Proceedings of ASSE

Proceedings of ASSE 2010
Argentine Symposium on Software Engineering

ISSN: 1850-2792

The 11th Argentine Symposium on Software Engineering (ASSE 2010) will bring together researchers, developers, and practitioners to discuss new ideas, problems and experiences in the field of Software Engineering. ASSE 2010 will be part of the 39th Argentine Conference on Informatics (JAIIO 2010). ASSE 2010 seeks original works in the wide spectrum of Software Engineering, from academic research to industrial and business applications with significant impact and lessons learned from application development. The symposium will feature invited talks, paper and poster sessions, and panels presenting both mature work and new ideas in research and applications.

Chairs

Santiago Ceria, Hexacta and UBA, Buenos Aires, Argentina

Claudia Pons, CONICET and UNLP, La Plata, Argentina

Program Committee

Silvia Teresita Acuña (Universidad Autónoma de Madrid, España)

Gabriela Arevalo (LIFIA, Facultad de Informática, UNLP, Argentina)

Maria Cecilia Bastarrica (Depto de Ciencias de la Computación, Universidad de Chile)

Alejandro Bianchi (Liveware)

Eduardo Bonelli (UNQ, ITBA, Argentina)

Victor Braberman (FCEyN, UBA, Argentina)

Marcelo Campo (ISISTAN, UNCPBA, Argentina)

Alejandra Cechich (GIISCo, UNComa, Argentina)

Omar Chiotti (Conicet, UTN Santa Fe, Argentina)

Emilce Chiricola (Siemens)

Juan José Cukier (Pragma Consultores)

Pedro R. D'Argenio (FaMAF, Universidad Nacional de Córdoba)

Andrés Flores Mir (GIISCo, UNComa, Argentina)

Marcelo Frias (FCEyN, UBA, Argentina)

Juan Gabardini (AgilAr, UBA)

Carlos Garcia Garino (ITIC & Universidad Nacional de Cuyo, Argentina)

Roxana Giandini (LIFIA, UNLP, Argentina)

Silvio Gonnet (Conicet, INGAR, UTN Santa Fe, Argentina)

María Paula González (Conicet, UNS, Argentina)

Silvia Gordillo (LIFIA, Facultad de Informática, UNLP, Argentina)

Nicolás Kicillof (FCEyN, UBA, Argentina)

Raúl Martínez (RMyA)

Pablo Michelis (Intel)

Germán Montejano (UNSL, Argentina)

Luis Olsina (UNLa Pampa, Argentina)

Ernesto Pimentel (Universidad de Málaga, España)

Daniel Riesco (UNSL, Argentina)

Contributions to ASSE

Proceedings of ASSE 2010
Argentine Symposium on Software Engineering

ISSN: 1850-2792

Sesión 1: Experience reports

Sesión 2: Medición y Evaluación

Sesión 3: Procesos de software

Sesión 4: Pruebas

Modelado y Derivación Automática de Código para Pruebas de Software Fernando Palacios, Claudia Pons LIFIA, Facultad de Informática, Universidad Nacional de La Plata. Buenos Aires, Argentin. (pdf) (p. 421-435)

Experience Report: Implementación de Servidor stub para la ejecución de pruebas unitarias en aplicaciones cliente basadas en XMPP Pablo Szyrko, Pablo Perotti. Nimbuzz Argentina. Córdoba, Argentina. (pdf) (p. 436-445)

Sesión 5: desarrollo de software

Un Marco de Software para Planificación Comportamental en Sistemas de Tiempo Real Leo Ordinez, David Donari, Rodrigo Santos, and Javier Orozco. Instituto de Investigaciones en Ingeniería Eléctrica. Universidad Nacional del Sur - CONICET- Bahía Blanca - Argentina. (pdf) (p. 446-460)

Compound and Non Homogeneous Poisson Software Reliability Models. Nestor R. Barraza. Facultad de Ingeniería. Universidad de Buenos Aires. Buenos Aires. Argentina. (pdf) (p. 461-472)

Automatización del Diseño de Bases de Datos Objeto-Relacionales basada en MDA y XML. María Fernanda Golobisky y Aldo Vecchietti. INGAR - Instituto de Desarrollo y Diseño – CONICET – UTN Santa Fe, Argentina. (pdf) (p. 473-487)

JOnE: JEE Online Development Environment Programacion en la Nube. Pablo Larrimbe , Pablo Salgado , Vanesa Maiorana. UADE - Universidad Argentina de la Empresa Facultad de Ingeniería y Ciencias Exactas, Ciudad Autónoma de Buenos Aires, Argentina. (pdf) (p. 488-499)

Short Paper: A Catalog and Classification of Fortan Refactorings. Mariano Mendez, Jeffrey Overbey, Alejandra Garrido, Fernando G. Tinetti and Ralph Johnson. Fac. de Informatica, Universidad Nacional de La Plata,, Buenos Aires, Argentina. Dept. of Computer Science, University of Illinois at Urbana-Champaign, Illinois, USA. (pdf) (p. 500-505)

Sesión 6: Arquitecturas

Reutilización de Soluciones Arquitectónicas empleando Razonamiento Basado en Casos. María Celeste Carignano , Silvio Gonnet, Horacio Leone. INGAR, Instituto de Desarrollo y Diseño. CIDISI, Universidad Tecnológica Nacional – Facultad Regional Santa Fe. CONICET, Comisión Nacional de Investigaciones Científicas y Técnicas. (pdf) (p. 506-520)

Mining Architectural Responsibilities and Components from Textual Specifications Written in Natural Language Agustin Casamayor, Daniela Godoy, Marcelo Campo. ISISTAN Research Institute, UNICEN University. Tandil, Bs. As., Argentina. CONICET, National Council for Scientific and Technical Research. (pdf) (p. 521-534)

El Papel de las Tecnologías del Acuerdo en la Definición de Arquitecturas de Software Adaptativas. J. Santiago Pérez, Carlos E. Cuesta and Sascha Ossowski. Centro para las Tecnologías Inteligentes de la Información y sus Aplicaciones (CETINIA), Kybele Grupo de Investigación, Depto. de Lenguajes y Sistemas Informáticos II. Universidad Rey Juan Carlos, Madrid, España. (pdf) (p. 535-549)

Application of Semantic Web Technologies in Requirements Engineering. Verónica Castañeda1, Luciana Ballejos, Ma. Laura Caliusco, and Ma. Rosa Galli. CIDISI Research Center, UTN-FRSF, Santa Fe, Argentina. (pdf) (p. 550-561)

Short paper: Análisis de Modelos de Variabilidad Ortogonal empleando Redes de Petri. Omar Cristian Martinez, Silvio Gonnet, Horacio Leone. INGAR, Instituto de Desarrollo y Diseño (UTN – CONICET). CIDISI, Universidad Tecnológica Nacional – Facultad Regional Santa Fe. (pdf) (p. 562-567)

Sesión 7 : Aspectos

Supporting Aspect Oriented Requirements Engineering for Large Documents. Arturo Zambrano, Julian Rousselot, Johan Fabry, and Silvia Gordillo. LIFIA, Facultad de Informática. Universidad Nacional de La Plata, La Plata, Argentina. Facultad de Ingeniería y Ciencias Exactas,Universidad Argentina de la Empresa, Ciudad Autónoma de Buenos Aires, Argentina. PLEIAD Lab, Computer Science Department (DCC) University of Chile, Santiago, Chile. (pdf) (p. 568-577)

Automatización de un Proceso de Refactorización para la Separación de Concerns.Santiago A. Vidal, Claudia A. Marcos. ISISTAN Facultad de Ciencias Exactas, UNICEN.Tandil, Buenos Aires, Argentina. CONICET (Consejo Nacional de Investigaciones Científicas y Técnicas), Argentina. (pdf) (p. 578-592)

Short paper: Exploring visual scenarios as an aspect-oriented modeling language. Fernando Asteasuain and Víctor Braberman. Facultad de Ciencias Exactas y Naturales - Universidad de Buenos Aires. (pdf) (p. 593-598)

Short paper: Integración de Reglas de Negocio con Conectores Aspectuales Spring. Graciela Vidal, Juan G. Enriquez y Sandra Casas. Universidad Nacional de la Patagonia Austal, Unidad Académica Río Gallegos - Argentina. (pdf) (p. 599 -609)

Sesión 8. Bof de Aspectos

Community Master: Danilo Zecchin - dzecchin@lifia.info.unlp.edu.ar

Diego Rubio (Motorola and UTN-FRC)

Alvaro Ruiz de Mendarzoqueta (Motorola and UTN-FRC)

Alvaro Soria (ISISTAN, UNCPBA, Argentina)

Sebastián Uchitel (FCEyN, UBA, Argentina)

Antonio Vallecillo (Universidad de Málaga, España)

Alejandro Zunino (ISISTAN, UNCPBA, Argentina)

Addicional reviewers

Pedro Campos y John W. Castro. Escuela Politécnica Superior, Universidad Autónoma de Madrid,

Carlos Gerardo Neil, Universidad Abierta Interamericana (UAI),

María Luciana Roldán, INGAR-CIDISI (UTN - FRSF),

María Celeste Carignano, INGAR-CIDISI (UTN - FRSF),

Agustina Buccella, GIISCo, UNCo,

Gabriela Aranda, GIISCo, UNCo,

Silvia Amaro, UNCo,

Patricia Bazán (LINTI, Facultad de Informática, UNLP, Argentina),

Leandro Antonelli (LIFIA, Facultad de Informática, UNLP, Argentina),

Eduardo Rodríguez. Practia Consulting España. Pragma Consultores.

Luis Roqué y Carlos Salgado, Universidad Nacional de San Luis.

Guillermo Covella y Maria de los A. Martin, Facultad de Ingeniería, UNLPam.

Community Master: Danilo Zecchin - dzecchin@lifia.info.unlp.edu.ar

CQA-Meth: una Metodología para la Evaluación de la Calidad de los Modelos Software

Moisés Rodríguez[1], Marcela Genero[2], Damiano Torre[2], Belen Blasco[2] y Mario Piattini[2]

[1] Alarcos Quality Center S.L. (AQC)
Ciudad Real, España
moises.rodriguez@alarcosqualitycenter.com

[2] Grupo de Investigación ALARCOS, Universidad de Castilla-La Mancha
Ciudad Real, España
(Damiano.Torre, Belen.Blasco)@alu.uclm.es (Marcela.Genero, Mario.Piattini)@uclm.es

Resumen. Las empresas se preocupan cada vez más de la calidad de sus productos software y reconocen que las técnicas de evaluación y aseguramiento de la calidad deberían aplicarse desde las primeras etapas del proceso de desarrollo. Esto se debe a que el foco de atención de la calidad ha pasado del código a los modelos. La calidad de los modelos ha ido ganando cada vez más relevancia desde que apareció el paradigma del Desarrollo Dirigido por Modelos (MDD). Aunque existen algunas metodologías para la evaluación de la calidad de los modelos software, todas son propuestas aisladas que se centran en un modelo en concreto y sólo proponene técnicas de evaluación específicas para ese modelo. No hay ninguna metodología genérica y flexible que permita la evaluación de la calidad de cualquier tipo de modelo software, y mucho menos una herramienta que la soporte. Para abordar este problema, en este artículo proponemos un entorno integrado llamado Entorno CQA (Continuous Quality Assessment), que consiste en una metodología, CQA-Meth, y la herramienta que la implementa, CQA-Tool. CQA-Tool se complementa con un conjunto de herramientas para evaluar la calidad de los distintos modelos UML. Además, CQA-Tool permite construir un catálogo de técnicas de evaluación que integre todas las técnicas de evaluación disponibles (por ejemplo, medidas, listas de comprobación, guías, convenciones de modelado, etc.) para cada modelo UML. El Entorno CQA es apropiado para: (i) empresas que ofrecen servicios de evaluación de la calidad del software, (ii) factorías de software que quieran evaluar sus propios desarrollos, (iii) organizaciones que hayan externalizado una parte o la totalidad de sus desarrollos software y quieran obtener una evaluación independiente de la calidad de los productos software que adquieren.

Keywords: calidad del software, evaluación de la calidad, modelos UML, metodología, herramienta.

1 Introducción

Las empresas ponen cada vez más y más atención en la calidad de sus productos software, centrándose no sólo en la calidad del producto final, sino también considerando la calidad como una parte integral desde el principio del ciclo de vida del desarrollo de software.

Esta importancia se ve en el boom de las certificaciones basadas en modelos, como CMMI (Capability Maturity Model Integration) [1], ISO 15504 [2], ISO 90003 [3], etc., donde se destacan las actividades de evaluación de la calidad de productos software dentro de las áreas clave de la madurez de una organización que desarrolla o mantiene software.

La aparición del Desarrollo Dirigido por Modelos (MDD) ha contribuido considerablemente a cambiar el foco de atención de las actividades de evaluación de la calidad desde el código hacia los modelos. En particular, la propuesta de la Arquitectura Dirigida por Modelos (MDA) [4] fomenta el uso de modelos a lo largo de todo el proceso de desarrollo. El código que implementa el producto de software final se obtiene a partir de sucesivas transformaciones entre modelos. Con este enfoque los modelos no son meras formas de describir el software, sino que son la piedra angular de su desarrollo. La calidad de los modelos es por tanto de gran importancia, ya que determinará la calidad de los productos software que finalmente se implementen. Dentro de MDD el Lenguaje Unificado de Modelado (UML) [5] está ganando aceptación en la industria como "EL" lenguaje de modelado estándar para expresar modelos software.

Aunque existen numerosas técnicas (listas de comprobación, heurísticas, métricas, etc.), estándares y herramientas para evaluar la calidad de modelos UML, todas ellas son propuestas aisladas que se centran en modelos específicos y que en genral aplican un sólo tipo de técnica, o bien métricas, o listas de comprobación, etc. Tampoco existe una metodología genérica y flexible que permita la evaluación continua de la calidad de cualquier modelo UML y menos aún una herramienta que la automatice.

Cómo se refleja en la revisión sistemática de la literatura presentada en [6], que considera las propuestas existentes en la literatura desde 1998 hasta 2009 en "calidad de modelos UML", este tema es de actualidad y relevancia debido a que todavía no tiene un nivel de madurez suficiente, indicado por la falta de evidencia empírica de la mayoría de las propuestas existentes.

Hemos analizado las siguientes propuestas: EMISQ [7, 8], IEEE 1012-1998 [9], IEEE 1028-2008 [10], IEEE Std 1061-1998 [11], CMMI (Capability Maturity Model Integration) [12], ISO/IEC 9126 [13], ISO/IEC 12207 [14], ISO/IEC 14598 [15], ISO/IEC 15504 [16], ISO/IEC 25000 [17].

Una vez analizadas todas estas propuestas y con el fin de paliar los problemas expuestos anteriormente, propusimos una solución integral, el Entorno CQA que consiste en:

1) Una metodología (CQA-Meth) para la evaluación continua de la calidad de los modelos software (en concreto en este artículo la aplicamos a modelos UML).
2) Un conjunto de herramientas que automatizan dicha metodología, que está compuesta por una herramienta horizontal (CQA-Tool) que soporta la propia metodología, junto con varias herramientas específicas para la evaluación de diferentes modelos UML, denominadas herramientas verticales, como CD-Tool

que evalua la calidad de diagramas de clases UML. También será posible conectar otras herramientas existentes a la herramienta genérica. Además, CQA-Tool permite construir un catálogo de técnicas de evaluación que integra todas las técnicas disponibles para cada modelo UML (por ejemplo, métricas, listas de comprobación, convenciones de modelado, guías, etc.).

El Entorno CQA ofrece la posibilidad de ser utilizado por empresas dedicadas a ofrecer servicios de evaluación de la calidad para fábricas de software, así como a los clientes que hayan externalizado la construcción de sus productos software, permitiendo de este modo la obtención de una evaluación de la calidad independiente a la que realice el fabricante. Las empresas de desarrollo de software también pueden utilizar la metodología para llevar a cabo evaluaciones de sus propios productos software.

Este artículo está organizado del siguiente modo: en la sección 2 se presenta una introducción a CQA-Meth, cuyos procesos se detallan en la sección 3. Se puede ver un ejemplo de aplicación de CQA-Meth y CQA-Tool en la sección 4. Finalmente, en la sección 5 se presentan las conclusiones y las líneas de trabajo futuro.

2 Características y descripción de la metodología

El principal objetivo de CQA-Meth es definir un marco de trabajo que permita determinar los procesos necesarios para llevar a cabo la evaluación de los modelos UML, así como facilitar la comunicación entre el cliente (patrocinador de la evaluación) y el equipo de evaluación.

Las principales características de CQA-Meth, se pueden resumir en los siguientes principios básicos:

- Está formada por un conjunto estructurado de procesos.
- Está orientada a la relación con el cliente y a la externalización de la evaluación de calidad.
- Está pensada para ser una metodología fácilmente adaptable.
- Está soportada por un conjunto de técnicas y herramientas.

CQA-Meth proporciona un marco de trabajo donde se identifica claramente el qué, cuándo, y el quién, de cada una de las fases y actividades de los procesos, así como la secuencia de pasos que se deben seguir a la hora de llevar a cabo la evaluación.

Además, está orientada a la relación con el cliente de manera que, en distintos puntos del proceso de evaluación, el cliente se encuentra involucrado en la toma de decisiones. Y por otro lado, la metodología contempla el modo de trabajo externalizado, de manera que para realizar la evaluación no sea necesario encontrarse en las instalaciones del cliente, sino que se pueda planificar, diseñar y realizar la evaluación externamente, poniéndose en contacto con el cliente en los momentos puntuales en los que sea necesario.

De igual manera, la metodología de evaluación está pensada para poder ser adaptada a las distintas necesidades del cliente, construyendo los catálogos de técnicas de evaluación apropiados para cada producto software, Estos catálogos almacenarán la severidad con la que se desea realizar la evaluación (alta, media o baja), las técnicas de evaluación (listas de comprobación, métricas, etc.) y las herramientas de evaluación que las automatizan.

CQA-Meth ha sido implementada con SPEM [27] y EPFC [28]. Por último, CQA-Meth se ha soportado por un conjunto de técnicas de evaluación y herramientas que agilizan la evolución de las fases de la metodología y permiten diseñar y realizar la evaluación de manera semiautomática.

2.1 Catálogo de roles

A continuación se detallan los roles y agrupamiento de estos que se identifican a lo largo del proceso de evaluación, así como las principales tareas y responsabilidades de cada uno de ellos.

En la figura 1 se pueden observar los roles involucrados en el proceso de evaluación según CQA-Meth, así como la relación que existe entre ellos. Tal y como se aprecia en la figura 1, la comunicación entre las empresas (cliente y evaluadora) es realizada a través del patrocinador y del evaluador jefe. Éstos a su vez serán los que se comuniquen directamente con sus equipos internos.

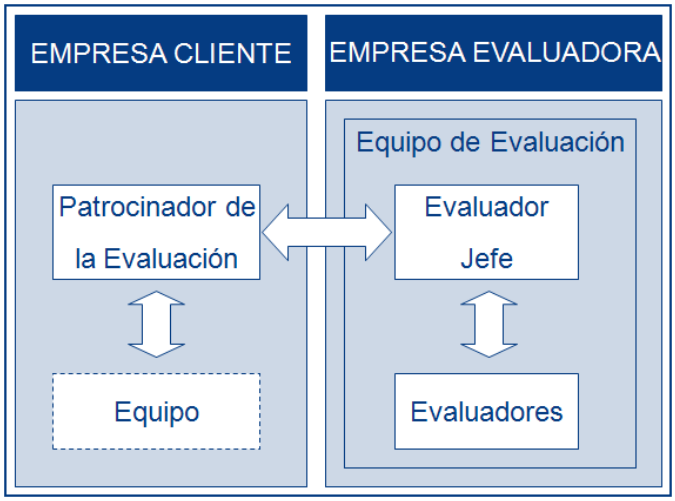


Fig. 1. Roles y relaciones en el proceso de evaluación de CQA-Meth.

A continuacón se detallan cada uno de los roles presentados en la figura 1:

- **Empresa cliente**. Representa a la organización que ha expresado su intención y/o necesidad de contratar los servicios de evaluación de la calidad de los modelos UML (bien desarrollados por ella o por otra empresa).
- **Patrocinador de la evaluación**. El patrocinador de la evaluación es el representante de la empresa cliente que se pone en contacto con la empresa evaluadora para expresar la necesidad de evaluación de la calidad de sus modelos UML (normalmente suele ser el jefe de proyecto o el responsable del departamento de calidad). En algunos casos, el patrocinador de la evaluación podrá disponer de un equipo de trabajo interno formado por personal propio de la empresa (ver figura 1, rectángulo punteado).

- **Equipo de la empresa cliente**. Sirve de apoyo al patrocinador de la evaluación. Su principal responsabilidad consiste en analizar los resultados reportados por la empresa evaluadora para detectar defectos o inconsistencias. La existencia de este equipo no es necesaria siempre, depende de la envergadura del proyecto a evaluar.
- **Empresa evaluadora**. Representa la organización que ha sido contratada para llevar a cabo el proceso de evaluación. La empresa evaluadora necesitará disponer de un evaluador jefe y uno o varios evaluadores que formaran el equipo de trabajo que llevará a cabo el proceso de evaluación.
- **Evaluador jefe**. Es el representante de la empresa evaluadora y líder del equipo de evaluación, cuyo objetivo es asegurar el correcto desarrollo del proceso de evaluación. Este rol debe ser ocupado por un profesional con conocimientos en el análisis y diseño con UML y con experiencia en la práctica de las principales normas y estándares sobre evaluación del producto y procesos software.
- **Evaluador**. Está encargado de realizar las actividades y tareas propias del proceso de evaluación. Este rol deberá ser ocupado por una persona con conocimientos de modelado UML y de la metodología de evaluación. Las principales responsabilidades son, realizar las actividades de evaluación asignadas por el evaluador jefe y recoger información del propio proceso de evaluación.
- **Equipo de evaluación**. Representa el equipo de la empresa evaluadora, formado por el evaluador jefe y un conjunto de uno o varios evaluadores de calidad (según los requisitos del proyecto). Su principal objetivo es cumplir con los requisitos de evaluación acordados con el patrocinador dentro de los tiempos planificados.

2.2 Catálogo de elementos

A continuación se detallan todos los elementos (documentos, informes, catálogos, modelos, etc.) que se identifican a lo largo de las actividades de evaluación. Estos elementos podrán ser de entrada y/o salida para las actividades del proceso de evaluación, siendo además algunos de ellos entregables finales del proyecto de evaluación.

- **Información de contacto**. Documento donde la empresa evaluadora recoge los datos de contacto de la empresa cliente.
- **Contrato de evaluación**. Este documento representa el primer entregable (documento de salida, se entrega a la empresa cliente) del proyecto de evaluación y será generado como resultado de la fase 1 (planificación). En él se recogen las condiciones bajo las que se realizará el proyecto de evaluación, es de carácter público para la empresa evaluadora y para la empresa cliente.
- **Modelo de calidad genérico**. Es un documento propiedad de la empresa evaluadora y contiene el conjunto de características, subcaracterísticas y atributos de calidad que se pueden evaluar para cada modelo software. Como se ha indicado anteriormente, estas características y subcaracterísticas de calidad se encuentran detalladas en la ISO 25010 [18].

- **Modelo de calidad específico**. Es un documento que solo contiene el conjunto de características, subcaracterísticas y atributos de calidad que la empresa cliente ha seleccionado para el proyecto de evaluación. Este documento es de carácter público para la empresa evaluadora y la empresa cliente, está integrado dentro de la especificación de la evaluación que representa el tercer entregable del proceso de evaluación.
- **Catálogo de herramientas de evaluación**. Es un documento propiedad de la empresa evaluadora y que permite identificar las herramientas necesarias y su modo de utilización, en función de las técnicas de evaluación que se vayan a utilizar para evaluar los modelos UML.
- **Plan de evaluación**. Este documento público para ambas partes, y representa el segundo entregable del proyecto de evaluación. Una primera versión de este documento se desarrolla previamente al inicio del proyecto de evaluación, incluida en la propuesta técnica del contrato de evaluación. Esta primera versión es refinada por ambas partes tras la aceptación del contrato, generándose así la versión definitiva del plan de evaluación, que guiará todos los procesos y actividades realizados durante el proyecto. Este documento es generado como resultado de la fase 1 (planificación) del proceso de evaluación, sin embargo estará sujeto a cambios durante todo el proyecto de evaluación y en él se deberán registrar los retrasos y re-planificaciones.
- **Conjunto de modelos a evaluar**. Entendiendo por modelos los modelos UML que la empresa cliente quiere validar, así como toda la documentación que sea necesaria para poder comprender correctamente la naturaleza y contenido de dichos modelos. Entre otras, es responsabilidad del patrocinador que los modelos a evaluar estén disponibles en las fechas planificadas. Y responsabilidad del evaluador jefe asegurar que los modelos entregados para ser evaluados cumplen con los requisitos impuestos por las técnicas y herramientas que se van a utilizar en el proceso de evaluación.
- **Especificación de la evaluación**. Este modelo representa el tercer entregable del proyecto de evaluación y como tal es público para ambas empresas. La especificación de la evaluación será generada como resultado de la fase 2 (especificación) del proceso de evaluación y estará formado por uno o varios documentos donde se recoja, una vez analizado el conjunto de modelos a analizar, el modelo de calidad específico que se va a analizar, qué técnicas se van a utilizar, qué métricas se esperan obtener, qué herramientas se van a manejar, etc. En definitiva un conjunto de características a evaluar por cada uno de los modelos UML, las técnicas y herramientas que se utilizarán y el listado de métricas e indicadores que se esperan obtener. Esta especificación posteriormente será incluida como apartado dentro del informe final de evaluación, conforme lo establece la norma ISO/IEC 14598 [19].
- **Informe de evaluación**. Este artefacto representa el cuarto entregable del proyecto de evaluación y como tal es público para ambas empresas. El informe de evaluación será generado como resultado de la fase 4 (conclusión) y estará formado por un documento donde se recogen, todos los resultados obtenidos mediante las técnicas de evaluación, las conclusiones obtenidas del análisis de dichos resultados y un conjunto de recomendaciones y acciones futuras para la mejora de los modelos UML, además de la especificación de la evaluación. El

informe va acompañado de una presentación de los resultados que va dirigida tanto al patrocinador de la evaluación como a los directivos de la empresa cliente y al equipo interno si lo hubiese. Una característica importante del informe es la retroalimentación que se produce posteriormente a la presentación de los resultados y en cuyo caso la empresa cliente reporta a la empresa evaluadora mediante el documento petición de modificación.

- **Petición de modificación**. Este artefacto representa el quinto entregable del proceso de evaluación y como tal es público para ambas empresas. Este documento podrá ser generado como resultado de la fase 4 (conclusión) y representa el documento mediante el cual la empresa cliente, manifiesta sus opiniones y posibles cambios respecto a los resultados presentados en el informe de evaluación. La empresa evaluadora recibirá el documento que evaluará, revisará y actualizará el informe de evaluación de acuerdo a los cambios sugeridos. El informe modificado será de nuevo entregado a la empresa cliente quien podrá solicitar una nueva reunión o reportar una nueva petición de modificación, actuándose de la misma manera que la primera vez. El resultado de esta petición no es más que un refinamiento del informe de evaluación así como la mejora del proceso de evaluación.
- **Información interna de evaluación**. Además de los propios entregables y resultados de la evaluación, durante la evaluación se genera un conjunto de documentación relacionada con el propio proceso. Esta información principalmente son informes del proceso de gestión y monitorización, informes de métricas de la propia evaluación (progreso, retrasos, costes/beneficios, etc.), informes de cambios y ajustes realizados durante la evaluación, informes de defectos y adaptaciones realizadas en la infraestructura de evaluación, etc. Posteriormente esta información será utilizada por los procesos **Gestión de la Evaluación** y **Gestión de la Infraestructura** de la presente metodología, para realizar una mejora continua del proceso de evaluación.

3 Procesos de la metodología de evaluación

La metodología de evaluación CQA se encuentra compuesta por tres procesos principales (ver figura 2).

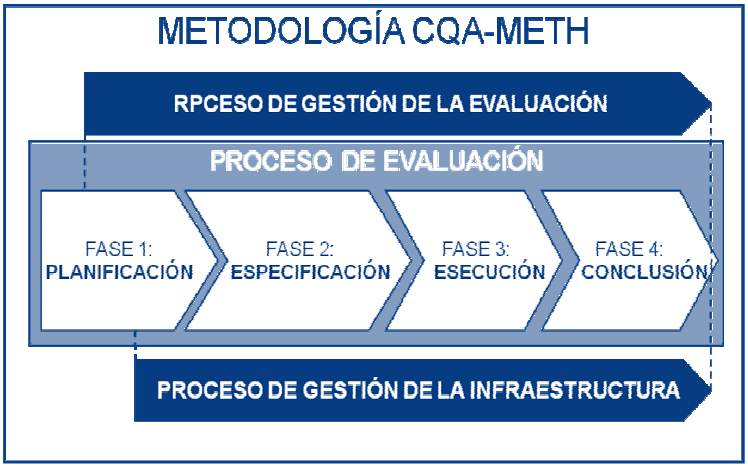


Fig. 2. Procesos y fases de la metodología de evaluación CQA-Meth.

3.1 Proceso de evaluación

Este proceso representa el esqueleto de la metodología de evaluación ya que contiene las fases de planificación, especificación, ejecución y conclusión del proyecto de evaluación. La ejecución de este proceso está directamente relacionada con los dos primeros objetivos de CQA-Meth.

El proceso de evaluación se descompone en cuatro fases:

- **Fase 1. Planificación:** el objetivo de esta fase es la elaboración del contrato y la obtención de un plan de evaluación de los modelos UML. Esta fase está formada por cuatro actividades: contratación, arranque del proyecto, planificación detallada y consolidación del plan.
- **Fase 2. Especificación:** el objetivo de esta fase es definir el alcance de la evaluación, determinando el conjunto de características que se van a evaluar para cada uno de los modelos UML (artefactos), las técnicas y herramientas que se van a utilizar para realizar la evaluación, los niveles de evaluación (bajo, medio o alto) que se van a aplicar a cada modelo y el listado de indicadores y métricas que se esperan obtener. Esta fase está formada por cuatro actividades: obtención y análisis de modelos a evaluar, selección del modelo de calidad y las técnicas, planificación interna de la evaluación y verificación de la especificación..
- **Fase 3. Ejecución:** el objetivo de la fase de ejecución es la aplicación de las técnicas de evaluación y el lanzamiento de las herramientas (utilizando como entrada los artefactos a evaluar) para obtener los resultados iniciales (métricas de más bajo nivel) sobre la calidad de los modelos UML. Una vez obtenidos los resultados iniciales, es necesario asegurar su veracidad (pueden existir defectos en los resultados, como los conocidos falsos positivos) y realizar su almacenamiento de una manera unificada que permita su posterior explotación. Esta fase está formada por tres actividades: aplicación de técnicas de evaluación, análisis de la ejecución y unificación de resultados.
- **Fase 4. Conclusión:** el objetivo de la fase de conclusión es la elaboración del informe de evaluación y la presentación de los resultados tanto al patrocinador como al resto de personas involucradas dentro de la empresa cliente. Esta fase

está formada por tres actividades: elaboración del informe de evaluación, presentación de resultados, corrección del informe y finalización de la evaluación.

3.2 Proceso de gestión de la evaluación

Es el proceso de soporte al proceso de evaluación, que permite controlar, evaluar y mejorar el propio proceso de evaluación. La ejecución de este proceso está directamente relacionada con el segundo y tercer objetivo de CQA-Meth.

Este proceso está formado únicamente por una fase, denominada de igual manera que el proceso. Esta fase a su vez se descompone en tres actividades bien diferenciadas: monitorización, documentación y ajuste y evolución.

3.3 Proceso de gestión de la infraestructura

Es el proceso de soporte al proceso de evaluación, que permite gestionar todo lo relacionado con la infraestructura necesaria para llevar a cabo el proceso de evaluación (técnicas y herramientas de evaluación). La ejecución de este proceso está directamente relacionada con el tercer y cuarto objetivo de CQA-Meth.

Este proceso está formado únicamente por una fase, denominada de igual manera que el proceso. Esta fase a su vez se descompone en tres actividades[2] bien diferenciadas: especificación de la infraestructura, mantenimiento de la infraestructura y adaptación y transferencia de la infraestructura.

4 Un ejemplo de aplicación

Hemos resaltado previamente que empezamos utilizando el Entorno CQA para evaluar la calidad de los modelos UML, concretamente de los diagramas de clases UML, dada la gran relevancia que dichos diagramas tienen en el desarrollo de software..

Como hemos comentado anteriormente, CQA-Meth se ha automatizado con la herramienta CQA-Tool, que permite evaluar la calidad de modelos UML a través de un conjunto de herramientas específicas. Concretamente hasta el momento hemos construido la herramienta CD-Tool para evaluar la calidad de los diagramas de clases UML. Las restricciones del espacio nos limitan a poder describir un ejemplo real y completo de uso de CQA-Meth. Para ilustrar el uso de CQA-Meth a través del uso de la herramienta CQA-Tool , mostraremos el resultado final, que es la publicación de un informa final que llamamos “Informe de Evaluación”. Este informe, conforme a la plantilla dada por la metodología, contiene los siguientes apartados: datos de identificación, requisitos de la evaluación, especificaciones de la evaluación, métodos de evaluación y resultados de la evaluación.

Evaluaremos la calidad del diagrama de clases mostrado en la figura 3, usando CD-Tool [20]. Esta herramienta provee a CQA-Tool la siguiente información “Epecificaciones de la Evaluación” y “Resultados de la Evaluacion”. La herramienta CD-Tool tiene un catálogo de técnicas para diagramas de clases UML que consiste en 3 listas de comprobación (con 40 items en total) 70 métricas. Estan técnicas permiten evaluar la calidad sintáctica, pragmática y semántica [21]. En la tabla 2, se muestra el

apartado “Especificaciones de la Evaluación” contenida dentro del “Informe de Evaluación”, generado por CQA-Tool. La primera columna representa el número de item dentro de cada lista de comprobación, la segunda columna describe el defecto que dicho item detecta, la tercera columna describe si se calcula automática o manualmente, la cuarta columna indica la referencia bibliográfica donde dicho item fue propuesto y también las referencias que muestran resultados basados en evidencia empírica de su utilidad, y la útima columna muestra el resultado de la aplicación de dicho item al ejemplo de la figura 3.

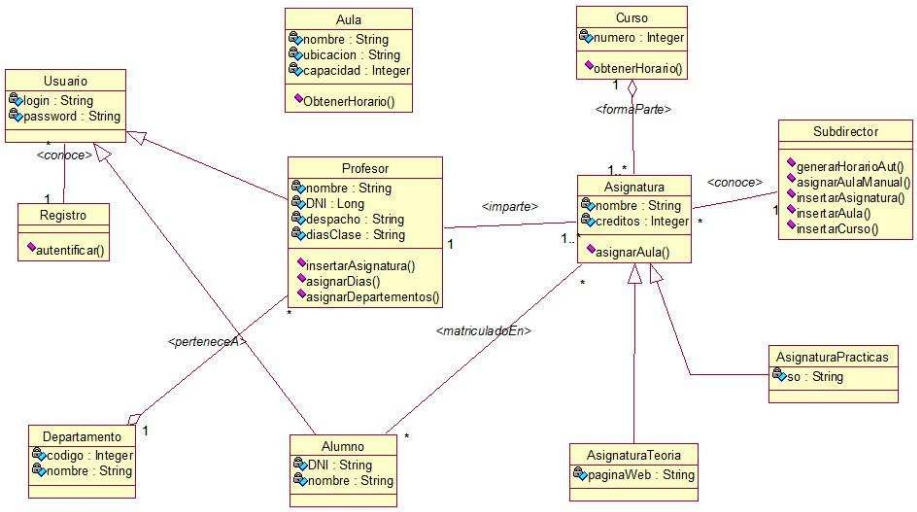


Fig. 3. Ejemplo de un diagrama de clases UML.

Tabla 1. Ejemplo de las listas de comprobación sintactica, semántica y pragmática (sólo se muestran algunos ítems).

Código	Descripción	A/M	Referencia	Resultado
Calidad sintáctica				
I.1	¿Tienen las clases un nombre único?	A	[22]	SI
I.2	¿Estan los nombres de las clases abstractas puestos en cursiva?	M	[23]	NO APLICABLE (No hay clases abstractas en el diagrama de clases)
I.3	¿Si es necesaria mas de una palabra para el nombre de la clase, están estas palabras combinadas?	M	[23]	SI
I.4	¿Estan las asociaciones especificadas mediante una línea recta?	M	[23]	NO
I.5	¿Esta la multiplicidad representada sólo para las asociaciones y las agregaciones?	M	[23]	SI

I.6	¿Empiezan los nombres de las clases con letra mayúscula?	A	[23]	SI
Calidad semántica				
I.31	¿Usan los nombres de las clases una terminología común?	M	[24]	SI
I.32	¿Represnta el significado de la clase un solo concepto lógico?	M	[23]	SI
I.33	¿Son atributos sustantivos basados en el dominio?	M	[24]	SI
I.34	¿Es el significado de las operaciones reflejado en su nombre y formato?	M	[23]	SI
Calidad pragmática				
I.35	¿El diagrama no contiene lineas cruzadas?	M	[22]	NO
I.36	¿Las clases no tienen referencias circulares?	M	[25]	SI
I.37	¿Los paquetes no tienen dependencias circulares a otros paquetes?	M	[26]	NO APLICABLE (No hay paquetes en el diagrama de clases.
I.38	¿Tiene cada clase como máximo 60 entre atributos y operaciones?	A	[23]	SI
I.39	¿Varia el número de métodos entre 10 y 21?	A	[23]	SI
I.40	¿La profundidad de la herencia no es más de tres?	M	[23]	SI

Del catálogo completo de las listas de comprobación incorporadas dentro de CD-Tool y del diagrama de la figura 3, se ha realizado una evaluación, teniendo en cuenta tanto las comprobaciones automáticas como las manuales. Los resultados generados por CD-Tool se almacenan en un fichero de XML, usado posteriormente como entrada a CQA-Tool, que realiza la tarea de unificar los resultados con los de otras herramientas para generar el informe final.

En el caso de nuestro ejemplo, los resultados de la evaluación se incluyen en el "Informe de Evaluación", en el apartado "Resultados de la Evaluación", que contiene la información mostrada en la figura 4. Como se puede observar, en primer lugar se presenta una tabla donde se resumen los resultados, expresados en porcentajes, para cada uno de los tipos de calidad (sintáctica, semántica y pragmática) evaluados.

En los resultados hemos sumado los "NO APLICABLE" junto con los "SÍ", porque creemos que no son defectos del diagrama de clases evaluado.

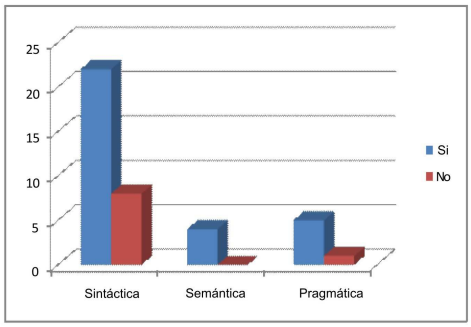
A continuación se presentan dos gráficos, un diagrama de barras donde se muestra la diferencia entre el número de ítems en cada lista de comprobacion evaluados positivamente y negativamente y un diagrama de Kiviat que resume la valoración final por tipo de calidad.

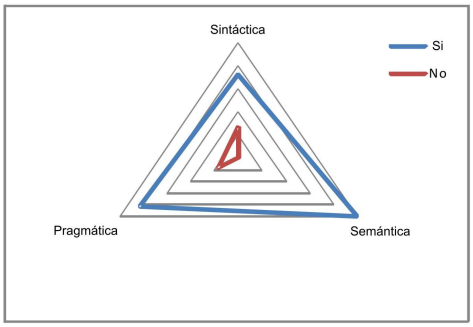
Finalmente, en este informe se presentan los porcentajes totales por tipo de calidad.

Resultados de la evaluación

Tipo de calidad	Resultados	
	Si	No
Calidad sintáctica	22 (72%)	8 (28%)
Calidad semántica	4 (100%)	0 (0%)
Calidad pragmática	5 (80%)	1 (20%)

Informe estadístico





RESULTADOS DE LA EVALUACIÓN DE LA CALIDAD DEL DIAGRAMA DE LA CLASES UML: **31 SI (77%) Y 9 NO (23%).**

Fig. 4. Ejemplo del apartado "Resultados de la Evaluacion" contenido en el "Informe de Evaluación".

5 Conclusiones y trabajos futuros

La contribución principal de este artículo es el Entorno CQA, que consiste en un componente metodológico, CQA-Meth, y un componente tecnológico, CQA-Tool.

Las ventajas principales de este entorno son:

- No se limita a la evaluación de calidad, sino también apoya el proceso de la gestión de configuración y algunos aspectos del planeamiento de proceso y también apoya la gestión de la infraestructura.
- Flexibilidad en la realización de las evaluaciones, permite que los usuarios elijan entre diversos modelos de la calidad y da la posibilidad de definir sus propios modelos. De esta manera es posible obtener resultados de la evaluación según un proyecto específico que se centre en características de calidad particulares.

- Se podría aplicar a cualquier artefacto software. Hasta ahora, lo hemos aplicado a la evaluación de calidad de los modelos UML, ya que son los primeros artefactos construidos en el ciclo de vida de un producto software. También hemos construido un catálogo de técnicas para la evaluación de los diagramas de clases UML con una herramienta. Sin embargo, podría fácilmente aplicarse a cualquier artefacto software, añadiendo nuevas herramientas verticales con sus respectivos catálogos para los diversos tipos de artefactos que querramos evaluar (código fuente, casos de prueba, etc.).

En los estudios futuros pretendemos extender el área de la evaluación de calidad alcanzada con CQA-Meth y las herramientas construidas hasta el momento, para permitir que evaluemos toda la gama de artefactos software generados desde las etapas iniciales del ciclo de vida hasta la entrega y la instalación del producto final (análisis, diseño, codificación, pruebas, etc.). De esta manera el nuevo entorno permitirá percibir cualquier anormalidad en el curso del proceso de desarrollo, sin tener que esperar a las pruebas finales.

Los cambios que apuntamos realizar en la metodología serán implementados en las respectivas herramientas. Además, una mejora importante será la adaptación de CQA-Tool para poder conectarla con herramientas existentes.

En el futuro inmediato aplicaremos el entorno CQA a proyectos reales realizados en fábricas del software instaladas en la región de Castilla-La Mancha en España.

Agradecimientos

Esta investigación es parte de los proyectos financiados por el "Ministerio de Ciencia e Innovación": EECCOO (TRA2009_0074), PEGASO (TIN2009-13718-C02-01) y EVVE (PTQ-08-03-07931 co-financiado por Fondo Europeo social), y los proyectos financiados por "Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha": IDONEO (PAC08-0160-6141) y MECCA (PII2I09-0075-8394).

Referencias

1. SEI, CMMI for Development, Version 1.2. Software Engineering Institute. (2006).
2. ISO, ISO/IEC 15504-2: 2003/Cor.1:2004(E). Information technology -Process assessment - Part 2: Performing an assessment. in International Organization for Standardization. Geneva, Switzerland. (2004a).
3. ISO, ISO/IEC 90003, Software and Systems Engineering - Guidelines for the Application of ISO/IEC 9001:2000 to Computer Software. International Standards Organization. Geneva, Switzerland. (2004b).
4. OMG, MDA Guide, from http://www.omg.org/docs/omg/03-06-01.pdf. Version 1.0.1. (2003).
5. OMG, UML The Unified Modeling Language™. (2006).
6. Genero, M., Fernández, A., Nelson, J., Poels, G., Piattini, M.:A Systematic Literature Review on the Quality of UML Models. In Submitted to the Journal of Database Management. (2009).
7. Plösch, R., Gruber, H., Hentschel, A., Körner, C., Pomberger, G., Schiffer, S., Storck, S.: The EMISQ method and It´s Tol Support-Expert Based Evaluation on Intrenal Software

Quality. Journal of Innovations in Systems and Software Engineering, Springer London, 4(1): p. 3-15. (2008).
8. Gruber, H., Kömer, C., Plösch, R., Schiffer, S.:Tool Support for ISO 14598 based code quality assessments. In: Sixth International Conference on the Quality of Information and Communication Technology. (2007).
9. IEEE Std. 1012-1998: Standard for Software Verification and Validation, Institute of Electrical and Electronic Engineers. (1998).
10. IEEE Std 1028-2008 Standard for Software Reviews and Audits, Institute of Electrical and Electronic Engineers. (2008).
11. IEEE Std 1061-1998. IEEE Standard for a Software Quality Metrics Methodology. (1998).
12. Chrissis, M.B., Konrad, M., Shrum, S.:CMMI: Guidelines for Process Integration and Product Improvement: Addison Wesley Professional. (2006).
13. ISO, ISO/IEC DTR 9126 Software Product Evaluation - Quality Characteristics and Guidelines for their Use, International Standards Organization. (2001).
14. ISO, ISO/IEC 12207. International Standard. Information technology - Software life-cycle processes. Amendedment 1, International Standards Organization. (2002).
15. ISO, ISO/IEC 14598: 1999-2001. Information Technology - Software Product Evaluation - Parts 1-6., International Standards Organization. (1999).
16. ISO, ISO/IEC 15504-5:2003, Information technology - Process assessment - Part 5: An exemplar Process Assessment Model, International Standards Organization. (2006).
17. ISO, ISO/IEC FDIS 25001, Software product Quality Requirements and Evaluation (SQuaRE) - Evaluation planning and management. In ISO/IEC FDIS 25001 Software and system engineering. Ginebra, Suiza: International Standards Organization. (2006).
18. ISO, ISO/IEC 25010 Software and system engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Quality model and guide. International Organization for Standarization. : Ginebra, Suiza. (2005).
19. ISO, ISO/IEC 14598: 1999-2001. Information Technology - Software Product Evaluation - Parts 1-6, International Organization for Standarization. (1999).
20. Torre, D., Blasco, B., Genero, M., Piattini, M.:CQA-ENV: An Integrated Environment for the Continuous Quality Assessment of Software Artifacts. In: SoMeT 2009. Prague: IOSPress. (2009).
21. Lindland, O., Sindre, G., Solvberg, A.:Understanding Quality in Conceptual Modelling. in IEEE Software. (1994).
22. Lange, C.F.J.:Assessing and Improving the Quality of Modeling, Technische Universiteit Eindhoven. (2007).
23. Unhelkar, B.:Verification and Validation for Quality of UML 2.0 Models, ed. I. John Wiley & Sons. (2005).
24. Ambler, S.W.:The Elements of UML™ 2.0 Style, ed. A. Modeling: Cambridge University Press. 200. (2005).
25. Wüst, J.:The Software Design Metrics tool for the UML SDMetrics. (2005).
26. Berenbach, B.:The evaluation of large, complex uml analysis and design models. In: ICSE '04: Proceedings of the 26th International Conference on Software Engineering. Washington, DC, USA. (2004).
27. OMG, SPEM Software & Systems Process Engineering Metamodel, v2.0, http://www.omg.org/spec/SPEM/2.0/Beta2/PDF. (2007).
28. EPFC, Eclipse Process Framework Composer, http://www.eclipse.org/epf/. (2004).