JISBD 2008

13th Conference on Software Engineering and Databases

XIII Jornadas de Ingeniería del Software y Bases de Datos

Gijón (Spain), October 7-10 2008

EDITORS: Ana Moreira
María José Suárez-Cabal
Claudio de la Riva
Javier Tuya

13th Conference on Software Engineering and Databases

XIII Jornadas de Ingeniería del Software y Bases de Datos

Gijón (Spain), October 7-10 2008

EDITORS: Ana Moreira
María José Suárez-Cabal
Claudio de la Riva
Javier Tuya

Edita:
Ana Moreira
María José Suárez-Cabal
Claudio de la Riva
Javier Tuya

Filmación e impresión:
Gráficas Rigel

Depósito Legal:
AS - 5.236 - 08

ISBN:
978-84-612-5820-8

Volume Editors Details

Ana Moreira
Departamento de Informática
Faculdade de Ciências e tecnologia
Universidade Nova de Lisboa
2829-516 Caparica, Portugal
E-mail: amm@di.fct.unl.pt
URL: http://ctp.di.fct.unl.pt/~amm/

María José Suárez-Cabal
Departamento de Informática
Universidad de Oviedo
33204 Gijón, Spain
E-mail: cabal@uniovi.es

Claudio de la Riva
Departamento de Informática
Universidad de Oviedo
33204 Gijón, Spain
E-mail: claudio@uniovi.es
URL: http://www.di.uniovi.es/~claudio/

Javier Tuya
Departamento de Informática
Universidad de Oviedo
33204 Gijón, Spain
E-mail: tuya@uniovi.es
URL: http://www.di.uniovi.es/~tuya/

Preface

Celebrating 13 Years of JISBD

With the 2008 edition in Gijón (October 7 to 10), the Conference on Software Engineering and Databases (JISBD) celebrates 13 years of existence. Born as a forum where the Spanish community would publish their work, meet to discuss potential research collaborations and evaluate the progress of research projects funded by the Spanish Ministery of Science and Technology, JISBD has long since moved beyond its initial boundaries and crossed several oceans.

Presently, the conference has become an important reference for younger researchers, as well as a forum which the more experienced do not wish to miss. In recent years, JISBD has broadened its radius, accepting papers also in English and Portuguese, in addition to Spanish. This change, not only brought more conference participants, but also significantly increased the number of submissions and, principally, the quality of the submissions accepted.

The JISBD community is now self-sustained and continues to expand. The quality of work accepted is equivalent to that of other relevant international events. In recent years, it has been possible to edit a special volume of IEEE LA with extended versions of the best conference papers and this also is happening with the current edition. This special issue, together with the conference proceedings with ISBN, is a showcase of the quality of the work of JISBD.

One of the highlights of this conference has been the excellence of its keynote speakers. Many of the most admired international researchers and professionals have already been invited to address the JISBD participants.

Within this rich framework for scientific and technological interchange, the conference includes several satellite events. In addition to the presentation of high quality original papers in the main conference, the program includes tutorials, tool demonstrations and workshops for the discussion of innovative ideas and work in progress, as well as a forum to bring to a wider audience research work already published in prestigious journals or conference proceedings (with an acceptance rate below 25% and an impact factor above 0.5).

It is no exaggeration to claim that JISBD has been consolidating its position as a reference event where researchers and professionals of Software Engineering and Databases can get together to discuss results and share ideas. JISBD has become an important forum for collaboration between different strands and research groups, while continuing to offer its participants a well organized event with exceptional hospitality.

About this Edition

The increased global reach of JISBD is evident in the origin of papers received. This year, in addition to the two Iberian and ten Latin-American countries, submissions arrived also from China, France, Germany, India, Iran and Pakistan.

Of a total of 115 abstracts, 112 papers were submitted for review. Most papers were reviewed by three PC members, and several were reviewed by four. The program Committee accepted 30 full papers and selected 12 for presentation as short papers. The acceptance rate for full papers was approximately 25%.

The increasing success of the conference implies greater responsibilities in terms of guaranteeing independent judgement and ensuring compliance with international standards of ethics. For this reason, a greater effort has been made in recent years to avoid double submissions, a task made

more difficult by the fact that the conference accepts submissions in three languages. This year, three good papers were rejected due to double submission, in different languages to different events.

In addition to the accepted papers, the conference includes five workshops, one tutorial, nine tool demos, an industrial panel and also a forum to discuss important relevant work already published elsewhere.

A highlight of the conference is, without doubt, the excellence of the invited keynote speakers. This year is no exception and we are honoured indeed to receive Bashar Nuseibeh and Bran Selic.

Bashar Nuseibeh is an academic and researcher at the Open University in the UK and invited professor in various other universities, including Japan's National Institute of Informatics. Bashar chairs several international committees and is recognized also for industry work, including organizations such as the UK's National Air Traffic Services (NATS), Texas Instruments, Praxis Critical Systems, Philips Research Labs, and NASA.

Bran Selic was, for many years, a distinguished engineer and researcher at IBM, and currently heads a global consultancy based in Canada. He is internationally known for his work in large-scale industrial systems, and for his pioneering work in Model-Driven Development and Real-Time Embedded Systems.

Bahsar's keynote is entitled "*The five W's (and one "H") of Security: ... Software Engineering of Secure Systems"* while Bran's is on "*Model-Based Software Engineering: Expected and Unexpected Challenges*".

Acknowledgements

A very special word of thanks is due to Bashar and Bran for having accepted my invitation and for sharing all the participants their knowledge, experience and refined wit. I sincerely hope JISBD was also for them a gratifying and unique experience.

Acknowledgements are due to a multitude of collaborators without whom the conference could not have been a success. Firstly, the paper authors for the trust placed in the quality of JISBD as a conference that merited their submissions. Secondly, to the PC members, whose diligent review work ensured that the authors' trust continues to be justified.

For managing the submission and review process, I was fortunate to have the constant help of Juan Hernández and José Javier Berrocal; they were my guardian angels, constantly alert to deadlines and ready to help as necessary. A special thanks for the contribution of my "Executive Program Committee", Antonio Vallecillo, Juan Hernández, Miguel Toro, Vicente Pelechano and Xavier Franch.

Acknowledgement is due to the main conference organizers, especially to Javier Tuya and his co-chair, Claudio de la Riva, for their efficient handling of the numerous tasks that a conference of this size and quality entails. Thanks also to those responsible for the satellite events (in alphabetical order), António Rito Silva, Antonio Vallecillo, Gustavo Rossi, João Araújo, João Falcão e Cunha, José Berrocal, José Corrales, José García-Fanjul, João Miguel Fernandes, Lidia Fuentes and María José Suárez-Cabal.

Finally, a special word of thanks to the sponsors of this conference, without whose contribution the event would have been somewhat less charming (not to mention gastronomically less satisfying).

Ana Moreira
Program Committee Chair

Prefácio

Celebrando 13 Anos de JISBD

Com a edição de 2008 em Gijón (7-10 Outubro), a Conferência em Engenharia de Software e Bases de Dados (JISBD) celebra 13 anos de existência. Apesar de ter nascido como um fórum onde a comunidade espanhola publicava os seus trabalhos e se reunia para discutir potenciais colaborações futuras de investigação, e até avaliar o estado de andamento dos projectos de investigação financiados pelo Ministério da Ciência e Tecnologia espanhola, há muito que extravasou essas fronteiras e cruzou oceanos.

Actualmente o JISBD é um marco importante na investigação dos mais jovens, mas também um fórum que os mais seniores não querem perder. Nos últimos anos a conferência abriu-se para o mundo inteiro, aceitando artigos escritos em Inglês, Espanhol e Português. Esta viragem trouxe não só mais participantes à conferência, mas também um aumento significativo do número de trabalhos submetidos e, principalmente, um aumento na qualidade desses trabalhos.

A comunidade do JISBD é agora auto-sustentada e em contínua expansão. A qualidade dos trabalhos aceites é equiparada à de muitos outros eventos internacionais de relevo. Por este motivo, nos últimos anos, foi-nos possível editar um volume especial no IEEE LA com uma versão estendida dos melhores trabalhos da conferência, o que acontecerá também nesta edição. Este volume, em conjunto com as actas formais da conferência com ISBN, é uma montra da qualidade do trabalho que aqui se discute.

Uma das características de excelência desta conferência tem sido, desde sempre, o gabarito dos seus palestrantes convidados. É um prazer ver que muitos dos mais admirados investigadores e profissionais internacionais já foram convidados a falar para os participantes do JISBD.

Neste enquadramento fecundo para divulgação científica e tecnológica, a conferência inclui vários eventos satélite. Além dos artigos seleccionados para apresentação na conferência, o programa inclui ainda tutoriais, demonstrações de ferramentas, workshops para discussão de ideias inovadores e trabalhos em andamento, assim como um evento para a disseminação de trabalho de investigação já publicado em revistas e actas de conferências de grande prestígio (onde o índice de aceitação é inferior a 25% e o factor de impacto superior a 0.5).

Assim, não é excessivo afirmar que o JISBD se tem vindo a consolidar como um evento de referência onde investigadores e profissionais em Engenharia de Software e Bases de Dados se encontram para discutir, disseminar e trocar ideias, partilhar experiências e resultados entre diversos sectores e grupos de investigação, num contexto de excelente organização e invulgar hospitalidade.

Sobre esta Edição

A atestar o crescimento e internacionalização do JISBD está a origem dos artigos que nos chegaram. Este ano, a nacionalidade dos autores foi surpreendentemente diversificada, pois para além dois países Ibéricos e de dez países Latino-Americanos, recebemos trabalhos também da Alemanha, China, França, Índia, Irão e Paquistão.

O número total de resumos foi de 115, sendo que destes, 112 artigos foram submetidos para avaliação. Cada artigo foi avaliado por pelo menos três revisores, sendo que vários foram avaliados por quatro. O Comité de Programa aceitou 30 artigos longos e escolheu 12 para apresentação como artigos curtos. Assim, o índice de aceitação de artigos longos foi de cerca de 25%.

Este sucesso acarreta responsabilidades acrescidas em garantir a independência de julgamentos e em fazer cumprir a ética e as normas internacionais. É por este motivo que, nos últimos anos, se

tem feito um esforço muito grande para evitar submissões duplicadas, tarefa nem sempre fácil para os membros do Comité de Programa, já que a conferência aceita três línguas de escrita. Este ano foram rejeitados três bons artigos avaliados como de submissão duplicada, em duas línguas, para eventos diferentes.

Para além dos artigos seleccionados, a conferência conta também com a organização de cinco *workshops*, um *tutorial*, nove demonstrações de ferramentas, um painel industrial e ainda um fórum onde se discutem trabalhos de relevo já publicados em revistas ou outras conferências.

Mas sem dúvida que os momentos mais altos da conferência são sempre marcados pelo admirável conjunto de palestrantes convidados. Este ano tivemos a sorte de receber Bashar Nuseibeh e de Bran Selic.

Bashar Nuseibeh é um académico e investigador da Open University, na Inglaterra, e professor convidado em várias outras universidades, incluindo o Instituto Japonês de Informática. Bashar preside vários comités internacionais e é admirado também pelo seu trabalho para a indústria, que inclui organizações como o National Air Traffic Services (NATS) do Reino Unido, Texas Instruments, Praxis Critical Systems, Philips Research Labs, e a NASA.

Bran Selic foi durante umas dezenas de anos engenheiro e investigador distinguido da IBM e actualmente preside uma empresa de consultoria internacional sediada no Canadá. É conhecido mundialmente pelos seus trabalhos em sistemas de larga escala industrial e também pelo seu pioneirismo nas áreas de desenvolvimento orientado a modelos e sistemas embutidos de tempo real.

A palestra do Bashar é intitulada "*The five W's (and one "H") of Security: ... Software Engineering of Secure Systems"*, enquanto que a do Bran é sobre "*Model-Based Software Engineering: Expected and Unexpected Challenges*".

Agradecimentos

Uma palavra especial de agradecimento ao Bashar e ao Bran por terem aceite o meu convite e por brindarem todos os participantes com a sua experiência, conhecimento e refinado sentido de humor. Espero que o JISBD tenha sido também para eles uma experiência agradável e diferente.

Agradecimentos são justamente devidos ao grande número de colaboradores, sem o contributo dos quais, a conferência não poderia ter tido êxito. Aos autores, claro, por confiarem na qualidade do JISBD e submeterem, por isso, os seus trabalhos. Aos membros do Comité de Programa cujas revisões asseguram que essa confiança continua a justificar-se.

Para gerir o sistema de submissão, contei com o apoio incondicional do Juan Hernández e do José Javier Berrocal. Eles foram os meus "anjos da guarda", sempre atentos a todos os prazos e prontos a dar todas as explicações. Um agradecimento particular ao contributo meu "Comité Executivo de Programa", Antonio Vallecillo, Juan Hernández, Miguel Toro, Vicente Pelechano e Xavier Franch. Obrigada pelo vosso apoio e sugestões.

Obrigada aos organizadores principais da conferência, em especial ao Javier Tuya, e ao seu vice-presidente, Claudio de la Riva, pela gestão eficaz das inúmeras tarefas que uma conferência desta dimensão exige. Um agradecimento é ainda devido, e por ordem alfabética, aos responsáveis dos eventos satélite, António Rito Silva, Antonio Vallecillo, Gustavo Rossi, João Araújo, João Falcão e Cunha, José Berrocal, José Corrales, José García-Fanjul, João Miguel Fernandes, Lidia Fuentes e María José Suárez-Cabal.

Finalmente, um agradecimento aos patrocinadores da conferência, sem o contributo de quem o evento teria tido menos charme (e uma gastronomia muito menos requintada).

Ana Moreira
Presidente do Comité de Programa

Prefacio

Con esta edición 2008 en Gijón (7 al 10 de Octubre), las Jornadas de Ingeniería del Software y Bases de Datos (JISBD) celebra 13 años de existencia. JISBD nació como un foro donde la comunidad española publicaba su trabajo, discutía potenciales colaboraciones en investigación y evaluaba el progreso de los proyectos de investigación financiados por el Ministerio de Ciencia y Tecnología, y en la actualidad ha traspasado fronteras y cruzado varios océanos.

Actualmente, la conferencia es una referencia importante para jóvenes investigadores, así como un foro de cita obligada para investigadores más experimentados. Durante los últimos años, JISBD se ha abierto al mundo, aceptando artículos en Inglés y Portugués, además de Castellano. Este cambio no solamente se ha traducido en más participantes, sino que ha incrementado significantemente el número de artículos enviados, y principalmente, la calidad de los artículos aceptados.

La comunidad JISBD está actualmente auto sustentada y continúa expandiéndose. La calidad de los trabajos aceptados es equivalente al de otros eventos internacionales relevantes. Durante los últimos años, ha sido posible editar un volumen especial de IEEE LA con versiones ampliadas de los mejores trabajos presentados en la conferencia, lo que sucederá también en la presente edición. Este volumen especial, junto con las actas de la conferencia con ISBN, es una muestra de la calidad de los trabajos de JISBD.

Una de las características más sobresalientes de la conferencia ha sido la calidad de los ponentes invitados. Varios investigadores y profesionales de reconocido prestigio internacional han sido invitados a participar como ponentes en JISBD.

Dentro de este marco científico y tecnológico, la conferencia incluye varios eventos relacionados. Además de la presentación de artículos originales de alta calidad en la conferencia principal, el programa incluye tutoriales, demostraciones de herramientas, talleres para la discusión de ideas innovadoras y trabajos en curso, así como la divulgación de trabajos de investigación publicados en revistas y conferencias de prestigio (con un ratio de aceptación por debajo del 25% y un factor de impacto por encima de 0,5).

No es una exageración afirmar que JISBD ha consolidado su posición como un evento de referencia donde investigadores y profesionales de la Ingeniería del Software y las Bases de Datos se reúnen para discutir resultados y compartir ideas. JISBD se ha convertido en un foro importante para la colaboración entre diferentes sectores y grupos de investigación, en un contexto de excelente organización y excepcional hospitalidad.

Sobre la presente edición

El crecimiento e internacionalización de JISBD se hace evidente analizando el origen de los artículos recibidos. En la presente edición, además de los artículos recibidos de los dos países de la Península Ibérica y los diez países Latinoamericanos, se han recibido artículos de China, Francia, Alemania, India, Irán y Pakistán.

De un total de 115 resúmenes previamente recibidos, finalmente se recibieron 112 artículos para su revisión. La mayoría de los artículos fueros revisados por tres miembros del Comité de Programa y varios por cuatro. El Comité de Programa aceptó 30 artículos largos y seleccionó 12 para su presentación como artículos cortos. El ratio de aceptación para los artículos largos fue de aproximadamente el 25%.

El éxito de la conferencia implica grandes responsabilidades en términos de garantizar la independencia de las revisiones y el cumplimiento de los estándares internacionales de ética. Por esta razón, durante los últimos años se ha realizado un mayor esfuerzo en aras de evitar envíos duplicados, una tarea especialmente dificultosa, ya que la conferencia acepta envíos en tres idiomas. En la

presente edición tres artículos fueron rechazados debido al doble envío en diferentes idiomas para diferentes eventos.

Además de los artículos aceptados, la conferencia incluye cinco talleres, un tutorial, nueve demostraciones de herramientas y foro para la discusión y divulgación de trabajos relevantes previamente publicados, así como una mesa redonda de carácter industrial.

Una característica importante de la conferencia es, sin ninguna duda, la excelencia de los ponentes invitados. La presente edición no es una excepción y estamos orgullosos de contar con la presencia de Bashar Nuseibeh y Bran Selic.

Bashar Nuseibeh es académico e investigador en la Open Unversity del Reino Unido y profesor invitado en otras muchas universidades, incluyendo el Instituto Nacional Japonés de Informática. Bashar preside varios comités internacionales y está reconocido igualmente por su trabajo industrial, incluyendo organizaciones tales como el Servicio Nacional de Tráfico Aéreo del Reino Unido (NATS), Texas Instruments, Praxis Critical Systems, Philips Research Labs y la NASA.

Bran Selic fué durante varios años un destacado ingeniero e investigador en IBM y actualmente lidera un consultora internacional con sede en Canadá. Es internacionalmente conocido por su trabajo en sistemas industriales a gran escala y por su trabajo pionero en Desarrollo Dirigido por Modelos y Sistemas Empotrados en Tiempo Real.

La conferencia de Bashar se titula "*The five W's (and one "H") of Security: ... Software Engineering of Secure Systems"* y la de Bran "*Model-Based Software Engineering: Expected and Unexpected Challenges*".

Agradecimientos

Un agradecimiento especial es para Bashar y Bran por haber aceptado mi invitación y por compartir con todos los participantes sus conocimientos, experiencia y refinado sentido del humor.

Agradecimientos también para la multitud de colaboradores sin los cuales el éxito de la conferencia no habría sido posible. En primer lugar, para los autores de los artículos por confiar en la calidad de JISBD y enviar sus trabajos. En segundo lugar, para los miembros del Comité de Programa, cuyas revisiones aseguran la calidad de los trabajos.

Para el proceso de gestión y revisión de los trabajos recibidos, fui afortunada por tener la ayuda constante de Juan Hernández y José Javier Berrocal. Ellos fueron mis ángeles guardianes, alertándome constantemente de las fechas límite y siempre preparados para ayudarme cuando lo necesitaba. Agradecimientos especiales por la contribución de mi "Comité de Programa Ejecutivo", Antonio Vallecillo, Juan Hernández, Miguel Toro, Vicente Pelechano y Xavier Franch.

Agradecimientos también para los organizadores de la conferencia principal, especialmente al presidente del comité organizador Javier Tuya y su vicepresidente Claudio de la Riva, por su manejo eficiente de las numerosas tareas que una conferencia de este tamaño y calidad conllevan. Agradecimientos también para los responsables de los eventos relacionados (en orden alfabético) António Rito Silva, Antonio Vallecillo, Gustavo Rossi, João Araújo, João Falcão e Cunha, José Berrocal, José Corrales, José García-Fanjul, João Miguel Fernandes, Lidia Fuentes y María José Suárez-Cabal.

Finalmente, palabras especiales de agradecimiento para los patrocinadores de la conferencia, sin cuya contribución el evento habría sido menos encantador (y con una gastronomía menos refinada).

Ana Moreira
Presidenta del Comité de Programa

Conference Committee

Program Committee Chair
Ana Moreira (Univ. Nova de Lisboa, Portugal)

Organizing Chair
Javier Tuya (Univ. Oviedo, Spain)

Organizing Co-Chair
Claudio de la Riva (Univ. Oviedo, Spain)

Permanent Committee Secretary
Mario Piattini (Univ. Castilla-La Mancha, Spain)

Tutorial Chair
António Rito Silva (Univ. Técnica Lisboa, Portugal)

Workshop Chair
João Araújo (Univ. Nova de Lisboa, Portugal)

Tool Demonstrations Chair
Lidia Fuentes (Univ. Málaga, Spain)

Relevant Papers Dissemination Chairs
Antonio Vallecillo (Univ. Málaga, Spain)
João Falcão Cunha (Univ. Porto, Portugal)

Proceedings Chair
María José Suárez-Cabal (Univ. Oviedo, Spain)

Cyber Chair
Jose Javier Berrocal (Univ. Extremadura, Spain)

Web Chair
José A. Corrales (Univ. Oviedo, Spain)

Publicity Chairs
Gustavo Rossi (Univ. La Plata, Argentina)
José García-Fanjul (Univ. Oviedo, Spain)
João Miguel Fernandes (Univ. Minho, Portugal)

Organizing Committee (Univ. Oviedo, Spain)

Javier Tuya
Claudio de la Riva
José García-Fanjul
Isabel Sevilla
María José Suárez-Cabal
José Ramón de Diego
Raquel Blanco
Eugenia Díaz Fernández
José A. Corrales
Marta Fernández de Arriba

SISTEDES Executive Board

President
Miguel Toro (Univ. Sevilla, Spain)

Vice President
Juan José Moreno (Univ. Polit. Madrid, Spain)

Secretary
Nieves R. Brisaboa (Univ. Coruña, Spain)

Treasurer
Javier Tuya (Univ. Oviedo, Spain)

Members
Pere Botella (Univ. Polit. Catalunya, Spain)
Ricardo Peña (Univ. Complutense Madrid, Spain)
Coral Calero (Univ. Castilla-La Mancha, Spain)
Manuel Hermenegildo (Univ. Polit. Madrid, Spain)
Ernesto Pimentel (Univ. Málaga, Spain)
María Ribera Sancho (Univ. Polit. Catalunya, Spain)
Natalia Juristo (Univ. Polit. Madrid, Spain)
Salvador Lucas (Univ. Polit. Valencia, Spain)

Submission and Review Support System
(Quercus Software Engineering Group)

Javier Berrocal (Univ. Extremadura, Spain)
Juan Hernández (Univ. Extremadura, Spain)

Secretariat

Fundación Universidad de Oviedo
C/ Principado 3, 4ª Planta
33007 Oviedo, Spain.
Tel: 34-985104927
Fax: 34-985104928

Executive Program Committee

Xavier Franch (Univ. Polit. Catalunya, Spain)
Juan Hernández (Univ. Extremadura, Spain)
Vicente Pelechano (Univ. Polit. Valencia, Spain)
Antonio Vallecillo (Univ. Málaga, Spain)
Miguel Toro (Univ. Sevilla, Spain)
Javier Tuya (Univ. Oviedo, Spain)

Program Committee

Albert Abelló (Univ. Polit. Catalunya, Spain)
Ana Paula Afonso (Univ. Lisboa, Portugal)
Ademar Aguiar (Univ. Porto, Portugal)
Jesús Aguilar (Univ. Sevilla, Spain)
José Aldana (Univ. Málaga, Spain)
Mauricio Alférez (U. Nova de Lisboa, Portugal)
Bárbara Álvarez (Univ. Polit. Cartagena, Spain)
Raquel Anaya (Univ. EAFIT, Colombia)
María José Aramburu (Univ. Jaume I, Spain)
Hernán Astudillo (U. T. Federico Santa María, Chile)
Orlando Belo (Univ. do Minho, Portugal)
Rafael Berlanga (Univ. Jaume I, Spain)
Paulo Borba (Univ. Federal Pernambuco, Brazil)
Pere Botella (Univ. Polit. Catalunya, Spain)
Rosana Braga (Univ. São Paulo, Brazil)
Nieves Brisaboa (Univ. Coruña, Spain)
Isabel Brito (Inst. Polit. Beja, Portugal)
Fernando Brito e Abreu (U. Nova de Lisboa, Portugal)
Coral Calero (Univ. Castilla-La Mancha, Spain)
Marcelo Campo (UNICEN, Argentina)
Carlos Canal (Univ. Málaga, Spain)
Valeria de Castro (Univ. Rey Juan Carlos, Spain)
Matilde Celma (Univ. Polit. Valencia, Spain)
Christina Chávez (Univ. Bahia, Brazil)
Rafael Corchuelo (Univ. Sevilla, Spain)
Dolors Costal (Univ. Polit. Catalunya, Spain)
Yania Crespo (Univ. Valladolid, Spain)
Carlos Delgado (Univ. Carlos III, Spain)
Oscar Díaz (Univ. País Vasco, Spain)
Javier Dolado (Univ. País Vasco, Spain)
Xavier Franch (Univ. Polit. Catalunya, Spain)
Pablo de la Fuente (Univ. Valladolid, Spain)
Mario Gaspar da Silva (Univ. Lisboa, Portugal)
Alessando García (Univ. Lancaster, UK)
Marcela Genero (Univ. Castilla-La Mancha, Spain)
Cristina Gómez (Univ. Polit. Catalunya, Spain)
Jaime Gómez (Univ. Alicante, Spain)
Alfredo Goñi (Univ. País Vasco, Spain)
Silvia Gordillo (UNLP, Argentina)
Pedro Guerreiro (Univ. Algarbe, Portugal)
Juan Hernández (Univ. Extremadura, Spain)
Jon Iturrioz (Univ. País Vasco, Spain)
Elena Jurado (Univ. Extremadura, Spain)
Natalia Juristo (Univ. Polit. Madrid, Spain)
Miguel Katrib (Grupo WEBOO, Cuba)
María Lencastre (Univ. Pernambuco, Brazil)
Antonia Lopes (Univ. Lisboa, Portugal)
Adolfo Lozano (Univ. Extremadura, Spain)
Esperanza Marcos (Univ. Rey Juan Carlos, Spain)
Henrique Madeira (Univ. Coimbra, Portugal)
Eduardo Mena (Univ. Zaragoza.Spain)
Ana María Moreno (Univ. Polit. Madrid, Spain)
Juan José Moreno (Univ. Polit. Madrid, Spain)
Juan Manual Murillo (Univ. Extremadura, Spain)
Oscar Pastor (Univ. Polit. Valencia, Spain)
Vicente Pelechano (Univ. Polit. Valencia, Spain)
Marcelo Pimenta(Univ. F. Rio Grande do Sul, Brazil)
Ernesto Pimentel (Univ. Málaga, Spain)
Mónica Pinto (Univ. Málaga, Spain)
Ángeles Places (Univ. Coruña, Spain)
Antonio Polo (Univ. Extremadura, Spain)
Claudia Pons (UNICEN, Argentina)
Tom Price (Univ. F. Rio Grande do Sul, Brazil)
Carme Quer (Univ. Polit. Catalunya, Spain)
Celia Ramos (Univ. Algarbe, Portugal)
Isabel Ramos (Univ. Sevilla, Spain)
Isidro Ramos (Univ. Polit. Valencia, Spain)
Claudio de la Riva (Univ. Oviedo, Spain)
José Riquelme (Univ. Sevilla, Spain)
José Luis Roda (Univ. La Laguna, Spain)
María José Rodríguez Fortis (Univ. Granada, Spain)
José Raúl Romero (Univ. Córdoba, Spain)
Antonio Ruiz (Univ. Sevilla, Spain)
Francisco Ruiz (Univ. Castilla-La Mancha, Spain)
José Samos (Univ. Granada, Spain)
Fernando Sánchez (Univ. Extremadura, Spain)
Juan Sánchez (Univ. Polit. Valencia, Spain)
Carla Silva (Univ. F. Pernambuco, Brazil)
Ernest Teniente (Univ. Polit. Catalunya, Spain)
Miguel Toro (Univ. Sevilla, Spain)
Ambrosio Toval (Univ. Murcia, Spain)
Juan Carlos Trujillo (Univ. Alicante, Spain)
Toni Urpi (Univ. Polit. Catalunya, Spain)
Antonio Vallecillo (Univ. Málaga, Spain)
Belén Vela (Univ. Rey Juan Carlos, Spain)

Referees

Álvaro E. Prieto Ramos
Amador Durán Toro
André L. Santos
Andrea Delgado
Ángel Herranz
Angélica Caro
Anna Grimán Padua
Antonio Jesús Roa Valverde
Antônio Oliveira Filho
Antonio Ruiz-Cortés
Arturo Zambrano
Carlos Bobed
Carlos D. Barranco González
Carlos Enrique Cuesta Quintero
Carlos Neil
Cecilia Delgado Negrete
César J. Acuña
Claudio Sant`Anna
Cristina Vicente Chicote
Daniel Rodríguez
Dante Carrizo
Diana Marcela Sánchez
Diego Alonso Cáceres
Diego Seco Naveiras
Domingo Savio Rodríguez Baena
Eduardo Rodríguez López
Elisa Yumi Nakagawa
Ellen Francine Barbosa
Encarna Sosa Sánchez
Fernando Molina Molina
Fran J. Ruiz Bertol
Francisco Javier Lucas Martínez
Francisco Luís Gutiérrez Vela
Francisco Martínez Álvarez
Ignacio García Rodríguez de Guzmán
Ismael Caballero
Ismael Navas Delgado
Ismael Sanz Blasco
Javier Pérez García
Joaquín Lasheras
Joaquín Nicolás
Jorge Gracia
Jorge Martínez Gil
José María Cavero Barca
Juan Ángel Pastor Franco
Juan M. Vara
Juan Manuel Pérez Martínez
Manuel Ángel Serrano Martín
Manuel Resinas
Márcio de Medeiros Ribeiro
Marcirio Chaves
Marcos López Sanz
Mari Carmen Otero
María Esperanza Manso Martínez
María Luisa Rodríguez Almendros
María Teresa Gómez López
María Visitación Hurtado Torres
Martin Solari
Miguel Ángel Laguna Serrano
Miguel Ángel Martínez
Miguel Rodríguez Luaces
Mª Ángeles Moraga de la Rubia
Nuno Cardoso
Orlando Avila-García
Oscar Dieste
Óscar Pedreira Fernández
Othmane Chniber
Pablo Inostroza
Pablo Trinidad
Paloma Cáceres García de Marina
Pedro Sánchez Palma
Raquel M. Crespo García
Raquel Trillo Lado
Roberto Almeida Bittencourt
Roberto Rodríguez Echeverría
Roberto Ruiz
Rui Lopes
Sascha Ossowski
Sergio Ilarri Artigas
Vicente Luque Centeno

Sponsors


Universidad de Oviedo
400
cuarto centenario


PCTI
ASTURIAS


Ayuntamiento de Gijón


GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA E INNOVACIÓN


GOBIERNO DEL PRINCIPADO DE ASTURIAS


TIC
CLUSTER ASTURIAS


csc


InterSystems


cajAstur


DEPARTAMENTO DE INFORMATICA
UNIVERSIDAD DE OVIEDO


Sistedes
Sociedad de Ingeniería del Software y Tecnologías de Desarrollo de Software


gijon.info

Table of Contents[1]

[1] The section headings below correspond to the conference program, but do not include all the presentations in each conference session (where short papers and dissemination papers on the same topic also were included). Thus, the sections here all contain fewer papers than the corresponding conference session; the short papers are listed separated in this volume, followed by a chapter with an overview of the dissemination papers.

Information Engineering

Model Engineering

Formal Methods

Maintenance and Testing

Data Mining, Data Streaming and Datawarehouses

Reengineering and Software Modernization

Quality, Measurement & Estimation of Products & Processes

Requirements Engineering

Keynote Address 2

Short Papers

Workshops, Tutorials, Demos and Dissemination

Priorización del Valor de Artefactos Software Basada en la Frecuencia de Uso

Daniel Cabrero[1], Javier Garzás[2] y Mario Piattini[3]

[1] Dirección General de Tráfico. Subdirección Gral. de Informática,
C/Josefa Valcárcel 44. Madrid (España)
dani_mas_d@yahoo.fr
[2] Universidad Rey Juan Carlos – Grupo de Investigación Kybele - Kybele Consulting SL,
javier.garzas@kybeleconsulting.com
[3] Grupo de Investigación ALARCOS. Universidad de Castilla la Mancha
Mario.Piattini@uclm.es

Resumen. El coste de mantenimiento de un sistema software es directamente proporcional a la cantidad de líneas de código y el número de funcionalidades que es necesario mantener. A mayor tamaño, mayor coste de mantenimiento. El mantenimiento tradicional de los sistemas no tiene en cuenta el "Valor" relativo de cada artefacto software a mantener, entendiendo como artefacto cada modulo, clase, método o atributo que compone un sistema. El presente trabajo propone asignar un Valor relativo a los artefactos a mantener en base a la frecuencia de utilización de los mismos, de manera que bajo determinadas circunstancias, los artefactos que no aportan valor al usuario puedan ser eliminados, optimizando así el coste de mantenimiento. Lo que se propone es, por tanto, un método de simplificación y reducción del tamaño del software basado en herramientas de libre distribución, y con un alto grado de automatización, que proporcione una orientación sistemática al frecuente problema del mantenimiento de grandes sistemas legados o no documentados.

Palabras Clave:, Ingeniería del Software Basada en Valor, Reingeniería, Frecuencia de Utilización, Priorización de artefactos Software

1 Introducción

Elliot Chikofsky comenzó la presentación inaugural de la primera conferencia del Ingeniería del Software basada en Valor (IEEE EQUITY) subrayando el problema que suponía el hecho de que los gestores de sistemas software enfoquen casi todo sus recursos a la creación de nuevos sistemas y funcionalidades, y prácticamente ningún esfuerzo a retirar sistemas y funcionalidades obsoletas o no utilizadas [1]. En su exposición, se hacía una analogía de los circuitos de carreras y la gestión de proyectos, donde en un circuito entraban nuevos coches (proyectos), pero nunca se retiraban.

La retirada de sistemas no utilizados es una tarea relativamente sencilla partiendo de estadísticas de utilización disponibles en muchos tipos de sistemas centralizados, incluidos los Sistemas Web o Mainframe. Desgraciadamente este no es el caso de la simplificación de subsistemas, módulos o incluso clases de diseño de un aplicativo software, donde las frecuencias de utilización no son monitorizadas por los equipos de explotación y sistemas.

Muy frecuentemente, en la industria del software se desarrollan productos de gran tamaño, que se corresponden con especificaciones funcionales simples, que sugieren desarrollos de menor tamaño. A lo largo de los años de experiencia de los autores de este artículo, han sido frecuentes interrogantes similares a:

- ¿400.000 líneas de código para algo tan sencillo?
- ¿Es realmente necesario que el sistema contemple esta funcionalidad?

En muchos casos, ciertamente la respuesta es no. El problema reside en la imposibilidad de identificar qué elementos no aportan valor alguno de entre una compleja maraña de referencias entre clases, habitualmente exentas de la documentación de análisis y diseño necesaria para comprender el sistema.

A una incorrecta orientación en el desarrollo, se le debe sumar el agravante del crecimiento inherente al software. Según los estudios realizados [2], el coste de la gestión de un sistema crece constantemente y raramente se reduce, reservando excepcionalmente la labor de simplificación y reorganización de dichos sistemas a complejas y costosas tareas de reingeniería global. Incluso en el caso de simplificación mediante reingeniería, los ingenieros del software no disponen de datos que permitan evaluar la utilización que los usuarios han hecho de cada uno de los artefactos que conforman el sistema a reingenierizar.

En este trabajo de investigación se propone un método que permite paliar las carencias expuestas. Para ello, **se propone utilizar la frecuencia de utilización de los distintos artefactos que componen el software como indicador de simplificación**. Como caso más sencillo y a modo de ejemplo, una clase que no ha sido utilizada durante todo un ciclo de vida de una aplicación corresponde a un artefacto candidato a ser eliminado.

Este artículo se organiza de la siguiente manera. La sección 2 introduce el problema del tamaño y la complejidad del software como base para justificar la utilidad que tiene una investigación más profunda sobre este tópico. La sección 3 introduce el concepto de "Valor" en el marco de un proceso de simplificación de software, e introduce un ejemplo simple que ilustra las líneas maestras del método propuesto. La sección 4 describe en detalle la metodología propuesta, así como las herramientas utilizadas durante el proceso. Finalmente, la sección 5 resume las conclusiones de la investigación y describe el trabajo futuro pendiente.

2 El problema del tamaño y la complejidad del software

El tamaño del software disminuye la mantenibilidad y la predictibilidad de los sistemas. McConell [3] coincide con otros expertos en que el tamaño de los sistemas aumenta en un proyecto software las posibilidades de retrasos en la entrega o cancelaciones del proyectos en un una media de un 15% y un 60% respectivamente [4]. La gran mayoría de los autores coinciden también en que el crecimiento de los sistemas es un problema a evitar de cara a un mantenimiento efectivo del software.

Si bien no existe acuerdo en la tasa media de crecimiento de un sistema a lo largo de su versionado, "*se podría considerar una tasa de crecimiento modesta aquella en que cada nueva versión añade tantas líneas de código como contenía la versión inicial*" [2]. Esto significaría que un sistema doblaría su tamaño en la primera versión, aumentaría un 33% en la segunda, un 25% en la tercera, y así sucesivamente. El aumento del tamaño de los sistemas hace que en general sean más difíciles de comprender, y más costosos de mantener a todos los niveles.

La complejidad del software también es un factor determinante en la mantenibilidad de los sistemas. Según el estándar ISO 9126 [5], la "*Analizabilidad*" de los sistemas es uno de los elementos claves que afectan a la Mantenibilidad. Cuanto más complejo sea un sistema, es decir, cuantos más elementos lo compongan y más relaciones existan entre los elementos, mayor será el esfuerzo que es necesario realizar para entender cómo funciona, y poder cambiarlo [6].

Podemos, según lo argumentado anteriormente, concluir que **la tendencia natural a lo largo del tiempo de la mayoría de los proyectos software es la de crecer y volverse más complejos**, y por ende menos mantenibles. Se parte de la base de que "*El mantenimiento consume la mayor parte de recursos del ciclo de vida de las aplicaciones*" [7, 8], ya que, según Parnas, "la Ingeniería del Software trata de construir aplicaciones multiversión" [9], o lo que es lo mismo, sobre cómo construir aplicaciones mantenibles.

El impacto económico del mantenimiento de los sistemas, unido a la tendencia natural de crecimiento del software, y la disminución de la mantenibilidad que dicho crecimiento conlleva,

justifican a nuestro juicio la realización de un esfuerzo de investigación en técnicas de simplificación y reducción de sistemas, que raramente son empleadas en la práctica industrial de desarrollo software.

Según nuestra experiencia, podemos concluir que ante la **carencia de métodos reproducibles que permitan solventar el problema de la optimización del tamaño de código**, la praxis más común en la industria del software es el mantenimiento de los sistemas sin optimizar, hasta un punto en que la reconstrucción total se hace imprescindible por la ineficaz mantenibilidad de los sistemas.

La imagen muestra gráficamente lo comentado anteriormente, donde el ciclo de vida de las aplicaciones hace que crezcan a lo largo de las distintas versiones. Una vez el sistema es difícil de mantener, se plantea una reingeniería, que tiene por objeto mejorar la mantenibilidad del diseño.

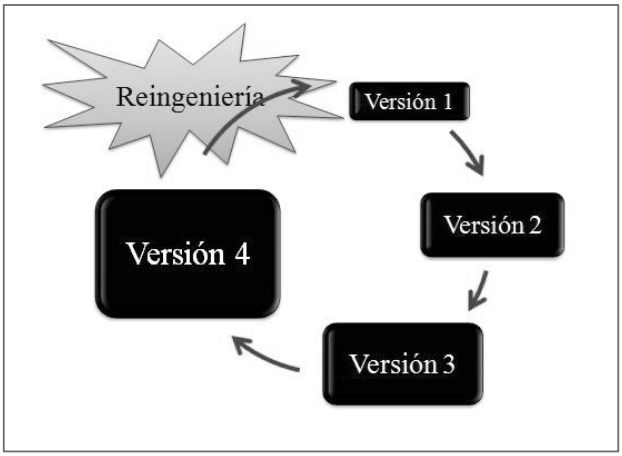

Fig. 1. Modelo clásico de evolución del tamaño de los sistemas software.

La novedad en la propuesta contenida en este trabajo consiste en desarrollar un método ágil, que permita realizar tareas de optimización menos pesadas que la reingeniería mostrada anteriormente La idea consiste en realizar tareas de simplificación del sistema, eliminando las funcionalidades que no son necesarias. Dicho método ha sido desarrollado dentro del contexto de organizaciones que desarrollan y mantienen una gran cantidad de software y con la automatización de tareas en mente, de manera que sea lo suficientemente rápido de utilizar para que pueda ser aplicado a la industria de desarrollo software.

Para hacer posible la aplicación de dicha técnica, es necesario tener en cuenta que debe tratarse de una metodología basada en herramientas que permitan un grado de automatización tal que el coste de simplificación de un sistema sea muy inferior al coste de mantenimiento extra que supondría mantener el sistema original

La siguiente imagen muestra la nueva orientación, donde en contraposición con el modelo clásico, se potencia la eliminación de artefactos software no utilizados a lo largo de la vida útil del sistema, de manera que el mantenimiento sea en todo momento lo más sencillo posible. Nótese que siguiendo esta nueva orientación de simplificación no es necesario, o al menos podremos evitar en muchos casos, el proceso de reingeniería que se describía en el modelo clásico de mantenimiento. En su lugar se llevan a cabo simplificaciones automáticas del código no utilizado en cada versión liberada a los usuarios.

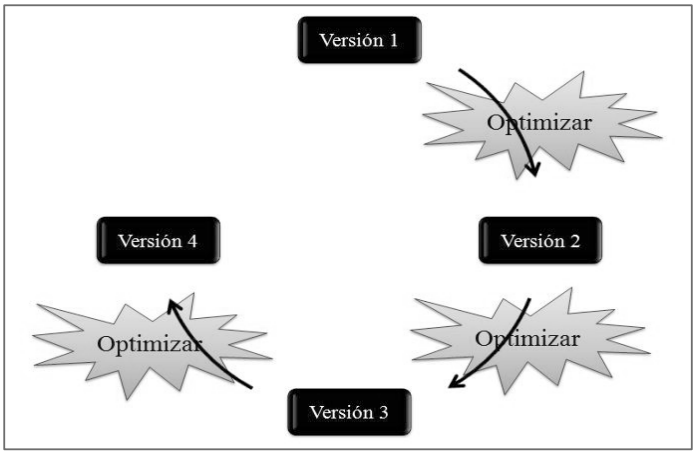

Fig. 2. Modelo propuesto de evolución del tamaño de los sistemas software.

3 Aplicando la Ingeniería Basada en Valor

Recientemente, Boehm [10] introdujo una agenda que pretende integrar las consideraciones de "Valor" dentro del conjunto de principios y prácticas existentes en la ingeniería del software. Uno de los siete elementos principales de esa agenda es la "Arquitectura, diseño y desarrollo basado en valor". A nivel de diseño, la aplicación de la orientación Basada en Valor consiste en asignar un valor relativo a cada uno de los artefactos de diseño, de manera que puedan priorizarse los trabajos centrando los esfuerzos en los elementos que aporten más valor al conjunto.

Existen muy pocas propuestas de priorización de artefactos de diseño. Una posibilidad es la asignación de valor para cada artefacto de diseño en función de los costes de mantenimiento. Esta propuesta se basa en el hecho de que "*el mantenimiento software consume la mayor parte de los recursos del ciclo de vida de las aplicaciones*" [7, 8]. En [11] se discute la priorización de los artefactos en base a la probabilidad de cambio que tienen con objeto de realizar refactorizaciones [12] donde sea más necesario, o dicho de otro modo, hacer más mantenibles los elementos de diseño que más cambiarán en un futuro. Es decir, se estima la cantidad de cambios que sufrirá cada artefacto software [13], y se modifica dicho artefacto para que sea más sencillo realizar cambios sobre éste, y de esta forma optimizar el mantenimiento.

Sin embargo, exceptuando la técnica anteriormente descrita, no se han encontrado propuestas que versen sobre la cuantificación del Valor relativo aportado por cada artefacto de cara al mantenimiento, seguramente como consecuencia del poco tiempo transcurrido desde la identificación de esta nueva línea de investigación denominada "*Ingeniería Basada en Valor*".

El trabajo que aquí se presenta pretende continuar con la idea de optimización de mantenimiento. A diferencia de la técnica anteriormente descrita, en lugar de centrarse en la probabilidad de cambio, se basa en la frecuencia de utilización de cada artefacto software.

Una manera de optimizar el mantenimiento es **seleccionar los elementos de diseño que aportan una mejor relación valor-coste a la solución**, y descartar aquellos que no cumplan dicha premisa. El enfoque aquí propuesto parte de esa base para estudiar el valor de diseño en función de los costes de mantenimiento asociados al mismo, en forma de coste asociado a elementos que nunca o casi nunca se utilizan.

En resumen, lo que se propone es identificar **los elementos que no se usan o que se usan muy poco** para posteriormente proceder a su eliminación, de manera que el sistema resultante cubra igualmente las necesidades del usuario pero sea más sencillo de mantener, al tratarse de un sistema

más pequeño, y que cubre únicamente la funcionalidad útil, es decir, la funcionalidad que aporta "Valor".

Indudablemente, para implementar el proceso de simplificación que aquí se plantea, es necesario conocer la frecuencia de utilización de cada módulo, clase, método y variable que componen nuestro sistema. Para ello existen actualmente dos aproximaciones claramente diferenciadas:

- **Examen estático del código**. Consiste en buscar métodos y variables no utilizadas desde ningún módulo del sistema, y que por tanto no serán referenciados en tiempo de compilación, por lo que su eliminación no tendrá repercusión alguna en la funcionalidad del sistema.

- **Examen dinámico del código**. Permite detectar artefactos de código que aun siendo referenciados en tiempo de compilación, no son frecuentemente utilizados. Para detectar qué artefactos no son utilizados en tiempo de ejecución, es necesario realizar un seguimiento de las llamadas que son realizadas dentro del sistema a lo largo de su vida útil.

Existen varias herramientas que soportan el examen estático del código como [14], [15], [16] o [17], por tanto parece claro que es una rama lo suficientemente madura como para realizar aportaciones de peso a nivel de investigación. Debido a ello, en el marco de este trabajo **se abordará esencialmente el examen dinámico** del código, es decir, la inclusión de factores de probabilidad de uso al diseño, campo en el que no se han encontrado referencias a herramientas ni métodos en la comunidad investigadora, y tampoco entre las prácticas industriales más comunes. Sí se han encontrado trabajos que abordan la detección de código muerto, en sistemas modelados a través de reglas lógicas, mediante la comprobación de condiciones que nunca esperaban cumplirse [18]. En este caso la aproximación al problema es radicalmente opuesta, dado que podemos aplicar la metodología a un sistema legado, que típicamente no está modelado, y mucho menos mediante reglas lógicas.

3.1 Ejemplo simple de simplificación de un sistema

A modo de ilustración del problema, se procederá a analizar un ejemplo sencillo de simplificación de un sistema. Imaginemos que tenemos un pequeño aplicativo que permite realizar consultas sobre distintos campos de una base de datos de vehículos y conductores. En particular, existe una clase en la capa de presentación (*ControladorPresentación*) capaz de invocar a clases de negocio de vehículos (*GestorVehículos*) y conductores (*GestorConductores*). El objetivo es gestionar la base de datos de conductores y vehículos respectivamente. La siguiente figura muestra un diseño en UML de las clases que componen este pequeño subsistema.

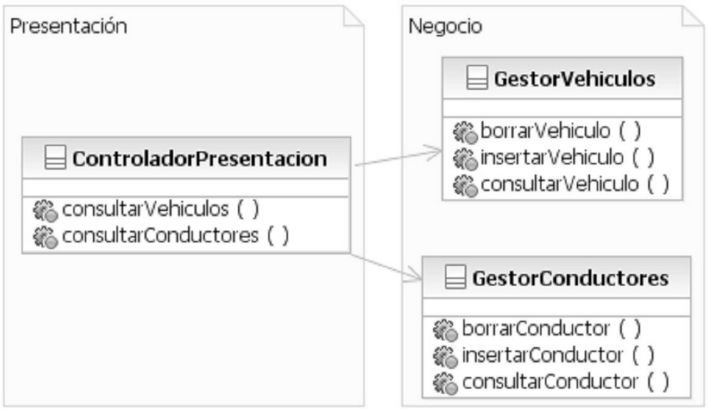

Fig. 3. Ejemplo simple de modelo de clases a optimizar

Imaginemos ahora que los usuarios únicamente estuvieran interesados en las consultas de vehículos, de forma que la clase "*GestorConductores*" únicamente es invocada unas pocas veces al año. Adicionalmente, la funcionalidad de "*insertarConductor*", incluida dentro de dicha clase, nunca ha sido invocada durante toda la vida de la aplicación, dado que el alta de nuevos conductores se realiza a través de otro aplicativo.

Obviamente, si el equipo de desarrollo hubiera conocido estas frecuencias de utilización a priori, probablemente no se hubiera desarrollado ninguna gestión de conductores, y casi con toda seguridad, no se habría construido ninguna inserción de conductores. Es importante resaltar también, que incluso pasado un tiempo prudencial de utilización del aplicativo por parte del usuario, probablemente tampoco se dispusiera de estos datos de gran importancia para priorizar los esfuerzos de análisis, diseño, construcción y pruebas, necesarios para poner en marcha cada funcionalidad.

Al término del proceso, y basándose en la frecuencia de utilización, se podría proponer la siguiente simplificación del sistema, que implicaría un coste de mantenimiento menor.

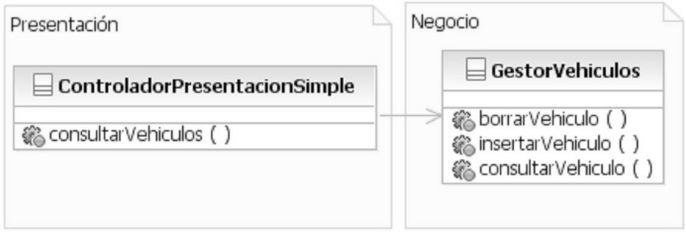

Fig. 4. Ejemplo simple de modelo de clases optimizado

Por tanto, esta información podría ser útil para varios fines:

- Al inicio del desarrollo de la siguiente versión del producto, ya que permite priorizar el desarrollo de la parte más utilizada, de forma que ésta estuviera disponible en un plazo menor de tiempo.
- Para guiar refactorizaciones, aplicando mayor esfuerzo a la parte más utilizada.

- Para simplificar y reducir el sistema, eliminando la parte que prácticamente no se utiliza.
- En la planificación de las pruebas, centrando el esfuerzo en asegurar el correcto funcionamiento de los artefactos más utilizados, y reduciendo el coste de las pruebas de regresión y del mantenimiento de los casos de prueba.
- Para ajustar un sistema a un presupuesto dado, descartando las funcionalidades menos esenciales.
- En definitiva puede aplicarse a la priorización de cualquier esfuerzo relacionado con un artefacto software.

El objetivo último de este ejemplo es mostrar cómo **la frecuencia de utilización puede ser de gran importancia para asignar un "Valor relativo"** a los distintos artefactos que componen nuestro diseño. Una vez se hayan asignado valores numéricos a cada artefacto software, será posible priorizar los esfuerzos.

De todos los posibles fines para los que puede ser utilizado el valor relativo de los distintos artefactos software, este trabajo se centra en la simplificación y reducción de sistemas grandes. Para ello, **se propone utilizar la metodología MERES**, que se describe en detalle en la siguiente sección.

4 Propuesta de Metodología: MERES (MÉtodo de REducción de Software)

El ejemplo anterior muestra el concepto de "Valor relativo" basado en la frecuencia de utilización de los distintos artefactos software. No obstante, la simplificación de sistemas reales de gran tamaño debe apoyarse en una metodología que defina de forma más detallada cada paso a seguir. En este epígrafe se expone en detalle la metodología propuesta para el proceso de optimización de software, basado en los conceptos presentados en el ejemplo anterior, añadiendo el detalle suficiente para que pueda ser también aplicado al desarrollo industrial de software.

Del mismo modo y tal y como ya se ha comentado, se cree crítico que el **método esté basado en herramientas que faciliten y automaticen su ejecución**. El éxito de un método cuyo objetivo es reducir los costes de mantenimiento de un software depende de forma directa de los recursos necesarios para su ejecución, ya que su aplicación misma supone un coste adicional en contra de dichos objetivos de ahorro de costes.

Los siguientes puntos presentarán en detalle la metodología y herramientas propuestas.

4.1 Pasos a seguir

Paso 1: Puesta en producción. Lógicamente, el primer paso en la aplicación de un método basado en la actividad de utilización del usuario es poner la aplicación a su disposición. En este punto es necesario activar alguna herramienta de monitorización de la ejecución. Más adelante se comentarán las distintas alternativas tecnológicas existentes para ello.

Paso 2: Obtención de la frecuencia de utilización de cada artefacto. La herramienta de monitorización proporcionará automáticamente un registro de las veces que un método ha sido ejecutado. Será necesario ordenar por frecuencia de utilización, de forma que se seleccionen los artefactos que nunca o casi nunca fueron utilizados. El producto obtenido de este paso será una tabla que ordene por frecuencia de ejecución las invocaciones realizadas sobre cada uno de los distintos artefactos.

La siguiente tabla muestra los datos aplicados al ejemplo anterior.

Tabla 1. Información relevante sobre la utilización de los artefactos.

Módulo	Clase	Método
Presentación	ControladorPresentación (14126)	consultarVehiculo (14023)
		consultarConductor (3)
Vehiculos	GestorVehiculos (21160)	borrarVehiculo (100)
		insertarVehiculo (500)
		consultarVehiculo (20560)
Conductores	**GestorConductores (20)**	**borrarConductor (0)**
		insertarConductor (0)
		consultarConductor (20)

Paso 3: Aplicación de criterios de decisión. El criterio general deberá ser la eliminación de módulos, seguido de clases, y por último de métodos. Esta heurística tiene por objetivo aprovechar la modularización inherente a los conceptos de módulo y clase para minimizar el posible impacto de la simplificación. Partiendo de los datos anteriores, será necesario además configurar los criterios de simplificación:

- Por debajo de qué límite de utilización un artefacto dejará de ser útil. Esto dependerá del tipo y cantidad de usuarios registrados, por ejemplo, si se trata de una aplicación web utilizada por miles de usuarios, el límite será mucho más alto que si se trata de aplicaciones departamentales, donde una funcionalidad puede ser muy importante aunque sea invocada por un tipo de perfil determinado de usuarios con poca frecuencia.

- Artefactos excluidos del análisis por motivos conocidos (utilidades de administración, o tareas "batch" que son lanzadas con poca frecuencia).

- Configuración de límites particulares para algunos módulos que el equipo de desarrollo identifica a priori como candidatos.

- Configuración del borrado automático de artefactos, es decir, especificar si el aplicativo debe preguntar antes de borrar o no.

Paso 4: Eliminación de los módulos/clases/métodos candidatos. Una vez configurado, se procede a la eliminación de los artefactos candidatos, asimismo se eliminan las llamadas a dichos artefactos, sustituyéndolas temporalmente por una funcionalidad de escritura en log. Este apunte contendrá una alerta de la ejecución de dicha funcionalidad, además de un identificador único mediante el que es posible encontrar el código borrado. El identificador único permitirá restablecer los elementos borrados en las siguientes versiones de la aplicación automáticamente si se considera necesario. En caso de consolidación definitiva, se procederá al borrado de los marcadores de escritura a fichero.

La aplicación del método de borrados temporales y consolidaciones permitirá realizar una **simplificación "adaptativa"** del código. Permitiendo reducir considerablemente el tamaño del software, pero teniendo la seguridad de que en caso de fallo provocado por la simplificación es posible volver automáticamente al estado original del artefacto que ha sido invocado.

Paso 5. Realización de pruebas. El último paso antes de la puesta en producción es la ejecución de una batería de pruebas funcionales (y si es posible, de regresión), además de pruebas de aceptación de usuario, que aseguren que la simplificación no afecta al funcionamiento habitual del aplicativo. La fase de pruebas es indudablemente siempre recomendable, pero adquiere una relevancia adicional tras la ejecución de cualquier simplificación.

Adicionalmente, es recomendable que la evolución y simplificación del software esté siempre basado en un **sistema de gestión de la configuración adecuado** (control de versiones), para poder volver al estado anterior en caso de detectar algún mal funcionamiento causado por el proceso de simplificación.

4.2 La automatización como factor crítico de éxito: Herramientas

En varios puntos del presente artículo se ha señalado la importancia del grado de automatización de la metodología de cara a que ésta sea aplicable en el mantenimiento real de aplicaciones. Por este motivo, las herramientas en las que descansa la automatización de la metodología es especialmente importante. A continuación se presentan las distintas opciones consideradas.

Herramientas de monitorización de actividad. En primer lugar y para que la metodología funcione es necesario utilizar una herramienta que permita monitorizar la actividad de los distintos usuarios, y que permita conocer la frecuencia de utilización de los distintos artefactos. Para ello, se han considerado principalmente dos opciones.

- Por un lado, existen en el mercado **herramientas que "instrumentalizan"** los servidores de aplicaciones. Se trata esencialmente de herramientas pensadas para la evaluación de la cobertura de pruebas funcionales o unitarias. Existen multitud de herramientas comerciales de cobertura de pruebas, y algunas de software libre como EMMA Coverage Tool [19], herramienta seleccionada para nuestros casos de estudio.

- Por otro lado, es posible utilizar la **Programación Orientada a Aspectos** (*AOP-Aspect Oriented Programming*), y que está actualmente siendo propuesta con éxito para diversos usos relacionados con el desarrollo software [20]. En el contexto de este trabajo se puede utilizar para interceptar las llamadas que se realizan dentro de las aplicaciones, y generar un log de invocaciones que posteriormente puede ser transformado en un informe de actividad de cada artefacto. Para ello, existen utilidades que facilitan el uso de aspectos como AspectJ [21] o Spring [22].

En principio, y salvo que el avance de los trabajos indique lo contrario, en el marco de este trabajo se utilizarán herramientas de cobertura, en particular se utilizará la herramienta EMMA Coverage Tool. Esta elección ha sido basada en la realización de pruebas de rendimiento, que arrojan unos índices menores de sobrecarga del sistema, y por tanto la hace más recomendable de cara a su utilización en un sistema en producción que las técnicas de AOP. Adicionalmente EMMA provee de una utilidad de exportación de informes a XML que permite tratar los informes de frecuencia de utilización sin necesidad de construir una aplicación que elabore un informe a partir de log de actividad.

Desarrollo de la herramienta de soporte a MERES. Para completar el grado de automatización necesario, se está construyendo una herramienta de ayuda a la simplificación, capaz de importar los datos extraídos de la monitorización en formato XML, agrupar la configuración de los criterios de simplificación presentados en el paso 3 de la metodología, y automatizar el proceso de borrado y chequeo de artefactos no referenciados.

Adicionalmente, y tal y como se sugería en la descripción de la metodología (en el paso 4. Eliminación de artefactos), la herramienta deberá ser capaz de guardar el código fuente borrado, e identificarlo de manera unívoca mediante una clave única. Las llamadas a dicho código serán sustituidas por anotaciones de log de dicha clave única. En caso de ser necesario, será posible identificar qué código fuente borrado fue invocado, mediante la clave única, y restaurar así el

artefacto eliminado. El objetivo final de este proceso es poder recuperar código fuente borrado que ha sido utilizado para la siguiente versión de un producto.

4.3 Aspectos adicionales a tener en cuenta

Es evidente que cualquier automatización sobre el código fuente ahorra por un lado una gran cantidad de recursos, pero añade un elemento de incertidumbre sobre la fiabilidad de los aplicativos que se ponen en producción. **¿Qué ocurre si el proceso automático elimina por error algún artefacto que afecte al funcionamiento de la aplicación?.** El desarrollo de la metodología no debe ser ajeno a este hecho. Para mitigar este riesgo, la metodología MERES prevé diversas acciones preventivas y correctivas.

En primer lugar, almacena todo el código fuente borrado, y utiliza un log que permite saber qué invocaciones iban dirigidas al código borrado. La metodología provee de un sistema de recuperación de código fuente borrado, si este es ejecutado. Se trata por tanto de un **"borrado lógico de código fuente"**, ya que en caso de invocación de ese código pueden tomarse acciones correctivas de recuperación de dicho código fuente.

Adicionalmente, la metodología MERES pone especial énfasis en la **gestión de la configuración**, que permitirá volver al estado anterior a la simplificación.

Por último, el paso 5 de la metodología se centra en la **realización de pruebas** que minimicen el riesgo de una puesta en producción de software defectuoso.

Otro aspecto importante a tener en cuenta son los periodos de prueba necesarios para considerar un artefacto no utilizado. **¿Cuánto tiempo puede considerarse válido para marcar un artefacto no utilizado?.** Ciertamente la respuesta varía en función del sistema que estemos analizando. Este es el motivo por el que se trata de un parámetro configurable a través de la herramienta. En general, si se trata de aplicaciones centralizadas utilizadas por multitud de usuarios con el mismo perfil (como podría ser una aplicación web o corporativa), se pueden establecer unas condiciones menos estrictas para la eliminación de artefactos, es decir, consideramos un tiempo de monitorización bajo, y un umbral de frecuencia de utilización alto. En el caso de aplicaciones más especializadas manejadas por pocos usuarios, se deberá ser mucho más exigente en lo que se refiere a la eliminación de artefactos, pues según nuestra experiencia es más habitual encontrar funcionalidades que son poco utilizadas pero resultan imprescindibles para los usuarios.

Por último, un aspecto importante que se planteaba era la **incidencia de la monitorización sobre el rendimiento** de las aplicaciones en producción. En este sentido se han realizado diversas pruebas de rendimiento sobre aplicativos de la Dirección General de Tráfico, y se han comparado los tiempos de respuesta obtenidos con y sin monitorización. El resultado ha sido satisfactorio para todas las pruebas realizadas, teniendo la monitorización una **incidencia inferior al 10**%. En cualquier caso, y para minimizar el riesgo de ralentización de los sistemas en producción, se propone utilizar en los casos de estudio un **grupo de aplicativos no exigentes** en lo referente a tiempos de respuesta.

5 Conclusiones y trabajo futuro

La simplificación automatizada del código fuente de grandes aplicativos constituye un reto de difícil solución en el que hay todavía un largo camino por recorrer. Se trata de un tema complejo para el que todavía no existe una rama de investigación madura, pero que aspira a resolver un problema de plena actualidad en la industria del desarrollo software y de gran impacto económico.

El principal problema de la simplificación de sistemas es el alto coste que supone si se realiza manualmente, y la escasa fiabilidad del software resultante si se realiza de forma automatizada. Con el objetivo de resolver este problema, se presenta en este artículo una nueva metodología llamada MERES (MÉtodo de REducción de Software), que utiliza la frecuencia de utilización del

sistema para asignar un valor relativo de cada artefacto software. Los artefactos que aporten menos valor a la solución serán borrados del sistema. La aportación del método reside en que se realiza un borrado lógico, y se deja un marcador que permite saber qué código borrado se ha intentado ejecutar, e incluso permite restablecer dicho código.

La metodología está soportada por la herramienta de software libre EMMA de cobertura, y complementada por un software a medida que se está desarrollando para proveer de un soporte automatizado. En este sentido es importante resaltar que una adecuada automatización es un factor crítico de éxito, dado que el motivo de que no se lleven a cabo las tareas de simplificación de software es el elevado coste manual que suponen.

El estado actual de los trabajos es el siguiente. Se han realizado comprobaciones manuales de la efectividad de la aproximación sobre proyectos reales de poco tamaño en el entorno de desarrollo de la Dirección General de Tráfico. Se ha comenzado el proceso de desarrollo de la herramienta MERES que soporte la automatización de la metodología para infraestructuras J2EE. Finalmente, se está en proceso de selección de una serie de proyectos candidatos, desarrollados por diez proveedores diferentes de software para la Dirección General de Tráfico, sobre el que se probará la metodología y se contrastarán los resultados obtenidos.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto ESFINGE (Ministerio de Ciencia y Tecnología (TIN 2006-15175-C05-05), el proyecto MELISA (PAC08-0142-3315), Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia, y el proyecto IDONEO project (PAC08-0160-6141). Junta de Comunidades de Castilla-La Mancha

Referencias

1. Chikofsky, E.: Engineering Management & Integration. In: 1st IEEE Computer Society Conference on Exploring Quantifiable Information Technology Yields.pp. IEEE Computer Society. Amsterdam (2007)
2. Wiederhold, G.: What is your Software Worth? In: Communications of the ACM. vol. 49(9): pp. 65-75. (2006)
3. McConnell, S.: Software Estimation: Demystifying the Black Art. Microsoft Press. Redmond, Wa (2006)
4. Jones, C., *Estimating Software Costs*, McGraw-Hill, Editor. 1998: New York.
5. ISO-9126. Software Product Quality. http://www.iso.org.
6. Garzás, J., D. Cabrero, and M. Piattini: El valor y el retorno de inversión de las TSI, In: Gobierno de las Tecnologías y Sistemas de Información, RA-MA. pp. 43-62. Madrid, España. (2007)
7. Bennet, K.H. and V.T. Rajlich: Software Maintenance and Evolution: a Roadmap. In: ICSE (Track on The Future of Software Engineering).pp. Limerick, Ireland (2000)
8. Pigosky, T.M.: Practical Software Maintenance. John Wiley & Sons. New York (EEUU) (1997)
9. Parnas, D.L.: Software fundamentals: collected papers by David L. Parnas. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA (2001)
10. Boehm, B.: Value-Based Software Engineering: Overview and Agenda, In: Value-Based Software Engineering Springer. pp. 3-14. Heidelberg, Germany. (2005)
11. Cabrero, D., J. Garzás, and M. Piattini: Maintenance Cost of a Software Design. A Value-Based Approach. In: 9th International Conference on Enterprise Information Systems (ICEIS).pp 384-389. Funchal, Madeira. Portugal (2007)
12. Fowler, M.: Refactoring. Addison Wesley Professional. (1999)
13. Cabrero, D., J. Garzás, and M. Piattini: Combining Different Change Prediction Techniques. In: 10th International Conference on Enterprise Information Systems (ICEIS).pp. Barcelona (Spain) (2008)
14. VisualCodeScan. http://windsweptsoftware.com/codetool/vcs6.htm.
15. JiveLint. http://www.sureshotsoftware.com/javalint/index.html.

16. Bodík, R. and R. Gupta: Partial dead code elimination using slicing transformations. In: ACM SIGPLAN 1997 conference on Programming language design and implementation.pp. ACM. Las Vegas, Nevada, United States (1997)
17. Eclipse. http://www.eclipse.org/.
18. Sims, S., et al.: Automated Validation of Software Models. In: 16th IEEE international conference on Automated software engineering.pp 91-96. IEEE Computer Society. San Diego, USA (2001)
19. EMMA. http://emma.sourceforge.net/.
20. Steimann, F.: The paradoxical success of aspect-oriented programming. In: Object Oriented Programming Systems Languages and Applications pp 481 - 497. ACM Portland, Oregon, USA (2006)
21. AspectJ. http://eclipse.org/aspectj.
22. Spring. http://www.springframework.org/.


Universidad
de Oviedo

400
cuarto
centenario

PCTI
ASTURIAS

Ayuntamiento de Gijón

GOBIERNO
DE ESPAÑA
MINISTERIO
DE CIENCIA
E INNOVACIÓN

Gobierno del
Principado de Asturias

TIC
CLUSTER ASTURIAS

CSC

INTERSYSTEMS

cajAstur

DEPARTAMENTO DE INFORMÁTICA
UNIVERSIDAD DE OVIEDO

Sistedes
Sociedad de Ingeniería del Software y
Tecnologías de Desarrollo de Software

gijon.info