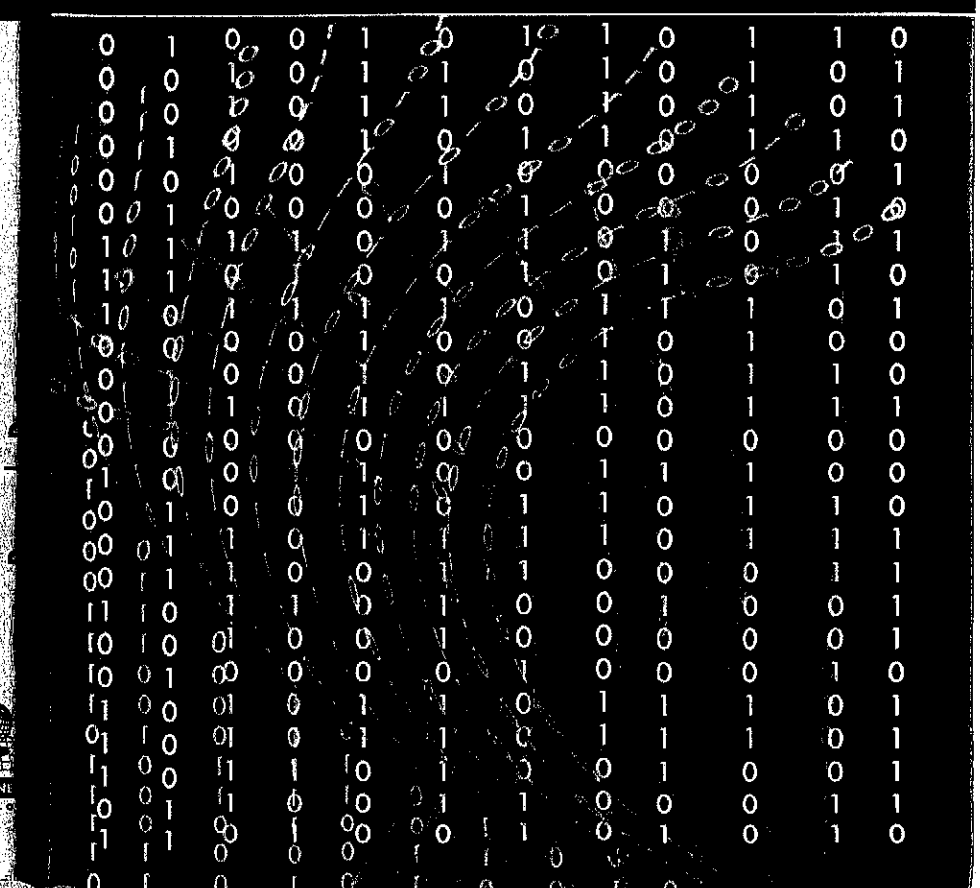
JISBD
2006
5008

Editores:
José C. Riquelme, Pere Botella

Ingeniería del Software y Bases de Datos

Editores:
José C. Riquelme, Pere Botella



UNIVERSITAT POLITÈCNICA DE CATALUNYA

FIB

gencat
www.gencat.net

INTERSYSTEMS

MINISTERIO DE EDUCACIÓN Y CIENCIA

Microsoft

CIMNE
BARCELONA

Ingeniería del Software

Ingeniería del Software y Bases de Datos

Actas de las
XI Jornadas de Ingeniería
del Software y Bases de Datos

Sitges, 3 al 6 de Octubre de 2006

Editores:

José C. Riquelme
Pere Botella

Publicado por

CIMNE
BARCELONA

Ingeniería del Software y Bases de Datos

Sitges, 3 al 6 de Octubre de 2006

Comité Ejecutivo JISBD 2006

Presidente del Comité Organizador
Pere Botella (Universitat Politècnica Catalunya)

Presidente del Comité De Programa
José C. Riquelme (Universidad de Sevilla)

Secretario Comisión Permanente
Mario Piattini (Universidad de Castilla-La Mancha)

Coordinador de Tutoriales
Xavier Franch (Universitat Politècnica deCatalunya)

Coordinador de Talleres
Antonio Ruiz (Universidad de Sevilla)

Ingeniería del Software y Bases de Datos

Primera edición, Septiembre 2006

Gran Capitán s/n, 08034 Barcelona, España
www.cimne.upc.es

Impreso por: Artes Gráficas Torres S.A., Morales17, 08029 Barcelona, España

Depósito legal: B-42461-2006

ISBN: 84-95999-99-4

Comité de Programa JISBD 2006

Jesús Aguilar (U. Sevilla)
José F. Aldana (U. Málaga)
Bárbara Álvarez (U. P. Cartagena)
María J. Aramburu (U. Jaume I)
Joao Araujo (U. Nova De Lisboa)
Orlando Belo (U. Do Minho)
Rafael Berlanga (U. Jaume I)
Pere Botella (U. P. Catalunya)
Nieves Brisaboa (U. Coruña)
Coral Calero (U. Castilla-La Mancha)
Carlos Canal (U. Málaga)
José M. Cavero (U. Rey Juan Carlos)
Matilde Celma (U. P. Valencia)
Rafael Corchuelo (U. Sevilla)
Dolors Costal (U. P. Catalunya)
Yania Crespo (U. Valladolid)
Carlos Delgado (U. Carlos III)
Oscar Díaz (U. País Vasco)
Javier Dolado (U. País Vasco)
Joao Falcão e Cunha (U. Porto)
Xavier Franch (U. P. Catalunya)
Pablo de la Fuente (U. Valladolid)
Lidia Fuentes (U. Málaga)
Mario J. Gaspar da Silva (U. Lisboa)
Marcela Genero (U. Castilla-La Mancha)
Juan Gómez (U. Alicante)
Alfredo Goñi (U. País Vasco)
Jon Iturrioz (U. País Vasco)
Elena Jurado (U. Extremadura)
Natalia Juristo (U. P. Madrid)
Antonia Lopes (U. Lisboa)
Adolfo Lozano (U. Extremadura)
Henrique Madeira (U. Coimbra)
Esperanza Marcos (U. Rey Juan Carlos)
Eduardo Mena (U. Zaragoza)
Ana Moreira (U. Nova De Lisboa)
Ana M. Moreno (U. P. Madrid)
Juan J. Moreno (U. P. Madrid)
Juan M. Murillo (U. Extremadura)
Oscar Pastor (U. P. Valencia)
Ernesto Pimentel (U. Málaga)
Ángeles Places (U. Coruña)
Antonio Polo (U. Extremadura)
Carme Quer (U. P. Catalunya)
Celia Ramos (U. Algarve)
Isidro Ramos (U. P. Valencia)
Isabel Ramos (U. Sevilla)
Antonio Rito (U. Técnica De Lisboa)
María J. Rodríguez (U. Granada)
Francisco Ruiz (Castilla-La Mancha)
Fernando Sánchez (U. Extremadura)
Juan Sánchez (U. P. Valencia)
Sofia Sousa Brito (I. P. Beja)
Ernest Teniente (U. P. Catalunya)
Miguel Toro (U. Sevilla)
Ambrosio Toval (U. Murcia)
Juan C. Trujillo (U. Alicante)
Javier Tuya (U. Oviedo)
Toni Urpi (U. P. Catalunya)
Antonio Vallecillo (U. Málaga)
Belén Vela (U. Rey Juan Carlos)

Comité Organizador
(U. P. Catalunya)

Alberto Abelló
Claudia Ayala
Xavier Burgués
Jordi Conesa
Dolors Costal
Cristina Gómez
Gemma Grau
Carme Martín
Joan Antoni Pastor
Paola Pizzi (Secretaría organización CIMNE)
Carme Quer
Angels Rius
Ruth Raventós
Toni Urpi

Revisores Adicionales

Alberto Abelló
Álvaro E. Prieto
Amparo Navasa
Ángel Herranz
Antònia Mas
Antonio Cesar Gómez
Antonio Ruiz
Arantza Illarramendi
Artur Boronat
Cesar J. Acuña
Clara Benac Earle
Cristina Vicente CHicote
Daniel Gomes
Daniel Jiménez
Dante Carrizo
Domingo S. Rodríguez-Baena
Dulce Domingos
Eduardo Pérez-Ureta
Encarna Sosa
Esperança Amengual
Fernando Molina
Fran J. Ruiz-Bertol
Francisco Gutiérrez
Francisco J. Lucas
Francisco J. García-Peñalvo
Francisco L. Gutiérrez
Herbert Kuchen
Isabel Nunes
Ismael Navas
Ismael Sanz
Javier Cámara
Javier Cubo
Javier Gutierrez
Jennifer Pérez
Jesús Arias
Joaquín Nicolás
Jordi Cabot
Jorge Martínez-Gil
José Luis Garrido
José Magno Lopes
José María Conejero
José Norberto Mazón
José Ramón Ríos
Juan Ángel Pastor
Juan Carlos Preciado
Juan Manuel Vara
Julia González
Lars-Åke Fredlund
Manuel Serrano
Marcirio Silveira Chaves
Mari Carmen Otero
María del Mar Roldán
María Esperanza Manso
María Isabel Sánchez Segura
María Teresa Gómez
María Visitación Hurtado
Marta Tabares
Martín Solari
Miguel Ángel Laguna
Miguel Á. Martínez-Aguilar
Miguel Á. Martínez-Prieto
Miguel Á. Pérez Toledano
Miguel Á. Rodríguez Luaces
Miguel Rodríguez Penabad
Mª Ángeles Moraga
Norberto Díaz-Díaz
Nuria Medina
Oscar Dieste
Paloma Cáceres
Pascal Poizat
Patricia Paderewski
Patricio Letelier
Pedro J. Muñoz
Pedro Sánchez-Palma
Pedro Valderas
Pepe Carsí
Rafael Ceballos
Raquel Trillo
Raúl Giráldez
Roberto Rodríguez-Echeverría
Santiago Melia
Sergio Ilarri Artigas
Toñi Reina
Toufik Taibi
Valeria de Castro
Vicente Luque
Vicente Pelechano
Xavier Ferré

Sistema Automático de Revisión
(Quercus Software Engineering Group)

Pablo Amaya — Universidad de Extremadura
Daniel García — Universidad de Extremadura

Entidades Patrocinadoras

FIB
Facultat d'Informàtica de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA
UPC

CIMNE
BARCELONA

Microsoft

INTERSYSTEMS

gencat

MINISTERIO DE EDUCACIÓN Y CIENCIA

Prólogo

La undécima edición de las Jornadas de Ingeniería del Software y Bases de Datos se celebró en Sitges (Barcelona) entre el 3 y el 6 de Octubre de 2006. Desde aquellas primeras ediciones del año 1996 en Sevilla y La Coruña, donde las Jornadas de Ingeniería del Software y las de Bases de Datos se celebraron por separado hasta la presente edición se ha recorrido un largo camino. La unificación de las dos líneas en un solo encuentro, primero con estructuras separadas y desde hace dos ediciones con un Comité de Programa único, ha servido para consolidar a la comunidad JISBD como una de las más dinámicas en las tecnologías informáticas, como se demostró en el número de inscritos de la última edición celebrada en el seno del Primer Congreso Español de Informática (CEDI).

Como viene ocurriendo desde la edición del 2001, las JISBD han acogido la celebración, en paralelo y compartiendo algunos actos, de PROLE, las VI Jornadas de Programación y Lenguajes. Ambos eventos son organizados bajo los auspicios de SISTEDES (Sociedad de Ingeniería del Software y Tecnologías de Desarrollo de Software), sociedad constituida en Granada durante la celebración de CEDI en Septiembre del 2005. Desde la edición de 2006, todas las personas inscritas en JISBD o PROLE serán miembros de SISTEDES hasta la celebración de las siguientes jornadas.

En los diez años transcurridos, las JISBD han servido de foro de encuentro para motivar y servir de acicate al esfuerzo investigador de los participantes. Este impulso ha incrementando de manera muy importante la presencia en foros internacionales de trabajos de investigación de grupos españoles y portugueses. Así se puede consultar en el *ISI Web of Science* que el número de trabajos con las palabras claves "Software Engineering" provenientes de España o Portugal en el año 1996 fue de 3, de 15 en el 2002 y de más de 30 en el 2005. Con palabras claves referidas a Bases de Datos los resultados de crecimiento que se obtienen son similares. Es muy posible que versiones previas de esos trabajos fueron presentadas y, a su vez enriquecidas, en ediciones anteriores de las JISBD. Parece justo pensar que sin la existencia de estas Jornadas no se hubiera conseguido este significativo avance en la presencia internacional de sus participantes.

El presente libro de actas contiene los trabajos seleccionados por el Comité de Programa para la edición de este año 2006. Se recibieron un total de 123 trabajos con la siguiente distribución geográfica: 25 de Latinoamérica, 6 de Portugal, 90 de España, 1 de Francia y 1 de India. El Comité de Programa realizó una ardua tarea de revisión, mediante la cual cada trabajo fue revisado por tres o cuatro expertos, abriéndose posteriormente un debate para los trabajos que presentaban disparidad de criterios. El número final de trabajos seleccionados para publicarse completos fue de 41, considerándose además como interesantes 14 trabajos para su presentación como artículos cortos de 6 páginas.

Como es habitual de ediciones anteriores fueron dos las conferencias impartidas durante esta edición. La lección inaugural de título *Software Architecture: Past, Present, and Future* fue dictada por el profesor David Garlan de la prestigiosa *Carnegie Mellon University*. El profesor

Garlan es considerado uno de los fundadores del campo de la Arquitectura Software y, en particular, es experto en representación formal y análisis de diseño de arquitecturas. La segunda conferencia titulada *Model Independent Schema and Data Translation* fue pronunciada por el profesor Paolo Atzeni de la *Università Roma Tre*. El profesor Atzeni trabaja en tópicos relacionados con Bases de Datos, ha sido presidente de la *EDBT Association* y actualmente es secretario de la *VLDB Endowment*. Asimismo las dos conferencias de PROLE impartidas por los profesores Eelco Visser y Krzysztof Apt, han sido incluidas en el programa de JISBD.

También como en ediciones anteriores y con una importante participación e interés se desarrollaron los talleres asociados durante el primer día de las Jornadas. Un total de ocho talleres con la presentación y debate de nuevas propuestas en líneas de trabajo diversas como software orientado a aspectos, pruebas del software, sistemas hipermedias, bases de datos o servicios web. Los talleres de JISBD representan la vanguardia de la investigación y semillero de ideas, convirtiéndose en un foro de encuentro imprescindible dentro de las Jornadas. Asimismo, se ha ofrecido un interesante tutorial sobre Líneas de Producto Software por parte de los profesores Oscar Díaz y Salvador Trujillo.

La celebración de las JISBD con tan alto número de partícipes obliga a una importante labor desinteresada por parte de muchas personas. En primer lugar a los investigadores que han considerado que las JISBD eran un foro adecuado para presentar sus trabajos y a los distintos organizadores y participantes de los talleres. A los miembros del comité ejecutivo y del comité organizador que han coordinado los talleres y tutoriales, así como los detalles de la organización y celebración del encuentro. A los miembros del grupo Quercus por su, cada año mejor, sistema de revisión de trabajos. También queremos dar las gracias al personal del CIMNE, en especial a Paola Pizzi, por su eficaz soporte en la organización del evento. Finalmente, no hay palabras para agradecer y reconocer el trabajo realizado por el Comité de Programa y los revisores adicionales. Se han realizado 380 revisiones y más de 20 discusiones o debates sobre artículos con discrepancias. Gran parte del éxito de estas Jornadas se debe al tiempo que estos investigadores le han dedicado a esta tarea.

La próxima edición de las JISBD en el 2007 volverá a celebrarse en común con el CEDI en Zaragoza. Les deseamos a sus responsables un nuevo éxito de convocatoria que refleje el buen momento que goza la comunidad investigadora ibero-americana en Ingeniería del Software y Bases de Datos.

Sitges, Octubre de 2006
José C. Riquelme, Pere Botella (Editores)

INDICE

CONFERENCIAS INVITADAS

INGENIERÍA DE PROCESOS

MODELADO DE DATOS I

MANTENIMIENTO SOFTWARE

MODELADO DE DATOS II

CALIDAD

GENERACIÓN AUTOMÁTICA

MINERÍA DE DATOS

ARQUITECTURAS SOFTWARE I

INGENIERÍA DE REQUISITOS I

INGENIERÍA DE REQUISITOS II

ARQUITECTURAS SOFTWARE II

MISCELÁNEA SOFTWARE

TRABAJOS CORTOS

ingeniero de aplicaciones será por lo tanto libre de usar la actividad de guía que más le convenga para asistirle en la creación del modelo PIM, ya sea un tutorial, una serie de ejemplos o una línea de productos de software.

4 Conclusión

En este trabajo hemos propuesto la integración de los modelos de procesos (SPEM) y la reutilización de activos (RAS) en nuestra herramienta MDA. En concreto, proponemos una combinación de MDA con RAS y SPEM para ofrecer facilidades de creación y manipulación de líneas de produtos de software no sólo para aplicaciones finales, sino para cualquier tipo de artefacto o producto de trabajo generado a lo largo del ciclo de desarrollo de software. Si pensamos en cualquier actividad del ciclo desarrollo como una actividad SPEM, que consume y produce artefactos, podemos entender los paquetes de activos como líneas de productos para generar artefactos concretos que podamos reutilizar en nuestros proyectos. Asímismo, si entendemos los procesos de modelado y transformaciones de MDA como actividades para producir artefactos, podremos empaquetar arquitecturas MDA como actividades de guía de activos reutilizables.

REFERENCIAS

[CN01] Paul Clements and Linda Northrop. *Software Product Lines: Practices and Patterns*. Addison Wesley, Aug 2001.

[Cza98] Krzysztof Czarnecki. *Generative Programming: Principles and Techniques of Software Engineering Based on Automated Conguration and FragmentBased Component Models*. PhD thesis, Department of Computer Science and Automation, October 1998.

[EPSR06] Antonio Estévez, Javier Padrón, E. Victor Sánchez, and José Luis Roda. ATC: A low-level model transformation language. In *MDEIS2006: Proceedings of the 2nd International Workshop on Model-Driven Enterprise Information Systems*, May 2006.

[GS04] Jack Greenfield and Keith Short. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. John Wiley and Sons, 2004.

[OMG03] OMG. MDA guide version 1.0.1. Technical Report omg/2003-06-01, OMG, Jun 2003. Online at http://www.omg.org/docs/omg/03-06-01.pdf.

[OMG04] OMG. Reusable asset specification. Technical Report ptc/04-06-06, OMG, Jun 2004. Online at http://www.omg.org/docs/ptc/04-06-06.pdf.

[OMG06] OMG. Software process engineering meta-model 2.0. Technical Report ad/06-04-05, OMG, Apr 2006. Online at http://www.omg.org/docs/ad/06-04-05.pdf.

XI Jornadas de Ingeniería del Software y Bases de Datos
JISBD 2006
José Riquelme - Pere Botella (Eds)

INVESTIGANDO LOS BENEFICIOS DE PAIR DESIGNING: UN ESTUDIO EMPÍRICO CON PROFESIONALES

Félix García[1], Corrado A. Visaggio[2], Gerardo Canfora[2] y Mario Piattini[1]

1: Grupo Alarcos
Departamento de Tecnologías y Sistemas de Información
Centro Mixto de Investigación y Desarrollo de Software UCLM-Soluziona
Universidad de Castilla-La Mancha
Paseo de la Universidad, 4, 13071 Ciudad Real, España
e-mail: {Felix.Garcia, Mario.Piattini}@uclm.es, web: http://alarcos.inf-cr.uclm.es

2: Research Centre on Software Technology (RCOST),
University of Sannio, viale Traiano 1,
Palazzo ex-Poste, 82100 Benevento, Italy
e-mail: {canfora, visaggio}@unisannio.it, web: http://rcost.unisannio.it

Palabras clave: Pair Programming, Pair Designing, Ingeniería del Software Empírica.

Resumen. *La técnica pair programming no sólo concierne a la etapa de codificación sino que puede ser también aplicada a otras fases del ciclo de vida del software como el análisis, el diseño y las pruebas. Las investigaciones previas han demostrado diversos beneficios de pair programming, destacando especialmente el incremento de la calidad y la reducción del esfuerzo dedicado a realizar las tareas de programación. El diseño y la codificación tienen muchas diferencias: tratan con distintas categorías de problemas, requieren diferentes habilidades y conocimiento y se basan en procesos distintos. Debido a esta asimetría, el hecho de aplicar pair programming a la fase de diseño ("pair designing") podría no producir los mismos beneficios que en la fase de codificación. Para contrastar dicha hipótesis se realizó un experimento en la empresa Soluziona Software Factory con el fin de determinar diversos factores de rendimiento del pair designing. El experimento proporcionó evidencia empírica de que pair designing podría incrementar la calidad del diseño obtenido y afectar a la productividad.*

1. INTRODUCCIÓN

La práctica de Pair Programming [2] requiere que dos desarrolladores trabajen "hombro con hombro" (*side by side*) en el mismo documento o pieza de código y sobre la misma máquina. Uno de los componentes del par, denominado "driver", escribe el código, que a su vez es revisado por el otro componente, denominado "observer". Estos dos roles son intercambiados durante la realización de la tarea siempre y cuando es necesario. El objetivo de la práctica es solapar la revisión y producción de código de un artefacto software con el fin de facilitar la detección de defectos. Ello implica que las estrategias de implementación podrían ser ajustadas continuamente durante la escritura del código. Sin embargo esta práctica no está sólo reservada a la fase de codificación, sino que puede ser aplicada a otras fases del proceso de desarrollo de software tales como el análisis, el diseño y las pruebas.

Algunos experimentos han demostrado que pair programming mejora el rendimiento de los equipos de proyecto en términos de reducción del tiempo necesario para realizar las tareas y mejora de la calidad del artefacto producido [5, 6, 7, 8]. Sin embargo, hay pocos estudios enfocados en la práctica de pair designing. Principalmente se han investigado aspectos de gestión de conocimiento, tal y como se presenta en [3] en los que se concluye que pair designing es un buen medio de mejorar el conocimiento individual y de transferir dicho conocimiento en el equipo del proyecto.

En este artículo, la práctica 'pair designing' se considera en un sentido ligeramente distinto a pair programming, ya que pair designing se refiere exclusivamente a la etapa de diseño, pero mantiene los fundamentos de pair programming en cuanto al solapamiento, creación y revisión del artefacto de forma conjunta con la posibilidad de intercambiar los roles .

El objetivo del presente artículo es investigar si pair designing podría replicar los beneficios de pair programming en relación a la calidad y el esfuerzo para lo cual se ha realizado un experimento con profesionales. El artículo se estructura de la siguiente forma: a continuación se describe el diseño del experimento y en el apartado tres se exponen y analizan los resultados. Finalmente se presentan las conclusiones.

2. DISEÑO DEL EXPERIMENTO

El objetivo del experimento expresado de acuerdo a la plantilla GQM [1] fue :

- *Analizar* la práctica de **Pair Designing**
- *Con el propósito de* **Comparar**
- *Con respecto a su* **Esfuerzo** y **Calidad**
- *Desde el punto de vista de* los **diseñadores**
- *En el contexto de* un grupo de **profesionales de desarrollo SW**.

A partir de este objetivo se plantearon las siguientes preguntas de investigación:

- Q.1. ¿Es Pair Designing más productivo que Solo Designing respecto al esfuerzo necesario?
- Q.2. ¿Es Pair Designing mejor que Solo Designing respecto a la calidad de los artefactos producidos?

Para responder estas cuestiones se llevó a cabo un experimento cuya visión general se

muestra en la Figura 1:

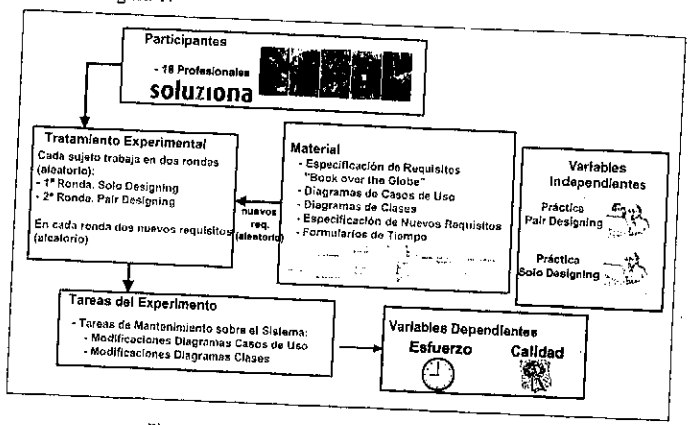


Figura 1. Diseño del Experimento sobre Pair Designing

El experimento se llevó a cabo en las instalaciones de la Fábrica de Software de Soluziona situadas en Ciudad Real (España). Los participantes fueron 18 empleados de la empresa con conocimientos en modelado de software (UML, base de datos, etc.) y programación. Todos habían participado previamente en varios proyectos de ingeniería de software y contaban al menos con un año de experiencia en la empresa. Para facilitarles su comprensión de la técnica pair designing se les impartió una sesión de entrenamiento.

El material fue preparado por los experimentadores y estaba compuesto por:

- **Especificación de requisitos textual** del sistema "Book over the Globe", un sistema de compraventa de libros de segunda mano a través de internet.
- **Diagramas de diseño** y de **análisis de requisitos** del sistema: 2 Diagramas de Casos de Uso y 2 Diagramas de Clases con sus tres capas (presentación, dominio y persistencia) junto con información textual adicional.
- **Especificación textual** de **nuevos requisitos,** agrupados en dos tareas de complejidad equivalente.
- **Formularios** de los tiempos empleados en resolver cada tarea en cada ronda.

Se pidió a los sujetos hacer tareas de mantenimiento sobre el diseño de un sistema software. En cada ronda se realizaba una tarea (con dos requisitos) consistente en modificar la lógica de negocio de la aplicación realizando modificaciones en los diagramas de casos de uso y/o en los diagramas de clases.

El proceso del experimento fue el siguiente:

1. Cada sujeto estudió la documentación individualmente durante 30 minutos.
2. Primera Ronda. Algunos sujetos ejecutaron pair designing y otros trabajaron individualmente. Los pares se formaron al azar y las tareas duraron dos horas.
3. Segunda ronda. Los sujetos realizaron nuevas tareas variando su modo de trabajo; aquellos que trabajaron en pares en la primera ronda lo hicieron individualmente en la segunda y viceversa. Del mismo modo, si un sujeto había trabajado con el formulario de asignación de la tarea 1 en la primera ronda, en la segunda trabajaba con el formulario de la tarea 2. Las asignaciones de las tareas a los sujetos también fueron aleatorias.

Antes de comenzar el experimento los sujetos participaron en sesiones de laboratorio para aprender pair designing. Se evitó el uso de herramientas CASE y se decidió usar sólo papel y lápiz, ya que algunos sujetos estaban más familiarizados con este tipo de herramientas y ello podía condicionar los resultados.

La **variable independiente** fue la práctica aplicada: **pair designing** o **solo designing**. Las **variables dependientes** fueron: el **Esfuerzo**, que fue medido como el tiempo empleado por los sujetos (en pareja o individual) para realizar las tareas del experimento y se obtuvieron los valores a partir de los formularios de asignación de tareas, en los que los sujetos tenían que anotar la hora de inicio y de finalización (incluyendo segundos) de los requisitos de la tarea; y la **Calidad,** que fue medida a partir de la evaluación de los diagramas que los sujetos modificaron para satisfacer los nuevos requisitos. Para reducir la subjetividad de esta evaluación, todos los artefactos se comprobaron por dos de los experimentadores a partir de un conjunto de soluciones acordadas por ellos mismos.

3. RESULTADOS DEL EXPERIMENTO

En la Figura 2 se muestra la media y desviación típica del esfuerzo que los sujetos emplearon para realizar las tareas en la primera y segunda rondas.

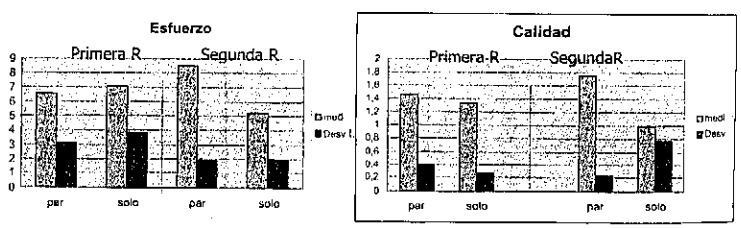


Figura 2. Estadísticas Descriptivas de la variables dependientes Esfuerzo y Calidad

Los datos mostrados en las Figura 2 indican que pair designing requirió menos tiempo que solo designing únicamente en la primera ronda. La primera ronda parece que permitió a los sujetos tener un mayor conocimiento del sistema lo que hizo que en la segunda ronda las discusiones de los pares requirieran más tiempo que en la primera. Por su parte, la desviación estándar de los pares fue menor en ambas rondas que la de los que trabajaron individualmente. En este sentido pair designing hace que el esfuerzo sea más predecible, lo que puede deberse a que el trabajo en pares tienda a compensar el rendimiento de los integrantes del par.

Respecto a la variable calidad, pair designing permitió obtener resultados de mejora de la calidad en ambas rondas. La diferencia de calidad entre los pares e individuos fue significativa en la segunda ronda: los pares obtuvieron una calidad de alrededor de un 45% mejor que el obtenido por los individuos. Es posible concluir por lo tanto que el pair designing necesita más tiempo para conseguir una mayor calidad.

Para contrastar estadísticamente las hipótesis planteadas, se aplicó el test de Mann-Whitney, ya que la distribución de los datos no era la normal y el nivel de significación α se fijó en 0.05. Como resultado se confirmaron las conclusiones obtenidas a partir de las estadísticas descriptivas, aunque sólo la segunda ronda permitió obtener resultados estadísticamente significativos. Por su parte también se concluyó que pair designing y solo designing no tuvieron diferencias significativas en cuanto al tiempo de esfuerzo en función del tipo de asignación de requisitos y que los efectos de pair designing fueron mayores en la segunda ronda que en la primera.

4. CONCLUSIONES

En muchas ocasiones se toman decisiones sobre qué métodos y herramientas emplear en un proyecto de desarrollo de software sin evidencia objetiva que confirme sus ventajas, limitaciones y riesgos. Por ello es importante proporcionar evidencias empíricas sobre la utilidad de estos métodos en la práctica.

Algunos investigadores han demostrado que pair programming puede incrementar la calidad del producto obtenido así como reducir el tiempo necesario para obtenerlo. Aunque habitualmente esta práctica se ha aplicado a la fase de codificación, también es posible su aplicación a las fases de diseño y pruebas y la replicación de estos beneficios a dichas fases no ha sido demostrada. En particular, en este artículo se ha abordado la utilidad de la técnica pair programming en la fase de diseño (pair designing) a la hora de realizar tareas de mantenimiento. Los resultados muestran evidencia estadística de que pair designing:

- Puede mejorar la calidad del trabajo.
- Requiere más tiempo que el tradicional solo designing.
- Asegura una mayor predecibilidad del esfuerzo que solo designing.

Para confirmar estas conclusiones se llevó a cabo una réplica con estudiantes en la que se obtuvieron resultados similares y se confirmaron las mismas hipótesis [4]. Los resultados de los experimentos confirman que la práctica de pair programming no está limitada sólo a la fase de codificación. Dicha práctica se puede aplicar de forma exitosa cuando se trabaja en la documentación del diseño e incluso no necesariamente sólo en el contexto de los métodos ágiles. Considerando que los sujetos que participaron en el experimento fueron profesionales

con un buen grado de experiencia las conclusiones de este trabajo pueden considerarse como suficientemente válidas para el contexto de la población de profesionales del software.

AGRADECIMIENTOS

Nos gustaría agradecer a los profesionales de la empresa Soluziona Software Factory de Ciudad Real su activa colaboración y participación en el experimento. Este trabajo ha sido financiado parcialmente por los proyectos: FAMOSO, financiado por el Ministerio de Industria, Turismo y Comercio, FIT-340000-2005-161 Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica 2004-2007 y "Fondo Europeo de Desarrollo Regional (FEDER)", European Union; y MECENAS (Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia, PBI06-0024).

REFERENCIAS

[1] Basili, V., Rombach, H. "The TAME project: towards improvement-oriented software environments", *IEEE Transactions on Software Engineering*, 14(6), 1988, pp. 728-738.

[2] Beck K. *Extreme Programming explained: Embrace change*. Addison-Wesley: Reading, Massachusetts, 1999.

[3] Bellini E., Canfora G., Garcia F., Piattini M., and Visaggio C.A."Pair designing as a practice for enforcing and diffusing design knowledge", *Journal of Software Maintenance and Evolution: Research and Practice*, Wiley Interscience, vol. 17, no. 6, 2005, pp. 401–423.

[4] Canfora, G., Cimitile, A., García, F., Piattini, M., Visaggio, C.A. Performances of Pair Designing on Software Evolution: a controlled experiment. *10th European Conference on Software Maintenance and Reengineering (CSMR'06)*, Bari, Italy, 22-24 March 2006, pp. 197-205.

[5] McDowell C., Werner L., Bullock H., and Fernald J. The Effects of Pair-Programming on Performance in an Introductory Programming Course, *Proc. of the 33rd SIGCSE technical symposium on Computer science education*, Cincinnati, Kentucky, USA, ACM, 2002, pp. 38-42.

[6] Nawrocki J. And Wojciechowski A. Experiment Evaluation of Pair Programming. *Proc. of the 12th European Software Control and Metrics Conference (ESCOM)*, 2-4 April 2001, London, pp. 269-276.

[7] Srikanth H., Williams L., Wiebe E., Miller C., and Balik S. On Pair Rotation in the Computer Science Course. *Proc. of Conference on Software Engineering Education and Training 2004*, Norfolk, Virginia, USA, IEEE CS Press, 2004, pp. 144-149.

[8] Williams L., Kessler R.R., Cunningham W., and Jeffries R. "Strengthening the Case for Pair Programming", *IEEE Software* 17(4), 2000, IEEE CS Press, pp.19-25.

XI Jornadas de Ingeniería del Software y Bases de Datos
JISBD 2006
José Riquelme - Pere Botella (Eds)

EXPERIENCIAS EN INTEGRACIÓN DE MÉTODOS CUALITATIVOS Y CUANTITATIVOS

María Lázaro[1]*, Esperanza Marcos[1] y Sira Vegas[2]

[1] Grupo de investigación Kybele
Universidad Rey Juan Carlos
C/ Tulipán s/n, 28933, Móstoles, Madrid
e-mail: marialaz@gmail.com, esperanza.marcos@urjc.es

[2] Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo s/n, 28660, Boadilla del Monte, Madrid
e-mail: svegas@fi.upm.es

Resumen. La experimentación se caracteriza por ser de naturaleza iterativa, es decir, que los experimentos no ocurren de forma aislada, sino que la interpretación de los resultados de un determinado experimento sirven para dar forma a las hipótesis del siguiente. En la actualidad, la etapa de interpretación de resultados experimentales en Ingeniería del Software (IS) es nula o muy pobre. Esto influye negativamente en el ciclo de experimentación/aprendizaje. Pensamos que los métodos de investigación cualitativa son una herramienta muy potente, que pueden mejorar esta etapa de interpretación, por lo que en este artículo proponemos su uso integrado con métodos cuantitativos. Para ello, se han utilizado métodos cualitativos en una serie de experimentos cuantitativos realizados desde hace unos años pudiendo apreciar que los métodos cualitativos ayudan a responder el porqué de los resultados cuantitativos.

Palabras clave: Experimentación, integración de métodos cualitativos y cuantitativos.

1. INTRODUCCIÓN

La experimentación se caracteriza por su naturaleza iterativa. A partir de una serie de hipótesis, se diseña y realiza un experimento para contrastarla. Los resultados arrojados por el experimento se interpretan, y llevan a modificar las hipótesis de partida, o a rediseñar el experimento, iniciándose un ciclo de experimentación/aprendizaje. La interpretación de resultados (y por tanto la continuidad del ciclo experimentación/aprendizaje) es una de las asignaturas pendientes de la experimentación en Ingeniería del Software (IS). Para mejorar la interpretación, proponemos integrar métodos cualitativos en experimentos cuantitativos.

En función del objeto de estudio, existen dos tipos de enfoque o métodos de investigación: cualitativos y cuantitativos. Los *métodos de investigación cuantitativa* se encargan de medir y analizar el grado de asociación o relación entre variables cuantificadas [2]. Están basados en la inducción probabilística del positivismo lógico, y son objetivos, ya que consideran que todos los fenómenos pueden ser reducidos a indicadores empíricos que representan la