Actas XVIII JENUI 2012, Ciudad Real, 10-13 de julio 2012
I.S.B.N. 10: 84-615-7157-6 | I.S.B.N. 13:978-84-615-7157-4
Páginas 209-214

Dos años y 31.000 líneas de código después: BabeliumProject, una experiencia práctica en el desarrollo de PFCs

Juanan Pereira, Inko Perurena, Silvia Sanz,
Julián Gutiérrez
Departamento de Lenguajes y Sistemas Informáticos
Universidad del País Vasco, Euskal Herriko Unibertsitatea
(UPV/EHU)
Paseo de Manuel Lardizábal, 1
20018 San Sebastián
{juanan.pereira,inko.perurena,silvia.sanz,julian.gutierrez}@ehu.es

Resumen

BabeliumProject parte con un claro objetivo: construir un sistema abierto que permita la práctica de la expresión oral en segundas lenguas.

En el desarrollo de este proyecto, realizado en gran parte a partir de PFCs, ha intervenido un equipo de 10 alumnos y 3 profesores y cuenta en la actualidad con cientos de usuarios que han practicado su expresión oral en 3 idiomas (Español, Euskera e Inglés).

En este trabajo se muestran las principales lecciones aprendidas a lo largo del período 2010-2012, usando para ello un proceso de mejora basado en un sencillo test de calidad de proyectos software, que han permitido gestionar con éxito 10 PFCs para obtener un proyecto de código abierto de las dimensiones de Babelium.

Summary

BabeliumProject has a clear goal: to build an open system that allows to practice speaking skills in second languages.

The development of this project has comprised a team of 10 students working in their Master Thesis and 3 teachers in the management and tutoring roles. As of today, Babelium has hundreds of users that have practiced their oral production in 3 languages (Basque, English and Spanish).

This paper shows the main lessons learned over the 2010-2012 period that have been applied by iterating over a simple process to improve the quality of software projects. This process has allowed us to successfully manage and deliver a considerably huge open source project like Babelium.

Palabras clave

Aprendizaje, idiomas, pfc, software libre, ingeniería software

1. Motivación

Los proyectos fin de carrera que defienden los alumnos de ingeniería informática (ahora proyectos fin de grado con los nuevos planes de estudio), suelen terminar en muchas ocasiones en las bibliotecas de las distintas facultades.

Por otro lado, en muchos casos, el objetivo es crear un nuevo programa desde cero, y la gran mayoría de las veces, estos trabajos son ejecutados por una única persona. Pero la realidad del mercado laboral actual no es ésta, y sería deseable que la formación de los ingenieros informáticos tuviese en cuenta este aspecto y se aprovecharan los nuevos grados, y en especial el trabajo fin de grado (antiguo PFC) para "entrenar" a los alumnos y enseñarles a desenvolverse en un entorno lo más parecido al mundo laboral real. En general, sería recomendable que los alumnos adquirieran competencias que les ayudaran a formarse como verdaderos ingenieros: supieran enfrentarse a código pre-existente, aprendieran a leerlo [8] y lograran desenvolverse en programas con decenas o cientos de miles de líneas de código fuente, para encontrar y depurar errores en ellos, documentar nuevas características y discutir las especificaciones , el diseño y su implementación en un equipo de trabajo.

Aunar los requerimientos de un PFC con las competencias indicadas no parece una labor sencilla. Y realmente, no lo es. Sin embargo, sí es viable hoy en día gracias a la existencia de múltiples proyectos de software libre. Nuestra tesis afirma que los alumnos de PFC mejorarán sus competencias trabajando sobre proyectos de software libre, mejorándolos mediante la corrección de fallos (*bugs*), mediante la adición de nuevas funcionalidades, aprendiendo del código fuente y buenas prácticas del resto de

desarrolladores del proyecto. Y todo ello en una lengua franca: el inglés.

En este trabajo mostramos una experiencia práctica, realizada durante 2 años, con 10 alumnos de PFC de la Facultad de Informática de San Sebastián, trabajando sobre un proyecto que, partiendo desde 0, ha llegado a tener cerca de mil usuarios y una base de código libre de 31.000 líneas. Basándonos en un conocido test de calidad de equipos de desarrollo software, se muestran las características que los alumnos implicados en proyectos de este estilo deberían tener para poder conseguir resultados satisfactorios, tanto en la calidad del proyecto final como en el aprendizaje de competencias por parte del alumno.

El artículo finaliza con las conclusiones sobre el trabajo realizado, haciendo especial hincapié en las posibles mejoras en el proceso de formación de los alumnos, durante la ejecución del PFC.

2. BabeliumProject

BabeliumProject nace en la Facultad de Informática de San Sebastián hace 2 años, con un claro objetivo, construir un sistema de código abierto (licencia GPLv3), basado en el trabajo con vídeos interactivos, que permita la práctica de la expresión oral en segundas lenguas siguiendo una forma de trabajo colaborativa.

BabeliumProject permite al usuario visualizar video-conversaciones grabadas en distintos idiomas (inicialmente Español, Euskera e Inglés), donde intervienen por lo general 2 ó 3 personas. El usuario, una vez visualizado uno de los vídeos, puede tomar el papel de uno de los personajes y realizar el doblaje del mismo. Es decir, una vez seleccionado el rol que quiere doblar, el vídeo comienza a visualizarse hasta que llega el turno de dicho rol; en ese momento se graba la voz del usuario (a través del micrófono y, si el usuario lo desea, también la imagen de su webcam). Tal y como ocurre en la vida real, el usuario dispone de un tiempo limitado para responder, exactamente el mismo que el tiempo utilizado por el rol del personaje en el vídeo original (ver Figura 1).

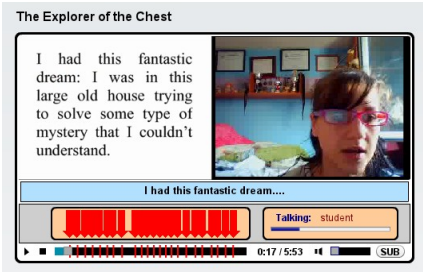

Figura 1. Una alumna trabajando la expresión oral con la aplicación BabeliumProject

Cuando el usuario termina el doblaje de la conversación, tiene la opción de visualizar su trabajo y volver a grabarlo tantas veces como quiera. En cuanto determina que la grabación es correcta, la puede subir al servidor Babelium de tal forma que queda disponible para su evaluación por parte de otros usuarios. Esta evaluación se realiza de forma colaborativa, es decir, si la grabación se ha hecho en euskera, serán usuarios euskaldunes los que evalúen el trabajo. Esta evaluación puede incluir, además de 5 ítems obligatorios que evalúan aspectos básicos del idioma, comentarios explícitos de texto y comentarios en vídeo, que pueden ser especialmente interesantes para correcciones fonéticas.

Babelium es una aplicación RIA (Rich Internet Application) donde la parte cliente se ha implementado en Flex (un framework de desarrollo de aplicaciones multimedia que hace uso del lenguaje ActionScript para la lógica de negocio y MXML para el interfaz gráfico y que al compilar genera una aplicación Flash ejecutable en el navegador) y la parte back-end mediante servicios PHP que hacen de interfaz entre el cliente y los servidores multimedia (Red5) y de bases de datos (MySQL).

La interacción con el usuario, a través de la grabación de su producción de voz con micrófono y webcam, ha sido una tarea compleja dada la necesidad de sincronizar dos fuentes de vídeo simultáneas: el vídeo-ejercicio que está disponible en la aplicación con los momentos en los que ha de grabarse la respuesta del usuario dentro de ese vídeo. Esta sincronización de vídeo y audio ha de hacerse en tiempo real, usando el protocolo RTMP (Real Time Multimedia Protocol), generando los streams de vídeo asociados en el servidor Red5 y guardando los metadatos en el servidor MySQL, de tal forma que toda esta información sea

accesible para el usuario que vaya a evaluar la producción oral de los alumnos.

El código fuente de Babelium, a día de hoy, acumula más de 31.000 líneas de código (15.000 en ActionScript, 10.000 en MXML y 6.000 en PHP), desarrolladas durante los últimos 2 años y medio, gracias al trabajo efectuado por parte de un equipo de 10 alumnos a través de sus PFC y 3 profesores en labores de gestión y dirección del proyecto. En la actualidad Babelium está implantado en varias escuelas de idiomas y asignaturas de inglés de la UPV/EHU, donde casi mil usuarios han practicado su expresión oral en los 3 idiomas disponibles.

Coordinar un equipo de 13 personas a lo largo de este tiempo, consiguiendo que el resultado de 10 PFC's haya generado una aplicación usada a nivel profesional, no ha sido fácil. Para conseguirlo, se han aplicado distintas técnicas y criterios de calidad. El seguimiento para un correcto desarrollo del proyecto se ha hecho usando el *test de Joel* para la medición de la calidad de un equipo de desarrollo software. Las siguientes secciones presentan dicho test y cómo se ha aplicado el mismo dentro del equipo Babelium.

3. Midiendo la calidad de los equipos de desarrollo: el *test de Joel*.

Joel Spolsky es autor de varios libros sobre ingeniería software [9][10] que recopilan artículos, entre otros, sobre la gestión de proyectos software. Spolsky fue también director de programación del equipo de Microsoft Excel entre 1991 y 1994. En 2001 publicó el conocido como Test de Joel, un test diseñado para evaluar el nivel de calidad de un equipo de desarrollo software. El test es sencillo, consiste en responder con un sí o un no a las siguientes 12 preguntas sobre el equipo de desarrollo:

1. ¿Usa un sistema de control de versiones?
2. ¿Puede construir el proyecto en un solo paso?
3. ¿Compila el proyecto todos los días?
4. ¿Tiene una base de datos de *bugs*?
5. ¿Arregla los *bugs* antes de escribir código nuevo?
6. ¿Tiene un calendario actualizado?
7. ¿Tiene una especificación?
8. ¿Los programadores trabajan en un entorno silencioso?
9. ¿Usa las mejores herramientas que el dinero puede comprar?
10. ¿Tiene *testers* (es decir, gente probando el software)?
11. ¿Los nuevos candidatos escriben código durante las entrevistas?
12. ¿Hace sencillas pruebas de usabilidad?

Según Spolsky, conseguir 12 puntos (1 punto por cada respuesta afirmativa) es perfecto, 11 tolerable, pero 10 o menos implicaría que en el equipo de desarrollo existen serios problemas. A pesar de la sencillez del test, al usarlo para medir la calidad de un equipo de trabajo en PFCs y aplicar las medidas correctoras consiguientes, la experiencia que describimos en este trabajo muestra que el nivel de calidad de los proyectos y las competencias alcanzadas por los alumnos mejoran considerablemente.

4. Aplicación del test de Joel en el equipo de PFCs de BabeliumProject

Los profesores del proyecto Babelium, tras modularizar las funcionalidades requeridas, ofrecieron el desarrollo de las mismas como PFCs a alumnos de Ingeniería Informática e Ingeniería Técnica Informática. Los alumnos que querían participar, tenían que pasar obligatoriamente una entrevista donde se les solicitaba que implementaran un pequeño problema relacionado con la tecnología que iban a usar durante el proyecto (punto 11 del test). No es necesario que los alumnos conozcan a fondo ningún lenguaje, ya que lo que esta prueba busca medir no es tanto la calidad del código (que también), sino la capacidad del alumno para autodirigir su formación, su capacidad de búsqueda de la información y resolución de problemas, así como su capacidad de comunicar resultados. Estas competencias genéricas de los candidatos son muy bien valoradas por las empresas y por cualquier equipo de desarrollo. Por lo tanto, si el alumno era capaz de entregar un resultado razonable en el tiempo acordado (junto con una explicación de cómo lo ha logrado, para evitar picarescas), era admitido en el grupo de desarrollo y se le asignaba un director de PFC.

La siguiente tarea consiste en formar al alumno en sistemas de control de versiones distribuidos (SCVD) - punto 1 del test -. Lamentablemente, encontramos que muy pocos alumnos habían trabajado previamente con un sistema de control de versiones y en aquellos casos en los que sí lo habían hecho, había sido con sistemas de versiones centralizados. En el equipo

Babelium creemos en la eficacia de estos sistemas dentro de un equipo de desarrollo, y lo consideramos prácticamente imprescindible en proyectos de grandes dimensiones como el que nos ocupa[1]. En concreto, no es viable seguir guardando versiones en memorias USB sin posibilidad de deshacer cambios o llevar una trazabilidad de quién ha hecho qué. Conseguir esto último permite además llevar un control de la velocidad del proyecto y por tanto, permite realizar estimaciones más cercanas a la realidad. El gestionar el código mediante un sistema de control de versiones ofrece además sentar las bases para poder responder afirmativamente al resto de preguntas del test.

Los alumnos tenían que dar cuenta de sus avances periódicamente, depositando sus aportaciones de código y asegurándose de que éstas compilaban, se integraban con el resto del código y pasaban las pruebas unitarias perfectamente. Por supuesto, todo este trabajo estaba automatizado a través de un sistema de integración continua (CI o Continuous Integration). En concreto, se ha usado el servidor de CI Jenkins [5]. Este sistema permite la compilación diaria (o cuando ocurre cierto evento, por ejemplo, un commit en el repositorio público del proyecto) y ejecución de pruebas de integración y unitarias, así como el despliegue de la aplicación en el servidor de destino (cumpliendo así los puntos 2 y 3 del test). Los alumnos aprenden así a trabajar en un entorno en el que no están solos, donde hay ciertas reglas que seguir para asegurar la calidad del código que generan y donde el esfuerzo de su trabajo se ve recompensado de forma inmediata mediante la exposición pública del mismo. Este hecho tiene además otra consecuencia positiva: los alumnos saben a ciencia cierta que su código funciona y disponen de una dirección pública para demostrarlo (por ejemplo, ante un tribunal de defensa de proyecto).

La experiencia nos dice, y los propios alumnos lo confirman, que la gestión de tareas de un proyecto era hasta el momento un mero trámite burocrático que debían pasar para aprobar ciertas asignaturas que lo requerían. Les bastaba con cumplimentar un diagrama de Gantt usando una herramienta como Microsoft Project, cuadrar las fechas para que parecieran razonables y a partir de ahí, gestionar las tareas como buenamente podían.

[1]El proyecto Babelium gestiona su código fuente utilizando el SCVD Mercurial, a través de la URL pública http://code.google.com/p/babeliumproject .

En Babelium no se les exige un diagrama de Gantt, pero sí que gestionen sus tareas a través de una herramienta de trabajo en grupo: el sistema de control de *issues* (tareas y *bugs*) de Google Project Hosting (http://code.google.com/hosting). En concreto, las tareas son descritas en detalle: quién la hará, qué hará, para cuándo (para qué iteración), cómo saber si la tarea se ha cumplido (tests de aceptación o unitarios asociados en caso de tratarse de código) y cuál es la prioridad de la tarea. A medida que el proyecto avanza, gran parte de estas tareas habrán generado *bugs* que formarán nuevos *issues*. La idea es que aquellos *issues* de mayor prioridad pasan a una pizarra compartida por el equipo de desarrollo, donde se genera una ficha por cada una de ellos (ver Figura 2).

Figura 2. Pizarra de tareas a realizar y ya finalizadas

Cualquier miembro del proyecto sabe exactamente en qué está trabajando cualquier compañero/a y cuánto tiempo le ha consumido cada tarea. En general, el director de PFC no permite que un desarrollador asuma nuevas tareas hasta que su lista de *bugs* esté finalizada (puntos 4 y 5) . El análisis de las tareas finalizadas ayuda también a estimar semanalmente la velocidad de desarrollo, así como a verificar la bondad de las estimaciones de esfuerzo que cada alumno realizó sobre cada tarea.

Los alumnos tienen que especificar en la lista de *issues* una estimación del tiempo que les llevará cada tarea y para qué iteración del proyecto se prevé que esté finalizada (punto 6). Estos datos son públicos y cualquier miembro del equipo puede rebatir o discutir esas estimaciones.

Los alumnos de ingeniería informática saben realizar correctamente especificaciones de las nuevas funcionalidades que hay que implementar, dado que han trabajado extensivamente en este aspecto durante las asignaturas de "Ingeniería del Software" y de "Análisis y Diseño de Sistemas de

Información". Las memorias que deben crear como soporte a la defensa del PFC reflejan este hecho. La novedad en Babelium es que toda nueva especificación tenía que ser consensuada previamente en la lista de distribución de los desarrolladores del proyecto. Esta lista sólo admite mensajes en inglés - salvo excepciones. Los archivos de esta lista sólo son accesibles para los miembros del equipo de desarrollo, y se muestran especialmente útiles para las nuevas incorporaciones (punto 7).

Respecto al entorno de trabajo silencioso (punto 8) se ha observado que de forma inconsciente suelen cumplirlo dado que por lo general suelen dedicarse al desarrollo del proyecto desde casa, como trabajo personal, en lugar de implementar en las salas de informática de la facultad, o en las zonas comunes de conexión habilitadas. Estas dos últimas localizaciones suelen albergar a multitud de alumnos simultáneamente, con el consiguiente alboroto y potenciales distracciones que harían que el alumno perdiera mucho tiempo y perdiera el foco de atención constantentemente.

¿Disponen los alumnos de las mejores herramientas que el dinero puede comprar? (punto 9). En general podríamos decir que así es, gracias a las licencias educativas que muchas empresas otorgan a sus productos (es el caso de Adobe Flash Builder, https://freeriatools.adobe.com/, la herramienta utilizada para el desarrollo en Flex y PHP, o el de Visual Paradigm, http://www.visual-paradigm.com/partner/academic/, aplicación usada para documentar los distintos diagramas que sugiere el proceso unificado de desarrollo software). Sin embargo, no siempre es posible adquirir las mejores herramientas que el dinero puede comprar y el equipo ha de saber adecuarse a las limitaciones que eso impone, tal y como sucede en la vida real. Por ejemplo, el servidor multimedia más usado y con más soporte para la emisión de vídeos en streaming es sin duda Flash Media Server (de la empresa Adobe, actualmente propietaria del formato de archivos Flash SWF y del formato de vídeo FLV), pero el coste por licencia de uso es de unos 3.000$ por servidor (http://cort.as/1bw3), algo prohibitivo para el alumno. Teniendo en cuenta, además, que para comprender y mejorar el funcionamiento interno de cualquier aplicación no hay nada mejor que disponer de su código fuente bajo una licencia que permita su estudio y modificación, la alternativa Red5 como servidor multimedia de código abierto, ha sido, en nuestro caso, de gran ayuda. La documentación de Red5 no es tan completa como la de FMS y a pesar de ser un producto con muchos años de desarrollo, aún no ha alcanzado una versión 1.0. Sin embargo, esto ha hecho también que los alumnos hayan tenido que interactuar - en inglés - con el grupo de desarrolladores de Red5, preguntando y aportando reportes de *bugs*, documentación y funcionalidades.

El código fuente de Babelium se ha liberado bajo una licencia libre GPLv3. El beneficio de poder trabajar en un PFC programando una aplicación de este estilo estriba en que cualquier empresa o entidad interesada puede reutilizar el código para su uso interno. Así ha sido el caso de Babelium, dado que desde el momento en que se programaron y probaron las funcionalidades básicas, comenzó a ser utilizado tanto por escuelas de idiomas como dentro de la asignatura de inglés de las universidades de Navarra (UPNA) y del País Vasco (UPV/EHU). Este hecho permitió disponer constantemente de decenas (o centenares en algunos momentos) de usuarios probando la aplicación a fondo. Estos usuarios hacían labores de "beta-testing" continuo (punto 10), encontrando errores y *bugs* en el código. Además, en ocasiones, los alumnos de PFC debían de lidiar directamente con el usuario final, normalmente mediante explicaciones por correo electrónico y con el ánimo de encontrar una forma de reproducir los errores detectados. Esto permitía desarrollar, también, la competencia de comunicación con el cliente, con los problemas que esto suele conllevar en ocasiones.

Algunos de los centros de enseñanza de idiomas en los que se ha implantado Babelium, han actuado realmente como *beta testers,* siendo totalmente conscientes de que estaban probando una aplicación generada por el trabajo fin de carrera de alumnos con escasa o nula experiencia profesional. De este modo, cada fin de mes, las personas responsables de implantar el proyecto en el centro se reunían por video-conferencia con el equipo de desarrollo de Babelium para sugerir mejoras tanto en el apartado de usabilidad como en las funcionalidades del proyecto. Estas peticiones de mejora pasaban a formar parte de las tareas a realizar, con su responsable correspondiente, la iteración en la que se esperaba tenerla disponible (en su caso) y la descripción del resultado esperado. Gracias a estas mejoras constantes en la usabilidad del proyecto (punto 12) se ha ido refinando el código y haciéndolo

más atractivo para su implantación en nuevos centros educativos.

5. Resultados obtenidos

Aplicando el test de Joel de forma iterativa y mejorando la puntuación obtenida por el grupo de alumnos de PFC en cada iteración, se ha conseguido mantener una base de código de nivel empresarial y cerca de 1.000 usuarios registrados como resultado. Los alumnos se han sentido muy motivados en su desarrollo y han conseguido finalizar sus objetivos, en general con excelentes resultados académicos.

Nos gustaría destacar algunas frases de las memorias del PFC de algunos alumnos que recogen esta información:

"La dinámica de trabajo en el proyecto ha sido otro de los puntos clave para realizar el proyecto. Normalmente, un proyecto llevado a cabo entre varias personas suele ser difícil y a veces pesado, pero no ha sido el caso del grupo de desarrolladores de BabeliumProject. Cualquier duda, bien fuera de nuestro código o de otro desarrollador, era debatida vía mail casi al momento, con varias personas ofreciendo su ayuda u opinión para resolverla. El ambiente de trabajo ha sido inmejorable y en ningún momento se tuvo la sensación de estar aislado." I.J. (2011)

"Respecto al grupo de BabeliumProject, decir que aunque por propia experiencia, trabajar en un proyecto en grupo puede llegar a ser más que pesado y difícil, los desarrolladores que hay detrás de este grupo han facilitado mucho la elaboración de las tareas y han agilizado la evolución del proyecto. He trabajado muy a gusto junto al resto de compañeros con los que he tenido que intercambiar conocimientos para resolver alguna que otra tarea (tanto mía como suya)." I.L.(2011)

Aunque Babelium comenzó su desarrollo a finales de 2009, la orientación de PFCs al desarrollo de funcionalidades en grandes proyectos de código abierto no es nueva [7]. De hecho, grandes empresas y organizaciones relacionadas con el mundo del software libre como [2], [11] y [3] promocionan el desarrollo de PFCs mediante concursos de programación con premios en metálico.

6. Oportunidades de mejora

En el futuro, el deseo de los autores de este artículo sería seguir con esta línea de trabajo, formando a los alumnos del Grado en Ingeniería Informática para que sepan desenvolverse en grandes proyectos de software libre.

Consideramos que cualquier alumno que sepa crear especificaciones y discutir su calidad en público, dentro de un equipo de desarrollo internacional, implementar las especificaciones sugeridas y/o corregir *bugs* en aplicaciones de cientos de miles de líneas de código (como LibreOffice, Firefox, Eclipse...), así como compilar, probar y desplegar una versión final de su trabajo para distintas plataformas y sistemas operativos y en distintos idiomas, estará más que capacitado para enfrentarse a cualquier otro trabajo relacionado con la ingeniería software fuera del ámbito académico.

Sería deseable que este esfuerzo no se desarrollara simplemente durante el trabajo fin de grado, sino que convendría establecer y acordar el contenido de distintas asignaturas y prácticas, de tal forma que se guiara al alumno desde el comienzo de su andadura académica en el Grado. Algunas experiencias fructíferas en este sentido son: el proyecto OpenOffice.org, que a través de uno de sus desarrolladores, ya ha trabajado con PFCs en el desarrollo de nuevas funcionalidades para ese paquete ofimático libre desde el año 2008, mediante acuerdos con la École Centrale de Nantes, Francia [1]. También el proyecto [6] mantiene la iniciativa "Open Source Comes to Campus" a través de la cual ayuda a los estudiantes universitarios a introducirse en el mundo del desarrollo de software libre, impartiendo de seminarios prácticos y gratuitos.

Otro posible aspecto de mejora sería el de establecer normas que estipularan un mínimo compromiso del alumno hacia proyecto. Estas normas podrían estar basadas en la experiencia de gestión de nuevos proyectos de la Fundación Apache (http://www.apache.org). En efecto, Apache dispone de una sección llamada Apache Incubator donde acoge nuevos proyectos de desarrollo, ofreciéndoles infraestructura y apoyo durante los primeros 2 años de lanzamiento. A cambio, solicita que cumplan ciertos requisitos para demostrar que el proyecto sigue adelante. Uno de ellos es el de generar periódicamente un informe de situación, que incluya, además, indicadores de avance, gestionados a través de un pane de control automatizado (http://incubator.apache.org/clutch.html) como son: número de días desde el último *commit* en el sistema de control de versiones, número de errores corregidos desde el último informe, nueva documentación de interés generada, etc.