# Adaptación de un Modelo de Desarrollo de Software para la Construcción de Software Usable con Calidad

Fabio Armando García Toribio[1], Juan Manuel Gómez Reynoso[1]

1Universidad Autónoma de Aguascalientes,
Centro de Ciencias Básicas, México.fgarc@live.com.m
*Paper received on 25/08/10, Accepted on 05/10/10.*

**Abstract.** Software is built in an intuitive way as a generalization of users; this approach makes the developer to ignore end-users except during analysis phase. Ignoring the user may result in systems that are not easy-to-use, as well as economic losses and high maintenance costs. This research proposes a way to evaluate software, enriched through the use of quality models, such as the proposed by McCall as well as the ISO/IEC 9126. This evaluation is made primarily as an assessment by end-users, since they made the final decision to accept or reject the system. This proposal intends to evaluate whether our approach can improve software development process using such models so that a user satisfaction can be improved.

**Keywords:** User-Centered Development, Usability, Quality, Quality Factors, Triangle McCall, ISO/IEC 9126.

## 1 Introducción

El software se ha vuelto un elemento indispensable para toda organización, un término común y entendido aun para aquellos ajenos al área. En la actualidad, es común encontrar sistemas informáticos en casi cualquier lugar, robustos sistemas bancarios, sistemas contables, y administrativos. Cada uno enfocado a un dominio en particular, haciendo de aquellos trabajos complejos actividades simples de realizar. Sin embargo, esto no es tan sencillo dado que un punto fundamental para la correcta aplicación de los sistemas informáticos estriba en el grado de eficiencia en el cual, el usuario puede interactuar y completar las tareas que desea de forma satisfactoria. Sin embargo, las prácticas de desarrollo de software en muchas ocasiones no contemplan a quiénes serán los responsables de operar el sistema, es decir, el usuario final. Todo esto provocando que los sistemas sean difíciles de manejar, y en ocasiones, colapsos totales debido a un software ineficiente o mal diseñado. Debido a todo esto, consideramos indispensable que el software sea sometido a una cuidadosa revisión por parte de los usuarios antes de su entrega definitiva. De esta manera, se


*Advances in Intelligent and Information Technologies.*
*Research in Computing Science 50, 2010, pp. 113-124.*

buscaría cumplir con una meta primordial en el software: Usuarios satisfechos en sus expectativas [1].

Para los desarrolladores del software es importante contar con modelos que aseguren la calidad, afirmando de manera eficiente aquello que los usuarios necesitan. Dichos modelos listan un conjunto de atributos, los cuales todo producto de software con alta calidad debe poseer (ej. confiabilidad, mantenibilidad, entre otros) [2].

## 2 Conceptos Clave

### 2.1 Calidad del software

Existen múltiples definiciones de calidad del software. Algunas de ellas son: ISO/DIS 9126 [3]:

"La totalidad de características de un producto de software que conlleva en su habilidad de satisfacer necesidades existentes o implícitas".

Dentro de la literatura computacional existen generalmente dos significados aceptados para el término calidad. El primero es que la calidad significa "conocer los requerimientos". Con esta definición, para tener un producto de calidad, éstos deben ser medibles, y pueden ser conocidos o no. Dentro de este enunciado, la calidad es un estado binario, esto quiere decir que un producto posee calidad o no.

Los requerimientos pueden ser complejos o simples, siempre y cuando sean medibles, puede determinarse en donde los requerimientos de calidad se han cumplido. Esta es la vista de calidad del productor, a través del cumplimiento de las especificaciones y requerimientos. Cumplir con los requerimientos se vuelve un fin en sí mismo [4].

### 2.2 Modelos de Calidad

Los modelos de calidad de software, se definen de la siguiente manera [6]:

"La examinación sistemática de la capacidad del software que llene requerimientos específicos de calidad". Así mismo, un modelo de calidad es una buena herramienta para evaluar la calidad de un producto de software. Firesmith [7] lo define de una manera más concisa:

"Es un modelo jerárquico (colección por capas de abstracciones relacionadas o simplificadas) para formalizar el concepto de calidad de un sistema en términos de:"

Factores de calidad.- Atributos de un sistema que capturan los mejores aspectos de su calidad (usabilidad, interoperabilidad, seguridad, confianza, etc).

Subfactores de Calidad.- Componente mayor de un factor de calidad que captura un aspecto subordinado de la calidad de un sistema (respuesta, tiempo, etc.).

Criterios de Calidad.- Descripción específica de un sistema que proporciona evidencia a favor o en contra de la existencia de un factor específico de calidad.

Medidas de Calidad.- Medidas que califican los atributos de calidad y los hacen medibles, objetivos e inambiguos (Precisión = número de resultados adecuados / número total de resultados regresados).

### 2.2.1 Modelo McCall

Uno de los primeros modelos de calidad fue propuesto por McCall Richards, y Walters [8]. Este modelo describe la calidad como nacida de una relación jerárquica entre los factores de calidad, criterios de calidad, y métricas de calidad. McCall propone una clasificación útil de los factores que afectan la calidad los cuales se concentran en tres puntos clave (ver Figura 1).


**Fig. 1.** Triángulo de McCall (adaptado de [9].)

El modelo establece tres áreas principales que intervienen en la calidad del software [10]:

- Revisión de la calidad del producto de software. Tiene como objetivo realizar revisiones durante el proceso de desarrollo para detectar los errores que afecten a la operación del producto.
- Transición del producto. Requiere de la definición de estándares y procedimientos que sirvan como base para el desarrollo del software.
- Calidad en la operación del producto. Requiere que el software pueda ser entendido fácilmente, que opere eficientemente y que los resultados obtenidos sean los requeridos inicialmente por el usuario.

Pese a que el modelo de calidad de McCall fue concebido a finales de los años 70s, sigue siendo en la actualidad un marco de trabajo reconocido para el establecimiento de estándares de calidad en materia de software.

### 2.2.2. ISO/IEC 9126

EL ISO/IEC es un estándar para la evaluación de software y uno de los más conocidos. Propone modelos como los artefactos que dan seguimiento a la aseguración de los factores de calidad que son de interés particular [11]. Típicamente la calidad interna del software se obtiene por medio de la revisión de documentos de especificación, modelos de revisión, o por análisis de código fuente. La calidad externa se refiere a las cualidades del software que interactúan con su entorno. En contraste la calidad en uso se refiere a la calidad percibida por un usuario final que utiliza el

software bajo un contexto específico [12]. De tal manera las métricas internas pueden utilizarse para predecir la calidad del producto final. Para modelar la calidad interna y externa ISO/IEC 9126 define el mismo marco. El estándar está basado en seis características: Funcionalidad, Confiabilidad, Usabilidad, Eficiencia, Mantenibilidad, y Portabilidad (ver Tabla 2).

**Table 1.** Caracteristicas ISO/ISEC 9126-1[14].

| CARACTERISTICAS | SUB-CARACTERISTICAS | EXPLICACIÓN |
|---|---|---|
| Funcionalidad | Conveniencia | ¿ Realiza las tareas deseadas? |
| | Exactitud | ¿Es el resultado esperado? |
| | Interoperabilidad | ¿Puede interactuar con otros sistemas? |
| | Seguridad | ¿Prevé acceso unautorizado? |
| Confiabilidad | Madurez | ¿La mayoría de los defectos se eliminaron en tiempo? |
| | Tolerancia a Fallas | ¿puede manejar errores? |
| | Recuperabilidad | ¿Puede continuar con el trabajo y restablecer los datos después de una falla? |
| Usabilidad | Entendimiento | ¿el usuario sabe cómo utilizar el sistema sin complicaciones? |
| | Facilidad de Aprendizaje | ¿el usuario aprende fácilmente a utilizar el software? |
| | Operabilidad | ¿El usuario utiliza el sistema sin esfuerzo? |
| | Atractivo | ¿La interface luce bien? |
| Eficiencia | Tiempo de Respuesta | ¿Qué tan rápido responde el sistema? |
| | Utilización de recursos | ¿Utiliza los recursos eficientemente? |
| Mantenibilidad | Análisis | ¿Las fallas pueden ser diagnosticadas? |
| | Cambio | ¿Puede ser fácilmente modificado? |
| | Estabilidad | ¿Continúa funcionando si algún cambio es realizado? |
| | Pruebas | ¿Es fácil de probar? |
| Portabilidad | Adaptabilidad | ¿Puede moverse a otros entornos? |
| | Instalación | ¿Es fácil de instalar? |
| | Conformidad | ¿Cumple con estándares de portabilidad? |
| | Remplazabilidad | ¿Pude reemplazar a otro software? |

Abran et al [13] argumentan que “Aunque se pensó, no es exhaustivo, estas series constituyen el modelo de calidad de software más extenso creado hasta la fecha”, es también un modelo simple de utilizar para aquellos no considerados como especialistas. Éste estándar incluye la visión del usuario e introduce el concepto de calidad en uso. La presente investigación se centra en el desarrollo de software con calidad, y por tanto, la importancia del usuario toma un papel importantísimo para la elaboración de software exitoso, por ello el modelo de calidad ISO/IEC 9126 es base para esta investigación.

## 3 Problemática

En la actualidad la mayoría de proyectos de software realizan tareas para implementar planes de calidad. Esto genera una forma ordenada y controlada de llevar a cabo el proceso de desarrollo de software [1]. Sin embargo, esto no determina la calidad en el producto final, debido a que tal afirmación depende del usuario final. Los sistemas de software se construyen de una manera intuitiva pensando en una generalización para el usuario, lo cual generalmente provoca que el desarrollador tienda a dejarlo de lado durante el análisis. No en pocas ocasiones el desarrollador realmente no conoce al usuario fanal y/o ha tenido contacto con ellos. Razones por las cuales, hacen una serie de suposiciones, las cuales no atienden a las necesidades reales y se ven reflejadas en el producto final. Existen aspectos propios del usuario tales como estilo de aprendizaje, herramientas favoritas, e incluso diferencias culturales, los cuales pueden impactar en la eficiencia del sistema en cuestión [15].El resultado pueden ser sistemas de información difíciles de utilizar por los usuarios. Algunos de ellos contienen textos técnicos que el usuario desconoce, en otros simplemente no se lograrán las metas que deseaba realizar al utilizar dicho sistema. Algunos problemas que acarrea el no realizar un diseño centrado en el usuario pueden ser los siguientes [15]:

- Pérdida significativa de tiempo en desarrollo, incluyendo en el sistema apartado que el usuario nunca utilizará porqué desconoce que existen o simplemente no sabe cómo hacer uso de ellos.
- Altos costos en diseño y mantenimiento ya que software mal construido requerirá futuras modificaciones que significan costos extra.
- Pérdidas económicas, debido a que los cambios efectuados en los sistemas para corregir errores sobrepasan los presupuestos establecidos al inicio.

Por lo anterior se puede decir que la calidad de un producto de software no depende únicamente del seguimiento en los procesos de desarrollo sino en el grado de satisfacción del usuario al utilizar dicho producto.

De acuerdo a Pfleeger [16] un sistema puede tener un desempeño excelente al realizar una función, pero si los usuarios no logran entender cómo utilizarlo, el sistema es un fracaso. De lo anterior reafirmarnos la importancia del aseguramiento de la calidad, de acuerdo a Lewis [4] De tal definición se puede decir que tales actividades y funciones de seguimiento no están especificadas, es decir varían según el tipo de proyecto, y personal que da seguimiento, sin embargo es importante destacar que en este seguimiento de calidad el usuario debe ser indispensable, propuesta que establece este trabajo de investigación, ya que la falta de calidad en el software (que determina el usuario final) acarrea varios inconvenientes.

Algunos ejemplos que la poca calidad en el software puede ocasionar son los siguientes [4]:

- Fallas frecuentes en el software.
- Las consecuencias de las fallas de software son inaceptables, desde lo económico a escenarios de vida.

- Frecuentemente el software no está disponible para lo que era su propósito original.
- Las mejoras del software suelen ser muy costosas.
- El costo de detectar y corregir errores es excesivo.

Por ello se han definido modelos de calidad que sirven como referencia para conocer las cualidades necesarias para alcanzar la calidad de un producto de software. Aunque los modelos de calidad definen criterios que evalúan la calidad de cierta forma, es necesario establecer métodos de evaluación cuantitativa que permitan obtener resultados de manera más objetiva, integrando tanto elementos técnicos a evaluar como involucrando al usuario en la determinación de la calidad del software [1].

Finalmente, desarrollar sistemas no es tarea simple ni barata. Asimismo, se ha reportado que el mantenimiento representa aproximadamente el 80% del costo del software [9], [18]. En consecuencia, es de vital importancia minimizar estos costos al detectar todo los posibles fallos que el sistema pueda tener. Además, Pressman [9] explica que generalmente los desarrolladores solo hacen pruebas "suaves" por lo que pueden tender a no ser exhaustivas. Creemos que el usuario final opera el sistema en condiciones normales –o extremas en no pocas ocasiones- por lo que la evaluación por ellos será más real y valiosa para obtener un producto final con calidad suficiente.

## 4 Desarrollo de la Investigación

Para llevar a cabo la presente investigación primero fue necesario definir un modelo de desarrollo de software base (ver Figura 2), para la realización del método desarrollado, dentro del cual se hace énfasis en el apartado *Pruebas de Calidad* (usuario/desarrollador) con el objetivo que al ser parte integral del proceso de desarrollo, la evaluación iterada por parte de los usuarios así como desarrollador permita alcanzar un producto final exitoso.


**Fig. 2.**Modelo de desarrollo base (adaptado de [17]).

Posteriormente en base a los modelos de calidad analizados (McCall e ISO 9126), se estableció que factores de calidad (ver Figura 3) deben evaluarse desde el punto de vista del usuario final, y cuáles del lado del desarrollador, así como la forma de medirlos y en qué términos.

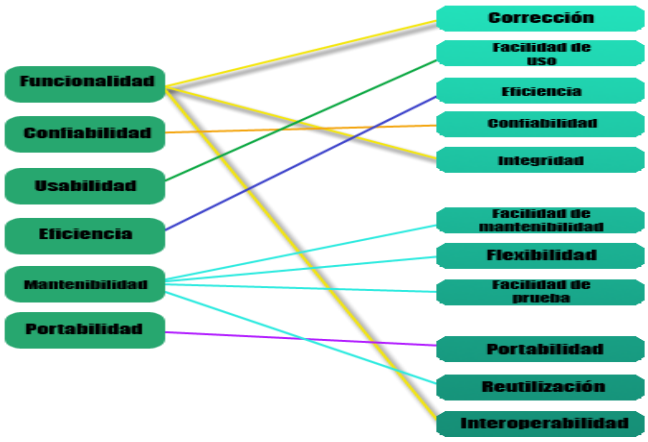


**Fig. 3.** Equivalencias McCall e ISO/IEC 9126 [1].

**Tabla 2.** Factores de calidad que se miden en la investigación

| ASPECTOS | ATRIBUTO | SUB ATRIBUTO | COMO SE MIDE |
|---|---|---|---|
| Operación del Producto | Confiabilidad | Madurez | USUARIO |
| | | Tolerancia a fallas | |
| | | Recuperabilidad | |
| | Usabilidad | Comprensibilidad | USUARIO |
| | | Facilidad de aprendizaje | |
| | | Operabilidad | |
| | | atractivo | |
| | Eficiencia | Tiempo de respuesta | USUARIO |
| | Funcionalidad | Conformidad | USUARIO |
| | | Exactitud | |
| | | Seguridad (No Aplica) | |
| Transición del Producto | Portabilidad | Adaptabilidad | USUARIO |
| | | Conformidad | DESARROLLADOR |
| | Mantenibilidad | Facilidad de análisis | DESARROLLADOR |
| | | Facilidad de cambio | |
| Revisión del Producto | Mantenibilidad | Facilidad de prueba | DESARROLLADOR |
| | | Estabilidad | |

### 4.1 Prueba Piloto

Una vez definidos los criterios, es necesario diseñar un instrumento de evaluación para medir los factores por parte del Usuario Final, para tal efecto se creó un cuestionario, que consiste de 7 preguntas demográficas y 34 preguntas sobre los factores de calidad deseados, utilizando la escala de Likert. Se utilizaron como base cuestionarios tales como: Questionnaire for User Interface Satisfaction (QUIS) [19], Computer System Usability Questionnaire (CSUQ) [19], System Usability Scale (SUS) [20]. Una vez terminado se realizó una prueba piloto para evaluar la efectividad de la herramienta creada por medio de la prueba de alfa de Cronbach que arrojó los siguientes resultados:

**Table 3.** Resultados prueba piloto.

| FACTOR | ALFA DE CRONBACH | MEDIA | VARIANZA |
|---|---|---|---|
| Operabilidad | 0.887 | 2.44 | 1.33 |
| Facilidad de Aprendizaje | 0.847 | 2.70 | 1.34 |
| Atractividad | 0.842 | 2.19 | 1.08 |
| Comprensibilidad | 0.804 | 2.45 | 1.60 |
| Funcionalidad | 0.810 | 2.51 | 1.60 |
| Exactitud | 0.905 | 2.38 | 1.15 |
| Madurez | 0.878 | 3.20 | 1.80 |
| Tolerancia a Fallos | 0.843 | 3.28 | 1.82 |
| Recuperabilidad | 0.947 | 3.18 | 1.80 |
| Tiempo de Respuesta | 0.868 | 2.51 | 1.44 |

Por lo tanto, debido a que los valores de alfa de Cronbach están por arriba de 0.8 se puede decir que la herramienta posee una consistencia interna adecuada, además al observar el histograma de cada una de las preguntas se aprecia que se cuenta con una distribución normal.

### 4.2 Métricas para el lado del desarrollador

Para las pruebas por parte de los desarrolladores se definieron métricas para medir los factores necesarios, a continuación se muestra como fueron definidas (ver tabla 5).

**Table 4.** metricas para medir el sistema del lado del desarrollador.

| ATRIBUTO | SUB ATRIBUTO | METRICA |
|---|---|---|
| Mantenimiento | Facilidad de Análisis | $\alpha = \frac{\sum n.v.}{T.p.}$ <br> $\beta = \frac{\sum e.c.}{T.p.}$ |

| | | |
|---|---|---|
| | | $\gamma = \frac{\sum c.c.}{T.p.}$<br>$\delta = \frac{\sum n.c.}{T.p.}$<br>$f.a. = \frac{\alpha + \beta + \gamma + \delta}{4}$<br>***Donde:***<br>Promedio Nombres de variables y métodos.<br>Promedio Estructura del Comentario.<br>Promedio Calidad del Comentario.<br>Promedio Número de comentarios.<br>***T.p.***= Total Programadores.<br>***f.a.*** = Facilidad de análisis. |
| | Facilidad de Cambio | El índice de madurez de software (IMS) :<br>***IMS* = [Mt – (Fa + Fc + Fd)]/ Mt**<br>***donde:***<br>***Mt*** = número de módulos en la versión actual.<br>***Fc*** = número de módulos en la versión actual que se han cambiado.<br>***Fa*** = número de módulos en la versión actual que se han añadido.<br>***Fd*** = número de módulos de la versión anterior que se han borrado en la versión actual. |
| | F. de prueba | Complejidad Ciclomática:<br>**CC = A - N + 2C**<br>***donde:***<br>***CC*** = *La complejidad ciclomática*<br>***A*** = *El número de aristas del grafo*<br>***N*** = *El número de nodos en el grafo*<br>***C*** = *El número de componentes conectados*<br>Si la función posee múltiples puntos de salida, entonces:<br>***CC = A - N + C + R***<br>R= número de declaraciones de salida. |
| | Estabilidad | Eficacia de la Eliminación de Defectos(EED):<br>**EED = E / (E + D)**<br>***donde:***<br>***E****= es el número de errores encontrados antes de la entrega del software*<br>***D****= es el número de defectos encontrados después de la entrega* |
| Portabilidad | Conformidad | ***Cdev(req,e1)* =**<br>***Cdes(req) + Ccod(req,e1) + Ctd(req,e1) + Cdoc(req,e1).***<br>**Donde:**<br>***Cdev(req,e1)****= Costo desarrollar software*<br>***Cdes(req)****= costo análisis y diseño*<br>***Ccod(req,e1)****=costo codificación*<br>***Ctd(req,e1)****= costo pruebas*<br>***Cdoc(req,e1)****=* costo documentación<br>***Cdevp(req,e1) = Cdev(req,e1) + Cpa(req)***<br>***Donde:***<br>***Cdevp(req,e1)****= costo rediseño.*<br>***Cpa(req)****= diseño portable*<br>***Cport(su,e2) = Cmod(su,e2) + Cptd(req,e2) + Cpdoc(req,e2).***<br>**Donde:**<br>***Cport(su,e2)****=portabilidad del software en nuevos ambientes*<br>***Cmod(su,e2)****= modificación manual* |

| | | |
|---|---|---|
| | | ***Cptd(req,e2)=*** *pruebas*<br>***Cpdoc(req,e2)=***documentación<br>***DP(su) = 1 - (Cport(su,e2) / Crdev(req,e2)).***<br>**Donde:**<br>***DP(su)=****conformidad en la portabilidad* |

### 4.3 Perfil del usuario

Para realizar el experimento se desarrolló un sistema web orientado al público en general, por esto debido a que es web las características del usuario serán acorde al perfil de los Internautas en México, las características relevantes para definir el perfil del usuario se toman en base los siguientes datos:

- **Edad**

La población de usuarios que hacen uso de Internet en México está conformada principalmente por Jóvenes, de acuerdo con cifras de INEGI [21] 77.3% de los Internautas Mexicanos tiene menos de 35 años; AMIPCI [22] establece que 61% de los jóvenes entre 20 y 24 años hacen uso del ciberespacio.

- **Escolaridad**

Conforme a un estudio realizado por INEGI [21] 7 de cada 10 jóvenes que cursan estudios de nivel superior hacen uso frecuente de la web

### 4.4 Muestra

Por lo anterior se realiza el estudio con estudiantes de licenciatura de la Universidad Autónoma de Aguascalientes, de la carrera de Ingeniería en Sistemas Computacionales, ya que al ser relacionadas al uso de tecnologías de la información hacen mayor uso de Internet, factor clave para la investigación. Por ello los grupos seleccionados de usuarios cumplen con las características antes mencionadas de forma adecuada.

### 4.5 Análisis de Datos

Con el fin de evaluar la calidad de cada aspecto y poder tomar decisiones, se considero como Excelente un valor promedio entre 1 y 1.5, 1.5 a 2 como muy buena, 2 a 2.5 mínima aceptable, y superior a 2.5 mala. [23]

En cada iteración se obtiene una evaluación adecuada del producto y en base a esto se realizan los ajustes necesarios al sistema que se evalúa. Ello permitirá mejorar en lo subsecuente hasta lograr el nivel indicado para su liberación. Para la evaluación de los factores se utiliza estadística descriptiva, y posteriormente un análisis de correlación entre los datos obtenidos.

## 5 Conclusiones

Aunque existen varios trabajos en nuestro país referentes a la calidad en los sistemas, pocos de ellos han intentando hacer una evaluación sistemática y donde el usuario final sea el principal crítico del producto de software entregado. Creemos que de esta forma se están atacado una serie de aspectos relevantes relacionados con la adopción de tecnología, tales como: reducir la resistencia al cambio, elevar la calidad, lo cual reduce costos de mantenimiento, entre otros. Adicionalmente, este trabajo se centra en la evaluación del producto, ya que otros se han centrado en la evaluación del proceso donde se sugiere usar Moprosoft, CMMI, entre otros. Esto no quiere decir que son estrategias rivales, más bien son estrategias complementarias para que al final los productos de software sean producidos bajo estándares y con la calidad esperada del usuario final.

Es importante recalcar que la calidad en el proceso de desarrollo de un producto de software no garantiza la calidad del producto final. Sommerville [18] argumenta que una suposición subyacente de la gestión de la calidad, es que la calidad del proceso de desarrollo afecta directamente a la calidad de los productos derivados.

### Referencias


1 A. Castañeda, et al.,"Método de Evaluación de Software Para Mejorar la Calidad del Producto Final," Universidad Autónoma de Aguascalientes, Aguascalientes, Ags.2009.

2 W. A. Ward and B. Venkataraman. 1999, Some Observations on Software Quality. School of Computer and Information Sciences, University of South Alabama.

3 ISO/IEC, "Information technology – software product evaluation – quality characteristics and guidelines for their use," in Technical Report 9126, ed: International Organization for Standardization, 1990.

4 W. E. Lewis, Software Testing and Continuos Quality Improvement, 3 ed.: CRC Press, 2009.

5 R. A. Khan, et al., Software Quality Concepts and Practices: Alpha Science, 2006.

6 S. R. Kalaimagal Sivamuni. 2008, A Retrospective on Software Component Quality Models.

7 D. Firesmith. 2005. Achieving Quality Requirements with Reused Software Components: Challenges to Successful Reuse.

8 J. A. McCall, et al., "Factors in software quality," US Rome Air Development Centre Reports1977.

9 R. Pressman, Ingeniería de Software, 6ª ed.: McGraw Hill, 2005.

1 M. Calero y Coral, "Modelos de Calidad. WQM, PQM, e-Commerce, portlets," Departamento de Informática, Universidad de Castilla-La Mancha, 2005.

11 J. P. Carvallo y X. Franch. 2006, Extending the ISO/IEC 9126-1 Quality Model with Non-Technical Factors for COTS Components Selection.

12 B. Zeiss y D. Vega. 2007, Applying the ISO 9126 Quality Model to Test Specifications. Institute for Informatics.

13 A. Abran, et al., "Usability Meanings and Interpretations in ISO Standards," Software Quality Journal, 2003.

14 SO, "ISO 9126-1:2001 Software engineering-Product quality," in Part 1: Quality Model, ed: International Organization for standardization, 2001.

15 D. Macracken y R. J. Wolfe, User-Centered Website Development: Prentice Hall, 2004.

16 Pfleeger y S. Lawrence, Ingeniería de software: teoría y práctica, 1a ed., 2002.

17 J. M. Gomez-Reynoso y M. R. Brisuela-Sandoval, "Participación de los usuarios en sistemas de información," presented at the Americas Conference on Information System, Toronto, ON, Canada, 2008.
18 I. Sommerville, Ingenieria de Software, 7 ed.: Addison Wesley, 2006
19 T. S. Tullis and J. S. Stetson, "A Comparison of Questionnaires for Assessing Website Usability," 2004.
20 J. R. Lewis y J. Sauro. *The Factor Structure of the System Usability Scale.*
21 INEGI. 2009, ESTADÍSTICAS A PROPÓSITO DEL DÍA MUNDIAL DE INTERNET.
22 AMIPCI. 2009, ESTUDIO AMIPCI 2009 Sobre hábitos de los Usuarios de Internet en México.
23 J. M. Gomez-Reynoso, *et al.*, "Utilizando el Modelo de Calidad McCall y el Estándar ISO-9126 para la Evaluación de la Calidad de Sistemas por los Usuarios," presentado en Proceedings of the sixteenth Americas Conference of Informations Systems Lima, Peru, 2010.