EVALUACION DE LA CALIDAD Y SEGURIDAD EN PRODUCTOS SOFTWARE

Daniel Mellado Fernández
Técnico Superior de Informática
Agencia Estatal de Administración Tributaria,
Paseo de la Castellana 108, 28046 Madrid (España)
damefe@esdebian.org

Moisés Rodríguez y Javier Verdugo
Consultores
Alarcos Quality Center S.L.
(moises.rodriguez, javier.verdugo)@alarcosqualitycenter.com

Mario Piattini y Eduardo Fernández-Medina
Catedrático y Profesor Titular,
Universidad de Castilla–La Mancha,
Instituto de Tecnologías y Sistemas de la Información
(Mario.Piattini, Eduardo.FdezMedina)@uclm.es

Resumen: La calidad y seguridad del software es un campo que ha tenido una prolífica actividad investigadora en los últimos años, si bien ha estado principalmente centrada en la calidad de los procesos que se siguen para desarrollar el software. Prueba de ello es la gran cantidad de modelos y estándares de referencia, evaluación y mejora de procesos software que han surgido durante las últimas décadas: ISO 90003, ISO 12207, ISO 15504, CMM, CMMI, IDEAL, SCAMPI, Competisoft, etc. Lo mismo sucede con la seguridad con el reciente auge de la familia ISO 27000 y ya anteriormente con la ISO 21827(SSE-CMM) o con los Criterios Comunes (ISO 15408).
En definitiva, pocas aportaciones se han hecho sobre la calidad y seguridad de los productos software y, aunque el área del testing (sobre todo de funcionalidad) si es un campo bastante trabajado, todavía no se han desarrollado las técnicas necesarias para evaluar de forma efectiva la calidad y la seguridad de un producto software. Con este fin presentamos el entorno MEDUSAS que en el marco de la cooperación público-privada permita ofrecer a las empresas y organismos públicos, un conjunto de servicios de evaluación y control de la calidad del software, de forma independiente y basados en la actual familia de normas ISO 25000, conocida como SQuaRE (Software Quality Requirements and Evaluation).

Palabras clave: Calidad del Software, Seguridad, ISO 25000, Evaluación de la Calidad.

1 Introducción

Las actividades relacionadas con la calidad de software, están cobrando cada vez más importancia debido principalmente a:

i) El crecimiento experimentado por la externalización (*outsourcing*) de software, cuyo mercado a nivel internacional, según estudios de Gartner Group, se espera que alcance los 1,3 billones de dólares en 2008. Hay que señalar además que España se está convirtiendo en uno de los centros de *nearshoring* [2] y *offshoring* [5], preferidos a nivel mundial con una gran cantidad de factorías de software instaladas [10]. Esto hace que, por un lado, las organizaciones que trabajan en "modo factoría" deban invertir recursos para "asegurar" la calidad del software que fabrican; mientras que, por otro, los clientes deban "controlar" la calidad del software que le suministran las factorías.

ii) La importancia creciente de las certificaciones basadas en modelos como CMMI (Capability Maturity Model Integration), ISO 15504, etc. que destacan las actividades de aseguramiento de calidad como clave para la madurez de una organización que desarrolle o mantenga software. En España, la importancia que está adquiriendo para las organizaciones los modelos de este tipo [9] se refleja en el elevado número de certificaciones respecto a otros países europeos así como

las ayudas oficiales que se han puesto en marcha para fomentar la obtención de este tipo de certificaciones.

Por otro lado, a pesar de que la evaluación de la calidad del software es un campo de gran actividad investigadora desde hace bastantes años, la mayor parte del esfuerzo realizado se ha centrado en la calidad de los procesos, habiéndose desarrollado una plétora de modelos y estándares de referencia, evaluación y mejora de procesos software: ISO 90003, ISO 12207, ISO 15504, CMM, CMMI, IDEAL, SCAMPI, Competisoft, etc. Con la premisa de que utilizando un proceso de calidad se obtiene un producto de calidad, las organizaciones y empresas se han preocupado de exigir a sus proveedores (y los proveedores de obtener) una certificación sobre la capacidad y madurez de sus procesos, principalmente ISO 9001 (aplicando la guía específica para software ISO 90003), CMMI-DEV o ISO 15504 (SPICE).

Análogamente, en la última década se han publicado diversas normas y estándares relativos a métricas de seguridad, como los Criterios Comunes [15], ISO/IEC 27004 [16], NIST 800-55 [24] o FIPS 140-1/2 [4]. Aunque estas normas y estándares son amplias y con definiciones imprecisas de métricas de seguridad o demasiado limitadas como para cubrir una variedad extensa de situaciones de seguridad [31] y poco centradas en cubrir la medición de la seguridad desde las primeras fases del desarrollo software, a pesar de que de que numerosos estudios coinciden en que abordar la seguridad en las primeras fases del ciclo de vida del software tiene un impacto muy significativo en la reducción de vulnerabilidades [27, 29], y consecuentemente es más eficaz respecto a los costes y tiene como resultado diseños más robustos y sistemas más seguros. Esto se debe en parte a que la medición de la seguridad, es decir, la definición de métricas de seguridad, se trata de una disciplina que está aún dando los primeros pasos, y de la que hasta ahora no había muchos recursos documentados o trabajos centrados en ella.

En definitiva, pocas aportaciones se han hecho sobre la calidad y seguridad de los productos software. El área más trabajada ha sido la de pruebas (*testing*) del software, sobre todo de funcionalidad, y de hecho recientemente se han instalado en España las primeras "factorías de *testing*". Sin embargo, todavía no se han desarrollado las técnicas necesarias para evaluar la calidad y seguridad de un producto software de un modo efectivo. Por otro lado, es cada día mayor el número de organizaciones y empresas que están más interesadas en la calidad y seguridad de los productos que adquieren que en la de los procesos de sus proveedores, ya que una vez que el producto ha sido implantado en sus instalaciones, se encuentran con graves problemas de calidad y seguridad. Y esta preocupación no sólo atañe al código, sino también a los modelos que se usan para desarrollarlo, y que permiten asegurar la calidad y seguridad desde las etapas más tempranas del ciclo de desarrollo del software. En este sentido cabe destacar la revisión sistemática realizada en [6], que contempla las propuestas existentes desde 1998 hasta finales de 2008 sobre la calidad en el modelado software, a través de más de 300 artículos de las principales bibliotecas digitales (IEEE, ACM, Science Direct, Wiley, Scopus, etc.), y que pone de manifiesto cómo el tema de la calidad de los modelos software carece todavía de la suficiente madurez científica.

En efecto, existen algunas propuestas aisladas para evaluar la calidad del software: métricas [7], listas de control [17, 25, 30] y gestión de la consistencia [18, 23], pero la mayoría no han sido validadas empíricamente (salvo las referidas a programas desarrollados en Cobol o C), y no abordan el problema de la calidad de modelos de manera integrada. Como se señala recientemente en [3]: "el estado del arte respecto al aseguramiento de la calidad no aporta respuestas concluyentes. Por ejemplo, no podemos determinar la combinación adecuada de técnicas basándonos en el conocimiento empírico existente". Así, por ejemplo, ni siquiera

existen umbrales "fiables" para las métricas de calidad (en entornos modernos como Java, .Net, etc.)

En lo que respecta a la seguridad, es cierto que además de las normas y estándares expuestos anteriormente, se han realizado propuestas de métricas como trabajos sobre métricas de seguridad basados en la puntuación de vulnerabilidades o debilidades (CVSS [22], CMSS [26], CWE [21]), basados en el análisis de código fuente [8], basadas en la medición de la seguridad de la arquitectura del sistema [19], basadas en la medición de la seguridad de los diagramas de clases orientados a objetos [1], o basadas en el riesgo ([28], o MAGERIT [20]). Sin embargo, aunque muchas de estas propuestas son muy interesantes, habitualmente tratan la seguridad de un modo parcial, y sin ofrecer un claro seguimiento de esos aspectos de seguridad a lo largo del proceso de desarrollo, y haciendo que la definición de métricas de seguridad a nivel de diseño haya recibido poca atención en estos últimos años [1].

Por lo tanto, existe una **gran necesidad de controlar y evaluar la calidad y seguridad de los desarrollos informáticos**, tanto por parte de los clientes como por parte de las factorías de software y otras empresas de desarrollo. Por todo ello, estamos llevando a cabo el proyecto MEDUSAS "MEJORA Y EVALUACIÓN DEL DISEÑO, USABILIDAD, SEGURIDAD Y MANTENIBILIDAD DEL SOFTWARE, financiado por el Centro para el Desarrollo Tecnológico Industrial (CDTI) con Fondos Tecnológicos (FEDER).

2 Entorno MEDUSAS

El objetivo del proyecto es construir un entorno (metodológico e instrumental) que permita ofrecer, a empresas y organismos públicos, servicios de evaluación y control de la calidad y seguridad del software, de forma independiente. El entorno MEDUSAS permitirá evaluar no sólo la calidad y seguridad del código (software), sino también la calidad y seguridad del diseño del mismo. Este entorno está basado en la nueva familia de normas ISO 25000, conocida como SQuaRE (Software Quality Requirements and Evaluation) [11], y que reemplazará a las actuales normas ISO 9126 e ISO 14598. A finales del año 2007 empezaron a aprobarse las primeras normas de esta familia [12-14] y se espera que durante el año 2010 se terminen de aprobar oficialmente el resto.

En la Figura 1 se puede observar un diagrama de los componentes del proyecto MEDUSAS, diferenciándose claramente un componente metodológico, un componente tecnológico y un componente de gestión y divulgación del proyecto.

- **Entorno Metodológico**: se encuentra formado por los siguientes componentes:
  - **Metodología de evaluación de la calidad**: que define las actividades, roles, entradas y salidas, necesarias para llevar a cabo el proceso de evaluación de la calidad software. Esta metodología tiene en cuenta los principales estándares de evaluación software.
  - **Modelos de calidad**: tres modelos diferentes teniendo en cuenta las tres características de calidad que se tratan en este proyecto (mantenibilidad, seguridad y usabilidad).
  - **Métricas**: un conjunto de métricas para la mantenibilidad, seguridad y usabilidad del software.
  - **Heurísticas**: un conjunto de heurísticas para la mantenibilidad, seguridad y usabilidad del software.
  - **Checklists**: un conjunto de listas de control para la mantenibilidad, seguridad y usabilidad del software.

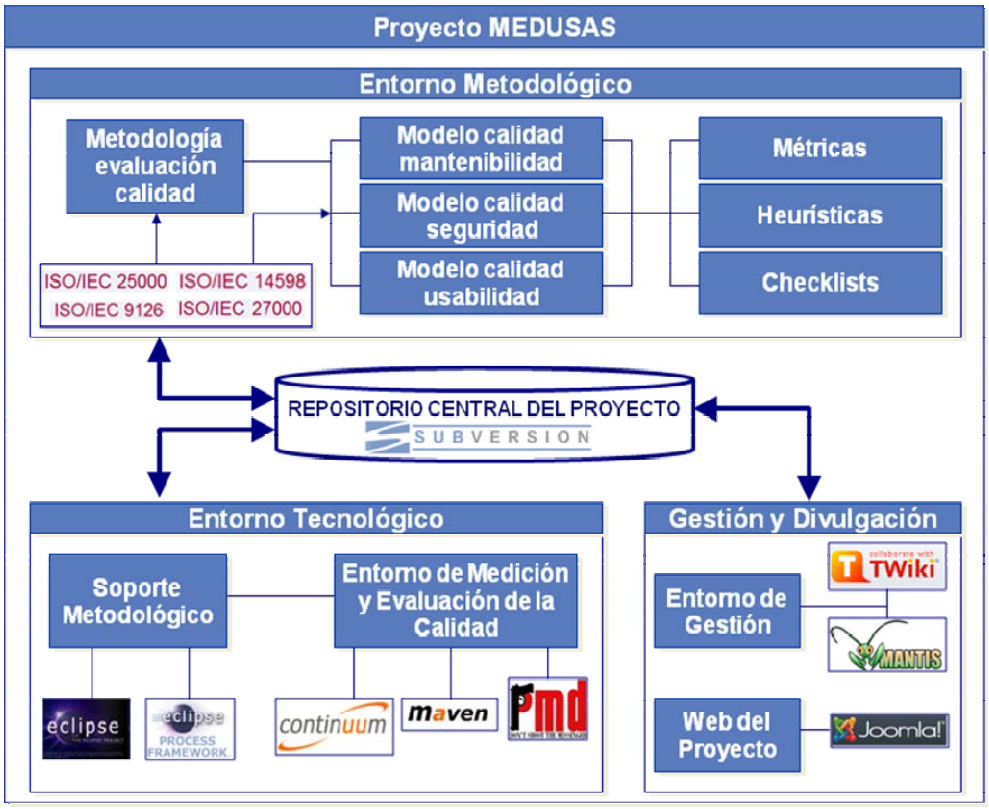

Figura 1. Componentes del Proyecto MEDUSAS

- **Entorno tecnológico:** que da soporte a todo lo definido en el soporte metodológico y que está formado por dos componentes:
  - **Soporte metodológico:** representa el entorno tecnológico que de soporte a la metodología y modelos de calidad desarrollados y que permita su aplicación práctica en proyectos.
  - **Entorno de medición y evaluación de la calidad:** que representa el conjunto de herramientas automáticas y la configuración de las mismas que permita llevar a cabo el proceso de medición de acuerdo a las métricas definidas en el entorno metodológico.
- **Entorno de gestión y divulgación:** formado por un conjunto de herramientas que permiten realizar la planificación, control y mejora del proyecto, así como desarrollar la tarea divulgativa y presentación de los resultados obtenidos.
- **Repositorio central del proyecto:** los resultados y evoluciones de los tres entornos anteriores se almacenan en un repositorio central que permite la comunicación entre las tres empresas integrantes y la gestión de configuración de las distintas versiones.

3 La metodología MEDUSAS

Como punto de partida para la elaboración de la metodología MEDUSAS se ha realizado una revisión de los principales estándares, normas y metodologías existentes en la actualidad que, en mayor o menor medida, se encuentran relacionados con la evaluación y el aseguramiento de la calidad del software.

El principal objetivo de la metodología MEDUSAS es definir un **marco de trabajo que permita determinar los procesos necesarios para llevar a cabo la evaluación de la calidad de los productos software**, así como facilitar la

comunicación entre la empresa cliente (patrocinador de la evaluación) y el equipo de evaluación.

Para alcanzar el objetivo principal, la metodología MEDUSAS presenta un conjunto de objetivos parciales, enumerados a continuación:

- Definir el conjunto de procesos necesarios para llevar a cabo la evaluación de la calidad de los productos software, así como las actividades en las que se descompone cada uno de ellos.
- Determinar los distintos participantes en el proceso de evaluación de la calidad, así como las fases en las que se produce comunicación entre ellos.
- Concretar los artefactos que se manejan durante el proceso de evaluación de la calidad (entradas al proceso, salidas o entregables, etc.).
- Identificar el catálogo de técnicas de evaluación aplicable durante el proceso de evaluación.

Las principales características de la metodología MEDUSAS 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 adaptativa.
- Está respaldada por un conjunto de modelos y herramientas de medición.

La metodología MEDUSAS está formada por un conjunto estructurado de procesos, proporcionando un marco de trabajo donde se identifica claramente el qué, cuándo, y el quién, de cada una de las fases y actividades, así como la secuencia de pasos que se debe seguir a la hora de llevar a cabo el proceso de evaluación de la calidad.

La metodología MEDUSAS está basada en 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 (en función de los objetivos de su empresa y del riesgo asociado a los activos evaluados). 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 de la empresa cliente, sino que se pueda planificar, diseñar y realizar la evaluación externamente, poniéndose en contacto con el cliente en los momentos puntuales donde sea necesario.

Además, la metodología de evaluación está pensada para poder ser adaptada a las distintas necesidades del cliente, existiendo un completo método de adaptación que permite, en función de las características y objetivos de la empresa cliente, implementar una versión propia que cumpla con su modo de trabajar y su filosofía de empresa.

Por último, la metodología MEDUSAS se encuentra respaldada por un conjunto de modelos de calidad y herramientas de soporte a la evaluación que definen los atributos de calidad que se pueden medir para los distintos artefactos software, agilizan la implementación de las fases de la metodología y permiten diseñar y realizar la evaluación de manera semiautomática.

3.1 Tipos de implementación

La metodología MEDUSAS ha sido elaborada para permitir dos tipos de implementaciones diferentes, de manera que la elección de una de ellas vendrá

determinada principalmente por la empresa cliente y por el modo de trabajo que mejor se adapte a sus intereses y modelo de negocio. Independientemente del tipo de implementación, la metodología MEDUSAS se compone de una serie de procesos, fases, actividades y tareas. Lo que diferenciará a una implementación de otra será principalmente el modo de comunicación entre la empresa cliente y la empresa evaluadora, así como la alineación entre los procesos de desarrollo de la empresa cliente y los procesos de la metodología MEDUSAS. A continuación se detallan los dos tipos de implementaciones por los que se puede llevar a cabo la evaluación de la calidad del producto software por medio de la metodología MEDUSAS.

3.1.1 Implementación adaptada a la empresa cliente (OnSite)

La primera modalidad de implementación para la cual se ha elaborado la metodología MEDUSAS consiste en adaptar el ciclo de vida de la metodología al ciclo de vida de los procesos de desarrollo de la empresa cliente. De esta manera, la metodología MEDUSAS pasa a ser un activo más de la empresa cliente, y el equipo de evaluación de la empresa evaluadora se convierte en parte del departamento de calidad de la empresa cliente.

La Figura 2 representa lo que podría ser un ejemplo de este tipo de implementación. Como se puede observar, dentro de los procesos que componen la metodología, lo que se hace es alinear el proceso de evaluación de la calidad con el ciclo de vida que siga la empresa cliente para el desarrollo de su proyecto software. De esta manera, de forma paralela a la planificación del proyecto se lleva a cabo la planificación del proceso de evaluación de la calidad, seguido de un proceso inicial de especificación común a todo el proyecto de evaluación. Posteriormente se realizarán una o varias iteraciones de las fases de la evaluación (tanto para los diseños como para el código fuente) relacionadas con la especificación, ejecución y conclusión de las tareas de evaluación. Finalmente, y de forma paralela a la fase de cierre del proyecto, se lleva a cabo la fase de conclusión, agrupando los resultados y certificando la calidad final del producto software desarrollado.

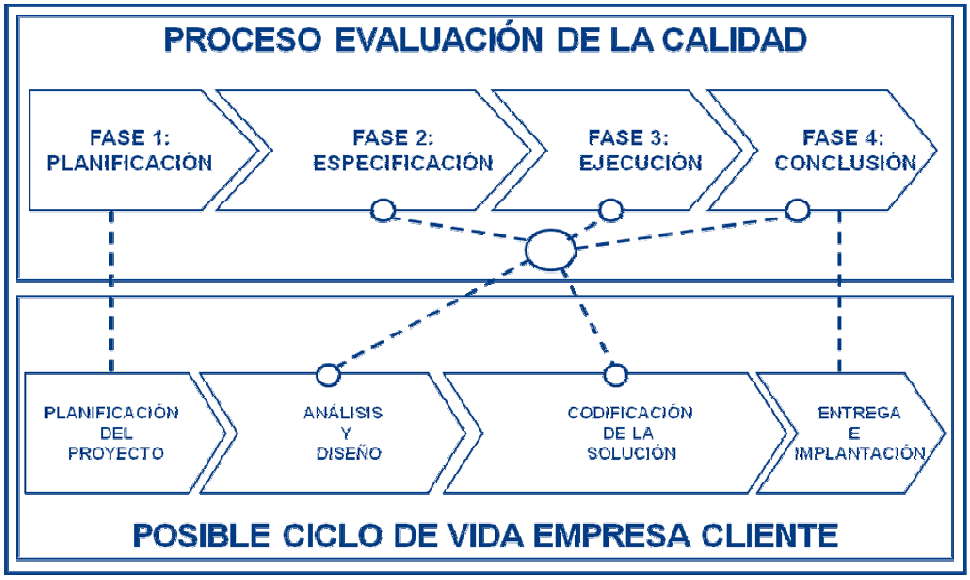

Figura 2. Ciclo de vida de la metodología MEDUSAS

Este tipo de implementación está recomendado para empresas desarrolladoras de software, con poca experiencia en el ámbito de la calidad y auditoría del producto,

que desean implantar un sistema para la evaluación de la calidad y carecen del personal necesario para alcanzar dicho objetivo.

2.2.2. Implementación externalizada (Outsourcing)

La segunda modalidad de implementación para la cual se ha elaborado la metodología MEDUSAS no requiere de un alineamiento con los procesos de la empresa cliente, teniendo como único requisito el realizar una correcta planificación de entregas de artefactos a evaluar sincronizada con el desarrollo del proyecto software, de manera que permitan la evaluación continua de la calidad.

Esta segunda modalidad se ha denominado "Implementación Externalizada" porque tiene lugar cuando una empresa externa es contratada (como es el caso de Alarcos Quality Center) para evaluar de manera continua la calidad de los artefactos software que, o bien son generados por la factoría o departamento de desarrollo que contrata los servicios, o bien son elaborados por uno de los proveedores de la empresa cliente que quiere evaluar la calidad de dichos artefactos de manera independiente.

En la Figura 3 se puede observar el modo de trabajo externalizado para el que está pensada también la metodología MEDUSAS.

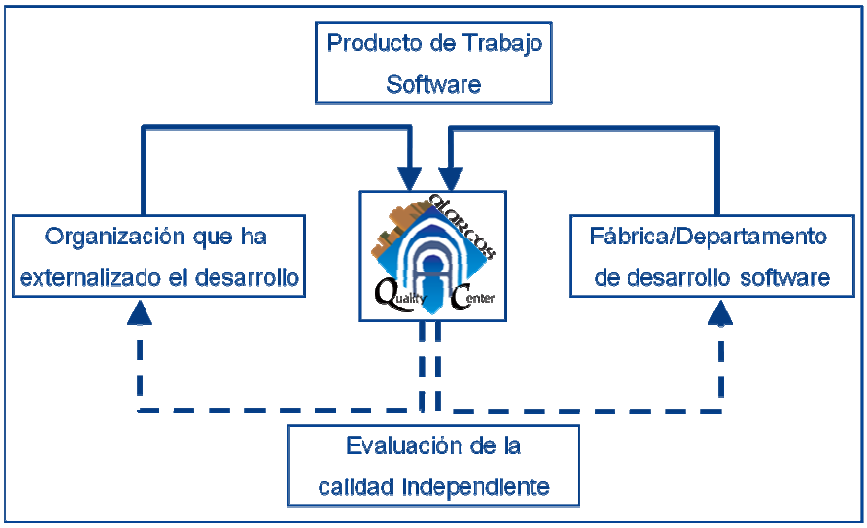

Figura 3. Implementación externalizada de la metodología MEDUSAS

En este caso, la empresa evaluadora (AQC en el ejemplo de la Figura 3) ofrece un servicio externalizado de evaluación de la calidad a sus clientes, fábricas de desarrollo software o empresas que externalizan el desarrollo, utilizando como entrada los distintos artefactos software (diseños y código) que se van generando a lo largo del ciclo de vida de desarrollo.

En cualquiera de las dos implementaciones detalladas anteriormente, la metodología MEDUSAS se encuentra preparada para llevar a cabo la evaluación de la calidad del producto software, a partir de los productos de trabajo generados principalmente en dos fases del ciclo de vida de desarrollo:

- **Productos de trabajo de la fase de análisis y diseño**: diagramas de casos de uso, transición de estados, clases, diseño de arquitecturas, etc.

- **Productos de trabajo de la fase de codificación**: código fuente del programa en los lenguajes Java, C#, Visual Basic, C y PHP, además de la documentación asociada a dicho código y los casos de prueba implementados.

Para ello, la metodología está complementada por un conjunto de métricas, checklists y heurísticas específicas para cada uno de los posibles productos de trabajo, además de encontrarse soportada por un conjunto de herramientas semi-automáticas que agilizan el proceso de medición, análisis y evaluación.

4 EVALUACION DE LA SEGURIDAD

En esta sección, por motivos de extensión máxima permitida, en esta comunicación únicamente se presentan de forma muy resumida la propuesta de modelo de seguridad y de métricas de seguridad, como parte del entorno MEDUSAS (que no solo incluye a la característica de seguridad sino también a las de mantenibilidad y usabilidad).

4.1 Modelo de seguridad

En la Tabla 1 se presenta el modelo de seguridad que proponemos, compuesto por dos grupos de características de seguridad. El primer grupo es relativo a propiedades de seguridad definidas para proteger al sistema ante ataques de seguridad, y el segundo grupo se refiere a propiedades de seguridad definidas para proteger al sistema de fallos y situaciones fortuitas. Cada característica define a su vez un conjunto de subcaracterísticas, que se refiere a algún matiz específico que aun estando relacionado con la característica a la que pertenece, tiene algún aspecto claramente diferenciador. Se puede comprobar que el modelo elaborado comparte cierta similitud con la ISO/IEC 25010, pero la enriquece ampliamente con aspectos identificados en propuestas más especializadas en seguridad.

	Características	Subcaracterísticas
Protección ante ataques	Autenticidad	Autenticación
		Identificación
	Confidencialidad	Anonimato
		Privacidad
	Conformidad	
	Detección de ataques	
	Disponibilidad	
	Integridad	Integridad de datos
		Integridad de hardware
		Integridad personal
		Integridad de software
		Protección física
	No Repudio	
	Trazabilidad	
Protección ante accidentes	Conformidad (*safety*)	
	Daño Comercial	
	Daño medioambiental	
	Seguridad y Salud del operador	
	Seguridad y Salud pública	

Tabla 1. Modelo de seguridad MEDUSAS

El modelo considera las siguientes características y subcaracterísticas, relativas a la protección de los sistemas de información ante ataques provocados:

- Autenticidad: Tiene que ver con el grado en el que se garantiza que los sujetos y recursos del sistema de información son auténticos.
  - Autenticación: Es relativo al grado en el que se verifica la identidad de los sujetos antes de interactuar con ellos.
  - Identidad: Es relativo al grado en que se identifican a los sujetos antes de interactuar con ellos.
- Confidencialidad: Es el grado en el que se asegura que la información es solamente accesible a sujetos autorizados.
  - Anonimato: Es el grado en que se impide el almacenamiento o descubrimiento de la identidad de los usuarios.
  - Privacidad: Es el grado en el que se asegura que la información de carácter personal, privado e íntimo es solamente accesible a sujetos autorizados.
- Conformidad: Es el grado en que los productos software se ajustan a los estándares, acuerdos, o regulaciones de leyes y otras recomendaciones similares relativos a seguridad.
- Detección de ataques: Es el grado en que los intentos de ataque o los ataques realizados con éxito son detectados, almacenados y notificados.
- Disponibilidad: Es el grado en que se asegura que los sujetos autorizados tienen acceso a los datos y aplicaciones en el momento en que lo requieran.
- Integridad: Es el grado en que se protege a los componentes de los sistemas de información de alteraciones intencionada por parte de sujetos no autorizados.
  - Integridad de datos: Concepto de integridad aplicado a los datos.
  - Integridad del hardware: Concepto de integridad aplicado a los componentes hardware del sistema.
  - Integridad del personal: Es el grado en que se protege la seguridad de las personas ante posibles reacciones del sistema provocados intencionadamente.
  - Integridad del software: Es el grado en que se protege los componentes de software de corrupción intencionada.
  - Protección física: Es el grado en que el sistema se protege a sí mismo y a sus componentes de ataques físicos.
- No repudio: Es el grado en que se impide que una parte de una interacción pueda repudiar algún aspecto de la interacción.
- Trazabilidad: Es el grado en que se asegura que las acciones de un sujeto pueden ser trazadas inequívocamente y asociadas a dicho sujeto.

El modelo de seguridad, para el caso de características de seguridad relativas a la protección de los sistemas de información ante accidentes no provocados, básicamente se heredan las propiedades definidas por la ISO/IEC 25010 en el modelo de Calidad en Uso

4.2 Métricas de seguridad

A modo de ejemplo, en esta sección se muestra un extracto de la propuesta de métricas de seguridad de diseño que se contemplan en MEDUSAS. Este conjunto de métricas tanto a nivel de diseño como posteriormente y de forma relacionada a

nivel de implementación permiten evaluar el nivel de cumplimiento deseado de los requisitos de seguridad que se hayan especificado en las etapas de análisis del software. Y que además, dichas métricas se integren con un modelo de seguridad (como componente de calidad) que claramente haya identificado una taxonomía de requisitos de seguridad para que éstos puedan ser identificados, modelados e implementados, junto al resto de requisitos, tanto funcionales como no funcionales. (Dicho modelo de seguridad ha sido definido en la sección anterior).

Tabla 2 Métrica para la integridad de la información

Campo	Dato
ID Métrica	Métrica de Integridad de la información 1
Características del modelo de seguridad	Integridad
Descripción Métrica	Porcentaje (%) de vulnerabilidades para las que se han aplicado parches o han sido mitigadas
Fórmula	(Nº de vulnerabilidades tratadas en alertas distribuidas para las que se han implementado parches, determinados como no aplicables, o concedidos un nº aplicable de vulnerabilidades identificados por las alertas y escáneres de vulnerabilidades) * 100

Tabla 3 Métrica de Identificación y Autenticación

Campo	Dato
ID Métrica	Métrica de Cuentas de Usuario 1
Características del modelo de seguridad	Autenticidad → Identificación y Autenticación (subcaracterísticas)
Descripción Métrica	Porcentaje (%) de usuarios con acceso a cuentas compartidas
Fórmula	(Nº de usuarios con acceso a cuentas compartidas / nº total de usuarios) * 100

Tabla 4 Métrica de Control de Acceso

Campo	Dato
ID Métrica	Métrica de Control de Acceso Remoto
Características del modelo de seguridad	Autenticidad → Identificación y Autenticación (subcaracterísticas)
Descripción Métrica	Porcentaje (%) de puntos de acceso remoto usados para obtener accesos no autorizados
Fórmula	(Nº de puntos de acceso remoto usados para obtener accesos no autorizados / nº total de puntos de acceso remoto) * 100

Agradecimientos

Esta comunicación es parte del proyecto BUSINESS (PET2008-0136) del Ministerio de Ciencia e Innovación (España), proyecto PEGASO/MAGO (TIN2009-13718-C02-01) Ministerio de Ciencia e Innovación (España) y el FEDER. Además, es parte de los proyectos SISTEMAS (PII2I09-0150-3135) y QUASIMODO (PAC08-0157-0668) de la Junta de Comunidades de Castilla – La Mancha y el FEDER así como parte del proyecto MEDUSAS (IDI-20090557) del Ministerio de Ciencia e Innovación (España - CDTI).

5 Bibliografía

1. B. Alshammari, C. Fidge, and D. Corney, *Security Metrics for Object-Oriented Class Designs.* 2009 Ninth International Conference on Quality Software, 2009: p. 11-20.
2. E. Carmel and P. Abbott, *Why "nearshore" mean that distance matter.* Communications of the ACM, 2007. **50**(10): p. 40-46.

3. F. Elberzhager and C. Denger, *A Comprehensive Planning Framework for Selecting and Customizing Quality Assurance Techniques*. 2007, IEEE Computer Society: Proc, of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007).
4. FIPS, *FIPS 140-2*, in *Security Requirements for Cryptographic Modules*. 2001, Federal Information Processing Standardization - National Institute of Standards and Technology
5. Gartner, *Analysis of Spain as an Offshore Services Location*. 2007: Garnet Research, noviembre 2007 ID Number: G00152674.
6. M. Genero, A. Fernández, J. Nelson, M. Piattini, and G. Poels, *A Systematic Literature Review on UML Conceptual Modeling Quality*. 2008: Technical Report UCLM.
7. M. Genero, M. Piattini, and C. Calero, *A Survey of Metrics for UML Class Diagrams*. 2005: Journal of Object Technology.
8. I. Chowdhury, B. Chan, and M. Zulkernine, *Security metrics for source code structures*, in *Proceedings of the Fourth International Workshop on Software Engineering for Secure Systems*. 2008, ACM: Leipzig, Germany.
9. INTECO. *Estudio sobre la certificación de la calidad como medio para impulsar la industria de desarrollo del software en España*. 2008 [cited March 2009]; Available from: http://www.inteco.es/Calidad_del_Software/estudios_e_indicadores/publicaciones/calidad_sw_estudios_e_informes/Calidad_software_32.
10. INTECO, *Estudio sobre el modelo de factorías de software con un enfoque nearshore.* . 2008 Laboratorio Nacional de Calidad de Software, Instituto Nacional de Tecnologías de Comunicación.
11. ISO, *ISO/IEC 25000 Software and system engineering – Software product Quality Requirements and Evaluation (SQuaRE) –Guide to SQuaRE*, in *International Organization for Standarization*. 2005: Ginebra, Suiza.
12. ISO, *ISO/IEC 25001 Software and system engineering – Software product Quality Requirements and Evaluation (SQuaRE) –Evaluation planning and management*, in *International Organization for Standarization.* 2007: Ginebra, Suiza.
13. ISO, *ISO/IEC 25020 Software and system engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Measurement reference model and guide*, in *International Organization for Standarization*. 2007: Ginebra, Suiza.
14. ISO, *ISO/IEC 25030 Software and system engineering – Software product Quality Requirements and Evaluation (SQuaRE) –Quality requirements*, in *International Organization for Standarization*. 2007: Ginebra, Suiza.
15. ISO/IEC, *ISO/IEC 15408:2005 Information technology - Security techniques - Evaluation criteria for IT security, (Common Criteria v3.0)*. 2005.
16. ISO/IEC, *ISO/IEC 27004:2009 - Information technology -- Security techniques -- Information security management -- Measurement.* 2009.
17. C. Lange, B. DuBois, M. Chaudron, and S. Demeyer, *An experimental investigation of UML modeling conventions.* 2006, Springer: Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2006). p. 27-41.
18. F. Lucas, F. Molina, and A. Toval, *A Systematic Review of UML Model Consistency Management*. 2008: Information and Software Technology.
19. P.K. Manadhata, K.M.C. Tan, R.A. Maxion, and J.M. Wing, *An Approach to Measuring A System's Attack Surface*. 2007, Carnegie Mellon University: Pittsburgh,.
20. MAP, *Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información (MAGERIT - v 2)*. 2005, (Ministry for Public Administration of Spain).
21. R.A. Martin, *Common Weakness Enumeration (CWE v1.8)*. 2010, National Cyber Security Division of the U.S. Department of Homeland Security.
22. P. Mell, K. Scarfone, and S. Romanosky, *A Complete Guide to the Common Vulnerability Scoring System (CVSS 2.0)*. 2007, NIST and Carnegie Mellon University.
23. J. Muskens, R. Bril, and M. Chaudron, *Generalizing consistency checking between software views*. 2005: 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05). p. 169-180.
24. NIST, *Special Publication 800-55 Revision 1 - Performance Measurement Guide for Information Security*. 2008, National Institute of Standards and Technologies.
25. P. Oman and C. Cook, *A taxonomy for programming style*. 1990: Proceedings of the 18th ACM Computer Science Conference. p. 244-250.
26. E.V. Ruitenbeek and K. Scarfone, *The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities*, in *NIST Interagency Report 7517*. 2009, National Institute of Standards and Technology.

27. A. Sachitano, R.O. Chapman, and J.A. Hamilton, *Security in software architecture: a case study*. Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshp, 2004: p. 370-376.
28. O.S. Saydjari, *Is Risk a good security metric?* Quality of Protection Workshop – Security Measurements and Metrics (QoP'06), 2006: p. 59-60.
29. E.A. Schneider. *Security architectre-based system design*. in *Proceedings of the 1999 workshop on New Security Paradigms*. 1999. Ontario, Canada: ACM.
30. B. Unhekkar, *Verification and Validation for Quality of UML 2.0 Models*. 2005: Wiley Interscience.
31. A.J.A. Wang, *Information Security Models and Metrics.* 43nd ACM Southeast Conference, 2005: p. 2-178 to 2-184.