Mejora de la mantenibilidad con un modelo de medición de la calidad: resultados en una gran empresa

Emanuel Irrazábal

Departamento de Informática. FaCENA.
Universidad Nacional del Nordeste
Av. Libertad 3400, Ciudad de Corrientes, Corrientes
Facultad de Ingeniería y Tecnología
Universidad de la Cuenca del Plata
Lavalle 50, 3400, Ciudad de Corrientes, Corrientes
emanuelirrazabal@gmail.com

Abstract. Ante un mercado cada vez más competitivo y en constante desarrollo, la calidad del software está tomando mayor importancia en las organizaciones, y con ello, la calidad del producto software, medida directamente en el código fuente. En un conjunto de artículos que el autor ha publicado anteriormente se presentó una recopilación de herramientas abiertas para el análisis del código fuente y la correspondencia entre las métricas proporcionadas con la mantenibilidad. Junto con ello se ha trabajado en un modelo de medición de la mantenibilidad del código fuente en base a las mediciones de las herramientas de análisis estático. En este artículo se presentan los resultados de la utilización del modelo de medición y de las herramientas durante diez meses en una gran empresa española. Se detalla la evolución de las mediciones, la satisfacción de los desarrolladores y como han ido evolucionandolas incidencias encontradas en las pruebas de aceptación.

Keywords: mantenibilidad, mejora, gran empresa, ISO 25010, herramientas

1 Introducción

Ante un mercado cada vez más competitivo y en constante desarrollo, la calidad del software está tomando mayor importancia en las organizaciones, y con ello, la medición software [1] ha adquirido mayor relevancia como principal herramienta para medir la calidad del software. Esta no sólo tiene influencia en los costos finales, sino que es también un factor clave para mejorar la imagen frente a los clientes y como elemento diferenciador de la competencia [2].

Aunque la calidad puede describirse desde diferentes perspectivas, la calidad software tradicionalmente se divide en calidad del producto y proceso [3]. Es decir, la calidad en el proceso de desarrollo para obtener el producto (actividades, tareas, etc. del desarrollo y mantenimiento del software) y la calidad del propio producto software. Debido a ello, se ha incrementado la tendencia hacia la institucionalización de buenas prácticas de calidad en organizaciones de desarrollo software. Dicha

tendencia se ha visto reflejada en un aumento constante del número de organizaciones que tienen la intención de alinear sus procedimientos con modelos de referencia de proceso ampliamente conocidos, como CMMI-DEV [4] o ISO/IEC 12207 [5].

Pero un proceso institucionalizado no necesariamente concluye con un producto de calidad. De hecho, son muchas las críticas que argumentan que estudiar solo la calidad del proceso puede llevarnos a obtener una visión parcial de la calidad software [6]. Puede ocurrir que la estandarización de los procesos y su uso incorrecto incluso institucionalicen la creación de malos productos. En esta línea, Maibaum y Wassyng [7], comentaban que las evaluaciones de calidad deberían estar basadas en evidencias extraídas directamente de los atributos del producto, no en evidencias circunstanciales deducidas desde el proceso.

Teniendo esto en cuenta, existen diferentes modelos de referencia para la calidad del producto. Sin embargo, es difícil llevar a la práctica la implantación de estos modelos. La principal barrera viene del carácter abierto de estas normas. Esto es, no muestran detalles sobre qué métricas utilizar o cuáles son las más adecuadas; cómo agrupar los valores de las métricas para obtener métricas de más alto nivel; o cuáles son los valores umbrales recomendados para cada métrica.

Con la intención de paliar los efectos de esta situación la Organización Internacional para la Estandarización (ISO) ha publicado una serie de estándares relacionados con la calidad del producto. Uno de estos estándares o normas es la ISO/IEC 9126:1991 Software Engineering – Product Quality [8]. El principal objetivo de esta norma es proporcionar un marco de referencia para definir y evaluar la calidad del producto software. Para ello la norma especifica un modelo de calidad identificando un conjunto de características de calidad. Así mismo, en 2005 se publica la serie de estándares ISO/IEC 25000. Esta serie, también denominada Software product Quality Requirements and Evaluation (SQuaRE) [9], la conforman un conjunto de normas internacionales orientadas a la calidad del producto; teniendo como objetivo organizar y unificar los estándares que especifican los requerimientos de calidad del software y su evaluación. Más recientemente, en marzo del 2011 se publica la norma ISO/IEC 25010 [10], que reemplaza a la norma ISO/IEC 9126 y define un modelo de calidad para los productos software.

Una de las características definidas en el modelo de calidad del producto de la norma ISO/IEC 25010 es la mantenibilidad. Históricamente la mantenibilidad ha sido reconocida como una de las características más relevantes del software debido a su impacto directo sobre el costo de desarrollo y el propio mantenimiento. De hecho, estudios previos señalan la fase de mantenimiento como la fase donde más recursos se invierten en el ciclo de vida del software, implicando dos veces el coste de la fase de desarrollo [11][12][13][14][15]. Por ejemplo, según [14], el tiempo que un programador invierte en mantenimiento es alrededor del 61% frente al 39% dedicado al desarrollo. Y en ocasiones la proporción de costo que supone el mantenimiento ronda el 90% [16].

En ese sentido, el modelo de calidad del producto descrito en la norma ISO/IEC 25010 no proporciona un conjunto consensuado de métricas concretas para calcular la mantenibilidad. Aun así, establece las propiedades que deberían tener estas métricas: han de ser objetivas, reproducibles, independientes, expresadas mediante escalas válidas y suficientemente precisas para realizar comparaciones fiables. Teniendo estas premisas en mente, y el hecho de que en la actualidad los desarrollos software suelen

tener un gran tamaño, con lo que el número de medidas a obtener es muy elevado, parece que el enfoque más conveniente sería disponer de herramientas que facilitaran la obtención automática de tales medidas.

Por todo ello, en un conjunto de artículos publicados anteriormente hemos presentado: una recopilación de herramientas abiertas para el análisis del código fuente, sus métricas y la correspondencia entre estas y la mantenibilidad [17][18][19]. Y un modelo de medición de la mantenibilidad que construye una medición abstracta de la mantenibilidad del código fuente en base a las mediciones de las herramientas comentadas en el punto anterior [20][21].

Como paso final de esta serie de trabajos aquí se presentan los resultados de la utilización del modelo de medición durante diez meses en una gran empresa española. Y, especialmente, se discuten los resultados de incorporar herramientas de análisis estático del código fuente en los equipos de desarrollo software de una empresa.

En la sección 2 de este artículo se presenta la estructura del modelo de medición. En la sección 3 se indican las herramientas y métricas utilizadas. En la sección 4 se detallan los resultados obtenidos en una gran empresa española. Por último, en la sección 5 se discuten las conclusiones finales.

2 El Modelo de Medición de la Mantenbilidad

En esta sección se describe brevemente la estructura elegida para el desarrollo de la herramienta implantada en la empresa. Para poder obtener valores indicadores de la calidad del producto software, es necesario establecer unos parámetros sobre los que realizar dicha medición. Para ello, el entorno metodológico de la herramienta se basa en la norma ISO/IEC 9126 e ISO/IEC 25010, que establecen unos atributos a partir de los cuales se obtendrán valores indicadores de calidad. En artículos anteriores se describe en detalle [20][21].

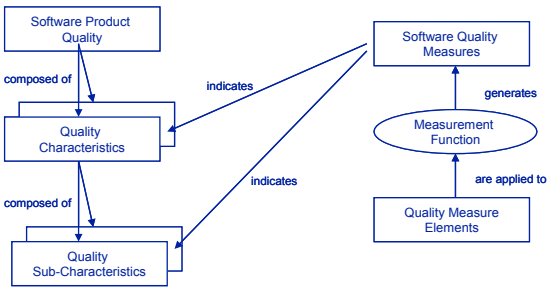


Fig. 1. Modelo de referencia de medición de la calidad del producto software (ISO/IEC 25000).

El valor de estas medidas de calidad software se obtiene por la aplicación de una función de medida (Measurement Function) a los elementos de medida de calidad (Quality Measure Elements). Aunque las normas ISO/IEC 9126 y 25010 establecen cuáles son las características de la calidad de un producto software y sus subcaracterísticas, no especifica qué medidas pueden utilizarse como indicador de una

subcaracterística (ver Fig. 1). Así, es necesario relacionar atributos del código fuente respecto de las diferentes características de calidad del producto software. Obtener una medición significa, por tanto, obtener el valor de un atributo. Si se trata de un atributo base, su valor se obtendrá leyendo el informe proporcionado por la herramienta que lo midió, y si es compuesto se calculará a través de una función de medición

3 Las Herramientas Seleccionadas y el Modelo de Medición de la Mantenibilidad Construido

A continuación se resumen las herramientas de medición y su relación con las características de calidad de la norma ISO 9126 e ISO 25010. Esto ha dado como resultado el modelo de medición de la mantenibilidad utilizado en la mejora de la empresa. La Tabla 1enumera las herramientas libres de análisis del código fuente y evaluación de pruebas unitarias seleccionadas en otros trabajos [17][18][19]. Se indica el nombre de la herramienta y las métricas básicas que pueden obtenerse.

Tabla 1. Selección final de herramientas y métricas básicas relacionadas.

Herramientas	Métricas
JavaNCSS	Sentencias de código fuente no comentadas (NCSS) Complejidad ciclomática (CC)
PMD/CPD	Evaluación del código fuente (posibles errores, código no utilizado, código duplicado)
CheckStyle	Validación de estilos
FindBugs	Evaluación del código fuente (posibles errores)
JDepend	Dependencias cíclicas (CDC) Acoplamiento aferente, acoplamiento eferente (Ca, Ce)
CCCC	Número de módulos (NOM) Líneas de código (LOC) Líneas de comentarios (COM) Complejidad Ciclomática (MVG) Peso del método por clase (WMC) Fan-In, Fan-In concreteness, Fan-Out visibility, Fan-Out (…) (FI,FIc,FIv,FO,FOc,FOv) Líneas de código (LOC) Líneas de comentarios (COM)
StyleCop	Validación de estilos
FxCop	Evaluación del código fuente (posibles errores)
Junit	Número de pruebas unitarias (UT)
CPPUnit	Número de pruebas unitarias (UT)
Nunit	Número de pruebas unitarias (UT)
EMMA	Cobertura de pruebas unitarias por paquetes (CBP) Resumen de cobertura de pruebas unitarias (OCS)

A continuación se muestra un resumen el modelo de medición de la mantenibilidad desarrollado en [20][21]. El primero de los elementos configurables del modelo de medición son las medidas de calidad software a partir de las cuales se obtienen los valores indicadores de calidad de las características y subcaracterísticas. La selección de medidas de calidad para la característica de mantenibilidad se muestra en la Tabla 2.

Tabla 1. Relación de medidas de calidad del modelo de medición respect de la mantenibilidad y sus sobcaracterísticas (ISO 9126).

Característica	Subcaracterística	Medida de calidad

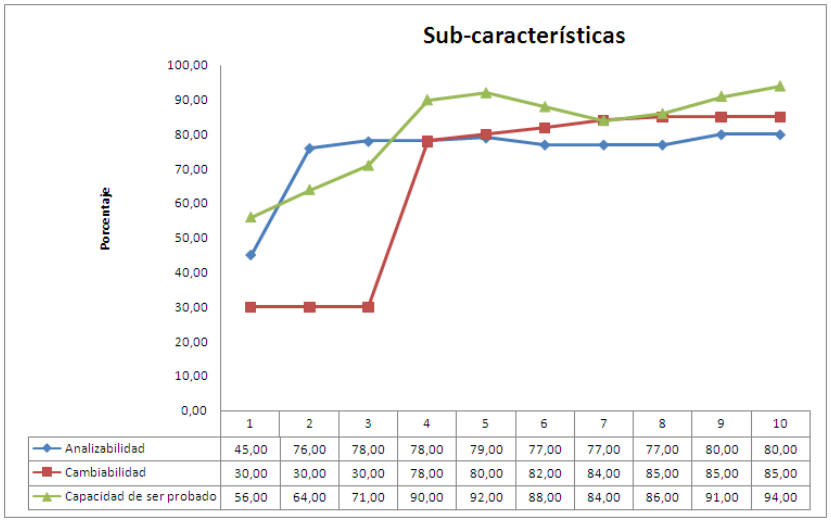
Mantenibilidad	Analizabilidad	Densidad de complejidad ciclomática
		Densidad de código repetido
		Densidad de comentarios
		Densidad de defectos de la capacidad para ser analizado
	Cambiabilidad	Densidad de defectos de la capacidad para ser cambiado
		Densidad de ciclos
	Estabilidad	Densidad de defectos de estabilidad
	Capacidad de ser probado	Densidad de defectos en pruebas unitarias
		Densidad de pruebas unitarias
		Cobertura de pruebas unitarias
		Tasa de errores de pruebas unitarias

4 Análisis de los Resultados al Implantar el Modelo de Medición

En este apartado se analizarán los resultados de la implantación y uso continuado del modelo de medición de la mantenibilidad. La implementación se realizó en una gran empresa española del sector público, con desarrollos en Java. .Se analizó la mejora de la mantenibilidad y se lo comparó con la evolución de las incidencias detectadas durante el desarrollo de diferentes proyectos.

Como se puede observar en la Figura 2, durante el primer mes la analizabilidad era bastante mala (en el orden de un 30%). Se realizó un programa detallado de mejora y se planificaron formaciones para aprender la utilización de las herramientas de análisis estático de código fuente. Por otro lado se comunicaron los objetivos y la manera en la que el modelo de medición penalizaba la ausencia de comentarios y las violaciones de las reglas implantas en las herramientas. Como resultado de estos cambios y la comunicación constante de los resultados se pudo mejorar la analizabilidad (especialmente debido a que las herramientas como PMD o FindBugs contienen ejemplos, explicaciones y otras características didácticas para favorecer el aprendizaje de los desarrolladores.

De igual manera, la subcaracterística de cambiabilidad ha tenido un período prolongado de estancamiento (unos 3 meses) hasta llegar a valores aceptables (superiores incluso al 80%) y mejores que la analizabilidad. En esos tres primeros meses se realizaron refactorizaciones para romper dependencias cíclicas y disminuir la complejidad ciclomática. A partir del tercer mes los nuevos desarrollos fueron adecuándose a las nuevas características de codificación.



	1	2	3	4	5	6	7	8	9	10
Analizabilidad	45,00	76,00	78,00	78,00	79,00	77,00	77,00	77,00	80,00	80,00
Cambiabilidad	30,00	30,00	30,00	78,00	80,00	82,00	84,00	85,00	85,00	85,00
Capacidad de ser probado	56,00	64,00	71,00	90,00	92,00	88,00	84,00	86,00	91,00	94,00

Fig. 2. Evolución de las mediciones. Subcaracterísticas de la mantenibilidad.

En el caso de la capacidad de ser probado el código fuente, existía ya una cultura de utilización de pruebas unitarias. En ese sentido, el problema era la cobertura de las pruebas unitarias y el tratamiento de los errores. Se consiguieron buenos resultados desde el inicio, incluso superando en los meses 4 y 5 el 90%. Durante los meses 7 y 8, al incorporarse nuevos desarrollos, los recursos necesarios para la realización de pruebas disminuyó.

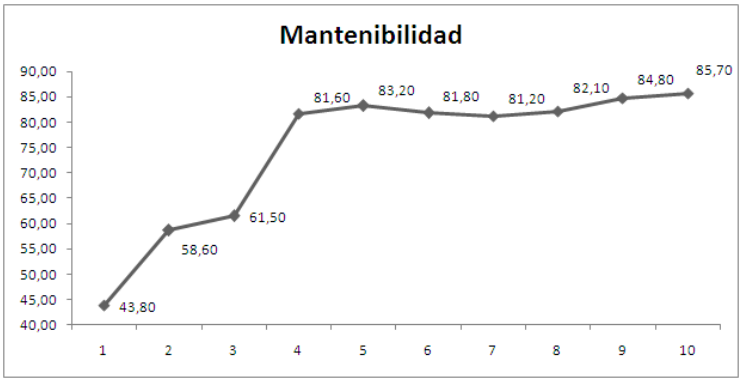


Fig. 3. Evolución de las mediciones. Característica de mantenibilidad

Como puede verse en la Figura 3, la mantenibilidad alcanzó registros aceptables (mayor de 80%) a partir del cuarto mes. Y se mantuvo en la franja de entre el 80% y el 85% en los siguientes seis meses. También en esta gráfica se advierte una pequeña disminución de la mantenibilidad, durante los meses 6, 7 y 8 debido al inicio del

desarrollo de nuevos módulos. Una de las ventajas evidentes al implementar las herramientas de análisis estático del código fuente ha sido la mejora en el nivel de conocimiento y productividad de los desarrolladores. Al final del quinto y del décimo mes se realizó una encuesta donde se preguntaba el grado de satisfacción al utilizar las herramientas de calidad.

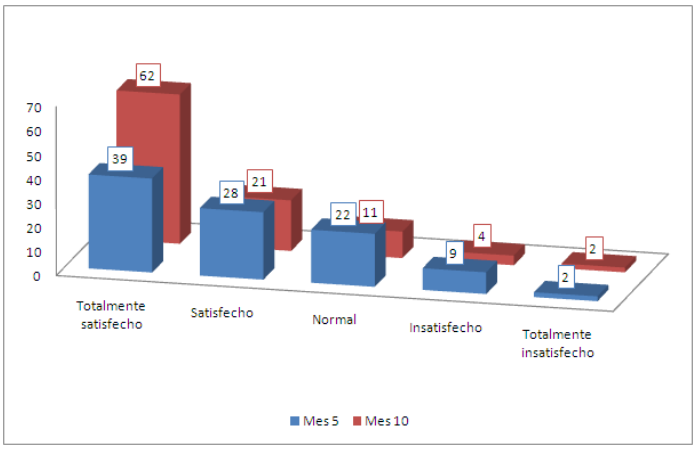
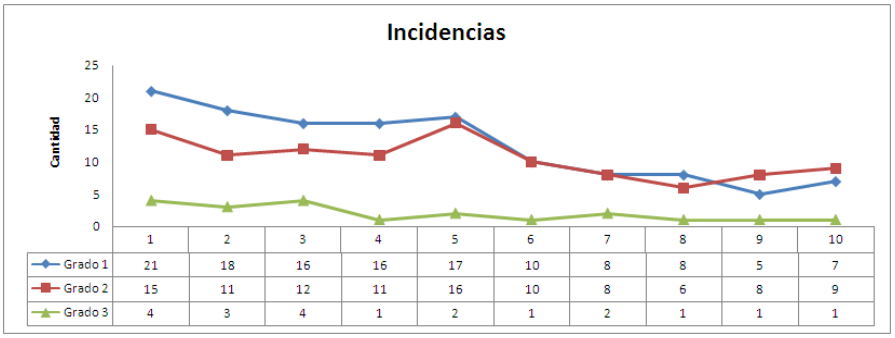


Fig. 4. Resultado de la encuesta de satisfacción realizada a los desarrolladores

Los resultados, que pueden verse en la Figura 4, muestran cómo se pasó de un 39% a un 62% de desarrolladores totalmente satisfechos. Así mismo, los desarrolladores no decididos respecto de estas buenas prácticas pasaron de un 22% a un 11%. Uno de los aspectos más importantes a destacar es que el 2% de los desarrolladores estaban completamente insatisfechos con las nuevas prácticas, permaneciendo esto hasta el décimo mes. Continuando con el análisis de los resultados se recopilaron datos de las incidencias encontradas y resueltas durante los diez meses de utilización de las herramientas de calidad. En este caso se ha considerado como incidencia cualquier error detectado en las pruebas de aceptación, ocasionado por problemas en el código fuente. Se descartaron los errores que provienen de una mala interpretación de los requisitos, y que, por otra parte, estuviesen bien codificados. Los errores se califican por grado, dependiendo el tiempo en que se puede resolver y otros aspectos técnicos que serán detallados a continuación:

- Grado 1: identificados con precisión y que pueden ser corregidos y probados en el día.
- Grado 2: errores que afectan más de una capa (lógica de negocio, comunicación, base de datos, etc.) y requiere más de 1 día de trabajo.
- Grado 3: errores de grado 2 que impiden el desarrollo y/o mantenimiento de otras partes del código fuente.

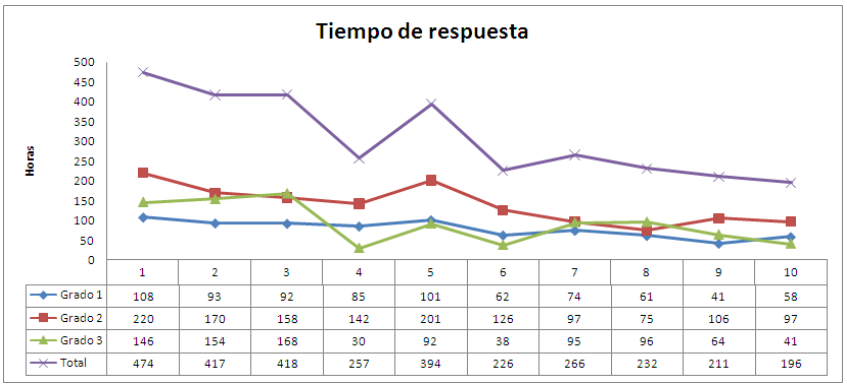
A continuación, y analizando el gestor de tareas de la empresa (utilizada desde antes de la implementación de las herramientas de calidad) se graficó la evolución del número de incidencias.



	1	2	3	4	5	6	7	8	9	10
Grado 1	21	18	16	16	17	10	8	8	5	7
Grado 2	15	11	12	11	16	10	8	6	8	9
Grado 3	4	3	4	1	2	1	2	1	1	1

Fig. 5. Evolución de las incidencias

Como se puede ver en la Figura 5, las incidencias han disminuido. Es importante recalcar aquí que no se aumentaron las incidencias en los tres primeros meses, aunque se realizaron gran cantidad de refactorizaciones para disminuir el acoplamiento entre clases. Las incidencias del tipo 1 y 2 tendieron a equipararse. Las conclusiones obtenidas con el departamento de calidad de la empresa indican que la disminución de incidencias del tipo 1 ha posibilitado la profundización en las inspecciones, y, por lo tanto, el descubrimiento de otro tipo de problemas. Esto se ha considerado ampliamente beneficioso.



	1	2	3	4	5	6	7	8	9	10
Grado 1	108	93	92	85	101	62	74	61	41	58
Grado 2	220	170	158	142	201	126	97	75	106	97
Grado 3	146	154	168	30	92	38	95	96	64	41
Total	474	417	418	257	394	226	266	232	211	196

Fig. 6. Evolución de los tiempos de respuesta de las incidencias.

En la Figura 6 se indican los tiempos de respuesta totales para cada tipo de incidencia. La tendencia general de los tiempos de respuesta ha ido en disminución. En el mes 5 se ha notado un aumento considerable (acompañando la cantidad de incidencias, vistas anteriormente, con la cantidad de horas invertidas en solucionarlo).

En este sentido, y analizándolo con el departamento de desarrollo de la empresa se ha concluido que las causas estaban relacionadas con la inclusión de nuevos desarrollos realizados por personal externo. Estos nuevos recursos no tuvieron la misma formación inicial; por el contrario se realizó una formación a distancia supervisada por los desarrolladores de la empresa cliente.

5 Conclusiones

La implementación del modelo de medición de la mantenibilidad se realizó en una gran empresa española del sector público, con desarrollos en Java. .Se analizó la mejora de la mantenibilidad y se lo comparó con la evolución de las incidencias detectadas durante el desarrollo de diferentes proyectos. Como primera conclusión y una de las ventajas evidentes al implementar las herramientas de análisis estático del código fuente ha sido la mejora en el nivel de conocimiento y productividad de los desarrolladores.

Se realizó un programa detallado de mejora y se planificaron formaciones para favorecer el aprendizaje de las herramientas. Por otro lado se comunicaron los objetivos y la manera en la que el modelo de medición penalizaba la ausencia de comentarios y las violaciones de las reglas implantas en las herramientas. Como resultado de ello, la mantenibilidad alcanzó registros aceptables (mayor de 80%) a partir del cuarto mes.

En cuanto a la percepción por parte del personal, se pasó de un 39% a un 62% de desarrolladores totalmente satisfechos. Así mismo, los desarrolladores no decididos respecto de estas buenas prácticas pasaron de un 22% a un 11%. Finalmente, la cantidad de incidencias ha disminuido, así como el tiempo de respuesta promedio.

La implantación de herramientas de análisis estático de código fuente en la empresa no sólo ha demostrado que proporciona visibilidad con sus indicadores; sino que gracias a ser soportado por un modelo de medición facilitó la personalización de las herramientas a las necesidades de la empresa. E incluso a las necesidades de distintos proyectos dentro de la propia empresa.

Como trabajo futuro se espera continuar con el estudio de la mantenibilidad desde el punto de vista de la deuda técnica, enfocando las mediciones a la manera en la cual evolucionan los intereses de la deuda.

Agradecimientos

Agradezco al proyecto F007-2009 “Modelos y métricas para la evaluación de la calidad del software” de la SECYT-UNNE. Y agradezco al grupo Kybele de la Universidad Rey Juan Carlos con quienes he trabajado en los primeros trabajos de esta línea de investigación.

Referencias

[1] Rifkin, S. (2009) Guest Editor's Introduction: Software Measurement. IEEE Software, 26, p. 70.

[2] Piattini, M., et al. (2008) Medición y estimación del software: técnicas y métodos para mejorar la calidad y productividad del software, Ra-ma.

[3] Ebert C. (2009) Guest Editor's Introduction: How Open Source Tools Can Benefit Industry. IEEE Software, 26, p: 50-51.

[4] SEI, Process Maturity Profile (2001) CMMI v1.2, SCAMPI v1.2, Class A Appraisal Results. 2011 Mid-Year Update, Septiembre 2011, Software Engineering Institute.

[5] ISO, ISO/IEC 12207:2008. Systems and software engineering - Software life cycle processes, in International Organization for Standarization.

[6] Kitchenham, B. and S.L. Pfleeger (1996) Software Quality: The Elusive Target. IEEE Software, 20(1), p: 12-21.

[7] Maibaum T, W.A. (2008) A Product-Focused Approach to Software Certification.

[8] ISO, Software Product Evaluation–Quality Characteristics and Guidelines for their Use. ISO/IEC Standard 9126, International Organization for Standarization.

[9] 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.

[10] ISO, ISO/IEC 25010 Software and system engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Quality model and guide. International Organization for Standarization. . 2005: Ginebra, Suiza.

[11] Frazer, A. (1992) Reverse engineering- hype, hope or here? Software Reuse and Reverse Engineering in Practice, p: 209-243.

[12] Pressman, R., (2002) Software Engineering: A Practitioner's Approach., Madrid: McGraw-Hill.

[13] Saiedian, H. and N. Carr (1997) Characterizing a software process maturity model for small organizations. , in ACM SIGICE Bulletin, p. 2-11.

[14] Janice, S. (1998) Practices of Software Maintenance, in Proceedings of the International Conference on Software Maintenance, IEEE Computer Society.

[15] Harrison, W. (2005) What Do Software Developers Need to Know about Business? IEEE Software, 22(5), p: 5-7.

[16] Erlikh L. (2000) Leveraging Legacy System Dollars for E-business. IT Professional, 2, 3, pp: 17-23.

[17] Irrazabal, E., Vara, J. M., Garzás, J., Santa Escolástica, R., & Marcos, E. (2010, June). Alignment of Open Source Tools with ISO norms for software product quality. In SOFTWARE MEASUREMENT EUROPEAN FORUM (p. 135).

[18] Irrazábal, E., Garzás, J., & Marcos, E. (2011). Alignment of Open Source Tools with the New ISO 25010 Standard-Focus on Maintainability. In ICSOFT (2) (pp. 111-116).

[19] Irrazábal, E., & Garzás, J. (2010). Análisis de métricas básicas y herramientas de código libre para medir la mantenibilidad. Innovación, Calidad e Ingeniería del Software, 6(3), 56.

[20] Quevedo A, Irrazábal E, Enríquez de Salamanca J, Vara J. M., Garzás J. Entorno de calidad KEMIS Congreso: XII QUATIC 2010, Industrial Track, . pp. 122-136; ISBN: 978-989-96867-0-0Oporto, (Portugal). 29 de Septiembre de 2010.

[21] Irrazábal, E. (2012). Construcción de un Entorno para la Medición Automatizada de la Calidad de los Productos Software.