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*** 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**GRADE ***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**.

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 963330992 <secreval@ati.es>
Administración y Redacción ATI Cataluña
Via Laietana 46, ppal. 1ª, 08003 Barcelona
Tlfn.934125235; fax 934127713 <secregen@ati.es>
Redacción ATI Andalucía
Isaac Newton, s/n, Ed. Sadiel,
Isla Cartuja 41092 Sevilla, Tlfn./fax 954460779 <secreand@ati.es>
Redacción ATI Aragón
Lagasca 9, 3-B, 50006 Zaragoza.
Tlfn./fax 976235181 <secreara@ati.es>
Redacción ATI Asturias-Cantabria <gp-astucant@ati.es>
Redacción ATI Castilla-La Mancha <gp-clmancha@ati.es>
Suscripción y Ventas <http://www.ati.es/novatica/interes.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: "Salida de la habitación 101" - Concha Arias Pérez / © ATI
Diseño: Fernando Agresta / © ATI 2003

Nº 190, noviembre-diciembre 2007, año XXXIII

sumario

Monografía del próximo número: "*Gobierno de las TIC*"

Aprendiendo a introducir una innovación en un proyecto basado en Software Libre

Christopher Oezbek, Lutz Prechelt
Institut für Informatik, Freie Universität Berlin, Alemania

<{oezbek, prechelt}@inf.fu-berlin.de>

Traducción: Ignacio Palomo Duarte (Escuela Superior de Ingeniería, Universidad de Cádiz).

Resumen: *nuestra propuesta trata de investigar la introducción de nuevas tecnologías o innovaciones en el campo de la ingeniería del software en proyectos basados en software libre, (1) para ayudar a los investigadores a evaluar sus herramientas, metodología y diseño de procesos en el mundo real, y (2) para contribuir a que los proyectos basados en Software Libre mejoren sus métodos de trabajo gracias a conocimientos más modernos e innovadores. Esta investigación tratará de ir más allá de la simple difusión de la innovación, adentrándonos en una introducción activa, para aumentar las posibilidades de que el proyecto adopte la nueva tecnología. También hablaremos de la metodología seguida en nuestra investigación, nuestros primeros resultados, las limitaciones de nuestra metodología y por qué los investigadores interesados en evaluar sus propias innovaciones deberían leer este estudio.*

Palabras clave*: adopción de innovaciones, comunidad de software libre, decisión de innovación, invención en Ingeniería de Software, proyecto anfitrión, proyecto de software libre.*

Autores

Christopher Oezbek es diplomado en Informática por la Universidad de Karlsruhe (2002), Master en Informática por el Georgia Institute of Technology (2004) y está cursando el doctorado en la Freie Universität de Berlín. Sus intereses investigadores incluyen los procesos de desarrollo de software libre, documentación de código fuente y los API de usabilidad. Es miembro de ACM y de la asociación de informáticos alemana GI (*Gesellschaft für Informatik*).

Lutz Prechelt es profesor de Informática de la Freie Universität de Berlín desde 2003. Hasta el año 2000, trabajó como investigador senior en la Escuela de Informática de la Universidad de Karlsruhe donde se doctoró en 1995. Después, trabajó en abaXX Technology, Stuttgart, primero como jefe de varios departamentos y después como Director de Tecnología. Sus intereses investigadores incluyen la Ingeniería del Software (usando enfoques empíricos), problemas de medida y comparación, y metodología de la investigación. Actualmente está investigando en desarrollo de software libre, métodos ágiles y plataformas de desarrollo web. Es miembro de IEEE CS, ACM y GI (*Gesellschaft für Informatik*). Asimismo, es editor del *Forum for Negative Results* (FNR) del Journal of Universal Computer Science (J.UCS).

1. Introducción

La mayor parte de las investigaciones en ingeniería del software da como resultado tecnología en forma de herramientas, metodologías o procesos que se aplican en la creación de sistemas software. Poco a poco se va teniendo conciencia de que es necesario evaluar empíricamente estas invenciones para conocer con exactitud los avances conseguidos y obtener reconocimiento fuera del ámbito académico [25][28].

Hay dos métodos clásicos para desarrollar una evaluación empírica: El primero de ellos consiste en pruebas de laboratorio; normalmente se trata de experimentos muy concretos desarrollados por estudiantes. El principal problema es que es muy difícil que estas pruebas sean lo suficientemente imparciales, realistas (sobre todo en las tareas a realizar) y objetivas como para que resulten creíbles, cuando su objetivo precisamente es que lo sean [19]. Los experimentos controlados en temas profesionales son más difíciles de abordar, pero a menudo difícilmente más creíbles. El segundo método consiste en pruebas en el ámbito empresarial, llevadas a cabo como casos de estudio en cooperación con una empresa. Este tipo de estudios gozan de un alto nivel de realismo, pero no están exentos de problemas: el coste y posible riesgo para la empresa hacen difícil encontrar empresas que deseen colaborar. Los acuerdos de confidencialidad hacen difícil poder describir de forma anticipada las hipótesis iniciales y los resultados, y además, la idiosincrasia propia de las empresas dificulta aceptar la generalización de los resultados.

Para muchos propósitos de evaluación, algunos investigadores opinan que realizar observaciones en el contexto de proyectos de software libre (OSS en su acrónimo en lengua inglesa[1]) es un tercer método de evaluación que proporciona unas condiciones ideales de estudio en muchos sentidos: su credibilidad es alta, son fáciles de observar, casi nunca existen restricciones en cuanto a divulgación de la información, las consideraciones sobre el riesgo son más relajadas y las objeciones de coste empresarial se sustituyen por meros obstáculos de voluntad de grupo.

Desafortunadamente, los proyectos OSS no están interesados en estudios, sino en desarrollar software. Así que si deseamos realizar un estudio sobre una nueva tecnología debemos conseguir que el proyecto acepte esa nueva tecnología y la incluya en su trabajo cotidiano. Sin embargo, cualquiera que haya intentado que un grupo de personas adopte una nueva invención (es decir, presentar una invención como una innovación) habrá descubierto que es una tarea realmente difícil.

Así que, en lugar de permitir que un gran número de investigadores prueben, fallen, vuelvan a intentar y al fallar de nuevo se frustren y abandonen la investigación, nuestra sugerencia es hacer el proceso de adopción el centro de nuestro estudio, para proporcionar a los investigadores una metodología probada para introducir una innovación en un proyecto de software libre.

En este estudio se utilizará el término *introducir* con el significado de comenzar un proceso de adopción planificado en una organización o sistema social. Se puede entender que la *adopción* es el punto de inflexión en el que las tecnologías se convierten en

[1] **Nota del Editor:** el artículo se refiere a Open Source Software (OSS) en su sentido más amplio, es decir software abierto y libre para ser modificado. De ahí que traduzcamos ese término por "software libre". En inglés existe el problema de que la palabra "free" es polisémica y puede significar tanto "libre" como "gratuito". De modo que muchos autores de habla inglesa prefieren referirse al software libre como "Open Source Software" en lugar de "Free Software" para que no se confunda con el software gratuito.

innovaciones que se utilizan de una forma común [7]. El término *introducción* contrasta con el término *difusión*, que conlleva connotaciones pasivas, y el verbo *diseminar*, que simplemente se refiere a la distribución de información o recursos.

Desde el punto de vista del investigador, unir la introducción de innovaciones y los proyectos OSS tiene muchas ventajas. Al contrario de lo que ocurre en el ámbito industrial, la visibilidad pública del proceso de creación, sus utilidades auxiliares y su comunicación, así como su apertura a contribuciones de personas externas permiten al desarrollador recoger más información y tener una mayor influencia. Al contrario que con la difusión y diseminación, el investigador puede (1) observar la adopción y utilización de la nueva tecnología en el momento en el que se produce, en lugar de hacer análisis a posteriori, (2) adaptar la tecnología a las particularidades del proyecto para solucionar problemas que a menudo se presentan en las primeras fases de la invención a estudiar y (3) elegir el proyecto para maximizar la perspectiva obtenida.

Por otro lado, desde el punto de vista de la comunidad OSS, este tipo de investigaciones incrementa sus posibilidades de beneficiarse de mejoras en la Ingeniería del Software, dado que los enfoques convencionales en la gestión de la mejora de procesos software como CMMI [5], o incluso enfoques especializados en OSS [8] no precisan cómo se debe realizar la introducción real de mejoras, y los mecanismos de éxito tradicionales como el compromiso y soporte de la dirección [24], es poco probable que funcionen.

El resto del artículo presenta nuestro enfoque investigador para obtener una mejor perspectiva en la introducción de nuevas tecnologías en proyectos OSS, así como nuestros resultados preliminares sobre las siguientes problemáticas de investigación:
1. Cómo seleccionar proyectos apropiados para introducir invenciones en Ingeniería del Software.
2. Cómo acceder a un proyecto para ofrecer una tecnología nueva.
3. Cómo interpretar las reacciones y tomar decisiones estratégicas y tácticas basadas en ellas en el transcurso del proceso de adopción.
4. Cómo reducir la implicación y abandonar el proyecto.
5. Cómo obtener los resultados de la evaluación durante y después de la introducción.

2. Enfoque de la investigación

Para comprender la problemática de la introducción de innovaciones llevaremos a cabo una serie de casos de estudio iterativos [27], utilizando una metodología acción-investigación [2], es decir, un proceso repetitivo y colaborativo de planificación, ejecución y reflexión. Este estudio se aplicará a tres tipos distintos de invenciones y a un gran abanico de proyectos de software libre. Evitaremos la introducción de varias mejoras en un mismo proyecto [9], con la finalidad de evitar posibles sinergias o canibalización entre mejoras [11].

Dentro de cada caso de estudio recabaremos una gran cantidad de datos cualitativos en relaciones causa-efecto y patrones recurrentes (utilizando la metodología de análisis comparativo *Grounded Theory* [6]) para obtener y comprender las claves de las interacciones que se dan a la hora de introducir una nueva tecnología.

Procuraremos minimizar los riesgos que puedan producirse en el proyecto, así como proteger la autonomía y objetividad de los sujetos del estudio [4]. Esto se conseguirá creando un buen ambiente de colaboración, implicación y participación entre el proyecto y el investigador, y garantizando la privacidad y la confidencialidad [3][13]. Aunque se sabe que los proyectos OSS son muy robustos frente a influencias negativas externas, los investigadores que evalúan sus invenciones en un proyecto han de tomar precauciones similares para asegurar que se produzca un comportamiento ético.

3. Cómo elegir el proyecto anfitrión

A la hora de elegir el proyecto OSS adecuado para desarrollar un estudio de este tipo es importante optar por un proyecto que sea (a) lo suficientemente típico como para que se pueda extrapolar la información a otros proyectos, (b) adecuado para la tecnología a implantar y (c) que tenga potencial para la creación de interacciones de interés alrededor de la introducción.

En concreto, el proyecto debe ser libre no sólo en cuanto a licencias de distribución, sino también por estilo de desarrollo: los participantes en el proyecto deberían estar geográficamente distribuidos en lugar de estar ubicados en una misma sede empresarial, la comunicación entre los participantes debe ser pública (y es recomendable que se almacene), debe permitirse la entrada a nuevos desarrolladores, y los procesos y herramientas básicos de trabajo (procesos de publicación, seguimiento de problemas y repositorio de versiones) deben estar bien definidos.

La distribución, observabilidad y apertura del proyecto aseguran que el investigador pueda estudiar la implantación de la tecnología objeto de estudio, mientras que la presencia de procesos y herramientas bien definidos indica que probablemente el proyecto cumple con los estándares profesionales básicos de la Ingeniería del Software, de modo que se puedan generalizar los resultados a otros proyectos de desarrollo de software. Afortunadamente, gracias a la existencia de proyectos anfitriones como SourceForge este tipo de procesos y herramientas son ahora estándar.

Respecto al tamaño del proyecto, éstos deben situarse en un tamaño medio, ni muy grandes ni muy pequeños. Los proyectos pequeños, de menos de tres o cuatro desarrolladores, suelen tener poca interacción, comunicación pobre entre miembros, herramientas y procesos ineficientes, o a veces no disfrutan de una metodología de trabajo bien definida, con lo cual no son recomendables a la hora de realizar un estudio de este tipo (excepto para invenciones muy básicas de ingeniería del software).

En cuanto a los proyectos grandes, de más de cincuenta desarrolladores, presentan el problema contrario: gozan de metodologías de desarrollo bien definidas, de modo que el síndrome del "*Not Invented Here*", el rechazo explícito, el lento proceso de búsqueda de consenso, la escasa percepción de los beneficios frente a los procesos establecidos y la saturación de la comunicación entre los miembros, pueden impedir que se preste atención a un investigador único. Por lo tanto, recomendamos que se opte por un proyecto de tamaño medio: entre cinco y cincuenta desarrolladores, de los cuales como mínimo cinco hayan estado desarrollando activamente en los últimos meses.

Como último requisito, estimamos oportuno elegir un proyecto que se haya mostrado receptivo al cambio (o que, al menos, no haya presentado rechazos de este tipo en el pasado). En muchos casos existe correlación entre esta propiedad y la capacidad del equipo de aceptar nuevos miembros, por lo que se recomienda estudiar los cambios e innovaciones adoptadas por el proyecto en el pasado, como por ejemplo la transición desde el sistema de control de versiones CVS al nuevo (y claramente superior) sistema SVN.

Para elegir un proyecto que cumpla estas características al tiempo que se mantiene cierta aleatoriedad en la elección, se recomienda visitar sitios web de noticias sobre proyectos como Freshmeat ,que agrega proyectos independientemente de su alojamiento, o SWiK, un directorio de proyectos. Ambos sitios permiten visitar aleatoriamente un proyecto de su lista. Mientras que SWiK muestra todo tipo de proyectos de Software Libre, la notable limitación de Freshmeat es que únicamente muestra proyectos OSS que se ejecuten bajo sistemas libres, por lo que no mostrará ningún proyecto que corra exclusivamente bajo plataforma Windows.

4. Cómo acceder a proyectos basados en Software Libre

Existe cierto conocimiento en la literatura sobre cómo acceder a un proyecto OSS [10][26]. Por un lado existe el concepto de "*gift culture*" [21] (cultura del altruismo), que explica que la influencia y el respecto que pueda tener un nuevo miembro del equipo de desarrollo es mayor cuanto más aporte esa persona al proyecto. Esto plantea la pregunta de si la invención en sí misma será vista como un regalo al diseminarse en el proyecto. Un caso de estudio al respecto, en el que se estudió la donación de una tecnología que requería cierto esfuerzo para ser integrada en el código base del proyecto, llegó a la siguiente conclusión: a menos que la donación tenga utilidad inmediata para el proyecto y sea comprensible inmediatamente por parte de los desarrolladores, las probabilidades de éxito serán realmente bajas [20]. Así que podemos suponer que el investigador debe plantearse emplear una considerable cantidad de esfuerzo generando estos beneficios hasta conseguir que la invención sea aceptada y adoptada.

En segundo lugar, el investigador necesita decidir si es recomendable entrar en contacto principalmente con los jefes de proyecto y desarrolladores principales, o es mejor tratar con la comunidad en general. Nuestra hipótesis sugiere que el tipo de acercamiento al proyecto depende de (a) el grado de independencia en la decisión de adopción de cada miembro del equipo, y (b) el nivel de beneficio que represente la innovación al proyecto. A continuación explicamos estos factores.

En "*Diffusion of Innovations*", Rogers distingue tres tipos de decisiones de innovación: *opcional*, en la que cada miembro del equipo de desarrollo es completamente independiente, *colectiva*, en la que se necesita cierto consenso, y *autoritaria*, que es tomada por un pequeño grupo de influencia dentro del proyecto [22].

Como ejemplo contémplese la adopción de un procedimiento en un equipo de desarrollo. La nueva norma que se debe aceptar es la siguiente: "es obligatorio realizar una revisión entre pares antes de enviar cambios (parches) al sistema de control de versiones". Este tipo de cambios suele empezar por una decisión de innovación colectiva para mejorar la calidad del código, dado que se necesita un consenso general para que cada miembro del proyecto envíe sus parches a una lista de e-mail general, y así la comunidad entera debe ser requerida para que promueva la adopción. Además, también implica una decisión de tipo opcional, ya que los miembros del equipo pueden participar y revisar los cambios enviados por los demás, y así puede ser también auspiciada por el investigador hablando individualmente con miembros del grupo. Como ejemplo del tercer tipo de decisión, y las implicaciones que conlleva para un investigador, consideremos el siguiente caso: dos semanas antes del lanzamiento de una nueva versión del proyecto se decide congelarla, es decir, se decide que a partir de este momento no se van a seguir integrando nuevas características, sino simplemente resolviendo fallos y depurando código. Esta decisión la pueden tomar los líderes y mantenedores del proyecto de una forma autoritaria, y soportada técnicamente mediante la creación de una rama local de la versión en el sistema de control de versiones. Los demás miembros del proyecto pueden discrepar de la decisión, pero realmente no pueden realizar ninguna acción al respecto. En este caso, el investigador debe comunicarse directamente con los jefes de proyecto.

La segunda gran característica de la nueva tecnología que puede afectar al acercamiento al proyecto es la estructura del beneficio que ofrece dicha innovación, es decir, la rentabilidad de la inversión o ventaja relativa [22] para cada participante del proyecto en contraste con la rentabilidad de la inversión para el proyecto completo. La documentación del proyecto, por ejemplo, no resulta muy rentable para el desarrollador experimentado que la escribe; en cambio, esa información puede ser tremendamente útil para los nuevos desarrolladores (lo cual puede generar grandes beneficios para el proyecto). Los inventores a menudo comprenden los retornos crecientes [1] prometidos por sus innovaciones, pero suelen pasar por alto que (a) los desarrolladores que conduzcan su introducción para ser compensados por el esfuerzo que realizan y que (b) puede ser realmente difícil medir el beneficio que supone, o que quizás ese beneficio solamente se presente a largo plazo.

Nuestra hipótesis es que el investigador debería empezar el acercamiento al proyecto a través de los miembros que más se puedan beneficiar a corto plazo de la implantación de la nueva tecnología. Así que en lugar de pedir a los miembros del proyecto que realicen tareas con una expectativa negativa a nivel personal, se recomienda que las lleve a cabo inicialmente el mismo investigador. Más tarde, una vez que estas tareas comiencen a generar beneficios visibles que afecten a otros individuos, el investigador tendrá muchas más posibilidades de implicar a otros desarrolladores y dejar de ocuparse de dichas tareas.

5. Cómo interpretar las reacciones y tomar decisiones tácticas y estratégicas

Cuando se introducen invenciones y novedades de cualquier tipo en un grupo social, el investigador debería esperar los siguientes tipos de reacción, tanto a nivel individual como colectivo: *rechazo*, *adopción* y *reinvención* [22].

El rechazo consiste en la decisión de no aceptar la innovación. Este rechazo puede ser activo (después de un estudio o toma de contacto) o pasivo, es decir, rechazar la innovación sin ningún motivo [22]. El rechazo pasivo, es decir la no obtención de respuesta, no es raro que se produzca, incluso en casos en los que el investigador expresa interés explícito en incorporarse al proyecto [26].

La reinvención ocurre cuando miembros del equipo de desarrollo utilizan la nueva tecnología de una forma totalmente inesperada por el investigador. La reinvención es tremendamente beneficiosa para el investigador, ya que puede abrir nuevas puertas a diferentes aplicaciones de la tecnología.

Por supuesto, todavía hay muchos aspectos y consideraciones referentes a la interacción entre desarrolladores, investigadores y tecnología hasta que se producen estos últimos pasos o reacciones. Existe mucha documentación en cuanto a ingeniería social, describiendo modelos que pueden encajar en estos desarrollos, como la teoría de campos (*theory of fields*) [12] o la teoría red-actor (*network-actor*) [17]. En nuestro caso hemos elegido el modelo de innovación desarrollado por Denning y Dunham [7]. En este modelo, el proceso de innovación comienza (1) detectando las posibilidades de cambio y (2) intentando saber qué se puede obtener de ese cambio. (3) ofreciendo esta visión a las personas afectadas (u otras unidades de adopción) y recibiendo su retroalimentación, permitiendo así transformar la idea inicial en algo que puede ser (4) derivando un producto, proceso o mejora social a partir de la ejecución e implementación de la idea. Solamente después de que la innovación haya sido (5) adoptada por la población objetivo y (6) se mantenga como una novedad positiva podremos decir que la introducción exitosa de la innovación ha sucedido. En el caso concreto que presenta este estudio, las dos primeras fases se centrarán en confeccionar y desarrollar el problema, la visión y la invención más que en intentar generar nuevas ideas e implementarlas.

6. Cuándo y cómo abandonar el proyecto

El momento de abandonar el proyecto será cuando se dé uno de los siguientes casos: el investigador ha tenido éxito implantando la innovación en un grupo de desarrollo determinado, y la tecnología se mantiene sin ayuda del investigador, o por el contrario la implantación ha fallado y no queda nada que arreglar o solucionar. En el caso de que hayamos tenido éxito, la retirada ha de producirse de forma gradual y no abrupta, so

pena de poner en peligro ese éxito y causar daño al proyecto. Por otra parte, después de una introducción fallida, el abandono del proyecto obliga al investigador a "limpiar", es decir revertir los cambios en el código base y realizar una retirada "ordenada".

7. Cómo obtener los resultados de la evaluación

Los resultados que se puedan obtener de este tipo de investigaciones dependen mucho de la naturaleza de la innovación a implantar y de cuál sea la meta final de la investigación. Para algunas innovaciones, la adopción con éxito puede ser suficiente. Para otras, quizás necesitemos comparar productos, procesos o métricas de uso con los valores previos a la introducción. Otro tipo de innovación puede que necesite que los desarrolladores rellenen una encuesta sobre su experiencia con la nueva tecnología.

Independientemente de estos tres enfoques, la forma con la que probablemente el investigador obtendrá más información útil, de calidad y práctica para mejorar la innovación será la comunicación directa con el equipo de desarrollo del proyecto. Un investigador que utilice la técnica de investigación-acción podría considerar esta información como su principal resultado.

8. Oportunidades, limitaciones y conclusión

Finalmente se nos plantea la pregunta de si la experiencia obtenida al introducir una nueva tecnología en un proyecto OSS se puede extrapolar a otros casos (validez externa). Por ejemplo, en proyectos de diferentes tamaños, dominios de aplicación, arquitecturas de software, con personal no voluntario, distintas configuraciones de gestión, distribución, otros ambientes de trabajo, experiencia previa en ingeniería del software, etc. El caso más típico es el estudio de un entorno corporativo dependiente de los ingresos. Los siguientes argumentos abogan porque la evaluación de resultados de proyectos OSS pueda transferirse a tales entornos: (1) Los desarrolladores de software libre se caracterizan por ser críticos ante los resultados académicos, (2) La existencia de liderazgo en la dirección y otras motivaciones externas (como compensaciones económicas) pueden a menudo impulsar la adopción y uso, y (3) los empleados a tiempo completo se beneficiarán más de las economías de escala y de los efectos del aprendizaje que desarrolladores de software libre trabajando a tiempo parcial.

La limitación más importante de nuestra investigación son las restricciones de las características concretas que debe presentar la nueva tecnología para que sea adecuada para la investigación. La literatura sobre difusión de la innovación señala varios atributos de la invención que afectarán a su tasa de éxito al ser introducida: (1) la compatibilidad de la invención con tecnologías, valores y creencias preexistentes (por ejemplo, los equipos de desarrollo de software libre pueden rechazar trabajar con herramientas que no estén a su vez licenciadas como software libre), (2) su complejidad técnica e intelectual, (3) la facilidad con que se puedan observar sus resultados, (4) la posibilidad de experimentar con la tecnología antes de ponerse a trabajar con ella, y (5) la incertidumbre sobre la innovación [22].

Halloran y Scherlis mantienen que los proyectos OSS diferencian mucho entre contribuciones de confianza y contribuciones que no generan confianza (metáfora del servidor amurallado, en inglés *"walled server"*), y que las invenciones necesitan preservar esta distinción para ser aplicables a un proyecto OSS [15]. Esto se puede interpretar de la siguiente manera: Una introducción exitosa de una tecnología puede significar que la tecnología es realmente valiosa; en cambio, una introducción sin éxito puede indicar que el proyecto OSS no presentaba las características idóneas (por ej., "el servidor amurallado"), y no que la tecnología no sea interesante.

Una segunda limitación que queremos remarcar es que, al contrario de otros trabajos de campo o estudios etnográficos llevados a cabo en grandes empresas (ver como ejemplo [18]), aquí será difícil estudiar las prácticas y procesos reales de cada participante en el proyecto, ya que sólo los resultados intermedios y los del proceso, tales como informes de errores, envíos a CVS y discusiones en listas de correo, son visibles al investigador. Para obtener información sobre cuánto se utiliza cada herramienta en cada ordenador de cada desarrollador es necesario instrumentarlo de forma adecuada [16][23].

Una tercera limitación del estudio se refiere a la velocidad de adopción de la tecnología. Los proyectos de software libre suelen ser desarrollados por personal voluntario, que suelen trabajar menos de 10 horas semanales en el proyecto, y se coordinan de forma asíncrona entre diferentes zonas horarias [14]. Por lo tanto, la velocidad de cambio es mucho más lenta que en un ambiente comercial, en el que los desarrolladores trabajan un número regular de horas y se comunican frecuentemente de forma síncrona.

Resumiendo, nuestra intención ha sido estudiar la introducción de innovaciones en el campo de la Ingeniería del Software, para ayudar a otros investigadores a evaluar herramientas, métodos y procesos desarrollados en un ambiente académico, y hemos ofrecido unos resultados preliminares. Mientras que la comunidad investigadora se puede beneficiar del acceso a entornos de trabajo real y de la posibilidad de interactuar con la comunidad del Software Libre, dicha comunidad recibe tecnología punta ajustada a sus problemas específicos por parte de los inventores.

Referencias

[1] W. B. Arthur. *Increasing Returns and Path Dependence in the Economy*. University of Michigan Press, 1994. ISBN: 0472064967.

[2] D. E. Avison, F. Lau, M. D. Myers, P. A. Nielsen. Action research. *Commun. ACM, 42(1):94–97*, 1999.

[3] M. Bakardjieva, A. Feenberg. Involving the virtual subject. *Ethics and Information Technology, 2(4):233–240*, 2001.

[4] J. Cassell. Ethical principles for conducting fieldwork. *American Anthropologist, 82(1):28–41*, Marzo 1980.

[5] CMMI Product Team. CMMI for development, version 1.2. Technical Report CMU/SEI-2006-TR-008, Software Engineering Institute, 2006.

[6] J. M. Corbin, A. Strauss. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology, 13(1):3–21*, Mar. 1990.

[7] P. J. Denning, R. Dunham. Innovation as language action. *Commun. ACM, 49(5):47–52*, 2006.

[8] S. Dietze. Modell und Optimierungsansatz für Open Source Softwareentwicklungsprozesse. Tesis doctoral, Universität Potsdam, 2004.

[9] G.W. Downs, L. B. Mohr. Conceptual issues in study of innovation. *Administrative Science Quarterly, 21(4):700–714*, 1976.

[10] N. Ducheneaut. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work (CSCW), V14(4):323–368*, Agosto 2005.

[11] M. L. Fennell. Synergy, influence, and information in the adoption of administrative innovations. *Academy Of Management Journal, 27(1):113–129*, 1984.

[12] N. Fligstein. Social skill and the theory of fields. *Sociological Theory, 19(2):105–125*, Julio 2001.

[13] M. S. Frankel, S. Siang. Ethical and legal aspects of human subjects research on the internet. Published by AAAS online , June 1999.

[14] R. A. Ghosh, B. Krieger, R. Glott, G. Robles, T. Wichmann. Free/Libre and Open Source Software: Survey and Study – FLOSS. Final Report, International Institute of Infonomics
University of Maastricht, The Netherlands; Berlecon Research GmbH Berlin, Germany, Junio 2002.

[15] T. J. Halloran, W. L. Scherlis. High Quality and Open Source Software Practices. En J. Feller, B. Fitzgerald, F. Hecker, S. Hissam, K. Lakhani, and A. van der Hoek, editors,
Meeting Challenges and Surviving Success: The 2nd Workshop on Open Source Software Engineering, pages 26– 28. ACM, 2002.

[16] P. M. Johnson, H. Kou, J. Agustin, C. Chan, C. Moore, J. Miglani, S. Zhen, W. E. J. Doane. Beyond the personal software process: metrics collection and analysis for the differently disciplined. En ICSE '03: *Proceedings of the 25th International Conference on Software Engineering, pages 641–*

646, Washington, DC, USA, 2003. IEEE Computer Society.
[17] J. Law. Notes on the theory of the actor-network: Ordering, strategy and heterogeneity. *Systems Practice, 5(4):379–393*, 1992.
[18] T. C. Lethbridge, J. Singer. Experiences conducting studies of the work practices of software engineers. En H. Erdogmus and O. Tanir, editors, *Advances in Software Engineering: Comprehension, Evaluation, and Evolution, pages 53–76*. Springer, 2001.
[19] D. E. Perry, A. A. Porter, L. G. Votta. Empirical studies of software engineering: a roadmap. In *Proceedings of the conference on The future of Software engineering, pages 345–355*. ACM Press, 2000.
[20] L. Quintela García. Die Kontaktaufnahme mit Open Source Software-Projekten. Eine Fallstudie. Bachelor thesis, Freie Universität Berlin, 2006.
[21] E. S. Raymond. *The Cathedral and the Bazaar*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1999. ISBN: 0596001088.
[22] E. M. Rogers. Diffusion of Innovations. Free Press, New York, 5th edition, August 2003. ISBN: 0743222091.
[23] F. Schlesinger, S. Jekutsch. ElectroCodeoGram: An environment for studying programming. En Workshop on Ethnographies of Code, Infolab21, Lancaster University, UK, Marzo 2006.
[24] D. Stelzer, W. Mellis. Success factors of organizacional change in software process improvement. *Software Process: Improvement and Practice, 4(4):227–250*, 1998.
[25] W. F. Tichy, P. Lukowicz, L. Prechelt, E. A. Heinz. Experimental evaluation in computer science: A quantitative study. *Journal of Systems and Software, 28(1):9–18*, Enero 1995.
[26] G. von Krogh, S. Spaeth, K. Lakhani. Community, joining, and specialization in open source software innovation: A case study. *Research Policy, 32:1217–1241(25)*, Julio 2003.
[27] R. K. Yin. *Case Study Research: Design and Methods*. Applied Social Research Methods. Sage Publications, Inc., 1988.
[28] M. V. Zelkowitz, D. R. Wallace. Experimental models for validating technology. *Computer, 31(5):23–31*, 1998.