DEPROVAL:Herramienta para la definición del proceso de validación en pequeñas organizaciones de software

DEPROVAL:*Tool for defining the validation process in small software organizations*

Jose Calvo-Manzano, Gonzalo Cuevas Agustin, Tomás San Feliu Gilabert
Dep. de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Universidad Politécnica de Madrid, UPM
Madrid, España
jacalvo@fi.upm.es, gcuevas@fi.upm.es, tsanfe@fi.upm.es

José de J. Jiménez Puello
Dep. de Informática
Universidad de Panamá, UP
Panamá, Panamá
jjimpue@hotmail.com

Resumen **— En el presesente artículo se presenta una herramienta para la definnición de los procesos de validación de software. La herramienta está implementada en base a un metamodelo de validación de software basado en el modelo CMMI. La herramienta facilita que las pequeñas organizaciones de software que presentan diferentes dificultades como es aplicar los modelos de mejora a su organización y falta de personal con conocimiento en pruebas, puedan establecer los procesos de validacion de sus productos a través de la herramienta.**

Abstract **— In this article, we present a tool for defining the software validation process. The tool is implemented based on metamodel validation software based on the CMMI model. The tool enables small software organizations that have different difficulties such as: applying the models to improve their organization, lack of resources and trained personnel and knowledge test, to establish the validation process of its products through this tool.**

I. INTRODUCCIÓN

En la actualidad la calidad es considerada como el elemento de mayor importancia y que presenta un impacto directo en cuanto al éxito de un producto software [22]. La baja calidad en el software puede ser el origen de algunos problemas como pueden ser la falta de fiabilidad y un mayor mantenimiento del producto [4]. El mercado valora, cada vez más, la calidad del software, por consiguiente, las compañías demandan la reducción de errores y sancionan los retrasos en entregas y las cancelaciones de proyectos [26]. A pesar de la importancia de las pruebas de software, muchas empresas tienen dificultad para generar el proceso de prueba [20].

En la práctica, la definición de los mecanismos que pueden adoptarse para mejorar y controlar las fases del proceso de pruebas y el orden que deben aplicarse en general, es un tarea difícil [14]. Un gran número de organizaciones reconocen que las iniciativas de mejoras de los procesos pueden resolver estos problemas [20]. Aunque el uso de buenas prácticas permite reducir el coste de pruebas fallidas o poco fiables, ya que, el personal de prueba garantiza que las pruebas funcionen de forma correcta [7].

Como mecanismo para mejorar los procesos, la industria y la academia han desarrollado modelos y estándares orientados a la mejora de procesos, mediante la aplicación de buenas prácticas, que incluyen el proceso de validación como son: el modelo de capacidad y madurez integrado para desarrollo versión 1.3 (CMMI) [30], ISO/IEC 12207 [12], ISO 9001 [11], IEEE-1012 verificación y validación [10] y el modelo de mejora de procesos del software brasileño (MPS.BR) [29]. En el caso de los modelos de mejora de procesos de pruebas tenemos: Test Process Improvement (TPI) [33], Test Process Improvement Next (TPI NEXT) [31], Test Maturity Model Integration (TMMI) [8], ISO/IEC/IEEE 29119[13] y Testing Maturity Model Enhanced (TMMe) [32]. Por otro lado, también se han desarrollado áreas de procesos de pruebas como el TestPAI [2] y el Software Testing process for the reference model of Competisoft (STPRMC) [28].

Sin embargo, una de las principales barreras es la dificultad de adaptar las pruebas a modelos de madurez para el entorno específico de una organización [1]. Por consiguiente, muchos enfoques de mejora de procesos de pruebas del software están disponibles, pero no todos son aplicables en general a la industria [3].

Por su parte, la mayoría de las empresas de desarrollo de software en todo el mundo corresponden a las pequeñas y medianas empresas [15] y [20].

Por consiguiente, las pequeñas empresas se enfrentan a varios obstáculos para implementar sus procesos de pruebas de software, debido a la falta de expertos en pruebas, la falta de recursos y las dificultades para adaptar los modelos de madurez relacionados con las pruebas a un contexto particular de la empresa [21]. Como señala [23] se reconoce que existe un problema por la baja madurez en pruebas de software en las organizaciones. A pesar de la importancia de las pruebas, en la industria del software, hay un vacío conceptual en pruebas por parte de los desarrolladores e ingenieros que deben aplicarlas [19].

Se hace evidente la necesidad de adaptar los modelos de mejoras en las organizaciones de desarrollo de software. Sobre todo, en las pequeñas empresas de desarrollo de software. Aunado a la falta de conocimiento del personal en pruebas de software. Por lo tanto, se desarrolló un metamodelo para validación de software basado en el modelo CMMI [16]. El metamodelo permite la definición del proceso de pruebas de validación, en base a un enfoque de mejora de procesos, que se adapta al contexto del proyecto a probar, como mecanismo para solventar la carencia en conocimiento en pruebas del personal de las pequeñas organizaciones de desarrollo de software.

Este artículo propone establecer una herramienta que defina el proceso de pruebas mediante un enfoque de mejora de procesos y que se adapte al contexto del proyecto a validar.

El artículo se divide en las siguientes secciones: Sección 1 ofrece una breve introducción al problema. En la sección 2 se presentan los trabajos relacionados con el problema. En la sección 3 Se presenta el metamodelo como solución al problema. En la sección 4 Se presenta la Herramienta desarrollada. En la sección 5 se presenta las principales conclusiones.. En la sección 6 Los agradecimientos. Y las referencias bibliográficas.

II. Trabajos Relacionados

En esta sección se describen diversos metamodelos, como resultado de una revisión sistemática. Los metamodelos descritos carecen del enfoque de mejora de los procesos de pruebas de software.

Un metamodelo es presentado por [9], para técnicas de validación de software en el entorno de la arquitectura dirigida por modelos. Los autores sugieren que a través de un modelo de pruebas se puede establecer técnicas de pruebas funcionales y estructurales. Incluso otros tipos pruebas como es la de mutación para mejorar la calidad de la prueba. Pero, resaltan que queda mucho por hacer hasta que no se normalice el lenguaje para el modelo de transformaciones. Por su parte en [27], establecen un metamodelo basado en generación de datos de pruebas denominados (modelos de pruebas) para modelos de transformación en el contexto de la arquitectura dirigida por modelos, como mecanismo de comprobar las transformaciones que tiene que realizar los modelos generados. Para ello, implementa un algoritmo que crea modelos de pruebas de un metamodelo. En [6] presentan un metamodelo que aplica un modelo de simulación para comprobar la definición de procesos implementados en SPEM, mediante las prácticas de validación del CMMI. En [34] proponen un metamodelo para pruebas de metamodelos, que determine la existencia de errores en los metamodelos. Para ello, se enfoca en un metamodelo de prueba automatizado basado en una especificación de prueba. En [35], establecen un metamodelo para soportar pruebas de regresión para aplicaciones Web, debido al rápido cambio en la tecnología, lo que genera que las aplicaciones que migran de forma continua no puedan validarse debido al cambio de las plataforma y de las pruebas. En [24] se propone un metamodelo para pruebas de unidad para el lenguaje de programación orientado a objetos, que facilita escribir pruebas de unidad dentro de las clases para que sean probadas. Por otra parte [5], define una metodología de pruebas para aplicaciones Web, mediante la integración de búsquedas de pruebas, a través de un metamodelo de pruebas que implementa dicha metodología.

Los metamodelos descritos no resuelven la dificultad de adaptar los modelos de mejora a las organizaciones de desarrollo de software. Además, aplican el concepto de metamodelo definido para la arquitectura dirigida por modelos [25]. Por consiguiente, el metamodelo es un soporte para crear y desarrollar modelos a partir de un modelo.

Por el contrario, en [16] se desarrolla un metamodelo que resuelve la problemática de definir la validación y de adaptar los modelos de mejora de procesos. El metamodelo consiste en una solución innovadora referente a los modelos y estándares existentes, ya que, proporciona "el cómo hacer" identificando y aplicando las prácticas de validación para la mejora del proceso de validación. El metamodelo no aplica el concepto de metamodelo de la arquitectura dirigida por modelos que trata sobre la transformación de modelos.

III. Metamodelo

La herramienta para definir el proceso de validación para para pequeñas empresas surge a partir de un metamodelo para validación de software como se describe en [16]. El metamodelo a su vez se organiza a partir de una taxonomía de proyectos de software [18] y un método para caracterización de pruebas de software [17]. Los resultados de la caracterización derivan en un plan de pruebas. El metamodelo desarrollado está basado en el modelo CMMI para desarrollo v.1.3, que al ser un metamodelo puede derivar otros modelos. El CMMI está integrado por áreas de procesos, el área de proceso seleccionada para la definición del metamodelo es el área de procesos de validación, y su meta específica preparar la validación que incluye las prácticas específicas (SP 1.1. Seleccionar los productos a validar, SP 1.2. Establecer el entorno de validación y SP 1.3 Establecer los procedimientos y criterios de validación). Estas prácticas específicas permiten organizar y establecer el proceso de preparación de la validación de software para luego ejecutar la validación.

El metamodelo basado en estas prácticas específicas, representa un marco para establecer los aspectos que debe considerar toda organización de desarrollo de software al momento de validar el software desarrollado. El metamodelo además, incluye plantillas de pruebas que se constituyen a

partir de las prácticas específicas del área de proceso de validación del CMMI, aplicadas al metamodelo.

IV. HERRAMIENTA: DEPROVAL

La heramienta desarrollada es de tipo escritorio y se implementó con Microsoft Access 2010. La herramienta es una aplicación bastante sencilla, versatil y que presenta costes bajos en cuanto a su desarrollo. La herramienta está alineada a la mejora de los procesos de validación de software del área de procesos de validación del CMMI. La misma está constituida por 8 funcionalidades que a continuación se describen.

Funcionalidades DEPROVAL

DEPROVAL esta conformado por las siguientes funciones:

- Producto: Se refiere al proyecto o componente de proyecto a probar, sus requisitos y restricciones.
- Entorno de validación: En el se describe cada elemento requerido y necesario para realizar la prueba. El entorno permite comprobar la funcionalidad de la aplicación o proyecto en un ambiente controlado, que simula el entorno real o de producción del proyecto.
- Equipo de prueba: Se identifica el personal de pruebas y sus responsabilidades en la validación de proyectos.
- Plan de Prueba: El plan reune la estrategía de prueba que será aplicada para comprobar los requisitos del proyecto.
- Calendario de programación de pruebas: Se establece el calendario de programación de cada requisito del proyecto que será validado.
- Procedimiento de validación: Describe el procedimiento para realizar la validación
- Criterios de validación: Se refiere a los aspectos que deben considerarse para la validación del proyecto.
- Procedimiento de prueba: Describe los casos de usos y de pruebas que serán aplicados para comprobar los requisitos.

1) Producto

La funcionalidad producto se refiere a la selección del proyecto a validar como lo específica la práctica específica 1.1. Seleccionar los productos a validar. El producto se selecciona a partir de la taxonomía de proyectos [18].

La taxonomía se organiza en categorías y subcategorías de proyectos software. La taxonomía está constituida por 9 categorías y 27 subcategorías. Las categorías que integran las taxonomía son: Aplicaciones de escritorio o independientes, Web, móviles, servicios, migración de sistemas, De procesos, De control de tiempo, almacenamiento y protocolos. A continuación se presenta la subcategorías con sus respectivas categorías: Escritorio: independientes, aplicaciones cliente /servidor, clientes enriquecidas, Web: páginas Web o sitios Web, Web tradicional, aplicaciones Internet enriquecidas (RIA), Web 2.0, Móvil: móvil nativa, móvil Web, Servicios: aplicación servidor, SOA, aplicaciones como servicios (SAAS), en la nube (Cloud), Grid, P2P, agentes, Middleware, Procesos SAP, De control de tiempo: sistema de tiempo real, sistemas críticos, embebidos, Legado:, Almacenamiento: Data Warehouse/BI, Big Data, Protocolos: protocolos de redes y protocolos de comunicaciones Para cada producto se debe seleccionar la categoría y subcategoría del proyecto al que pertenece el producto. Además, se identifica el alcance del proyecto y si el producto es un proyecto o componente de proyecto. Las restricciones que presenta el producto y se establecen los requisitos funcionales y no funcionales que deben comprobarse en el producto (Fig. 1) Además, la herramienta facilitará que se puedan añadir, modificar y eliminar categorías y subcategorías de proyectos.

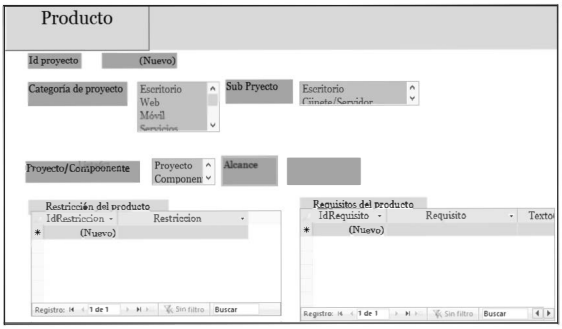


Figura 1. Producto a validar

2) Entorno de validación

El entorno de validación establece los elementos que deben considerarse para probar los requisitos del producto, según el proyecto o componente del proyecto. La herramienta permite registrar los diferentes elementos requeridos para la implentación del entorno de validación (Fig. 2).

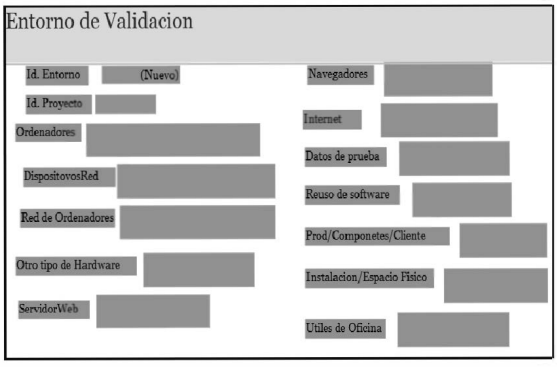
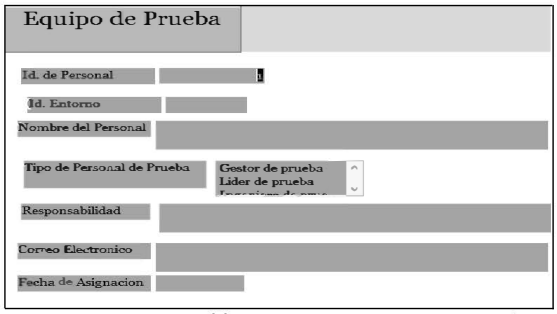


Figura 2. Entorno de validación

3) Equipo de prueba

Se refiere al personal asignado para llevar a cabo las diferentes pruebas de cada requisito del producto. Se determina y



establece la responsabilidad del personal asignado al equipo de prueba y a que entorno de prueba corresponde (Fig. 3).

Figura 3. Equipo de prueba

4) Plan de Prueba

El plan de prueba es el resultado de la aplicación de un método para caracterizar la pruebas de un proyecto software [17]. El plan establece lo siguientes elementos a cosnsiderar: Tipo de prueba, nivel de prueba, características a probar, pruebas específicas aplicables al proyecto, las técnicas de pruebas que se pueden aplicar al proyecto, las herramientas de pruebas según la prueba específica. El plan es el referente para que el equipo de prueba, tenga certeza de las pruebas que puede y deben aplicar al proyecto (producto) de forma objetiva (Fig. 4).

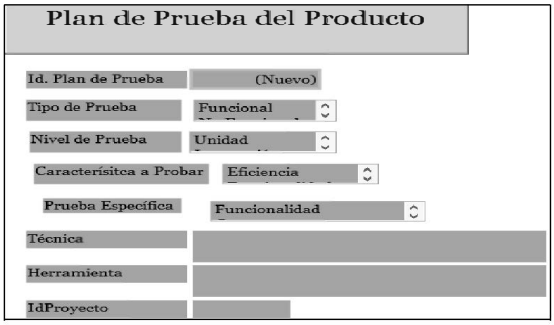


Figura 4. Plan de prueba del proyecto (producto)

5) Calendario de programación de pruebas

El calendario permite establecer la programación de las pruebas del proyecto. Para cada requisito se programa la fecha y hora estimada para la realización de las pruebas. De igual forma se programa el uso del entorno por parte del probador. Por medio del calendario se define como se desarrollaran las diferentes pruebas según los requisitos del proyecto (producto) (Fig. 5).

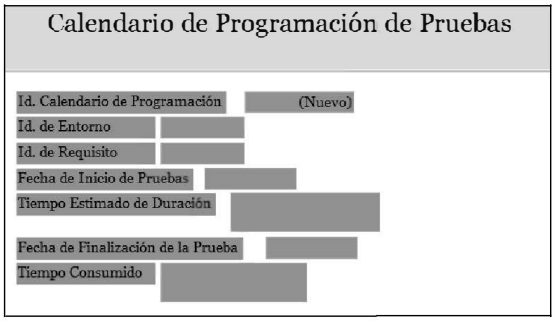


Figura 5. Calendario de programación de pruebas

6) Procedimiento de validación

El procedimiento de validación describe como se realiza la validación del proyecto (Fig. 6).

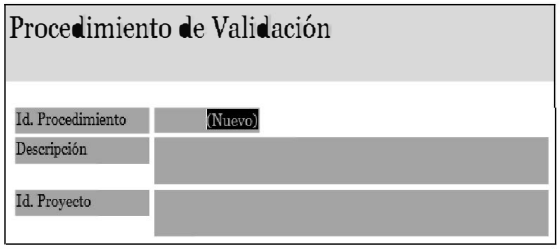


Figura 6. Procedimiento de validación

7) Criterios de validación

Los criterios permiten determinar si la prueba del requisito pasó o falló, según los datos de entrada y salida que se obtienen con la prueba. Los criterios permiten describir el fallo y si la prueba fue suspendida o reanudada. Además, si la fecha corresponde a dicho evento. El criterio está estrechamente relacionado con el procedimiento de prueba. Lo que ocurra con el caso de prueba repercute en los criterios de pruebas de validación (Fig. 7).

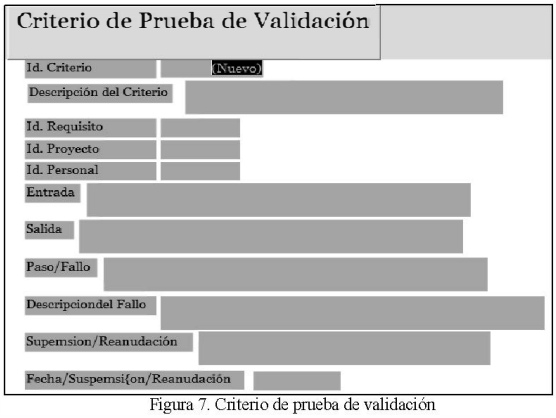


Figura 7. Criterio de prueba de validación

8) Procedimiento de prueba

El procedimiento de prueba describe los casos de usos y los casos de pruebas, que deben ser utilizados para probar los requisitos del producto. Por medio del procedimiento de prueba el probador comprueba si el requisito cumple o no con el resultado esperado (Fig. 8). Además, para los casos de pruebas el probador debe considerar el plan de pruebas y seleccionar las pruebas y técnicas que deben ser aplicadas a los casos de pruebas respectivos.

Figura 8. Procedimiento de prueba

9) Evaluación de la herramienta

Para evaluar la utilidad de la herramienta se tiene planificado aplicarla en organizaciones de desarrollo de software pequeñas. La característica principal de las organizaciones es que dispongan de poco personal. Que el personal sea a su vez desarrollador y probador. Además, que el personal tenga poco o casi nulo conocimiento en prueba de software. La evaluación permitirá determinar si los probadores consideran que la herramienta cumple con el objetivo de desarrollar el proceso de validación de prueba de software. Antes de aplicar la herramienta se tendrá que realizar una jornada formativa con el personal para que conozcan el funcionamiento de la herramienta. La evaluación por parte de los probadores se realizará mediante la técnica de encuesta a través del instrumento cuestionario. De los resultados que se obtengan se realizaran mejoras y se tiene proyectado cambiar la herramienta a un entorno Web.

V. CONCLUSIONES

Se ha desarrollado una herramienta para la definición del proceso de validación de software basado en un modelo de mejora de procesos. La herramienta está basada en un metamodelo de validación de software, basado en un modelo de mejoras de procesos como es el CMMI. La herramienta está aún en fase de desarrollo para luego ser implementada la fase experimental.

Con la herramienta se busca solucionar la problemática que tienen las pequeñas organizaciones de desarrollo de software, que presentan dificultad en adaptar y por ende, de aplicar los modelos de mejora de procesos a su organización. Sumado a la falta de conocimiento y formación del personal en pruebas de software. La herramienta es una guía para el desarrollo del proceso de validación en las pequeñas organizaciones de desarrollo de software, de forma tal, que puedan garantizar la funcionalidad y fiabilidad del producto que desarrollan.

VI. AGRADECIMIENTOS

Este trabajo es patrocinado por Everis Aeroespacial y Defensa, y la Universidad Politécnica de Madrid, a través de la Cátedra de Investigación de Mejora de Procesos de Software para España y la Región de América Latina.

REFERENCIAS BIBLIOGRÁFICAS

[1] A. Rodrigues, A. Bessa and P.R. Pinheiro, "Barriers to implement test process in small-sized companies," Organizational, Business, and Technological Aspects of the Knowledge Society. Springer Berlin Heidelberg. pp. 233-242, 2010.

[2] A. Sanz, J. Saldaña, J. García and D. Gaitero, "Test PAI: A testing process area integrated with CMMI," Proceedings of the workshops of the Conference on Software Engineering and Databases, 2008. "TestPAI: Un área de proceso de pruebas integrada con CMMI," Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, 2008.

[3] A. Wasif, A. Snehal, G. Kerstin and R. Torkar, "Software test process improvement approaches: A systematic literature review and an industrial case study," Journal of Systems and Software, vol. 111, pp. 1-33, January 2016.

[4] A.A.M. Elhag, M.A. Elshaikh, R. Mohamed, M.I Babar, "Problems and future trends of software process improvement in some Sudanese software organizations," in Computing, Electrical and Electronics Engineering (ICCEEE), International Conference on , vol., no., pp.263-268, 26-28, Aug. 2013.

[5] B. Marin, T. Vos, G. Giachetti, A. Baars and P. Tonella, "Towards testing future Web applications, Research Challenges in Information Science", pp.1—12, 2011.

[6] D. Sadilek and S. Weißleder, "Testing Metamodels", Model Driven Architecture Foundations and Applications, Lecture Notes in Computer Science, vol. 5095, pp. 294—309, 2008.

[7] E. Kerry and S. Delgado, "Applying software engineering practices to produce reliable, high-quality and accurate automated test systems," AUTOTESTCON, IEEE, vol., no., pp.69-71, Sept. 2009.

[8] E. van Veenendaal, Test Maturity Model Integration (TMMi), version 2.0, TMMi foundation, [Online], Available, http://www.tmmifoundation.org/downloads/tmmi/TMMi%20Framework.pdf

[9] F. Fleurey, J. Steel and B. Baudry, "Validation in model-driven engineering: testing model transformations, Model, Design and Validation," Proceedings. First International Workshop, pp. 29-40, 2004.

[10] IEEE, "Standard for System and Software Verification and Validation," IEEE Std 1012-2012 (Revision of IEEE Std 1012-2004), vol., no., pp.1-223, May 25 2012.

[11] ISO 9001, Quality management systems – Requirements, 2015.

[12] ISO/IEC 12207: Systems and software engineering – Software life cycle processes international organization for standardization/ international electrotechnical comission. Geneve: ISO, 2008.

[13] ISO/IEC/IEEE 29119 Software Testing, 2015. [Online], Available: http://www.softwaretestingstandard.org/news.php

[14] J. Andersin, TPI- a model for test process improvement. Seminar. University of Helsinki, Helsinki – Finland, 2004.

[15] J. Aranda S. Easterbrook and G. Wilson, "Requirements in the wild How small companies do it," 2007. Online], Available: http://www.cs.toronto.edu/~jaranda/pubs/REintheWild-RE07.pdf

[16] J. Calvo-Manzano, G. Cuevas, T. San Feliu Gilaber and J.J. Jiménez Puello, "Metamodel for software validation based on the CMMI" CIMPS'15 (4th International Conference on Software proceess Improvement) Mazatlan, Sinaloa, Mexico, 2015. "Metamodelo para

validación de software basado en el CMMI", CIMPS'15 (4th International Conference on Software Proceess Improvement) Mazatlan, Sinaloa, Mexico, 2015.

[17] J. Calvo-Manzano, G. Cuevas, T. San Feliu Gilabert, and J.J. Jiménez Puello, "Characterization of tests focused on validation of software Project," 22nd EuroAsiaSPI Conference, Turkish Standards Institute, Ankara, Turkey, 2015.

[18] J. Calvo-Manzano, G. Gonzalo Cuevas, T. San Feliu Gilabert, and J.J. Jiménez Puello, A Taxonomy for Software Testing Projects. Sistemas e Tecnologias de Informação, Atas da 10ª Conferência Ibérica de Sistemas e Tecnologias de Informação. Águeda, Portugal. AISTI | Universidade de Aveiro Vol. I Artigos Tomo 1, pp 289-294. Editores Álvaro Rocha, Arnaldo Martins, Gonçalo Paiva Dias, Luis Paulo Reis, Manuel Pérez Cota., 2015.

[19] J.A. Caicedo, J. Garces Tabares, and M.C. Camacho, "Proposed application of testing based on BS 7925-2 standard for degree projects focused on development software," in Computing Colombian Conference (9CCC), vol., no., pp. 121-127, Sept. 2014.

[20] K. G. Camargo, F. C. Ferrari and S. CPF. Fabbri, "Characterising the state of the practice in software testing through a TMMi-based process," Journal of Software Engineering Research and Development, vol. 3:7, pp. 1-24, December2015.

[21] L. M. da Rocha, Aplicação de um subset de práticas do TMMi em núcleo de práticas em informática da UFC Campus Quixadá / Laisa Morais da Rocha, Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2014

[22] M. Faizan, S. Ulhaq and M.N.A. Khan, "Defect Prevention and Process Improvement Methodology for Outsourced Software Projects," Middle East Journal of Scientific Research vol. 19 (5), pp.: 674-682, 2014.

[23] M. Grindal, J. Offutt, and J. Mellin, "On the Testing Maturity of Software Producing Organizations," in Testing: Academic and Industrial Conference - Practice And Research Techniques, Proceedings , vol., no., pp.171-180, 29-31 Aug. 2006.

[24] M. Lévesque, "A Metamodel of Unit Testing for Object-Oriented Programming Languages," 2009. http://arxiv.org/pdf/0912.3583.pdf

[25] M. Muñoz, Metamodel for creating applications in Next Generation Networks, "Doctoral Thesis, Charles III University of Madrid, Department of Telematic Engineering Doctorate in Communications Technology, Leganes, 2003 "SCMM. Metamodelo para la Creación de Aplicaciones en Redes de Siguiente Generación,"Tesis Doctoral, Universidad Carlos III De Madrid, Departamento De Ingeniería Telemática, Doctorado en Tecnologías de las Comunicaciones, Leganés, 2003.

[26] M. Piattini, F. García, I. García Rodriguez and F. Pino, Quality Information Systems, Second edition, Madrid, Rama, 2014. M. Piattini,

[27] F. García, I. García Rodriguez y F. Pino, Quality Information Systems, Second Edition, Madrid, Rome, 2014. Calidad de Sistemas de Información, Segunda edición, Madrid, Rama, 2014.

[28] N-L. Hsueh, W-H. Shen, Z-W. Yang and D-L Yang, "Applying UML and software simulation for process definition," verification, and validation. Information and Software Technology, 50, (9–10), pp. 897–911, 2008.

[29] P. Cruz, R. Villarroel, F. Mancilla and M. Visconti, "A Software Testing Process for the Reference Model of Competisoft," Chilean Computer Science Society (SCCC), 2010 XXIX International Conference of the, vol., no., pp.51-59, 15-19 Nov. 2010.

[30] Softex, "Guía General MPS de Software", MPS. BR -Mejora de Proceso del Software Brasileño, 2012, [Online], Available: http://www.softex.br/wp-content/uploads/2013/10/MPS.BR_Gu%C3%ADa_General_Software_2012.pdf

[31] Software Engineering Institute, "CMMI for Development" Version 1.3 [Online], Available: http://www.sei.cmu.edu/library/assets/whitepapers/Spanish%20Technical%20Report%20CMMI%20V%201%203

[32] Sogeti, Test Process Improvement Next, UTN Publishers, 2009.

[33] T. C. Staab, "Introduction to the Testing Maturity Model Enhanced (TMMe)," Wind Ridge International,LLC, 2010. [Online], Available: http://www.windridgeinternational.com/documents/Introduction%20to%20the%20Testing%20Maturity%20Model%20Enhance.pdf

[34] T. Koomen and M. Pol, "Test process improvement: a practical step-by-step guide to structured testing" Addison-Wesley Professional; 1 edition, June 1999.

[35] Y. Hernandez, M. King, J. Pava and P. Clarke, "A Meta-Model to Support Regression Testing of Web Applications", http://users.cis.fiu.edu/~clarkep/research/areas/REUPubs/HernandezKingPavaClarke.pdf