

Propuesta de Modelo para Especificar la Calidad de Herramientas de Pruebas

María Pérez, Edumilis Méndez, Kenyer Domínguez y Luis E. Mendoza

Laboratorio de Investigación en Sistemas de Información (LISI), Departamento de Procesos y Sistemas,
Universidad Simón Bolívar, Apartado 89000, Baruta, Caracas 1080-A, Venezuela.

{ovalles, emendez, kdoming, lmendoza}@usb.ve

Resumen. Las pruebas son esenciales durante el proceso de desarrollo de software ya que permiten cumplir con actividades de aseguramiento de la calidad. Por el mismo dinamismo de las empresas desarrolladoras de software, éstas reconocen que las pruebas son necesarias pero no las aplican por diversas razones, entre las cuales prevalece el tiempo. Las pruebas representan un proceso complejo en el cual deben considerarse, además del tiempo, aspectos como confiabilidad, costo y calidad con la finalidad de verificar y validar la calidad del producto de software antes de entregarlo al cliente. El objetivo de este artículo es proponer un modelo para especificar la Calidad para Herramientas de Pruebas y presentar su aplicación a 10 herramientas de Pruebas, considerando dentro de las categorías a evaluar la Funcionalidad, la Mantenibilidad y la Usabilidad. Para ello, se utilizó el Framework Metodológico Sistémico. Los resultados de esta investigación en progreso constituyen un primer paso para especificar la calidad de las herramientas de pruebas y agregan dinamismo al proceso de pruebas al proveer a las empresas desarrolladoras de software de un instrumento para soportar la decisión sobre cómo elegir una herramienta de prueba en particular.

Palabras Clave: Herramientas de Pruebas, Pruebas de software, Calidad del Software, Modelo de Calidad, Evaluación de Herramientas de Pruebas.

1 Introducción

Las pruebas son un proceso que intenta proporcionar confianza en el software [12,13] tanto desde el punto de vista de los desarrolladores del sistema como del punto de vista de los clientes, ya que el software debe satisfacer tanto los requerimientos funcionales como los no funcionales para su uso operacional o puesta en producción. Es decir, la confiabilidad de las pruebas impacta la confiabilidad del producto de software. Por esta razón se debe garantizar la calidad mínima esperada por el cliente según los criterios de aceptación que hayan sido previamente acordados.

Un primer paso es garantizar la funcionalidad del producto de software. En este sentido, se propuso en un trabajo anterior un método que especifica Casos de Pruebas a partir de Casos de Uso, denominado *Método para Especificar Casos de Prueba* (MECAP) [6], que permite hacer un chequeo de verificación de trazabilidad, consistencia, completitud y correctitud del producto de software y del proceso de pruebas [7].

Un segundo paso es evaluar la Calidad de herramientas que apoyan las pruebas de software. Motivado a ello se propone un Modelo para especificar la Calidad de Herramientas de Pruebas que permita evaluar este tipo de herramientas de acuerdo a tres categorías de calidad: *Funcionalidad*, *Mantenibilidad* y *Usabilidad*. La herramienta con mejor nivel será la base para la realización del siguiente paso.

Un tercer paso es automatizar MECAP a través de una herramienta que permita agregar eficiencia al proceso de las pruebas. Este aspecto escapa del alcance de este artículo ya que se encuentra en

desarrollo.

En resumen, este artículo presenta el resultado de dos objetivos específicos:

- (a) Proponer un modelo para especificar la Calidad para herramientas de Pruebas.
- (b) Evaluar un conjunto de herramientas de pruebas y estimar su calidad con el propósito de aplicar y comprobar la efectividad el modelo propuesto.

Esta investigación en progreso utilizó el Framework Metodológico Sistémico (FMS) del Laboratorio de Investigación en Sistemas de Información (LISI) [10], el cual se basa en el Método Investigación Acción [3] y la Metodología DESMET [5]. El FMS contempla iterar n veces, si es necesario, para obtener el producto de investigación buscado, pero para efecto de esta investigación se consideró una sola iteración. Adicionalmente, se empleó el enfoque Goal Question Metric (GQM) [2] para la operacionalización del modelo propuesto.

La estructura del artículo con el fin de mostrar los resultados asociados a cada objetivo específico es la siguiente: en la segunda sección se presenta el modelo propuesto para especificar la Calidad para Herramientas de Pruebas; seguidamente se presenta la evaluación de un conjunto de herramientas de pruebas; después se discuten los resultados obtenidos de la aplicación del modelo; y, finalmente, se presentan las conclusiones y el trabajo futuro.

2 Modelo para Especificar la Calidad de Herramientas de Pruebas

El modelo propuesto para la Especificación de la Calidad en herramientas de Pruebas está fundamentado en el MOdelo Sistémico de la CALidad (MOSCA) [9]. Este modelo está integrado por tres perspectivas: Producto, Proceso y Humana. MOSCA consiste de 4 niveles:

Nivel 0 – Dimensiones: incluye aspectos internos y contextuales del producto, proceso y humana.

Nivel 1 – Categorías: presenta 14 categorías de las cuales 6 están relacionadas con el producto, 5 relacionadas con el proceso y 3 con la parte humana.

Nivel 2 – Características: este nivel corresponde a un conjunto de características que definen las áreas claves a ser satisfechas para alcanzar la calidad del producto, del proceso, o humana.

Nivel 3 - Métricas: por cada característica, se proponen un conjunto de métricas para medir la calidad sistémica.

Para efectos de esta investigación, se trabajó con la perspectiva Producto, la cual se basa en el estándar ISO/IEC 9126 [4]. Para llevar a cabo la adecuación se utilizó la guía de adaptación descrita en [11] y el algoritmo para evaluar la calidad del software a través de MOSCA propuesto en [8].

De las 6 categorías que establece MOSCA: Funcionalidad, Eficiencia, Mantenibilidad, Confiabilidad, Portabilidad y Usabilidad, se deben seleccionar 3 categorías para la estimación de la calidad. La Funcionalidad es una categoría obligatoria y sus características deben superar el 75% de satisfacción para considerarla ‘aceptada’ y continuar con la evaluación de las restantes categorías [8]. Las otras 2 categorías consideradas pertinentes para evaluar la calidad de las herramientas de pruebas y adaptar el modelo MOSCA fueron Mantenibilidad y Usabilidad; para estas categorías se mantiene el mismo criterio de aceptación para considerar que está presente. Es importante resaltar que se ratificó la selección de las categorías Mantenibilidad y Usabilidad de acuerdo a otras investigaciones realizadas sobre herramientas en software libre [1]. A continuación se definen cada una de estas categorías, y se mencionan las características que se corresponden para cada una de ellas y que permitieron la adecuación de MOSCA para las herramientas de Pruebas.

La *Funcionalidad* es la capacidad del producto de software para proveer funciones que cumplan con necesidades específicas o implícitas cuando el software es utilizado bajo condiciones específicas. Sus características son [4, 9]: Ajuste a los propósitos, Precisión, Interoperabilidad, Correctitud y Encapsulado.

Mantenibilidad, según [4, 9] es la capacidad del producto de software para ser modificado; las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software ante cambios del ambiente o de requerimientos y especificaciones funcionales. La herramienta debe cumplir con esta categoría ya que es la que va a permitir implementar mejoras, de ser necesarias. Se tomaron en cuenta

las siguientes características: Capacidad de Análisis, Capacidad del Cambio, Estabilidad, Acoplamiento, Cohesión y Atributos de madurez del software.

Usabilidad es la capacidad del producto de software para ser atractivo, entendido, aprendido y utilizado por el usuario bajo condiciones específicas [4, 9]. Como las herramientas de pruebas pueden ser construidas por un grupo especializado (en técnicas y/o métodos de pruebas) o por los mismos usuarios finales del sistema, se requiere de un conjunto de características que verifiquen su uso: Facilidad de comprensión, Interfaz gráfica, Operabilidad, Efectivo y Auto-descriptivo.

Una vez seleccionadas las categorías y las características correspondientes a cada una, se formularon las métricas que permiten medir sus niveles de presencia en el software, logrando así la adecuación de MOSCA para herramientas de pruebas. Las métricas que se formularon utilizando el enfoque GQM durante esta investigación fueron las asociadas a la Funcionalidad y se presentan de forma resumida en la Tabla 1. Para las categorías Mantenibilidad y Usabilidad fueron evaluadas según las características, subcaracterísticas y métricas propuestas en [1], las cuales son el resultado de una adecuación previa de MOSCA.

Tabla 1. Nuevas subcaracterísticas y métricas de *Funcionalidad*.

Característica	Subcaracterística	Pregunta	Formulación
Ajuste a los Propósitos (FUN 1)	Tipos y Niveles de Pruebas (FUN 1.1)	¿Permite realizar Pruebas de Entrega?	5 = Si, 1 = No
		¿Permite realizar Pruebas de Rendimiento?	5 = Si, 1 = No
		¿Permite realizar Pruebas de Recuperación?	5 = Si, 1 = No
		¿Permite realizar Pruebas de Seguridad?	5 = Si, 1 = No
		¿Permite realizar Pruebas de Resistencia?	5 = Si, 1 = No
		¿Permite realizar Pruebas de Sensibilidad?	5 = Si, 1 = No
		¿Permite realizar Pruebas de Desempeño?	5 = Si, 1 = No
		¿Permite realizar Pruebas de Unidad?	5 = Si, 1 = No
		¿Permite realizar Pruebas de Integración?	5 = Si, 1 = No
	Detalles de Software (FUN 1.2)	¿Permite ingresar nombre de software a probar?	5 = Si, 1 = No
		¿Permite ingresar la versión del software?	5 = Si, 1 = No
		¿Permite verificar varios lenguajes de programación?	5 = Si, 1 = No
Precisión (FUN 2)	Detalles de pruebas (FUN 2.1)	¿Permite especificar el nombre de las pruebas?	5 = Si, 1 = No
Correctitud (FUN 5)	Completo (FUN 5.1)	¿Permite realizar todos los tipos de pruebas?	5 = Si, 1 = No
		¿Permite realizar todos los niveles de pruebas?	5 = Si, 1 = No
		¿Qué nivel de realización de tipos y niveles de pruebas dispone?	1 = Básico, 3 = Medio, 5 = Alto
	Consistencia (FUN 5.2)	¿Existe consistencia en cuanto a las pruebas realizadas?	5 = Si, 1 = No
		¿Es consistente con el lenguaje de la aplicación a evaluar?	5 = Si, 1 = No
Encapsulado (FUN 7)	Taxonomía (FUN 7.1)	¿Dispone de interfaz gráfica para verificar las pruebas?	5 = Si, 1 = No
		¿Dispone de varias vistas de evaluación?	5 = Si, 1 = No
		¿Dispone gráficos comparativos de resultados deseados y esperados?	5 = Si, 1 = No

En resumen, la adecuación de MOSCA para Herramientas de Pruebas propone 3 categorías (Funcionalidad, Mantenibilidad y Usabilidad), 15 características (4 para Funcionalidad, 6 para Mantenibilidad y 5 para Usabilidad) y 83 métricas en total: 50 del modelo original [9], 11 tomadas de [1] y 22 nuevas métricas para Funcionalidad que se formularon durante esta investigación. Éstas últimas se presentan en la Tabla 1.

3 Evaluación de Herramientas de Pruebas

Para la probar el modelo propuesto y hacer una evaluación preliminar del mismo, se utilizó el Método

de Análisis de Características de DESMET [5]. Se analizaron y evaluaron 10 herramientas de pruebas, de las cuales 7 son propietarias (Checking, Performance Optimizer, Jtest 3.0, C++ Test, Insure++, QACenter, QALoad) y 3 son software libre (JUnit, CPPUnit y PHPUnit). Siguiendo las pautas del Método de Análisis de Características [5], las características analizadas para cada herramientas corresponden a las categorías, características y subcaracterísticas, cuyos valores se obtuvieron de la medición de las métricas formuladas como resultado de la adecuación de MOSCA. Los resultados de la medición se presentan en la Tabla 2.

Tabla 2. Evaluación de 10 herramientas según MOSCA para Herramientas de Pruebas.

Categoría	Característica	Subcaracterística	Herramientas									
			Checking	Performance Optimizer	Jtest 8.0	C++Test	Insure++	QACenter	QALoad	JUnit	CPPUnit	PHPUnit
Funcionalidad	Ajuste a los propósitos	Tipos y niveles de Prueba	10%	10%	10%	10%	40%	10%	10%	10%	10%	10%
		Detalles del software	67%	33%	33%	33%	67%	33%	33%	0%	0%	0%
	Precisión	Detalles de prueba	0%	0%	0%	100%	100%	0%	0%	0%	0%	0%
	Correctitud	Completo	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
		Consistencia	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	Encapsulado	Taxonomía	100%	67%	67%	67%	67%	100%	67%	0%	0%	0%
Porcentaje de satisfacción			46%	35%	35%	52%	62%	41%	35%	18%	18%	18%
Manteniabilidad	Capacidad de Análisis	Capacidad de Análisis	100%	100%	50%	50%	100%	100%	100%	50%	50%	50%
	Capacidad de Cambio	Capacidad de Cambio	80%	60%	80%	40%	80%	60%	60%	60%	60%	60%
		Licencia	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%
	Estabilidad	Estabilidad	75%	75%	75%	100%	100%	75%	100%	75%	75%	
	Acoplamiento	Acoplamiento	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	Cohesión	Cohesión	100%	100%	100%	100%	100%	100%	100%	100%	100%	
Atributos de madurez del software	Atributos de madurez del software	66%	44%	55%	66%	88%	66%	66%	33%	33%	33%	
Porcentaje de satisfacción			74%	68%	66%	65%	81%	72%	75%	74%	74%	74%
Usabilidad	Facilidad de Comprensión	Facilidad de Comprensión	80%	80%	60%	80%	100%	60%	80%	20%	60%	60%
	Interfaz Gráfica	Interfaz Gráfica	87%	100%	87%	75%	75%	87%	63%	25%	25%	38%
	Operabilidad	Operabilidad	86%	76%	57%	50%	67%	74%	50%	36%	36%	36%
	Efectivo	Efectivo	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	Auto-descriptivo	Auto-descriptivo	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	Porcentaje de satisfacción			91%	91%	81%	81%	88%	84%	79%	56%	64%
Porcentaje Total			70%	65%	61%	66%	77%	66%	63%	49%	52%	53%

El criterio de selección para las herramientas propietarias fue su disponibilidad a través de demos y documentación. La cantidad de herramientas evaluadas se vió limitado por restricciones de tiempo y personal. La evaluación fue realizada por el LISI en colaboración con otra empresa que por razones de confidencialidad se denomina SysTest. A través de la aplicación de los cuestionarios elaborados con las métricas formuladas en la adecuación de MOSCA y dirigido a desarrolladores y/o probadores a todas las herramientas seleccionadas, se obtuvieron dos logros: (a) estimar la calidad de las herramientas de pruebas seleccionadas y (b) hacer una evaluación preliminar del modelo de calidad propuesto.

Con relación al primer logro, en la Tabla 2 se muestran los resultados generales para cada categoría. Además, en la Tabla 2 se resalta por categoría una fila de porcentaje de satisfacción que muestra el promedio que alcanzó cada herramienta para cada categoría según los resultados en cada subcaracterística.

Con respecto a los resultados de las herramientas propietarias se observa que ninguna alcanzó el porcentaje mínimo aceptado (75%) para la Categoría Funcionalidad. La herramienta que obtuvo el puntaje más alto en esta categoría fue Insure++ con un 62%. Con respecto a Mantenibilidad, de las 7 herramientas: sólo 2 superaron la evaluación (Insure++ con un 81% QALoad con un 75%). En relación a la Usabilidad todas las herramientas superaron el 75% quedando en igual de condiciones con la puntuación más alta Checking y Performance Optimizer con un 91%.

Con respecto a las herramientas en software libre y que son de nuestro interés debido a (a) un proyecto que se está llevando a cabo para PyMEs (Pequeñas y Medianas Empresas) y (b) que pueden ser modificadas/adaptadas para incorporar el método MECAP: Junit, CppUnit y PHPUnit, se puede observar que las tres tuvieron la misma puntuación en Funcionalidad y Mantenibilidad, con un 18% y 74% de aceptación respectivamente. Estos porcentajes quizás obedezcan a que son de la misma familia de XUnit, las 3 estén en un framework y realizan pruebas unitarias a sus respectivos lenguajes. En cuanto a Usabilidad, PHPUnit superó levemente a las Junit y CppUnit con un 67%. Cabe destacar que ninguna de las 3 herramientas en software libre superó el 75% de aceptación como mínimo para considerar que estas tres categorías están presentes en ellas.

En resumen, en ninguna de las 10 herramientas está presente la Funcionalidad. Se consideró continuar con la evaluación de las 2 categorías restantes para complementar la evaluación del modelo instanciado en cuanto a las métricas propuestas para Mantenibilidad y Usabilidad. La Mantenibilidad está presente sólo en las herramientas propietarias Insure++ y QALoad. Por su parte, la Usabilidad está presente en todas las herramientas propietarias que fueron evaluadas. Para el caso de las herramientas de software libre, todas presentan un nivel de satisfacción del 74%, muy cercano al mínimo establecido por MOSCA. Sin embargo, los niveles de satisfacción de la Usabilidad, son los más bajos de la evaluación.

4 Discusión de Resultados

Después de haber aplicado el modelo propuesto se puede decir que las herramientas en software libre adolecen de una Usabilidad aceptable, las subcaracterísticas que deben mejorarse considerablemente en éstas son Facilidad de Comprensión, Interfaz Gráfica y Operabilidad. Estas mismas subcaracterísticas deben ser mejoradas sólo en algunas de las herramientas propietarias. En cuanto a la Mantenibilidad, todas las herramientas en software libre deben mejorar en un gran porcentaje las subcaracterísticas de Capacidad de Análisis, Capacidad de Cambio y Atributos de madurez del software. Con respecto a las propietarias y exceptuando la subcaracterística de Licencia, deben hacer mejoras menores en las mismas subcaracterísticas de Mantenibilidad. En cuanto a la Funcionalidad, exceptuando la subcaracterística de Consistencia, todas las herramientas deben hacer mejoras considerables en todas las demás subcaracterísticas (sólo Checking y QACenter obtuvieron 100% en taxonomía).

En cuanto a la evaluación preliminar del modelo de calidad (segundo logro de la aplicación), como ninguna de las 10 herramientas igualó o superó el porcentaje mínimo de aceptación en Funcionalidad, se debe hacer una revisión del modelo propuesto en todas las características y subcaracterísticas de Funcionalidad a fin de valorar su pertinencia, nivel de profundidad, escala y factibilidad. Sin embargo, se evidenció su fácil aplicación y su utilidad, como una primera aproximación, para estimar la calidad de las herramientas de prueba.

5 Conclusiones y Trabajo Futuro

En este artículo se presenta un modelo propuesto para especificar la Calidad para Herramientas de Pruebas. Se logró hacer una evaluación preliminar de su efectividad a través de la aplicación del modelo a 10 herramientas. La evaluación preliminar del modelo indica que éste es factible de aplicar,

es sencillo y que en su intento de especificar la calidad para este tipo de herramientas, pudo lograrlo en esta primera versión. Como próximos pasos se refinará este modelo a través de su aplicación en otras herramientas, y la revisión y ajuste de las métricas para Funcionalidad, dado el bajo resultado que han dado todas las herramientas a las cuáles se les aplicó el modelo. Tomando en consideración la herramienta en software libre que quede mejor evaluada con la versión mejorada del modelo, se propondrá una herramienta que automatice el método MECAP, que permita su uso por parte de PyMES y demás organizaciones desarrolladoras de software.

Agradecimientos

Esta investigación fue financiada por el Fondo Nacional de Ciencia, Tecnología e Innovación (FONACIT), Venezuela, a través del Proyecto G-2005000165. Los autores agradecen a Domenico Iovine su colaboración para la realización de este trabajo.

Referencias

1. Alfonzo, O., Domínguez, K., Rivas, L., Pérez, M., Mendoza, L., Ortega, M. Quality Measurement Model for Analysis and Design Tools based on FLOSS 19th Australian Software Engineering Conference (ASWEC 2008). Proceedings of the 19th Australian Software Engineering Conference (ASWEC 2008). Vol. 1. pp. 258 - 267, Perth, Australia, (2008)
2. Basili, V.: Software modeling and measurement: the goal/question/metric paradigm. Technical Report CS-TR-2956. University of Maryland, (1992)
3. Baskerville, R.: Investigating Information Systems with Action Research. Communications of the Association for Information Systems, 2, 19, 1-32, (1999)
4. ISO/IEC 9126-1: Software engineering - Product quality - Part 1: Quality model. First edition, (2001)
5. Kitchenham, B.: Evaluating Software Engineering Methods and Tools. Part 1: The Evaluation Context and Evaluation Methods. ACM SIGSOFT - Software Engineering Notes, 21, 1, 11- 14. (1996)
6. Méndez, E., Pérez, M., Mendoza, L.: Aplicación de un Método para Especificar Casos de Prueba de Software en la Administración Pública. PRIS 2007: Taller sobre Pruebas en Ingeniería del Software. Actas de Talleres de Ingeniería del Software y Bases de Datos (JISBD 2007), Zaragoza, España, (2007)
7. Méndez, E., Pérez, M., Mendoza, L.: Improving Software Test Strategy with a Method to Specify Test Cases (MSTC). 10th International Conference on Enterprise Information Systems (ICEIS 2008). Barcelona, España. Proceedings of the Tenth International Conference on Enterprise Information Systems (ICEIS 2008). Vol. DISI. pp. 159 – 164, (2008)
8. Mendoza, L., Pérez, M., Grimán, A., Rojas, T.: Algoritmo para la Evaluación de la Calidad Sistémica del Software Memorias de las 2das. Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC 2002) Salvador, Brasil, (2002)
9. Mendoza, L., Pérez, M., Grimán, A.: Prototipo de Modelo Sistémico de Calidad (MOSCA) del Software. Computación y Sistemas, 8, 3, 196-217, (2005)
10. Pérez, M., Grimán, A., Mendoza, L., Rojas T.: A Systemic Methodological Framework for IS Research, In Proceedings of AMCIS-10, New York, New Cork, (2004)
11. Rincón, G., Mendoza, L., Pérez, M.: Guía para la Adaptación de un Modelo Genérico de Calidad de Software. IV Jornadas Iberoamericanas en Ingeniería de Software e Ingeniería del Conocimiento - JIISIC, Madrid, España, (2004)
12. SWEBOK: Guide to the Software Engineering Body of Knowledge 2004 Version. IEEE Computer Society, (2004)
13. Utting M., Legeard B.: Practical Model-based Testing. Morgan Kaufmann and Elsevier Publisher (2007)