Revista
Española de

Innovación, Calidad e Ingeniería del Software

REICIS

Volumen 4, Número 2 (especial X JICS), septiembre, 2008

Web de la editorial: www.ati.es/reicis
E-mail: editor-reicis@ati.es
ISSN: 1885-4486

Publicado por la Asociación de Técnicos de Informática
www.ati.es

Revista Española de Innovación, Calidad e Ingeniería del Software (REICIS)

Contenidos

REICIS

Editorial **4**

Luís Fernández-Sanz, Juan J. Cuadrado-Gallego

Presentación **5**

Luis Fernández-Sanz

Hacia la gestión cuantitativa en la gestión de proyectos en el ámbito de las pymes **7**

Jose A. Calvo-Manzano, Iván García y Magdalena Arcilla

Problemas de las pymes en el nivel 2 de madurez. Una muestra sesgada **20**

Juan José Cukier

Mejora de procesos organizativos: análisis estadístico **33**

Izaskun Santamaria, Teodora Bozheva, Iñaki Martínez de Marigorta

Revisiones de código en el contexto del aseguramiento de calidad. Un caso práctico **46**

María José Escalona, Manuel Pérez-Pérez, O. González-Barroso, J. Ponce, J. M. Correa, A. I. Merino

Diagnóstico de la situación de la calidad del software en la industria española **58**

Elena Argüelles, Antonio Sepúlveda

ACCESIBILIDAD WEB: un vistazo a tres webs de administraciones públicas en España **70**

Jorge Sánchez, Tanja E.J. Vos

Infraestructura de pruebas para una plataforma de inteligencia de negocios: lecciones aprendidas de una experiencia académica **82**

Ruth Alarcón, Carla Basurto, Abraham Dávila

Perfiles del ciclo de vida del software para pequeñas empresas: los informes técnicos ISO/IEC 29110 **96**

José A. Calvo-Manzano, Javier Garzás, Mario Piattini, Francisco J. Pino, Jesús Salillas, José Luis Sánchez

Estudio experimental de la conversión entre las unidades de medición funcional del software puntos de casos de uso e IFPUG **109**

Juan J. Cuadrado-Gallego, María J. Domínguez-Alda, Marian Fernández de Sevilla, Miguel Ángel Lara

Making Software Process Management Agile **122**
José Manuel García, José Javier Berrocal, Juan Manuel Murillo

La norma ISO/IEC 25000 y el proyecto KEMIS para su automatización con software libre **135**
José Marcos, Alicia Arroyo, Javier Garzás y Mario Piattini

Modelo de calidad para herramientas FLOSS que dan apoyo al modelado de procesos del negocio **148**
Leslibeth Pessagno, Kenyer Domínguez, Lornel Rivas, María Pérez, Luis E. Mendoza, Edumilis Méndez

Editorial

REICIS

El grupo de Calidad del Software de ATI ha consolidado su posición como principal promotor de la disciplina de ingeniería y calidad del software con la décima edición de las Jornadas sobre Innovación y Calidad del Software (las tradicionales JICS). Estas X JICS pretenden además potenciar la presencia iberoamericana en este foro de promoción de la cultura de la calidad del software y de la innovación en el desarrollo de sistemas y aplicaciones por lo que constituyen la promoción de una I Conferencia Iberoamericana de Calidad del Software (CICS). Por otra parte, las X JICS incorporan la presencia de la ponencia de un destacado experto europeo en la disciplina de ingeniería de software como es Darren Dalcher, Director del UK National Centre for Project Management en la Middlesex University y editor de la revista Software Process Improvement and Practice.

Por otra parte, queremos resaltar la línea de calidad de los trabajos, eminentemente prácticos pero rigurosos, aceptados entre los remitidos en la convocatoria de contribuciones: las ponencias aceptadas (con una tasa de rechazo del 40%) han sido sometidos a un completo proceso de revisión por el comité de programa así como a una cuidadosa labor de revisión de estilo, de terminología y de ortotipografía para garantizar el mejor resultado para nuestros lectores. Por supuesto, no cabe olvidar el apoyo de los patrocinadores (Telelogic, Steria, Deiser, GESEIN y SOGETI) no sólo aportando recursos sino también interesantes presentaciones de experiencias prácticas de sus expertos. Los debates promovidos en las mesas redondas así como la promoción de las actividades de comunicación y *networking* entre los participantes, tanto a nivel presencial como a través de la lista de distribución, los medios electrónicos y la nueva oferta formativa con plataforma *e-learning*. En definitiva, el evento más completo con toda la información disponible en la página del grupo de Calidad del Software (www.ati.es/gtcalidadsoft) acorde a la trayectoria pionera en España que, desde 1997, está proporcionando, a través de la Asociación de Técnicos de Informática, el apoyo para la productividad y la calidad en los proyectos de software. Este perfil ha sido reconocido por el apoyo del Ministerio de Industria, Turismos y Comercio con su apoyo institucional dentro de la convocatoria de la orden ITC/390/2007. Por último, debemos resaltar la aportación de datos de gran importancia no sólo mediante los eventos organizados sino también a través de la realización de estudios específicos (por ejemplo, sobre las prácticas de pruebas, el diseño de casos y los factores que dificultan su implantación eficiente y eficaz en las organizaciones) que permiten un mejor conocimiento de la práctica real de esta disciplina en España.

Luis Fernández Sanz
Juan J. Cuadrado-Gallego
Editores

Presentación

REICIS

En este número especial de septiembre de 2008 de REICIS, por primera vez en la historia de nuestra revista, esta publicación se convierte en el vehículo de difusión del evento decano en España en el ámbito de la ingeniería y la calidad del software: las Jornadas de Innovación y Calidad del Software (JICS) que alcanzan así su décima edición desde su inicio en 1998. En esta ocasión, el Grupo de Calidad del Software de ATI (www.ati.es/gtcalidadsoft) no sólo ha querido cumplir con esta decena de ediciones sino que ha apostado por una apertura a nuevos retos como la presencia de eminentes ponentes invitados de gran presencia internacional y la potenciación de los vínculos iberoamericanos para convertir a este evento en la referencia sobre calidad del software en la amplia comunidad latina. Los trabajos aceptados han sido sometidos a un completo proceso de revisión por el comité de programa así como a una cuidadosa labor de revisión de estilo, terminología y ortotipografía para garantizar la mejor calidad para nuestros lectores. Este número especial constituye en definitiva la publicación de las actas de las X JICS y, por ello, cuenta con un tamaño mayor del habitual. Esperamos repetir este número especial el próximo año con la undécima edición de las Jornadas de Innovación y Calidad del Software. Agradecemos la labor del comité de programa coordinado por la Dr. M.Idoia Alarcón (Universidad Autónoma de Madrid) y compuesto por la siguiente lista de expertos:

- Antonia Mas (Universitat de les Illes Balears)
- Luis de Salvador (AGPD)
- Ricardo Vargas (Universidad del Valle de Méjico)
- Javier Tuya (Universidad de Oviedo)
- Antonio de Amescua (Universidad Carlos III de Madrid)
- María Moreno (Universidad de Salamanca)
- José Antonio Calvo-Manzano (Universidad Politécnica de Madrid)
- José Antonio Gutiérrez de Mesa (Universidad de Alcalá)
- Isabel Ramos (Universidad de Sevilla)
- Esperança Amengual (Universitat de les Illes Balears)
- José Ramón Hilera (Universidad de Alcalá)
- Mercedes Ruiz (Universidad de Cádiz)
- María Teresa Villalba (Universidad Europea de Madrid)
- Adolfo Vázquez (INSA)
- María José Escalona (Universidad de Sevilla)
- Ana Araújo (Ministerio de Medio Ambiente)
- Antonio Rodríguez (Telelogic)
- Gurutze Miguel (TQS)
- Beatriz Pérez (Centro de Ensayos de Software, Uruguay)
- José Javier Martínez (Universidad de Alcalá)
- José Díaz (SSQTB)

Luis Fernández Sanz

Revista Española de Innovación, Calidad e Ingeniería del Software, Vol.4, No. 2, 2008

Infraestructura de pruebas para una plataforma de inteligencia de negocios: lecciones aprendidas de una experiencia académica

Ruth Alarcón, Carla Basurto, Abraham Dávila
Departamento de Ingeniería. Pontificia Universidad Católica del Perú
{ruth.alarcong, cbasurto, abraham.davila}@pucp.edu.pe

Abstract

The software testing is a process that is regularly conformed by repetitive tasks, that is why automating them becomes attractive and convenient in different situations. In our country many enterprises develop software with adjusted schedules, not considering enough time to perform tests adequately. A project for developing a platform for business intelligence should consider test automation and the implementation of a testing infrastructure of the platform which will have many versions. This paper contains some experiences about the implementation of a test infrastructure for business intelligence platform; both were developed in academic and business environments.

Key words: software testing, web testing, functional testing, business intelligence, automation software testing.

Resumen

La prueba de software es un proceso que, por lo general, tiene tareas que se repiten y cuya automatización resulta atractiva y conveniente en diversas situaciones. En nuestro medio, muchas empresas desarrollan software a medida con plazos ajustados que no les permite realizar pruebas de manera adecuada. La construcción de una plataforma de inteligencia de negocios es un proyecto en el que resulta muy conveniente automatizar las pruebas y la implementación de una infraestructura de pruebas de la plataforma que tendrá sucesivas versiones. Este artículo recoge algunas experiencias surgidas al implementar una infraestructura de pruebas para una plataforma de inteligencia de negocios, desarrolladas ambas en entornos académicos y empresariales.

Palabras clave: pruebas de software, pruebas en web, pruebas funcionales, inteligencia de negocios, automatización de pruebas de software.

1. Introducción

La prueba de software, según IEEE [1], es una actividad en la que un sistema o componente se ejecuta bajo condiciones específicas, se observa o registran los resultados y se realiza

una evaluación de un aspecto del sistema o componente. Para el SWEBOK [2], se trata de una actividad que se realiza para evaluar y mejorar la calidad del producto a través de la identificación de defectos y problemas. En general, podemos decir que las pruebas contribuyen a obtener un mejor producto o un producto con menos defectos.

A partir de lo presentado por varios autores, como Presmman [3], Sommerville [4] y Futrell [5], se entiende que las pruebas se realizan en las etapas finales de cada iteración, o incluso a finales del proyecto; esto es casi independiente del ciclo de vida de construcción del software. Asimismo, muchas organizaciones que desarrollan software a medida no cuentan con tiempo suficiente para poder ejecutar las pruebas de manera adecuada [6]; esto provoca que los productos tengan una cantidad significativa de defectos que salen a la luz cuando el software es utilizado por los usuarios. El desarrollo de un producto de software a nivel comercial, para una gran cantidad de usuarios y con varias evoluciones en el futuro, suele seguir el mismo proceso que cualquier otro proyecto de desarrollo de software; sin embargo, dada su orientación de producto comercial, es razonable pensar que habrá más interés en la automatización de las pruebas con respecto a otros proyectos, como el desarrollo de software a medida.

Por otro lado, un grupo de profesores y estudiantes, con el apoyo de la empresa ACKLIS SAC, que cuenta con el apoyo de la Universidad, desarrolló una plataforma de inteligencia de negocios [7] en dos esfuerzos sucesivos. En el primer esfuerzo del proyecto se desarrolló la versión en cliente-servidor de la plataforma y en un segundo esfuerzo, la versión para web. Durante este segundo esfuerzo se planteó la necesidad de contar con una infraestructura de pruebas que permitiera apoyar el trabajo de todo el equipo, sobre todo de cara a las futuras evoluciones del producto.

En este artículo se presentan las experiencias recogidas durante la construcción de una infraestructura de pruebas de software que se ha desarrollado para una plataforma de inteligencia de negocios. La infraestructura permitirá realizar pruebas de regresión con casos de prueba funcionales y pruebas de carga. Tanto la plataforma como la infraestructura han sido desarrolladas por un equipo de estudiantes y profesores bajo el esquema de dos proyectos separados.

El trabajo se organiza así: en el segundo epígrafe se describe la plataforma de inteligencia de negocios; en el tercero se describe la infraestructura de pruebas desarrollada;

en el cuarto se recogen algunas buenas y malas prácticas encontradas en el desarrollo de la plataforma y que impactaron en el desarrollo de la infraestructura; finalmente, se presenta una discusión final sobre el tema.

2. BUFEO, la plataforma de inteligencia de negocios

La plataforma de inteligencia de negocios BUFEO es un proyecto que se inició en el año 2003 y se desarrolló en varias fases y cada fase, en varias etapas. La primera fase consistió en desarrollar una primera versión sobre una arquitectura cliente-servidor para permitir que el equipo de desarrollo se familiarizase con los aspectos funcionales de la inteligencia de negocios y técnicos de la programación. El proyecto se dividió en tres frentes de trabajo para cubrir los módulos de análisis, extracción y explotación. La integración de los módulos se realizó principalmente a nivel de archivos, pues los procesos no dependían entre sí. La segunda fase consistió en desarrollar la versión para una arquitectura web; en este caso se planteó un proyecto integrado desde el principio, teniendo en cuenta el conocimiento adquirido sobre inteligencia de negocios y la parte técnica. La segunda fase contó con el apoyo parcial del fondo del programa PROCOM del Consejo Nacional de Ciencia y Tecnología del Perú (CONCYTEC) (http://www.acklis.com/bufeo/).

La plataforma cubre los servicios propios de inteligencia de negocios desde la extracción, transformación y carga de datos al cubo de análisis, así como su posterior utilización (explotación de datos) a través de reportes y consolidados a diversos niveles; incluye herramientas de modelado y análisis, no siendo necesarios otros productos para que una empresa pueda utilizar inteligencia de negocios a nivel básico. La empresa, al utilizar BUFEO, se tiene que preocupar de realizar el modelado de negocios, las reglas de transformación y las salidas (reportes) en la plataforma.

La plataforma BUFEO está dominada principalmente por los tres componentes funcionales definidos al principio: análisis, extracción y explotación. En la Figura 1 se presentan, además de los componentes principales, los componentes control, gráficos y común que han sido desarrollados por el proyecto y otros de uso libre. En la figura 2 se muestran algunas clases de análisis, en especial de empresas por su carácter multiempresarial; proyecto para las distintas iniciativas de inteligencia de negocios de una

organización; tema, tablas (hechos y dimensiones) y campos para la definición de los almacenes de datos (Data Warehouse).

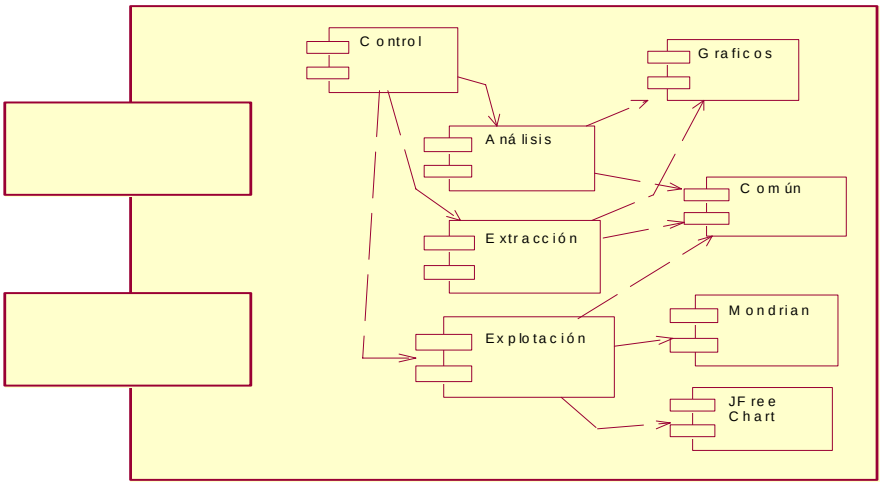

Figura 1. Arquitectura de BUFEO.

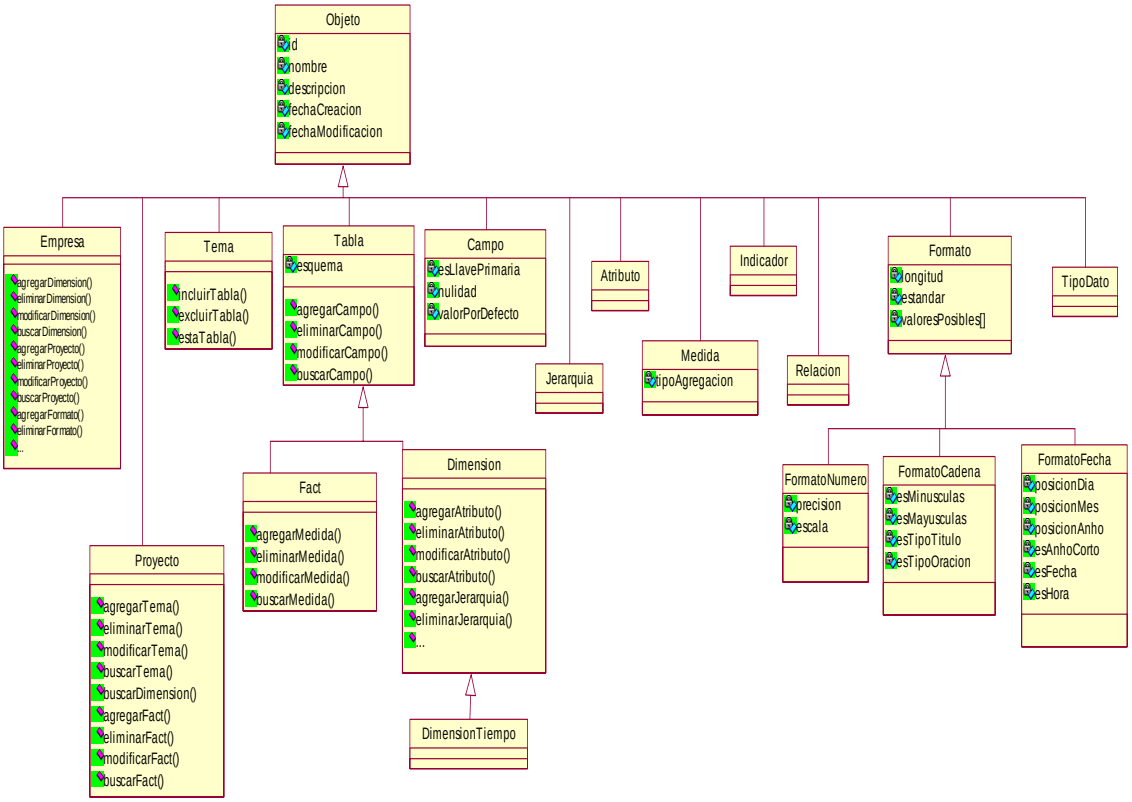

Figura 2. Diagrama de clases. Vista parcial de análisis.

3. tBUFEO, la infraestructura de pruebas de BUFEO

tBUFEO es la infraestructura de pruebas desarrollada expresamente para la plataforma de inteligencia de negocios BUFEO, con el objetivo de automatizar la ejecución de las pruebas de software. BUFEO está en su primera versión [7]; se trata de un producto comercial que evolucionará (crecerá y mejorará), por lo que resulta necesario y conveniente contar con una infraestructura de pruebas cuyo crecimiento se corresponda con la plataforma. El nombre de la infraestructura tiene una *t* al inicio para indicar que se trata de *testing* (pruebas); con *BUFEO* se indica que es exclusivo a BUFEO.

En la primera etapa del proyecto, las pruebas del software se realizaron de manera manual, a partir de un catálogo de pruebas registrado en una hoja electrónica. Para la segunda etapa se decidió automatizar la ejecución de las pruebas, desarrollándose el proyecto de la infraestructura de forma paralela al desarrollo de la plataforma. Esta decisión se basó en el hecho de que algunas pruebas se tornaron repetitivas —introducir los mismos datos para apreciar idénticos resultados—, porque eran sensibles a la automatización y porque, en algunos cambios de la plataforma, había que volver a probar varios aspectos del producto (pruebas de regresión) y pruebas de carga, tal como las entienden Pezzé [8] y Culbertson [9]. La infraestructura de pruebas, dentro de lo planteado, permitirá probar la plataforma BUFEO incluso a nivel de los modelos de negocios que más adelante se tiene previsto desarrollar sobre la plataforma.

En la construcción de tBUFEO se han incorporado, en primer lugar, las pruebas de caja negra (funcionales) para comprobar que se cumple los requisitos del producto; en segundo lugar, las pruebas referidas con grandes cantidades de datos (stress, volumen, carga) y, por último, un grupo de pruebas referentes a modelos de negocios que se desarrollarán sobre la plataforma (estas últimas pruebas están pendientes, porque el soporte a los modelos aún no ha sido desarrollado). Considerando la arquitectura por capas de BUFEO (interfaz, negocio y persistencia), las pruebas en la infraestructura se han desarrollado principalmente para la capa de negocios, dejando a un lado la capa de interfaz gráfica de usuario.

La infraestructura tBUFEO se planteó a partir de los siguientes requisitos (se enumeran los más relevantes):

- tBUFEO debe ser una infraestructura de software que haga posible ejecutar las pruebas de la plataforma de inteligencia de negocios BUFEO y que permita crecer en número de casos de prueba.
- tBUFEO debe permitir realizar pruebas funcionales y de manejo de grandes cantidades de datos, utilizando una definición de casos de prueba en archivos basados en XML.
- tBUFEO ha de posibilitar administrar la ejecución de los casos de prueba que se aplicarán sobre la plataforma.

A partir de aquí, se planteó un conjunto de casos de uso de tBUFEO teniendo como actor a un informático (probador), tal como se aprecia en la Figura 3. La arquitectura de la infraestructura se basa en un esquema que corresponde a la implementación de conductores de pruebas (test-drivers), de manera paralela a la propia arquitectura de la plataforma (Pezzé [8], McGregor [10]). Los programas que son los test-drivers se escriben de manera correspondiente con cada clase y los datos, al ser usados en las pruebas, se obtienen de archivos de datos en XML.

La infraestructura (ver Figura 3) tiene tres servicios principales: (i) ejecutar los casos de prueba que se encuentren registrados en la infraestructura —lo cual incluye seleccionar los casos que se van a probar y los casos que no se van a probar—; (ii) incorporar nuevos casos de prueba a la plataforma, siguiendo un esquema de adición de archivos en formato XML donde se tengan los datos que se usarán; (iii) eliminar casos de prueba que hayan quedado obsoletos a través de los sucesivos cambios de la plataforma. Este esquema se ha seguido para los tres grandes componentes de la plataforma BUFEO. Los casos de pruebas a nivel de unidad van desde probar algunos métodos complejos hasta probar grupos (cluster) de clases, e incluso componentes. A modo de ejemplo (ver Figura 4) se presenta la clase puEmpresa01, que se corresponde con una clase que probará a las clases asociadas a la clase Empresa, como son las clases ControlEmpresa y GestorEmpresa. Igualmente, se presenta la clase puJob01 (ver Figura 5), que corresponde a la clase que probará a las clases del BEJob, como son las clases Empresa, Proyecto, BEControlJob, ControlTema, ControlProyecto y ControlEmpresa; ambas clases de pruebas tienen como ámbito de prueba

un cluster de clases. Las Figuras 4 y 5 corresponden a los módulos de extracción y explotación. En el apéndice se presenta un extracto de dos clases que implementan test-drivers para probar empresas.

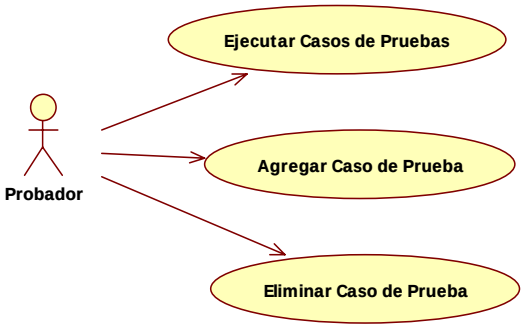

Figura 3. Diagrama de casos de uso de la infraestructura de tBufeo.

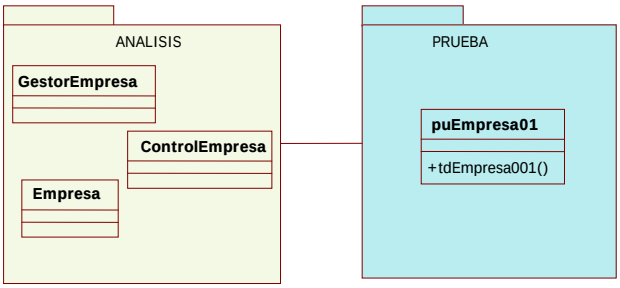

Figura 4. Test-driver de la clase Empresa.

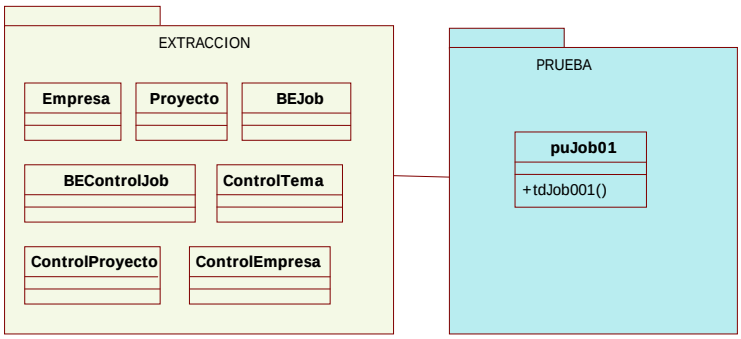

Figura 5. Test-driver de la clase Job.

4. Situaciones encontradas y lecciones aprendidas

Durante la implementación progresiva de tBUFEO se han realizado las pruebas (ejecución de tBUFEO) de la plataforma de inteligencia de negocios BUFEO con los resultados que siguen.

4.1. Situaciones encontradas

- Las funcionalidades del software que implican registro de datos a través de pantallas tipo formularios (o que implican un registro simple de datos) son las primeras que se han implementado. Para estos casos, el desarrollo de los test-drivers ha sido relativamente fácil y rápido, pues los desarrolladores de BUFEO han seguido los conceptos de tres capas a través del patrón Modelo Vista Controlador. Los test-drivers de la infraestructura implementan clases para manipular las clases Controladoras o Gestoras de acuerdo con cada situación.
- Las funcionalidades del software que implican el uso de una interfaz gráfica más elaborada, como suelen ser los servicios de modelado (por ejemplo, diagrama estrella o diagramas de transformación de datos), han representado una mayor complejidad que el caso anterior. Esta complejidad se debe al hecho de que la implementación de estos diagramas manipulan directamente la generación de los datos (almacenados en archivos en formato XML), dejando de lado la parte de las clases Controladoras o Gestoras.
- Las funcionalidades que implican la fijación de parámetros (criterios) para consultas (o reportes) son relativamente similares al primer caso. Sin embargo, la complejidad viene del lado de la comprobación de los resultados. En algunos casos se ha manejado una comprobación de resultados finales y en otros se ha hecho la comprobación de los datos que aparecen en la consulta (reporte). La automatización es posible, pero se ha dejado para una siguiente etapa la comprobación de los datos.
- En las pocas ocasiones en las que se ha tenido documentación no actualizada, inconsistente o inexistente, pues se trata de dos proyectos (la plataforma y la infraestructura) con relativa independencia y en paralelo, ha habido mayor trabajo, pues ha sido necesario revisar y entender de manera constante el código fuente para implementar los test-drivers.

4.2. Lecciones aprendidas

- Un programa bien desarrollado, que respete el diseño y que se encuentre debidamente documentado, permite trabajar con los test-drivers con mayor facilidad, rapidez y agrado; en caso contrario surge dificultad, lentitud y molestia. Ambas situaciones se encontraron en la plataforma. El equipo supervisó el proyecto para asegurarse de que el programa estuviese adecuadamente desarrollado en todos los casos. En los casos sencillos, los desarrolladores sí respetaron buenas prácticas, pero en algunos casos más complejos no todos lo hicieron.
- Para desarrollar una infraestructura de pruebas es necesario que el equipo que la construye tenga competencias técnicas en pruebas de software y, al menos, las mismas competencias que el equipo que desarrolló la plataforma (BUFEO). Esto es determinante para lograr construir test-drivers que interactúen con las clases que se utilizarán para manipular la porción de la plataforma que interesa, manteniendo la independencia entre ambos productos (plataforma e infraestructura de pruebas).
- Un programa con documentación deficiente supone que se tiene que revisar el código fuente y acarrea pruebas en falso, es decir, intentos de probar y encontrar que el test-driver no ha sido implementado de manera adecuada. Todo esto se traduce en demoras de trabajo y molestias para las personas involucradas en la construcción de la infraestructura.
- Los test-drivers que corresponden a casos con persistencia deben diseñarse de tal manera que sea posible volver a ejecutarlos y permitan hacer la comparación de manera automática. Por eso los test-drivers se han desarrollado, en primer lugar, fijando los datos en la persistencia y luego, realizando la operación con persistencia, verificando los valores finales con los valores esperados y, finalmente, borrando la operación.

5. Discusión final y trabajo futuro

Construir la infraestructura ha permitido identificar algunas lecciones sobre cómo se debe desarrollar y qué cosas se deben cuidar en un desarrollo de este tipo de producto, que

evolucionará en el tiempo. Para algunos casos ha sido posible trabajar directamente con la capa de Negocio, pero en otros casos no ha sido tan fácil porque no se han seguido de manera disciplinada estas separaciones en algunas porciones de BUFEO.

Una infraestructura de pruebas basada en test-drivers es útil para un software relativamente grande como es la plataforma de inteligencia de negocios. Más adelante se espera hacer uso de jUnit (http://www.junit.org/) y comparar el esfuerzo sin él. De manera análoga, para el caso de pruebas de volumen de datos se utilizará un software comercial y se comparará frente a lo desarrollado. En ambos casos se contará con equipos diferentes al que construyó tBUFEO.

Agradecimientos

Este proyecto ha sido parcialmente apoyado por PROCOM-CONCYTEC R.P. N° 080-2006-CONCYTEC-P, el Centro de Innovación y Desarrollo Emprendedor de la Pontificia Universidad Católica del Perú, y la empresa ACKLIS SAC.

Referencias

[1] IEEE, *IEEE Std. 610-12:1990. Glossary of Software Engineering Terminology*, IEEE, 1990.

[2] IEEE, *Guide to the Software Engineering Body of Knowledge*, IEEE, 2004.

[3] Presmman, R., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 2005.

[4] Sommerville I., *Software Engineering*, Addison Wesley, 2000.

[5] Futrell, S., Shafer, D. y Shafer, L., *Quality Software Project Management*, Prentice Hall, 2002.

[6] Valencia, G., *Subcontratación de las pruebas de software*, GreenSQA. http://www.greensqa.com/index.php?option=com_content&task=view&id=13&Itemid=30 (consultado en junio de 2008)

[7] González, D., Dávila, A., Basurto, C., Flores, L. y Morales, L., *Proyecto Sistema Software para la Aplicación de Inteligencia de Negocios en Organizaciones Medianas y Pequeñas*. http://www.acklis.com/bufeo/ (consultado en enero de 2008)

[8] Pezzé, M. y Young M., *Software Testing and Analysis, Process, Principles and Techniques*, Wiley, 2008.

[9] Culbertson, R., Brown C. y Cobb G., *Rapid Testing*, Prentice Hall, 2002.

[10] McGregor, J., *CMU/SEI-2001-TR-02. Testing a Software Product Line*, Software Engineering Institute. Carnegie-Mellon University, 2001.

Apéndice A

Programa ejemplo de los test-drivers de la infraestructura.

```
/* METODO: tdEmpresa001
 * OBJETIVO: Crear la Empresa.
 * @return boolean, true si la prueba fue correcta, caso contrario es false. */
public boolean tdEmpresa001() {
 boolean blnResultadoObtenido = true;
 long lngIdEmpresa = 0;
 try {
  /*1.- Creando Empresa */
  Empresa objEmpresa1 = cargardatosEmpresa.cargarDatosEmpresa001("XEMP");
  lngIdEmpresa = objEmpresa1.getId();
  /*2.- Obteniendo datos de Empresa creada*/
  GestorEmpresa objGestorEmpresa = new GestorEmpresa();
  Empresa objEmpresa2 = objGestorEmpresa.buscarEmpresa(objEmpresa1.getId());
  /*3.- Comprobando*/
  if((objEmpresa1.getId()==objEmpresa2.getId())&&
  (objEmpresa1.getNombre().equals(objEmpresa2.getNombre()))&&
  (objEmpresa1.getDescripcion().equals(objEmpresa2.getDescripcion()))&&
  (objEmpresa1.getDireccion().equals(objEmpresa2.getDireccion()))&&
  (objEmpresa1.getRUC().equals(objEmpresa2.getRUC()))&&
  (objEmpresa1.getRubro()== objEmpresa2.getRubro())&&
  (objEmpresa1.getRepresentanteLegal().equals(objEmpresa2.getRepresentanteLegal())))
  { blnResultadoObtenido = true;
  } else {
  blnResultadoObtenido = false; }
  /*4.- Eliminando datos cargados*/
  eliminardatosEmpresa.eliminarDatosEmpresa(objEmpresa1);
 } catch (Exception e) {
  e.printStackTrace(); }
```

```
 return blnResultadoObtenido;
}
/**
 * METODO:  cargarDatosEmpresa001
 * OBJETIVO: Crear la Empresa.
 * @param astrIdentificadorEmpresaxPrueba.
 * @return Empresa, devuelve la Empresa creada. */
 public static Empresa cargarDatosEmpresa001 ( String astrIdentificadorEmpresaxPrueba )
throws Exception{
 AppletServlet objAppletServlet = new AppletServlet();
 /*1.- Creando Empresa*/
 Empresa objDatosEmpresa = null;
 long lngIdEmpresa = 0;
 try {
  /*2.- Obtiene Empresa*/
objDatosEmpresa=ControlEmpresa.agregarEmpresa("BUFEO"+strIdentificadorEmpresaxP
rueba, "descripcion", "dir","111", 3456, "www.bipucp.pucp.edu.pe", "su abogado",1);
  lngIdEmpresa = objDatosEmpresa.getId();
} catch (Exception e) {
  e.printStackTrace();
}
/*3.- Devolviendo la empresa creada*/
  return objDatosEmpresa;
}
```