Facultad de Ingeniería
ISSN: 0121-1129
revista.ingenieria@uptc.edu.co
Universidad Pedagógica y Tecnológica de Colombia
Colombia

Escobar-Sánchez, Milton Eduardo; Fuertes-Díaz, Walter Marcelo
Modelo formal de pruebas funcionales de software para alcanzar el Nivel de Madurez Integrado 2
Facultad de Ingeniería, vol. 24, núm. 39, julio-diciembre, 2015, pp. 31-42
Universidad Pedagógica y Tecnológica de Colombia
Tunja, Colombia

ISSN 0121-1129
eISSN 2357-5328

Modelo formal de pruebas funcionales de software para alcanzar el Nivel de Madurez Integrado 2

A formal model for the functional test of software to achieve maturity integrated level 2

Modelo formal de provas funcionais de software para alcançar o Nível de Madureza Integrado 2

Fecha de Recepción: 10 de Noviembre de 2014
Fecha de Aceptación: 03 de Febrero de 2015

Milton Eduardo Escobar-Sánchez*
Walter Marcelo Fuertes-Díaz**

Resumen

Las aplicaciones de software son cada vez más importantes para las organizaciones debido a que permiten llevar a cabo eficientemente sus tareas primordiales; por ello es mandatorio realizar las pruebas de calidad de software. Esta investigación se enfocó en diseñar un modelo formal para desarrollar pruebas funcionales de software que permitan alcanzar el nivel de calidad 2 del Modelo de Madurez de Pruebas Integrado (TMMI). El proceso se inició con un diagnóstico situacional, aplicando la norma ISO-9001-2000; luego, se evaluaron diversos modelos de prueba de calidad de software, como el ISO/IEC 9126, el TMM, el TMMI, el Proceso de Mejoramiento de Pruebas (TPI) y el Enfoque de Gestión de Pruebas (TMAP), realizando una comparación bajo algunos criterios como año de publicación, licenciamiento, niveles, factorías y riesgos. Con esta información se diseñó el modelo propuesto, que es independiente del proceso de desarrollo de software. Concretamente, se fundamentó en el ciclo de prueba, y se compone de cuatro fases: Especificación, Planificación, Ejecución y Evaluación, en el que se contrasta en forma real el comportamiento esperado del software. Como caso de estudio y validación se aplicó este modelo a una PYME; los resultados mostraron la eficiencia del modelo y revelaron que es preciso desarrollar una cultura de calidad organizacional en esta empresa.

Palabras clave: Modelo formal de pruebas funcionales, Nivel 2 de TMMI, Proceso de desarrollo de software.

* M.Sc. Universidad de las Fuerzas Armadas ESPE (Latacunga, Ecuador). meesacobar1@espe.edu.ec.
** Ph.D. Universidad de las Fuerzas Armadas ESPE (Latacunga, Ecuador). wmfuertes@espe.edu.ec.

Abstract

Software applications are becoming increasingly important for organizations because they allow to accomplish its core tasks efficiently. Therefore, it is compulsory to test the software quality. This research focuses on designing a formal model for the development of a functional software testing, that would achieve level 2 of Test Maturity Model Integrated (TMMI). To carry out this work, we started with a situational analysis using ISO-9001-2000. Then, several software quality models as ISO/IEC 9126, were reviewed. Also the main quality standards for software testing as TMM, TMMI, Test Process Improvement (TPI), and Test Management Approach (TMAP) were compared, based on some criteria such as the year of publication, licensing, defined levels, factories and risks. With this information a proposed model which is independent of the software development process was designed. It is based on the test cycle and it has been divided of four parts: Specification, Planning, Execution, and Evaluation. To validate this model, it was applied to a SMEs as a case of study. The results show the model efficiency. Also reveal that it is necessary to develop an organizational culture of quality within the company.

Keywords: Formal model of functional tests, level 2 TMMI, software development process.

Resumo

As aplicações de software são cada vez mais importantes para as organizações devido a que permitem levar a cabo eficientemente suas tarefas primordiais; por isso é obrigatório realizar as provas de qualidade de software. Esta pesquisa se enfocou em desenhar um modelo formal para desenvolver provas funcionais de software que permitam alcançar o nível de qualidade 2 do Modelo de Madureza de Provas Integrado (TMMI). O processo se iniciou com um diagnóstico situacional, aplicando a norma ISO-9001-2000; logo, se avaliaram diversos modelos de prova de qualidade de software, como o ISO/IEC 9126, o TMM, o TMMI, o Processo de Melhoramento de Provas (TPI) e o Enfoque de Gestão de Provas (TMAP), realizando uma comparação sob alguns critérios como ano de publicação, licenciamento, níveis, fatores e riscos. Com esta informação se desenhou o modelo proposto, que é independente do processo de desenvolvimento de software. Concretamente, se fundamentou no ciclo de prova, e se compõe de quatro fases: Especificação, Planificação, Execução e Avaliação, no que se contrasta em forma real o comportamento esperado do software. Como caso de estudo e validação se aplicou este modelo a uma PYME; os resultados mostraram a eficiência do modelo e revelaram que é preciso desenvolver uma cultura de qualidade organizacional nesta empresa.

Palavras chave: Modelo formal de provas funcionais, Nível 2 de TMMI, Processo de desenvolvimento de software.

Cómo citar este artículo:
[1] M.E. Escobar-Sánchez & W.M. Fuertes-Díaz, “Modelo formal de pruebas funcionales de software para alcanzar el Nivel de Madurez Integrado 2”, Fac. Ing., vol. 24 (39), pp. 31–41, Mayo-Ago. 2015.

I. Introducción

En la actualidad, las empresas en la región, por lo general, no acostumbran a llevar procesos de prueba formales en el desarrollo de productos de software, a pesar de la existencia de un sinnúmero de modelos especializados, como el Modelo de Madurez de Pruebas Integrado (TMMI), el Proceso de Mejoramiento de Pruebas (TPI) y el Enfoque de Gestión de Pruebas (TMAP); la razón principal de esta falencia radica en que estos modelos plantean cómo mejorar el proceso de pruebas a través del cumplimiento de objetivos y prácticas específicas, sin embargo, no plantean cómo llegar a realizarlas. Como alternativa de solución, esta investigación propone un modelo para pruebas funcionales de software, con el fin de alcanzar el nivel 2 TMMI, que permita en forma sistemática mejorar la calidad de las aplicaciones finales a través del proceso de pruebas, el mismo que estará reflejado en tiempo, confiabilidad, usabilidad, pertinencia y costo.

Algunos investigadores, como Glenford Myers [1], proponen separar el desarrollo del software de la verificación y validación; desde este enfoque y de conformidad con Edwards Deming [2], una opción para cumplir esta premisa es articular el ciclo en la mejora de la Calidad Total (i.e., Planear, Hacer, Verificar y Actuar) en las pruebas de software, de la siguiente manera: en una primera etapa, se orientan las pruebas a la solución de errores; Alan Turing [3] escribe el primer artículo basado en pruebas de software. En una segunda etapa, las pruebas son orientadas a la demostración; Howden [4] publica la primera aproximación teórica de cómo diseñar métodos sistemáticos que puedan ser utilizados para construir pruebas funcionales. En una tercera etapa, las pruebas van hacia la detección; según Myers, "El proceso de ejecutar un programa con la intención de encontrar errores" [1] tiene como objetivo demostrar que si un programa falla, los datos de prueba deberían tener una alta probabilidad de detectarlo. En una cuarta etapa, la orientación de las pruebas es hacia la evaluación y prevención; la IEEE [5] publica el estándar ANSI/IEEE STD 1008-1987, cuya aplicación da como resultado una metodología conocida como "El proceso de evaluación y de pruebas sistemáticas" (STEP); así mismo, Hetzel & Hetzel [6] definieron un sistema de tareas de pruebas, productos y roles con el fin de dar consistencia y salvar costos a la hora de alcanzar los objetivos propuestos en las pruebas.

Tomando como referencia este análisis preliminar, la presente investigación se enfoca en diseñar un modelo formal para desarrollar pruebas funcionales de software que permita alcanzar el nivel de calidad 2 del Modelo de Madurez de Pruebas Integrado (TMMi). Para llevarlo a cabo se aplicó, desde una perspectiva holística, la norma ISO-9001-2000, que "especifica los requisitos para un sistema de gestión de la calidad, cuando una organización necesita demostrar su capacidad para proporcionar de forma coherente productos que satisfagan los requisitos del cliente y los reglamentarios aplicables"[8]; posteriormente, se evaluó el modelo de prueba para la calidad de software ISO/IEC 9126; luego, el TMM, el TMMI, el TPI y el TMAP, que son modelos especializado en prueba de software. Sobre esta base, se diseñó un modelo que sea independiente del proceso de desarrollo de software y que, concretamente, se fundamente en el ciclo de prueba. Como caso de estudio y validación se aplicó este modelo a una PYME; los resultados demostraron la eficiencia del modelo.

El artículo ha sido organizado de la siguiente manera: La sección 2 describe el marco teórico que sustenta esta investigación, la sección 3 presenta el diseño del modelo propuesto, la sección 4 muestra la aplicación y evaluación de resultados, la sección 5 describe los trabajos relacionados, y, finalmente, en la sección 6 se establecen las conclusiones sobre la base de los resultados obtenidos y se describe el trabajo futuro.

II. Marco teórico

En esta sección se proporciona una breve descripción de los modelos que apalancan el marco conceptual de esta investigación, representado en la Fig. 1:

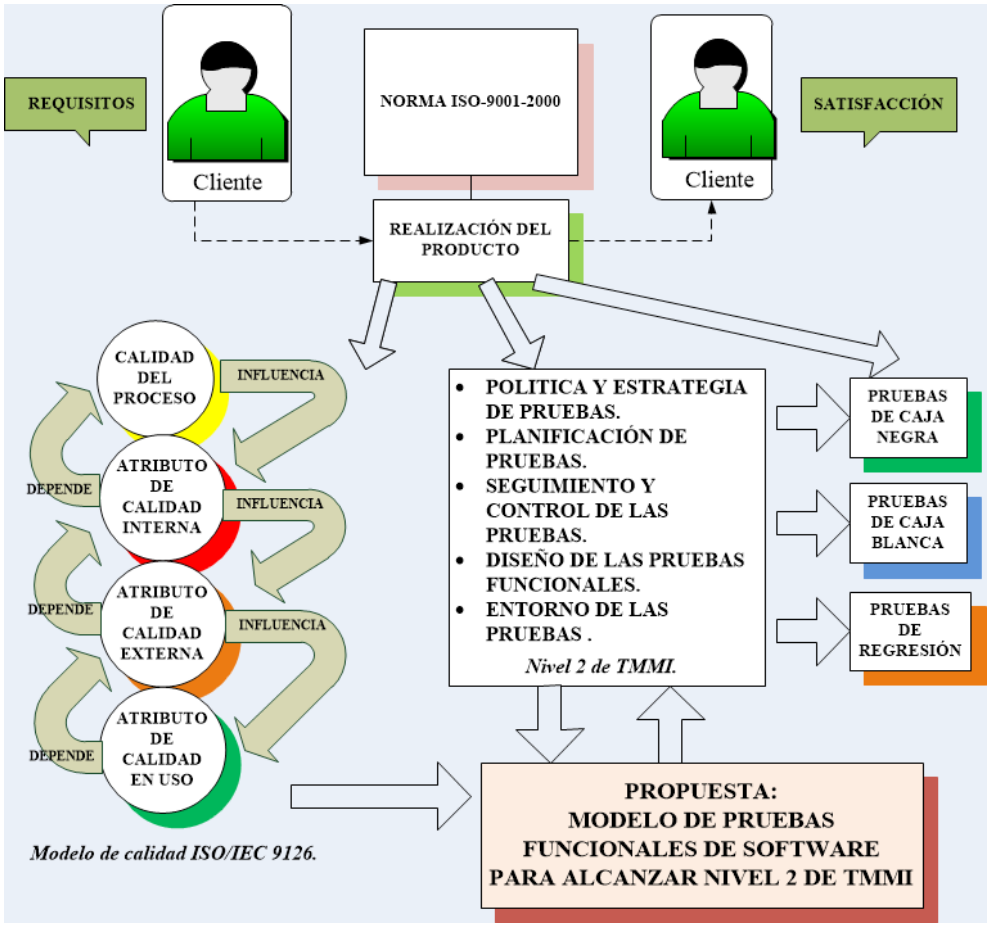


Fig. 1. Marco teórico referencial de esta investigación

De acuerdo con [8], la ISO-9001-2000 es una norma internacional que analiza el sistema de gestión de calidad en lo que respecta a la responsabilidad de la dirección, gestión de los recursos, realización del producto, medición, análisis y mejora. En este proyecto de investigación fue aplicada esta norma para realizar el diagnóstico de una PYME; posteriormente, se aplicó la norma ISO/IEC 9126, que, según Padayachee, Kotze & van Der Merwe [9], es un modelo de calidad enfocado al software tomando en consideración tres aspectos: calidad interna, calidad externa y calidad de uso; esta norma clasifica la calidad del software en un conjunto estructurado de características y subcaracterísticas como funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y transportabilidad; en esta investigación fue la base del modelo propuesto, y permitió definir los siguientes criterios: publicación, licenciamiento, niveles, factorías, características importantes y riesgos.

Continuando con la Fig. 1, el presente trabajo se enfocó en los modelos TMM y TMMI para determinar la calidad del producto; ambos tienen cinco niveles de madurez, sin embargo el último permite mejorar el proceso de prueba, por tanto, es una mejora del TMM. En el nivel 2 (planificación) las pruebas se realizan de manera formal, pues se tiene definido un proceso adecuado que permite su mejor depuración; aquí se

determinan estrategias y un plan de pruebas basado en un análisis de riesgo previo. Este documento señala cuándo, cómo y quién probará el software; además, se hace un seguimiento y control de lo planificado a fin de tomar correcciones, de ser necesario; aquí se confirma que el producto cumple con sus requisitos. Sus procesos son: política y estrategia, planificación, seguimiento y control, diseño (pruebas funcionales), ejecución y entorno de pruebas. Los procesos de verificación y validación (actividad o etapa que pertenece al proceso de construcción del software) son muy importantes (distintos el uno del otro) para determinar la calidad de un producto.

Entre los tipos de pruebas utilizados en este modelo se puede señalar: (1) La prueba estructural, que se conoce como caja blanca (código fuente del software); su objetivo es determinar los caminos del programa a ser evaluados en las pruebas; (2) La prueba funcional, conocida como caja negra (especificaciones del software); su objetivo es validar si cumple con sus especificaciones (enfoque del usuario), realizando entradas y observando sus salidas. Las pruebas funcionales utilizan la especificación del producto para diseñar los casos de prueba (entrada–salida esperada), existiendo técnicas como partición de equivalencia, análisis del valor límite, grafo causa-efecto y conjetura de errores; de acuerdo con Myers [11], la prueba funcional debe mostrar errores y disconformidades con la especificación, y no enfocarse en determinar si el programa cumple con su especificación. (3) Hay otros tipos de pruebas, denominadas de regresión, cuyo objetivo es verificar que los cambios realizados a los programas no causen nuevos defectos (alterando su calidad); pruebas de humo cuyo objetivo es verificar que cada nueva versión del programa cumple con sus funcionalidades básicas (según sus especificaciones).

Sobre la base de los antecedentes, en este modelo se articularon los estándares de calidad ISO-9001-2000 e ISO-9126 y los modelos TMMI y TMM; posteriormente, se realizó un diagnóstico del nivel de TMMI que la empresa tiene, a fin de conocer el nivel en el que se encuentra, para luego presentar una propuesta de un modelo formal de pruebas que permita alcanzar el nivel de madurez integrado 2 de TMMI.

III. Diseño de la investigación

A. Diagnóstico situacional

En virtud de que el estándar ISO-9001-2000 establece que: "La organización debe planificar y desarrollar los procesos necesarios para la realización del producto", es pertinente identificar si la planificación de la realización del producto está siendo coherente con los requisitos de los otros procesos del sistema de gestión de la calidad. En este contexto, esta investigación inició con el diagnóstico del sistema de gestión de calidad que tiene una PYME de desarrollo de software, tomando en consideración la responsabilidad de la dirección, la gestión de los recursos, la realización del producto, la medición, el análisis y la mejora. Para este propósito se utilizó la lista de chequeo basada en la ISO-9001-2000, que especifica los requisitos de un Sistema de Gestión de Calidad, y permite evaluar y demostrar su capacidad, a fin de cumplir con los requerimientos de los clientes y su satisfacción. A continuación se enumeran los resultados del diagnóstico realizado:

(1) El número de inconformidades es mayor a las conformidades. (2) No está establecido de manera formal un sistema de gestión de calidad en la organización; existe una falta de compromiso de la Dirección. (3) Hay falta de planificación para la capacitación del personal, falta de motivación, no hay evaluación continua, falta de infraestructura adecuada y se observa una escaza gestión de los recursos. (4) En la realización del producto no hay un sistema de gestión de calidad, no se han automatizado la mayoría de sus procesos. (5). No existen políticas que definan los criterios para la selección y evaluación periódica de los proveedores de hardware y software requeridos por analistas, entre otros.

B. Comparación de los modelos de pruebas de software

A fin de evaluar los modelos más importantes de procesos de pruebas de software con TMMI, se determinaron algunos criterios, llegándose a obtener los resultados que se muestran en la Tabla 1. Luego de la evaluación se obtuvieron las siguientes deducciones:

Tabla 1
Evaluación de los principales modelos de pruebas de software

Criterio/Modelo	TMM	TMMI	TPI	TMAP
Publicación	1996	2008	1998	1995
Licenciamiento	Propietario	Propietario	Propietario	Propietario
Definido	Sí	Sí	Sí	Sí
Niveles	Sí	Sí	Sí	Sí
Factorías	No	No	No	No
Riesgos	Sí	Sí	Sí	Sí

a) **Ninguno de los modelos** tiene un proceso adecuado y flexible de verificación y validación que permita utilizarse por todo tipo de empresas de software, sean pequeñas, medianas o grandes.

b) **TMMI y TMM** son modelos para mejorar los procesos de pruebas de software, tienen 5 niveles de madurez; su debilidad es la validación y verificación, dependiendo, por tanto, de la empresa para su realización. Además, para pequeñas empresas no es aconsejable por su complejidad, pudiéndose utilizar si se recortan los procesos por cumplir. Otra desventaja es que solo se tienen directrices de su utilización, debiendo las empresas definir los aspectos de su realización.

c) **TPI** es un modelo de mejora del proceso de pruebas, mediante acciones, actividades y tareas; pero no es un mecanismo que permita estructurar o formar uno desde cero.

d) **TMAP** es un modelo orientado a la gestión de las pruebas a través de actividades y tareas, pero no es un modelo de procesos de pruebas de software; no es aconsejable para pequeñas o medianas empresas, debido a su complejidad y extensión.

e) **Todos los modelos** de mejoramiento del proceso de pruebas toman en consideración la gestión de riesgos.

C. Estructura del modelo de pruebas funcionales propuesto

El modelo de pruebas propuesto en la Fig. 2 se compone de cuatro fases, cada una de las cuales tiene su propio objetivo: (1) Especificación, (2) Planificación, (3) Ejecución y (4) Evaluación. A continuación se describen cada una de las partes definidas en el modelo de pruebas propuesto:

a) **Especificación:** El objetivo es analizar las funcionalidades más importantes del software que se va a probar, con el fin de determinar el alcance de las pruebas y, así, establecer un cronograma de los ciclos correspondientes. Luego, si el cliente está de acuerdo, se procede con la planificación; caso contrario, se vuelve a realizar la especificación, donde se analizan los inconvenientes encontrados por el cliente, lo cual se documenta y guarda para tener una referencia para futuros proyectos.

b) **Planificación:** El objetivo es planificar y diseñar las pruebas, ya aceptadas por el usuario. Se establecen los ciclos, las funcionalidades y el análisis de riesgo; en esta parte se genera un plan de pruebas.

c) **Ejecución:** El objetivo es realizar las pruebas de una versión del software, relacionándolas con el ciclo correspondiente.

d) **Evaluación y resultados:** El objetivo es determinar el informe final sobre las pruebas realizadas, la satisfacción del usuario con respecto al software probado y al proceso utilizado, a fin de realizar una mejora continua. Cabe destacar que esta información es almacenada, a manera de histórico, para pruebas futuras.

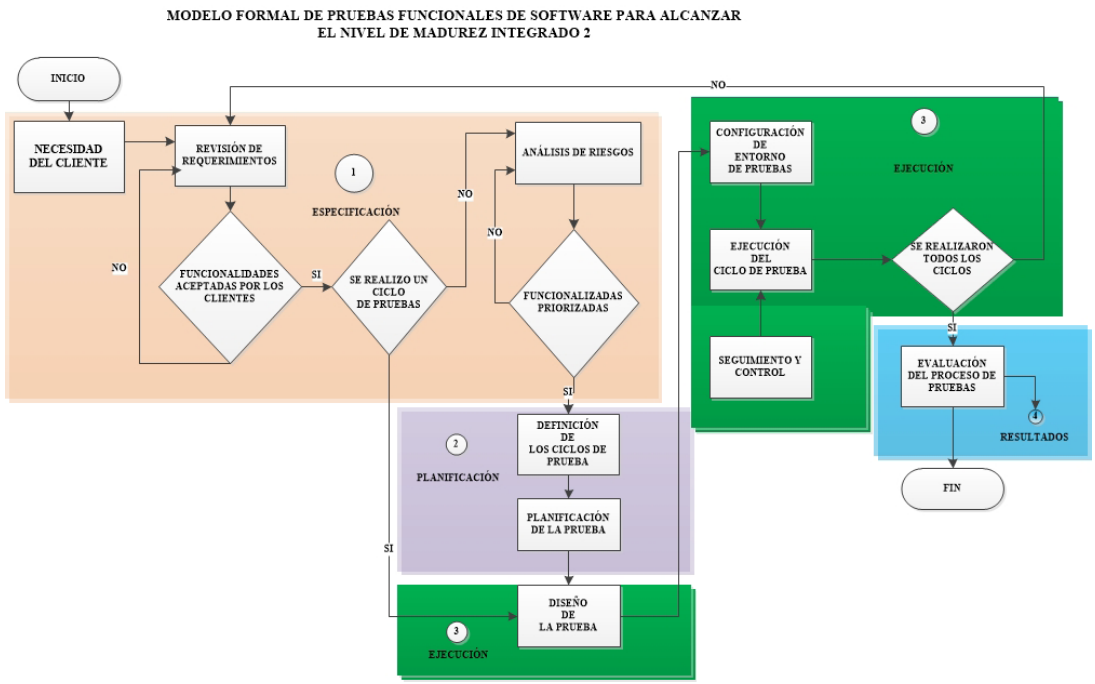


Fig. 2. Modelo propuesto para Pruebas Funcionales de Software

IV. Aplicación del modelo y evaluación de resultados

Tal como lo ilustra la Fig. 2, se cumplió con la Especificación, lo que permitió analizar las funcionalidades más importantes del software a probar, determinándose el alcance de las pruebas y un cronograma de los ciclos correspondientes. Luego que el cliente estuvo de acuerdo, se procedió con la Planificación, donde se analizaron los inconvenientes encontrados por el cliente, al planificar y diseñar las pruebas; con este ajuste se procedió con la Ejecución y la evaluación de una versión del software; finalmente, se obtuvo el informe final sobre las pruebas realizadas. Las pruebas se desarrollaron en un sistema modular de administración educativa, y como resultado se evidenció la satisfacción del usuario con respecto al software probado y al proceso utilizado. Esta información será almacenada y utilizada en pruebas futuras, además, permitirá realizar una mejora continua. A continuación una breve descripción:

A. Especificación de la prueba

En la Tabla 2 se determinó el alcance de la especificación de la prueba, acordando con el cliente 20 funcionalidades de las 40 especificadas al inicio.

Tabla 2
Alcance de la especificación de la prueba

Prioridad	Complejidad	Criticidad	Número de funcionalidades
Alta	Alta	Alta	5
Alta	Alta	Alta	5
Alta	Media	Media	5
Media	Media	Media	5
		Total	20

B. Planificación

En la Tabla 3 se muestra una comparación entre la planificación del proyecto y la realización de los ciclos de prueba; el cliente estuvo de acuerdo con lo realizado en el ciclo de pruebas 2 y con lo verificado en el plan de pruebas. Se evaluó en una reunión la satisfacción del cliente. En la Tabla 4 se observa una parte de la encuesta realizada; se establecieron como parámetros: 1 malo y 5 excelente.

Tabla 3

Planificación inicial versus planificación real de las pruebas

FASE -SEMANA	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Planificación de las Pruebas																		
Estudio Preliminar																		
Planificacion del Proyecto																		
Ciclo Prueba 1																		
Ciclo Prueba 2																		
Evaluación																		
Planificación Real de las Pruebas																		
Estudio Preliminar																		
Planificación del Proyecto																		
Ciclo Prueba 1																		
Ciclo Prueba 2																		
Evaluación																		

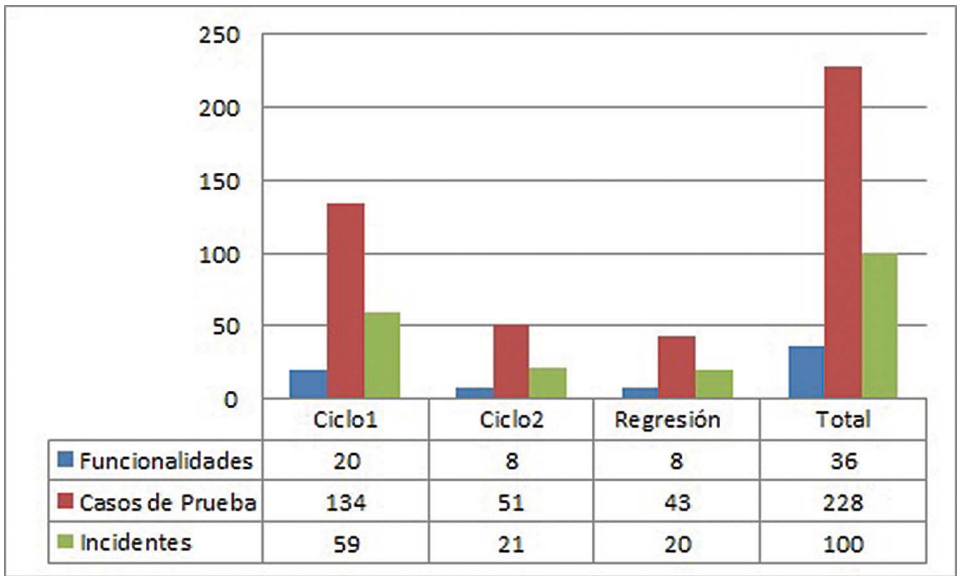
Tabla 4

Encuesta de satisfacción del cliente

Cree usted que fueron suficientes los ciclos de pruebas realizados?	1	2	3	4	5
Se sintió satisfecho con las reuniones realizadas con el equipo de pruebas ?	1	2	3	4	5
Se sintió satisfecho con su participacion en las pruebas realizadas?	1	2	3	4	5
Se siente satisfecho con el producto software obtenido ?	1	2	3	4	5
Cree que la priorizacion de funcionalidades del producto probado es adecuado	1	2	3	4	5

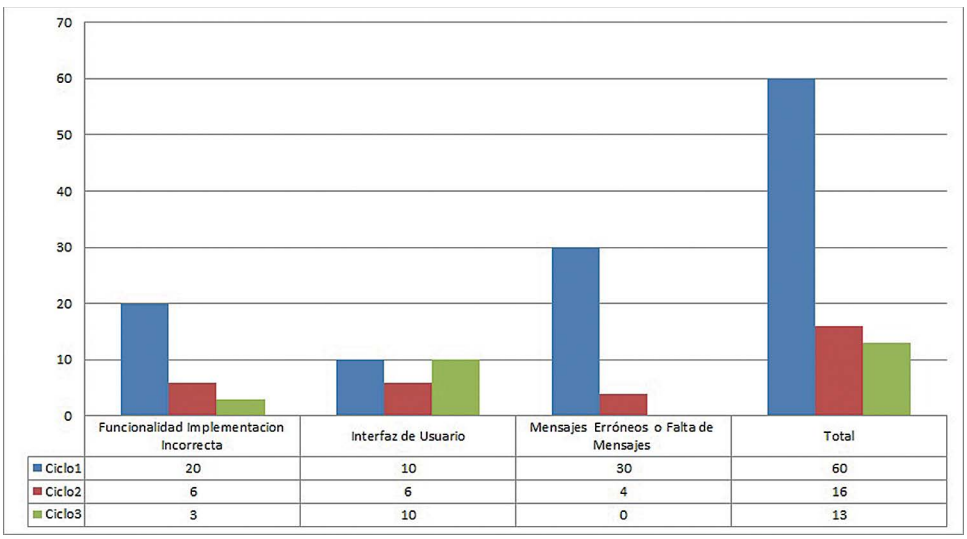
C. Ejecución

En las Figuras 3 y 4 se observan las funcionalidades que se probaron, los casos de pruebas ejecutados y los incidentes que se encontraron en cada ciclo; además, se muestran las pruebas de regresión realizadas en la evaluación del software.



	Ciclo1	Ciclo2	Regresión	Total
Funcionalidades	20	8	8	36
Casos de Prueba	134	51	43	228
Incidentes	59	21	20	100

Fig. 3. Funcionalidades, casos de prueba e incidentes en cada ciclo



	Funcionalidad Implementacion Incorrecta	Interfaz de Usuario	Mensajes Erróneos o Falta de Mensajes	Total
Ciclo1	20	10	30	60
Ciclo2	6	6	4	16
Ciclo3	3	10	0	13

Fig. 4 Incidentes por tipo según su ciclo

D. Evaluación y resultados

En la Fig. 5 se determina la criticidad de los incidentes encontrados durante la ejecución de las pruebas; como se puede observar, son incidentes de baja importancia, que se reducen significativamente al realizar la segunda versión del producto.

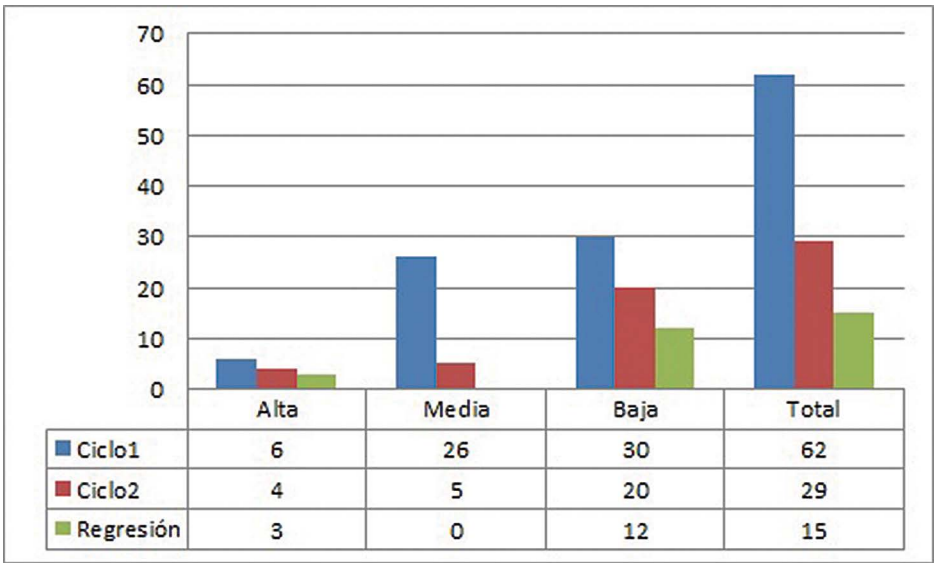


Fig. 5. Incidentes según importancia

En la Fig. 6 se muestra el porcentaje de incidentes según su ciclo de pruebas; se observa un incremento importante en el segundo ciclo, ya que el producto software probado tuvo alteraciones importantes en su funcionalidad.

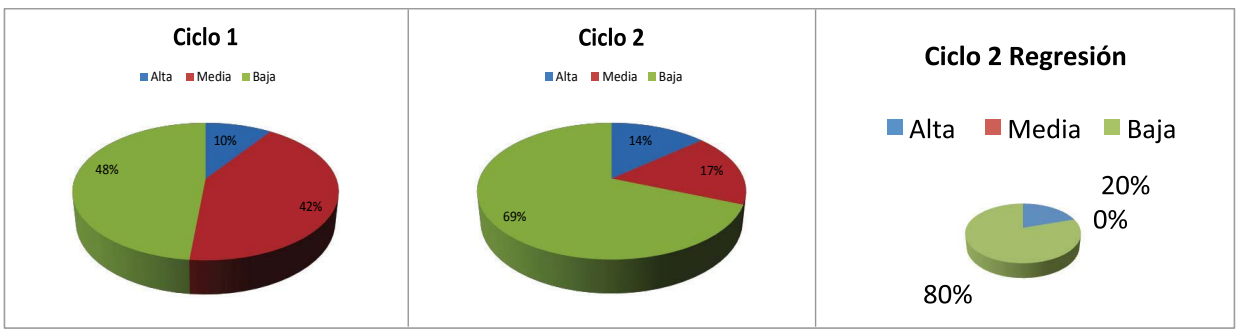


Fig. 6. Porcentaje de incidentes según la importancia

En la Fig. 7 se muestran indicadores en cada uno de los ciclos de prueba; como se puede apreciar, el modelo establecido permite identificar los casos por funcionalidad, los incidentes por funcionalidad y los incidentes por caso ejecutado.

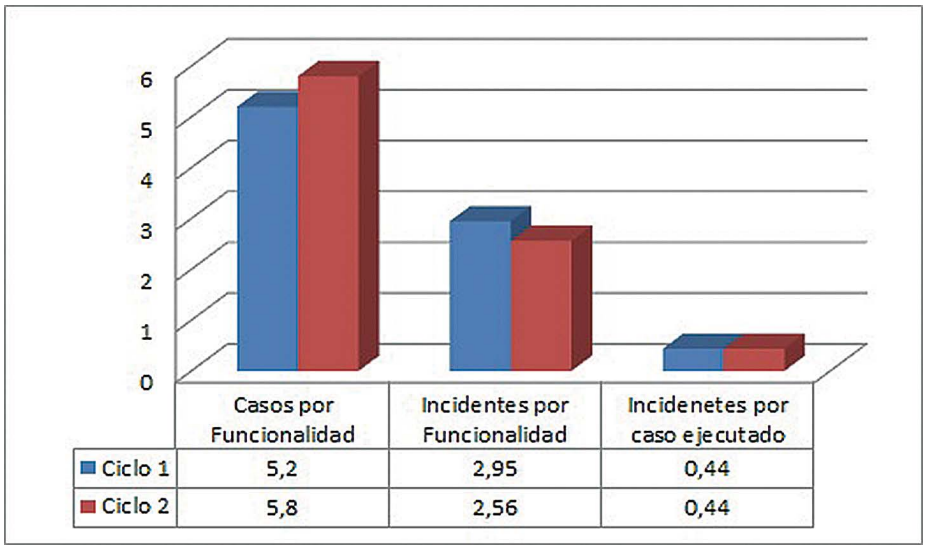


Fig. 7. Indicadores por ciclo de pruebas

V. Trabajos relacionados

Existen algunos trabajos relacionados con los principales métodos de pruebas de software TMM, TMMI, TPI y TMAP; sin embargo, existen limitados trabajos relacionados con la manera de crear un modelo que permita alcanzar el nivel 2 de TMMI. Algunos trabajos relevantes son: el trabajo propuesto por Sánchez Melchor [12], que permite analizar las ventajas y desventajas de los principales modelos de prueba de software en lo relacionado con un marco metodológico para la mejora de las actividades de verificación y validación de productos de software; el trabajo de Esteban [13], que permite comprender la mejora del proceso de pruebas de software y cómo esto repercute en la calidad de los productos, y el trabajo propuesto por Swebok [14], que ayuda a tener una visión adecuada de la Ingeniería de Software en los proyectos de desarrollo y sus procesos. En lo que se refiere a técnicas y tácticas para la comprobación exitosa y eficiente de las pruebas de software, en el trabajo de Myers [11] se establecen principios y estrategias de pruebas básicas, así como su planificación y control; Zamora & Hernández [15] establecen una guía de buenas prácticas para realizar las pruebas de software, tomando en consideración modelos de calidad, modelos de mejora del proceso de pruebas, las estructuras organizativas, las competencias y los perfiles profesionales.

Todos estos trabajos han sido insumo para el diseño del presente modelo formal que permita alcanzar el nivel 2 de TMMI, el cual resuelve algunas falencias de los anteriores, y que es independiente del proceso de desarrollo de software.

VI. Conclusiones y trabajo futuro

La presente investigación se enfocó en proponer y aplicar un modelo formal de pruebas de software que permita alcanzar el nivel 2 de TMMI, el cual es independiente del proceso de desarrollo de software y valida las especificaciones funcionales, evaluando los riesgos y realizando ciclos de prueba. Las actividades de la especificación permitieron analizar la necesidad del cliente; la planificación permitió establecer el alcance de las pruebas y un cronograma que duraría dos meses; el ejecutable del producto evaluado como caso de estudio permitió entender mejor la calidad del producto, lo cual redujo tiempos y costos en mejorar los requerimientos. Las pruebas de las 20 funcionalidades se realizaron en dos ciclos (1 y 2), que duraron 4 semanas, respectivamente, y la evaluación del proyecto se realizó en la última semana. Los casos de prueba se generaron utilizando las técnicas de caja negra de partición de equivalencia y valor límite. Se hizo una matriz que permita establecer la relación que existe entre funcionalidades y los casos de prueba generados.

Los resultados muestran la eficiencia del modelo, y revelan que es preciso desarrollar una cultura de calidad organizacional en esta empresa.

Como trabajo futuro se plantea aumentar al modelo propuesto el análisis de los requisitos no funcionales.

Referencias

[1] G. J. Myers. *The Art of software Testing*. New York: John Wiley & Sons. 1979.

[2] Out of the crisis. Cambridge, MA: Massachusetts Institute of Technology. Center for Advanced Engineering Study, 1986.

[3] A. M. Turing. *Computing machinery and intelligence*. Mind, 433-460, 1950.

[4] W. E. Howden. "Methodology for the generation of program test data". *IEEE Transactions on Computers*, 24(5), 554-560, 1975.

[5] "ANSI/IEEE STD 1008-1987 Standard for Software Unit Testing". Institute of Electrical and Electronic Engineers, New York, 1988.

[6] W. C. Hetzel & B. Hetzel. *The complete guide to software testing*: QED Information Sciences Wellesley, MA, 1988.

[7] J. Tuya, I. R. Román & J. D. Cosín. *Técnicas cuantitativas para la gestión en la ingeniería del software*: NetBiblo, 2007.

[8] ISO-9001 (2014), Norma Internacional ISO 9001-2000, Spanish Translation Task Group, Disponible en: http://www.ccoo.us.es/uploads/descargas/documentacion/NormaInternacionalISO9001.pdf

[9] I. Padayachee, P. Kotze & A. van Der Merwe. *ISO 9126 external systems quality characteristics, sub-characteristics and domain specific criteria for evaluating e-Learning systems*. The Southern African Computer Lecturers' Association, University of Pretoria, South Africa, 2010.

[10] M. A. Ampuero & Y. López Trujillo. "Creando un profesional con disciplina en el proceso de desarrollo de software". *Ingeniería Industrial*, 27(1), 4 pág, 2010.

[11] Myers. The art of software testing (2nd edition ed.), 2004.

[12] S. Sánchez Melchor. Una revisión y comparativa de Modelos de Procesos de Pruebas, 2010.

[13] A. S. Esteban. *Marco metodológico para la mejora de las actividades de verificación y validación de productos software.* Universidad Carlos III de Madrid, 2012.

[14] SWEBOK, Guide to the Software Engineering Body of Knowledge, 2004 Disponible en: http://www.swebok.org

[15] J. Zamora Hernández. *Análisis de los procesos de verificación y validación en las organizaciones software*, 2011.