos como en los robots humanoides requiere un esfuerzo considerable.

Un proyecto de desarrollo de software que integre distintos sistemas suele presentar habitualmente una considerable dificultad para realizarlo correctamente, en el tiempo, con el esfuerzo y todas las funcionalidades planificadas. Si a esto le añadimos que se ha de integrar perfectamente dentro de una estructura física también en desarrollo y con distintos dispositivos electrónicos formando un único sistema que ha de percibir y actuar con objetos reales, creo que no queda ninguna duda del gran reto ante el que nos encontramos. Un reto insalvable sin dos ingredientes fundamentales: creatividad y capacidad innovadora.

Aunque probablemente otros retos sean más complejos de resolver, como las cuestiones éticas. Actualmente, en la industria militar, la mayoría de los llamados robots están realmente telecontrolados, pero muchos de ellos también están preparados para funcionar en modo autónomo. En Corea del Sur, uno de los países más avanzados en robótica, ya existen robots armados que vigilan la frontera con Corea del Norte. ¿Es ético que un robot pueda decidir por sí solo cuando puede disparar a personas?

Creatividad e innovación

En mi opinión, el mayor atractivo de la robótica es su enorme poder para despertar y mejorar nuestro potencial creativo y de innovación; sin ello sería imposible realizar la integración y coordinación de disciplinas tan diferentes, en la cual cada especialista se ve obligado a comprender, al menos, los conceptos básicos de las otras.

Pero no sólo es fundamental para la robótica. En un mundo donde la industrialización y las nuevas tecnologías están progresando rápidamente, no sólo en los países más avanzados sino también en países emergentes como Brasil, Rusia, India o China (BRIC), nuestro futuro creo que está muy vinculado a nuestra capacidad creativa y de innovación.

Estas capacidades se pueden aprender y mejorar, dentro del sistema educativo y en las empresas y entidades que componen todo el tejido social.

Lista orientativa de temas que vamos a tratar.

- Robótica en la educación.
  - Como instrumento en los distintos niveles educativos.
  - Como especialidad.
- Desarrollo de software.
  - Coordinación de proyectos multidisciplinares.
  - Arquitecturas robóticas.
  - Creación de comportamientos: percepción, memoria, deliberación y actuación.
- Plataformas robóticas.
  - Frameworks y herramientas software.
  - ¿Componentes o kits?
- Robótica y empresas.
  - Componentes electrónicos: IGEP.

Fuentes

Wikipedia. *Robotics.* <http://en.wikipedia.org/wiki/Robotics>.
Wikipedia. *Unimate.* <http://en.wikipedia.org/wiki/Unimate>.
Encyclopedia Británica. *Robot.* <http://www.britannica.com/EBchecked/topic/505818/robot/248208/Industrial-robots?anchor=ref849962>.
Encyclopedia Británica. *Robotics.* <http://www.britannica.com/EBchecked/topic/1384950/robotics>.
Jean Kumagai. *A Robotic Sentry For Korea's Demilitarized Zone.* IEEE Spectrum, <http://spectrum.ieee.org/robotics/military-robots/a-robotic-sentry-for-koreas-demilitarized-zone>.
Grupo Educativa. *¿Por qué enseñar Robótica en las escuelas?* <http://grupoeducativa.blogspot.com/2010/08/porque-ensenar-robotica-en-las-escuelas.html>.
Fundación Gabriel Piedrahita Uribe. *Robots en la educación.* Eduteka, <http://www.eduteka.org/reportaje.php3?ReportID=0018>.

José Cortés Arena (coordinador de la sección técnica "*Robótica*")

* * * * * *

Utilidad de los procesos ágiles en el desarrollo de software

Uno de los elementos clave dentro de la ingeniería del software es la

definición del proceso de desarrollo por el grupo de trabajo encargado de realizar el sistema software. Mucho se ha hablado en la literatura sobre las ventajas o desventajas de utilizar procesos clásicos, estructurados o rígidos en el desarrollo de software, frente a procesos "más desordenados" (vulgarmente hablando). Vamos a referirnos a continuación a dos aspectos importantes sobre la utilidad de los procesos ágiles: los aspectos críticos de los mismos y la interacción con la arquitectura software. Parecería que los proponentes de una arquitectura software rígida serían contrarios a una propuesta de métodos ágiles para el desarrollo, pero ambos puntos de vista no están enfrentados y van apareciendo las primeras evidencias sobre la utilidad de las metodologías de desarrollo ágiles. Mientras que desde el punto de vista teórico el uso de procesos ágiles estaría enfrentado con determinados productos software "estructurados", recientes estudios nos muestran que no hay contradicción entre ambas posturas y que los distintos aspectos pueden ser reconciliados en la práctica.

Como el mismo término "ágil" indica, los procesos software que se engloban bajo ese término poseen características de flexibilidad y de respuesta a condiciones cambiantes. Los supuestos básicos de los métodos ágiles son: prestar más importancia a los individuos y sus interacciones que a los procesos y las herramientas, preocuparse más por el funcionamiento del software que por la documentación, tener una mayor colaboración con el cliente que sobre las cuestiones formales y, sobre todo, orientarse a una respuesta inmediata frente al seguimiento de un plan estricto.

Los métodos ágiles constituyen una reacción a los métodos tradicionales planificados que enfatizan un enfoque excesivamente ingenieril y racional y que asumen que todos los problemas se pueden especificar completamente. El enfoque "tradicional" propugna una planificación intensa, procesos estrictos y rigurosa reutilización, de tal modo que la actividad de desarrollo se asemeja a una ingeniería estricta. Por el contrario el enfoque "ágil" promueve una respuesta rápida a entornos cambiantes, cambios en los requisitos y modificaciones en los plazos de entrega. Los acrónimos que se incluyen bajo el paraguas de método ágil son variados e incluyen términos como *Extreme Programming* -XP-, Scrum, Crystal, *Lean Software Development* -LD- y otros.

Las principales críticas que se atribuyen a los enfoques ágiles son: a) sus ideas no son nada innovadoras, pues es lo que se hacía desde hace tiempo; b) la falta de enfoque hacia la arquitectura del software va a generar productos de menor calidad; y c) no existen demasiadas pruebas de la utilidad de estos métodos, además de la falta de replicabilidad de esas prácticas.

Un reciente estudio bastante sistemático realizado por Dyba y Dingsoyr ha mostrado algunos de los beneficios y limitaciones de la introducción de métodos ágiles en el desarrollo. Así, se ha encontrado que XP ha sido más difícil de introducir en organizaciones complejas y que era más adecuado para pequeños grupos que para grandes proyectos. La adopción de métodos ágiles es fácil en muchos casos y se encuentran beneficios en la colaboración con el cliente, en el tratamiento de los errores y en algunos aspectos de gestión, incluso en la estimación. También se encuentra cierta mejora en la percepción de los clientes sobre los efectos de la comunicación, aunque si el contacto es muy continuo se percibe cansancio. Igualmente se puede destacar que el uso de metodologías ágiles va acompañado de distintas prácticas de gestión de proyectos. Por ejemplo, dado que los métodos ágiles son dependientes de las personas, un modelo de gestión que permita intercambiar personas presentará grandes limitaciones. La productividad medida en líneas de código también aumenta si se utiliza XP. Con respecto a la calidad del producto, ese estudio indica, sin grandes convicciones, que la calidad del producto puede incrementarse, pero no es concluyente. Tampoco se puede establecer una correlación entre la satisfacción en el trabajo de los programadores y las prácticas ágiles.

Los factores de éxito con evidencias cuantitativas en la implementación de este tipo de procesos son: la utilización de una estrategia correcta de entrega de productos, el uso de una práctica adecuada de las técnicas de ingeniería del software ágiles, la integración de un grupo de trabajo de alta cualificación junto con una alta implicación del cliente y un proceso de gestión adecuado. El principal problema para la gestión de métodos ágiles es la correcta cuantificación y evaluación del estado real del proyecto. No existen muchas conclusiones absolutamente claras y generalizables, pero podemos decir que no se ha detectado ninguna evidencia en contra de la utilización de estas metodologías ágiles. Quizá el punto más delicado sería el hacer compatible los conceptos de "arquitectura software" con el de "agilidad". La arquitectura del software, según el *Rational Unified Process*, se define como "*el conjunto de decisiones sobre la organización de un sistema software, la selección de los elementos estructurales y las interfaces de los que se compondrá el sistema, junto con el comportamiento que especifican esos elementos. La arquitectura del software no se preocupa sólo de la estructura y del comportamiento, sino también del uso, de la funcionalidad, del rendimiento, reutilización, comprensibilidad y de las restricciones económicas y tecnológicas*".

Diversos autores indican que estos conceptos no son incompatibles puesto que se puede construir una arquitectura con distintos niveles de detalle, de tal modo que quienes utilicen métodos ágiles puedan establecer sus requisitos de modo flexible y con una comunicación adecuada. Además, ya existen propuestas "ágiles" que no reniegan de la existencia de una documentación sobre la arquitectura. El mayor problema reside en los requisitos no funcionales, que son los que quedan al margen en las propuestas ágiles. Sin embargo, ya existen alternativas para expresar los conceptos más importantes de una arquitectura e incorporarlos a la comunicación de los grupos de desarrollo y ya se ha hecho para Scrum. Podemos concluir que, aunque no existan resultados concluyentes sobre la gran superioridad de los métodos ágiles, tampoco se evidencian desventajas con respecto a los métodos clásicos en los aspectos críticos del desarrollo de arquitecturas software.

Referencias

P. Abrahamsson, M.A. Babar, P. Kruchten. Agility and Architecture: Can They Coexist?. *Software, IEEE, vol.27, no.2*, pp.16-22, March-April 2010.

Tsun Chow, Dac-Buu Cao. A survey study of critical success factors in agile software projects. *The Journal of Systems and Software 81* (2008), pp. 961-971.

Tore Dyba, Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology 50* (2008), pp. 833-859.

D. Falessi, G. Cantone, S.A. Sarcià, G. Calavaro, P. Subiaco, C. D'Amore. Peaceful Coexistence: Agile Developer Perspectives on Software Architecture. *Software, IEEE, vol.27, no.2*, pp.23-25, March-April 2010.

Javier Dolado Cosín **y** ***Daniel Rodríguez García*** **(**coordinadores de la sección técnica "*Ingeniería del Software*"**)**

* * * * * *