Automation of Software Testing: Experience and Lessons Learned

Leticia Laura-Ochoa, Mg[1]
[1]Universidad Nacional de San Agustín, Perú, llaurao@unsa.edu.pe

Abstract– ***With the aim of improving the quality, reliability and ensuring the proper functioning of the software products; This paper describes the experience of the use of open source tools for the automation of software tests such as Specflow and Selenium, which have allowed to perform functional tests and have the respective documentation of the software tests. Having a team dedicated to perform these tests in parallel to the development contributes to improving the quality of the software product and decreases the errors of the deliverables in the milestones established with the client.***
Keywords- Software testing, software quality, test tools, test automation, test cases

Digital Object Identifier (DOI):http://dx.doi.org/10.18687/LACCEI2018.1.1.213
ISBN: 978-0-9993443-1-6
ISSN: 2414-6390

Automatización de Pruebas de Software: Experiencia y Lecciones Aprendidas

Leticia Laura-Ochoa, Mg[1]
[1]Universidad Nacional de San Agustín, Perú, llaurao@unsa.edu.pe

***Resumen*– Con el objetivo de mejorar la calidad, fiabilidad y asegurar el correcto funcionamiento de los productos de software; en este trabajo se describe la experiencia del uso de herramientas open source para la automatización de pruebas de software como Specflow y Selenium, las cuales han permitido realizar pruebas funcionales y tener la documentación respectiva de las pruebas de software. Contar con un equipo que se dedique a realizar estas pruebas de forma paralela al desarrollo contribuye a mejorar la calidad del producto de software y disminuye los errores de los entregables en los hitos establecidos con el cliente.**

***Palabras claves*– pruebas de software, calidad de software, herramientas de pruebas, automatización de pruebas, casos de prueba.**

***Abstract*— With the aim of improving the quality, reliability and ensuring the proper functioning of the software products; This paper describes the experience of the use of open source tools for the automation of software tests such as Specflow and Selenium, which have allowed to perform functional tests and have the respective documentation of the software tests. Having a team dedicated to perform these tests in parallel to the development contributes to improving the quality of the software product and decreases the errors of the deliverables in the milestones established with the client.**

***Keywords*— Software testing, software quality, test tools, test automation, test cases**

I. Introducción

Ante un mercado cada vez más competitivo y en constante desarrollo, la calidad del software está tomando mayor importancia en las organizaciones, y con ello, la calidad del producto software [1]. La calidad de los productos de software está fuertemente influenciada por la calidad del proceso que los generó; particularmente, el proceso de prueba contribuye a la calidad del producto y representa un esfuerzo significativo en proyectos de desarrollo de software [2]. La industria del software reconoce hoy la importancia de llevar a cabo pruebas de software, como instrumento para asegurar la calidad de los productos desarrollados [2].

El software debido a lo ajustado de los tiempos de entrega o a recursos humanos con poco entendimiento del proceso de desarrollo o las tecnologías contiene mala calidad y esta queda oculta para los usuarios finales; esta mala calidad invisible y difícil de entender en términos de negocio se conoce como deuda técnica [3]. Deuda técnica es una metáfora utilizada para representar los problemas que ocurren cuando una de las dimensiones de la gestión de proyectos es priorizada por sobre otra, típicamente presiones de calendario por sobre requerimientos de calidad [4]. Todas las etapas en el proceso del desarrollo de software son sumamente relevantes, pero, quizás la etapa de pruebas o testing sea la menos sistematizada y tenida en cuenta en ese proceso [5]. Las pruebas son una etapa del proceso de construcción del software que determina, junto a las actividades de validación, la calidad del nuevo sistema de software a construir [6].

En el proceso de desarrollo de software, las actividades relacionadas con la etapa de testing suelen llevarse a cabo luego de finalizada la implementación, y, en muchos casos, terminan realizándose en paralelo con puestas en producción. Esto trae las obvias consecuencias de sistemas de actualizaciones y parches constantes que permiten reparar problemas que podrían haber sido salvados con una mejor revisión [5]. Los sistemas o productos de software son creados por seres humanos por ende no se puede garantizar que no se hayan cometido errores durante la implementación de los mismos. Asimismo, el costo asumido por los errores encontrados en el sistema, luego de haberse adquirido, supone pérdida de dinero y tiempo de los recursos asignados para el proceso de desarrollo [7]. Las pruebas del software pueden realizarse en distintas etapas del proceso de desarrollo. Es importante que las mismas se realicen en etapas tempranas, pudiendo esto, obviamente, condicionar el posterior desarrollo. Si se realizan en etapas tempranas es posible mejorar la calidad del producto, es menos costoso encontrar errores y resolverlos en las primeras etapas que al final del desarrollo (se deben encontrar errores antes de que los encuentre el cliente) [8].

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 [9]. Las pruebas son muy costosas por lo que se dejan para las últimas etapas del proyecto y no se realizan con la calidad necesaria [10]. Hay que considerar que las pruebas de software son una parte importante en el proceso de desarrollo, pero si estas no se realizan de forma adecuada, el costo de corregir un error en un sistema aumenta a medida que se avanza en el desarrollo del mismo, además gran parte del presupuesto del

Digital Object Identifier (DOI): http://dx.doi.org/10.18687/LACCEI2018.1.1.213
ISBN: 978-0-9993443-1-6
ISSN: 2414-6390

proyecto es asignado para la etapa de prueba, sin dejar a un lado la verificación y validación que se debe realizar al software [11]. Para lograr productos de calidad se debe buscar el mejoramiento permanente de las pruebas y el proceso de desarrollo, y probar nuevos enfoques y metodologías. Por lo tanto, ese es el momento para ensayar las herramientas de prueba de código abierto [12].

La automatización de las pruebas, implica una disminución del costo de las mismas, no sólo porque se ejecutan rápidamente, sino además porque las podemos programar para que se ejecuten sin intervención humana [13]. Se convierte en un importante mecanismo de control para asegurar la precisión y la estabilidad del producto a través del ciclo de vida [14]. La automatización de las pruebas funcionales reduce significativamente el esfuerzo dedicado a las pruebas de regresión en productos que se encuentran en continuo mantenimiento [15].

En este trabajo se describe la experiencia adquirida con el uso de herramientas open source como Specflow[16] y Selenium[17] para la automatización de pruebas en una empresa que se dedica al desarrollo de productos de software.

II. Marco Conceptual

A. *Calidad del Software*

El tema de la calidad de software es muy importante en la actualidad, pero ha sido poco trabajada por parte de las empresas desarrolladoras. El obtener software de calidad involucra utilizar procedimientos estándares para análisis, diseño, programación y pruebas, que permitan compensar los requerimientos de trabajo para lograr confiabilidad, mantenibilidad y facilidad de pruebas, así como elevar la productividad, tanto en el desarrollo como en la gestión de la calidad del software [11]. Se define como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente [18].

B. *Pruebas de Software*

Las pruebas de software son procesos realizados concurrentemente a través de las etapas del desarrollo de software, su objetivo es apoyar la disminución del riesgo de aparición de fallas y faltas en operación [10]. Comprenden una fase del proceso de desarrollo que se centra en asegurar la calidad, fiabilidad y robustez de un software, dentro de un contexto o escenario donde está previsto que este sea utilizado. Un subconjunto de estas pruebas corresponde con las que tienen, como principal objetivo, asegurar el correcto funcionamiento de las interfaces de usuario (GUIs) [19]. Constituyen un elemento importante en el aseguramiento de la calidad de software y la mejora de procesos en cualquier empresa [20].

El concepto de pruebas de calidad de software permite en las empresas con áreas afines a los sistemas, la computación y la informática, brindar productos con altos estándares de calidad y con una disminución de fallos en estos [21].

C. *Pruebas Funcionales*

Reciben el nombre de pruebas funcionales, aquellos procesos de evaluación del software enfocados en la validación del cumplimiento de los requisitos funcionales de un sistema informático, certifican que el producto de software ejecute de forma correcta las actividades para las cuales fue desarrollado [22]. Son aquellas que validan las especificaciones definidas por el usuario, teniendo en cuenta a la funcionalidad como una caja opaca en la cual se ingresan datos/valores y se debe controlar los valores/datos de salida [8]. Se orientan en el comportamiento externo de un producto o aplicativo software, en las pruebas de caja negra [21].

D. *Casos de Prueba*

Los casos de prueba especifican los requisitos de la aplicación, por lo que cada requisito debe estar cubierto por un mínimo de un caso de prueba. Cada caso de prueba está compuesto por varios pasos a ejecutar, dependiendo de la complejidad del caso, y cada paso está compuesto por una acción, que será realizada por el tester, y un resultado esperado. Para que un caso de prueba resulte exitoso, todos los pasos deben cumplir el resultado esperado. Si uno de los pasos no lo cumple, todo el caso de prueba resultará fallido [23]. Existen diferentes técnicas de derivación de casos de prueba para probar una unidad de software. Estas se pueden dividir en dos, de caja negra (funcional) o de caja blanca (estructural). Las técnicas de caja negra, para derivar los casos de prueba, se basan únicamente en la funcionalidad que debe proveer la unidad. Por otro lado, las técnicas de caja blanca consideran la implementación para derivar los datos de prueba [22].

Los casos de prueba son esenciales para todas las actividades de pruebas [24]: (a) Son la base para diseñar y ejecutar los procedimientos de pruebas. (b) La profundidad de las pruebas es proporcional al número de casos de pruebas. (c) El diseño y desarrollo, y los recursos necesarios son gobernados por los casos de pruebas requeridos. (d) Si los casos de prueba no son correctos, la calidad del sistema se pone en duda y las pruebas dejan de ser confiables.

E. *Automatización de Pruebas*

La automatización de pruebas es implementada mediante herramientas de software que se encargan de realizar validaciones que no requieran intervención manual y que puedan ser ejecutadas sin vigilancia o monitoreo constante por parte del recurso humano [22]. Las pruebas automatizadas son sólo un tipo de prueba que utiliza scripts para ejecutar automáticamente una serie de procedimientos en el software bajo prueba para comprobar que los pasos se codifican

adecuadamente [14]. Este proceso de automatización es ideal para la realización de operaciones altamente repetitivas y para el trabajo con aplicativos que requieren mantenimiento continuo [22].

Las pruebas automatizadas tienen como propósito aligerar el proceso de ejecución de pruebas, sin embargo traen consigo limitantes, como la alta dependencia que se tiene con las herramientas de automatización y la complejidad que conlleva su configuración a la hora de interactuar con ellas [22]. Se requiere que el personal dedicado a la automatización de pruebas tenga el conocimiento adecuado sobre el uso de estas herramientas.

F. SpecFlow

SpecFlow se usa para definir, administrar y ejecutar automáticamente pruebas de aceptación legibles y de fácil comprensión en proyectos .NET, es de código abierto y se proporciona bajo una licencia BSD. SpecFlow es una herramienta basada en lenguaje Gherkin y es compatible con el framework .NET, Xamarin y Mono [16].

Utiliza el lenguaje Gherkin (Given-When-Then) para la definición de escenarios de pruebas lo que facilita la creación de los mismos por parte de las personas del negocio [25].

G. Selenium

Selenium automatiza los navegadores, se usa para la automatización de aplicaciones web para fines de prueba. Selenium cuenta con el apoyo de algunos de los proveedores de navegadores más grandes que han tomado (o están tomando) medidas para hacer que Selenium sea una parte nativa de su navegador. Es la tecnología core en innumerables herramientas de automatización de navegadores, APIs y frameworks [17].

Selenium es un conjunto de utilidades que facilita la labor de obtener juegos de pruebas para aplicaciones web. Para ello nos permite grabar, editar y depurar casos de prueba, que podrán ser ejecutados de forma automática e iterativa posteriormente [26]. El potencial de esta herramienta puede ser utilizado para la grabación de las pruebas funcionales durante la generación de pruebas de regresión. Con este servicio se consigue obtener una batería de pruebas automatizadas que podrán ser utilizadas cuando sea necesario repetir las pruebas [7].

III. DESCRIPCIÓN DE LA EXPERIENCIA

La empresa donde se realizó la implementación de pruebas automatizadas utilizando las herramientas open source Specflow y Selenium, se dedica al desarrollo de productos de software en la ciudad de Arequipa-Perú.

Las empresas desarrolladoras de software están dedicadas a producir sistemas informáticos para automatizar las principales actividades de las organizaciones y se han visto en la necesidad de aplicar metodologías y técnicas para asegurar la calidad de sus productos [11]. El director o gerente del proyecto debe estar centrado en que los clientes buscan productos que satisfagan sus necesidades, y la calidad es el factor principal; por eso se tienen que seguir unos principios y clasificaciones, que a través de un plan de pruebas, sean guía para el equipo de desarrollo, y que este realice una optimización de las herramientas, técnicas y conocimientos [21], es por esta razón que en la empresa se utilizan nuevas herramientas como apoyo a las pruebas de productos de software.

Actualmente, gran parte del esfuerzo de la prueba requerido en un proyecto está apoyado por herramientas. La prueba manual es laboriosa y propensa a errores, y no soporta el mismo tipo de controles de calidad que son posibles con las pruebas automatizadas [14].

SpecFlow y Selenium se utilizaron para generar las pruebas automáticas para la interfaz de usuario, en este trabajo se describe la experiencia del uso de estas herramientas bajo la plataforma de Microsoft .NET [27].

SpecFlow utiliza un lenguaje Gherkin, el cual no es un lenguaje técnico y puede entenderse fácilmente tanto por los programadores como los clientes del negocio (Stakeholders). Se utiliza para crear los escenarios de prueba en lenguaje natural y a partir de este generar de forma paralela la documentación respectiva como casos de prueba y registro de observaciones internas a los desarrolladores. Al crear los escenarios con SpecFlow se puede utilizar el lenguaje de origen, sea español, inglés, etc.

En la Fig. 1 se muestra un ejemplo de creación de escenario para automatizar una prueba de flujo normal de creación de clientes.

MClientes.feature

```
7   Scenario: Crear Cliente
8       Given que estoy en la ventana de Maestro de Clientes
9       When hago clic en el botón Crear Cliente
10      And muestra la ventana donde ingreso los datos
11      Then hago clic en el botón Guardar
```

Fig. 1 Escenario crear cliente

De la misma forma se definen pruebas para flujos de error cuando por ejemplo se ingresan datos incompletos o incorrectos para comprobar que muestren mensajes de error o validación de datos.

Features son los archivos con los que se trabaja en SpecFlow en donde se definen los escenarios, los cuales describen la secuencia de pasos para la automatización de pruebas de flujo normal y flujo de error, deben ser independientes. Los escenarios pueden definirse a partir de los casos de uso.

En la Tabla I se muestra un ejemplo de la estructura del archivo Feature.

TABLA I
ESTRUCTURA DEL ARCHIVO FEATURE

Feature (Se especifica el nombre y texto de la descripción)
Escenario 1 Paso 1 Paso 2 … Paso n
Escenario 2 Paso 1 Paso 2 … Paso n
Escenario n Paso 1 Paso 2 … Paso n

Luego se realiza la implementación de los pasos definidos en lenguaje natural de los escenarios creados. En la Fig. 2 se muestra el archivo MClientesSteps.cs, cuya estructura se genera de forma automática para empezar a programar las definiciones de los pasos declarados en el escenario.

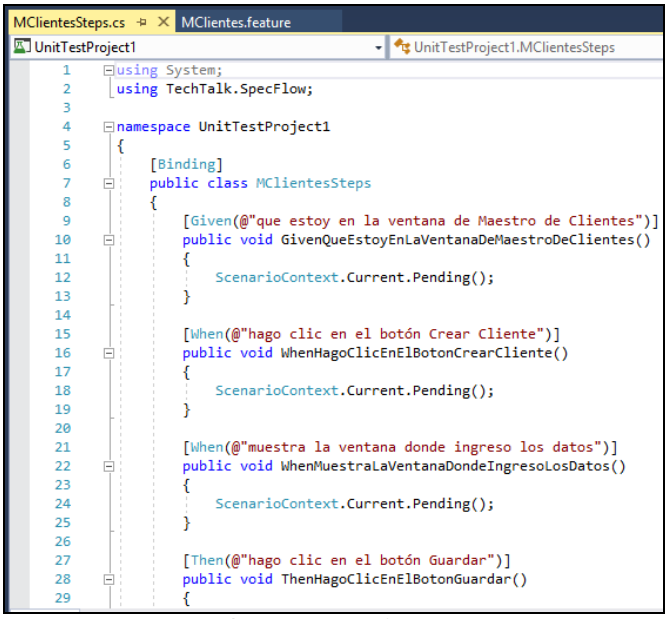

Fig. 2 Generación de los pasos definidos en el escenario

SpecFlow permite crear los escenarios en lenguaje natural y Selenium es el código de automatización que se puede utilizar en Java, C#, Python.

Selenium WebDriver es una herramienta que sirve para automatizar pruebas de interfaz de usuario UI de las aplicaciones Web conectándose directamente con el navegador. Permite trabajar con múltiples navegadores como por ejemplo Firefox, Chrome, Internet Explorer, que son los mayormente utilizados.

Para la implementación de pruebas de interfaz de usuario para las aplicaciones Web se debe tener conocimiento de HTML para acceder a los elementos de la interfaz de usuario como cuadros de texto o botones mediante el ID, nombre o XPath.

Las interfaces gráficas (GUIs) representan un elemento fundamental y crítico de las aplicaciones de hoy en día, llegando a acaparar incluso hasta el 60% del código. Por lo tanto, probar la funcionalidad de las GUIs se presenta como una tarea imprescindible para asegurar la robustez, usabilidad y calidad del sistema completo [19].

Al estar trabajando en Visual Studio, se probó también otras herramientas como code metrics, el cual es un analizador de código estático que favorece la calidad de nuestro desarrollo generando métricas de código que nos ayudan a tomar decisiones sobre la forma en que se diseña y escribe el código [28]. Se pudo obtener indicadores como:

- Índice de mantenimiento
- Complejidad ciclomática
- Profundidad de herencia
- Acoplamiento de clases
- Líneas de código

Estas métricas nos permiten determinar la facilidad de mantenibilidad de los productos de software que se desarrollan, así como también el nivel de dependencia que existe entre clases y complejidad del código.

En el 2017, se desarrolló una encuesta a uno de los equipos de desarrollo de software de la empresa, en la cual el 67% del equipo de desarrollo encuestado opinó que las observaciones realizadas por el equipo de analistas de pruebas han permitido mejorar bastante su calidad de desarrollo de software, y al 33% restante le ayudo un poco estas observaciones para mejorar su calidad de desarrollo. También un 67% del equipo de desarrollo encuestado opinó que el área de calidad le ha permitido disminuir completamente (17%) y bastante (50%) sus errores al momento de presentar sus entregables con el cliente, y al 33% restante le ayudo un poco a disminuir sus errores, no encontrándose opiniones desfavorables al respecto [29].

IV. LECCIONES APRENDIDAS

Contar con un equipo que se dedique a realizar pruebas de software de forma paralela al equipo de desarrollo contribuye a mejorar la calidad del producto de software y disminuye los errores de los entregables en los hitos establecidos con el cliente.

Al realizar las pruebas automatizadas se pueden detectar fallas y realizar el registro de observaciones para ser corregidas por los programadores según su nivel de prioridad.

Las pruebas automatizadas se pueden ejecutar en cualquier momento y de forma rápida a comparación de las pruebas manuales y es menos propenso a errores.

Con las herramientas de automatización de pruebas se pueden grabar flujos normales para verificar el correcto funcionamiento de las aplicaciones Web y flujos de errores para comprobar el manejo de excepciones, mensajes de error y validación de campos.

Si se producen cambios en las aplicaciones Web a causa de nuevos requerimientos por el cliente, es más fácil y rápido actualizar las pruebas automatizadas para probar el producto de software con las nuevas modificaciones.

El equipo que realiza las pruebas de software debe tener una buena comunicación con el equipo de desarrollo y conocer los requisitos y funcionamiento de las aplicaciones que se desarrollan.

SpecFlow al utilizar un lenguaje natural en la creación de escenarios es comprensible incluso para el cliente. También es más fácil realizar la documentación de casos de prueba a partir de los escenarios definidos.

Selenium WebDriver permite trabajar con múltiples navegadores para la automatización de pruebas de interfaz de usuario de las aplicaciones Web.

El uso de herramientas en las pruebas de software apoya el trabajo manual, por lo que ambas son necesarias. Es recomendable la actualización e investigación del uso de nuevas herramientas que contribuyan a mejorar la calidad de los productos de software

El tipo de pruebas de las interfaces de usuario GUI representan un paso crítico antes de que un software sea aceptado por el usuario final y sea puesto en funcionamiento [19].

V. Conclusiones

La automatización de pruebas de software utilizando herramientas open source como Specflow y Selenium permitieron grabar y reproducir flujos normales así como flujos de error para verificar el correcto funcionamiento de los productos de software pudiendo ejecutarse en cualquier momento, estas herramientas sirvieron para realizar pruebas funcionales de las interfaces de usuario en las aplicaciones Web y tener la documentación respectiva.

La implementación de un proceso de pruebas de calidad de software en las empresas o universidades instituye un gran avance en el intento por garantizar márgenes de calidad en los productos de desarrollo software, y es un elemento estratégico para la imagen corporativa o institucional [21].

Referencias

[1] E. Irrazábal, "Mejora de la mantenibilidad con un modelo de medición de la calidad: resultados en una gran empresa", in *Proc. XXI Congreso Argentino de Ciencias de la Computación*, 2015.

[2] C. García, A. Dávila, and M. Pessoa, "Test process models: Systematic literature review", *Communications in Computer and Information Science*, vol. 477, pp. 84-93, 2014.

[3] E. Flores, "Cuantificación de la deuda técnica en aplicaciones bancarias, mediante el método sqale", Tesis para optar el grado de Maestro en Ciencias en Informática, Instituto Politécnico Nacional, México, 2015.

[4] S. Matalonga, A. Villar and C. Nacimiento, "Deuda técnica: ¿Cuáles son los límites de la metáfora?", in *Memorias de la XVI Conferencia Iberoamericana de Ingeniería de Software CIBSE 2013*, pp. 33-46, 2013.

[5] F. J. Díaz, C. M. Banchoff, A. S. Rodríguez and V. Soria, "Metodologías para la evaluación de herramientas Free/Open Source para pruebas de software", In *XIII Workshop de Investigadores en Ciencias de la Computación*, 2011.

[6] G. Kaplan, J. Doorn, W. Panessi, C. Ortiz and E. Cespedes, "Derivación de casos de prueba a partir de escenarios", In *XIX Workshop de Investigadores en Ciencias de la Computación WICC (WICC 2017, ITBA, Buenos Aires)*, pp. 475-479, 2017.

[7] E. Chinarro, M. E. Ruiz and E. Ruiz, "Desarrollo de un Modelo de Pruebas Funcionales de Software Basado en la Herramienta SELENIUM", *Revista Industrial Data*, vol. 20, no. 1, pp. 139-148, 2017.

[8] J. Díaz, C. Banchoff, A. Rodríguez and V. Soria, "Herramientas open source para testing de aplicaciones Web", In *XV Congreso Argentino de Ciencias de la Computación*, 2009.

[9] M. E. Escobar-Sánchez and W. M. Fuertes-Díaz, "Modelo formal de pruebas funcionales de software para alcanzar el Nivel de Madurez Integrado 2", *Revista Facultad de Ingeniería*, vol. 24, no. 39, pp. 31-42, 2015.

[10] G. J. Myers, T. Badgett, and C. Sandler, "The art of software testing 3rd. ed.", New Jersey, USA: JohnWiley & Sons, 2011.

[11] M. G. Vinueza, "Análisis de la aplicación de los modelos de calidad de software en las empresas desarrolladoras asentadas en Quito y Guayaquil", *Ciencia Unemi*, vol. 5, no. 8, pp. 93-101, 2015.

[12] E. Serna and A. Serna, "Una evaluación a las herramientas libres para pruebas de software", *Revista Virtual Universidad Católica del Norte*, no. 37, 2012.

[13] P. A. Vaca, C. Maldonado, C. Inchaurrondo, J. Peretti, M. S. Romero and M. Bueno, "Estudio de Test-Driven Development en el proceso de desarrollo de Software.", In *XVI Workshop de Investigadores en Ciencias de la Computación*, 2014.

[14] J. Francis, "Los mitos en la automatización de las pruebas", *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software RACCIS*, vol. 4, no. 2, pp. 57-60, 2014.

[15] I. Esmite, M. Farías, N. Farías and B. Pérez, "Automatización y gestión de las pruebas funcionales usando herramientas open source", In *XIII Congreso Argentino de Ciencias de la Computación*, 2007.

[16] SpecFlow, "What is SpecFlow?", http://specflow.org/, Revisado en Febrero del 2018.

[17] Selenium, "What is Selenium?", http://www.seleniumhq.org/, Revisado en Febrero del 2018.

[18] R. Pressman, *Ingeniería del Software. Un enfoque práctico*, España: McGraw-Hill, 2010.
[19] P. L. Mateo, G. Martínez and D. Sevilla, "Open HMI Tester: un Framework Open-source para herramientas de pruebas de software", *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*, vol. 3, no. 4, 2009.
[20] R. C. Castilla, O. Barrera, L. G. Fernández, M. Cabrera and L. González, "Proceso de pruebas y suite de herramientas de soluciones informáticas para la salud", *Revista Cubana de Informática Médica*, vol. 7, no. 1, pp. 56-72, 2015.
[21] J. A. Mera-Paz, "Análisis del proceso de pruebas de calidad de software", *Ingeniería Solidaria*, vol. 12, no. 20, pp. 163-176, oct. 2016. doi: http://dx.doi.org/10.16925/in.v12i20.1482
[22] S. M. Giraldo and J. E. Giraldo, "Sistema de generación automática de scripts de ejecución para pruebas unitarias en aplicaciones web", *Revista Politécnica*, vol. 9, no. 17, 2013.
[23] Globe, "Pruebas funcionales y casos de prueba", https://www.globetesting.com/2012/01/pruebas-funcionales-y-casos-de-prueba/, Revisado en Febrero del 2018.
[24] P. Kruchten, The Rational Unified Process: An Introduction. Second Edition 2nd, Addison-Wesley, 2000.
[25] Testing en español, "Arrancando con Specflow", https://josepablosarco.wordpress.com/2014/09/23/arrancando-con-specflow/, Revisado en Febrero del 2018.
[26] Selenium, "What is Selenium?", http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/381, Revisado en Febrero del 2018.
[27] Plataforma Microsoft .NET, www.microsoft.com/net, Revisado en Febrero del 2018.
[28] L. Aldazabal, Code2Read, "code metrics – Visual Studio Code Metrics ¿Cómo aseguras la calidad de tu código?", Revisado en Febrero del 2018.
[29] L. Laura, P. Mamani and R. Arisaca, "Automatización de Pruebas y Uso de Métricas en una Empresa de Desarrollo de Software: un caso de estudio", In *Conferencia Latin American Women in Technology - LATINITY 2017*, Arequipa, Perú, 2017.