Innovación y Software
ISSN: 2708-0927
ISSN: 2708-0935
facin.innosoft@ulasalle.edu.pe
Universidad La Salle
Perú

Pérez Vera, Yasiel; Gallegos Valdivia, Juan José; Zapata Quentasi, Sandra María; Ccama Yana, Doris Marcela; Choque Apaza, Rosa Elvira

Design Thinking en la Planificación de Pruebas de Software

Innovación y Software, vol. 1, núm. 2, 2020, Septiembre-Febrero, pp. 40-51
Universidad La Salle
Perú

Revista Innovación y Software
Vol. 1, No. 2, Mes Septiembre-Febrero, 2020
ISSN: 2708-0935
Pág. 40-51
https://revistas.ulasalle.edu.pe/innosoft

Tipo de artículo: Artículos originales
Temática: Calidad de software
Recibido: 15/08/2020 | Aceptado: 04/09/2020 | Publicado: 30/09/2020

Design Thinking en la Planificación de Pruebas de Software

Design Thinking in Software Testing Planning

Yasiel Pérez Vera [1][0000-0001-9421-9529]*, Juan José Gallegos Valdivia [2], Sandra María Zapata Quentasi [3], Doris Marcela Ccama Yana [3][0000-0002-4994-4884], Rosa Elvira Choque Apaza [5]

[1] Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú. yperezv@unsa.edu.pe
[2] Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú. jgallegosv@unsa.edu.pe
[3] Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú. szapataq@unsa.edu.pe
[4] Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú. dccamay@unsa.edu.pe
[5] Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú. rchoqueapaz@unsa.edu.pe

* Autor para correspondencia: yperezv@unsa.edu.pe

Resumen
Este artículo presenta la posibilidad de trabajar en el desarrollo de software utilizando no solo metodologías ágiles para los procesos, sino también el uso de *Design Thinking* para la planificación del Plan de Pruebas del Software. Ya que durante el proceso de gestión del software existe la dificultad de comprenderlo, para una de las etapas de la gestión de software, en este caso la planificación de las pruebas del software, el que ejecuta las pruebas necesita saber lo que el usuario requiere, debido a que el producto de software es intangible y difícil de medir, el que realiza las pruebas debe escuchar al usuario final y realizar las pruebas de acuerdo a los requerimientos que se solicita. Para mejorar el proceso de comprensión de lo que el usuario realmente requiere, se utiliza sus pilares como: empatía, colaboración y creación de prototipos para una mayor interpretación, la cual permitirá obtener una actitud asertiva al equipo de pruebas. Por otro lado, tenemos que realizar el plan de pruebas del software, en base a esto se realizó un estudio sobre la posibilidad de utilizar *Design Thinking* para ayudar al proceso de planificación de pruebas de software.

Palabras clave: *Design Thinking*, Planificación, Pruebas de Software.

Abstract
This article presents the possibility of working on software development using not only agile methodologies for processes but also the use of Design Thinking for the planning of the Software Test Plan. Since during the software management process there is the difficulty of understanding the software, for one of the stages of the software management, in this case, the planning of the software tests, the tester needs to know what the user requires, due to Because the software product is intangible and difficult to measure, the tester must listen to the end-user and perform the tests according to the requirements requested. To improve the process of understanding what the user really requires, its pillars are used as empathy, collaboration, and prototyping for a better interpretation, which will allow

obtaining an assertive attitude to the tester. On the other hand, we have to carry out the software test plan, based on this a study was carried out on the possibility of using Design Thinking to help the software test planning process.

Keywords: *Design Thinking, Planning, Software Testing.*

Introducción

En los proyectos de desarrollo de software, es común que surjan situaciones que comprometan el éxito o la continuación de los proyectos debido a diferentes factores. Usualmente los problemas surgen debido a demoras, sobrecostos no previstos y mala calidad del producto final. Es responsabilidad de los jefes de proyecto anticiparse a estas situaciones a fin de reducir la probabilidad de ocurrencia, y minimizar el impacto en sus proyectos [1].

Design Thinking permite concebir proyectos informáticos para dar soluciones a los problemas del mundo real. Se centra en las personas, considerando multidisciplinariedad, colaboración y concreción de pensamientos y procesos, los cuales son caminos que llevan a soluciones innovadoras su base en la resolución de problemas, desde el punto de vista del usuario y con el apoyo de prototipos [2]. Estudios sugieren que este enfoque utiliza métodos basados en soluciones para explorar valores centrados en el ser humano a lo largo del proceso de diseño de ingeniería. Estos hallazgos se reflejan en muchas aplicaciones del pensamiento de diseño: la creación de prototipos, un método basado en soluciones, a menudo se cita como una forma útil de fomentar la inspiración, la ideación y el aprendizaje organizacional, todos valores centrados en el ser humano [3].

En este artículo se presenta una aplicación de *Design Thinking* en el área de planificación de pruebas de software, ya que este enfoque aplicado a pruebas de software significa desarrollar métodos de prueba innovadores, tipos de prueba y estrategias de prueba, tanto como las pruebas. Esto se puede lograr mediante un cambio fundamental en el pensamiento y la acción. Todos los integrantes del equipo de pruebas de software, desarrolladores y gerentes deben participar en este proceso de cambio ya que práctica tan innovadora requiere empleados de mente abierta, gerentes y la propia empresa [4]. El objetivo de las pruebas es mejorar y mejorar la calidad del software. Por lo tanto, las pruebas siempre están estrechamente vinculadas a la búsqueda de ideas creativas e innovadoras.

Materiales y métodos

Materiales

a. Metodología ágil

Las metodologías ágiles son un formato más contemporáneo para el desarrollo de software donde funciona el compromiso activo del cliente, la comunicación recurrente entre el equipo de desarrollo y la entrega rápida del valor del cliente, de acuerdo con el Manifiesto Ágil [5]. Los métodos ágiles permiten un trabajo de *sprint* corto, que pueden ser características lanzadas en siete o catorce días, por ejemplo, entregar valor al cliente que puede probar durante el período de desarrollo. Sin embargo, debido a que tiene *sprints* cortos, el software falla en algunos puntos, como la falta de documentación o la documentación deficiente del software o software desarrollado que se lanza con cada *sprint*.

Al tener un modelo de comunicación interna y externa muy fácil, ya que las reuniones de lanzamiento y planificación se llevan a cabo al comienzo de cada sprint, el cliente siempre tiene fácil acceso al prototipo desarrollado y le permite iterar entre lo que se está haciendo. desarrollado y el usuario final. Por otro lado, el usuario tiene acceso solo cuando se finaliza el prototipo, sin ser escuchado antes de que comience el sprint, lo que puede hacer que las fallas, que son predecibles durante el período de desarrollo, sean más frecuentes de lo esperado.

b. Design Thinking

Es un modelo mental de desarrollo empresarial innovador que permite enfocar todo el proceso de producción en cómo las personas pueden llegar a lo que se está desarrollando. Aunque es un modelo para el desarrollo empresarial y las ideas de diseño, *Design Thinking* se puede aplicar de muchas maneras dentro de otros procesos, como la gestión de proyectos de ingeniería de software.

El proceso de *Design Thinking* se basa en tres pilares importantes [6]: empatía, colaboración y experimentación. El pilar de la empatía busca colocar a todas las personas que participan en el proceso de desarrollo en lugar de las personas que se verán afectadas por el desarrollo de este proyecto. De esa manera, los participantes del proyecto deben descifrar todo lo que la gente necesita para comprender el proceso, quién debe ser el público objetivo de una solución determinada. El pilar de la colaboración busca comprender cómo cada una de las personas que participan en el proyecto puede ayudar en cualquier momento dado, facilitando el proceso de desarrollo y trabajando eficazmente en equipos para obtener el mejor resultado posible. El pilar de la experimentación propone que cuando se comprende un estudio del requisito y las necesidades de un usuario, los participantes del proyecto deben crear un prototipo que minimice o resuelva el requisito del usuario, lo ponga en uso y brinde retroalimentación, generando aprendizaje sobre el proceso.

Revista Innovación y Software
Vol. 1, No. 2, Mes Septiembre-Febrero, 2020
ISSN: 2708-0935
Pág. 40-51
https://revistas.ulasalle.edu.pe/innosoft

Al usarlo como método de desarrollo, también parte de la premisa de que lo que se desarrollará debe ser innovador, pero más que eso debe ser: deseable, económicamente viable y técnicamente posible [6].

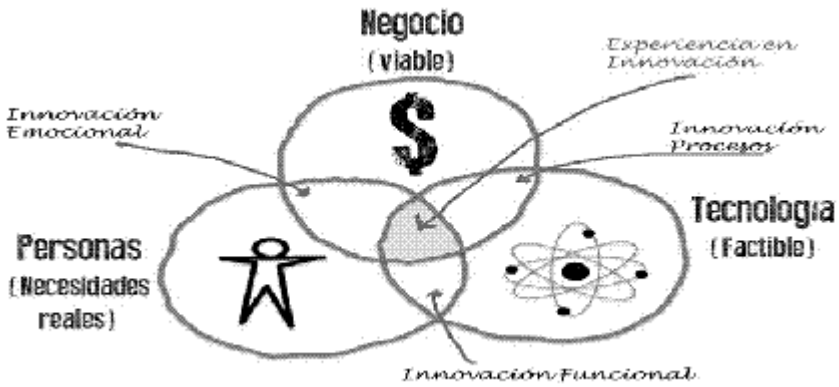

Figura 1. *Design Thinking* [6].

Design Thinking funciona con cinco fases básicas para que un proyecto tenga éxito, siendo ellas:

1. Empatía: donde la empatía es el punto más importante del proceso. Los actores del proceso deben comprender y diferir en cómo los usuarios sufren un problema particular que debe abordarse. En este momento, se recopilan todos los posibles problemas que el usuario pueda tener con respecto al problema en cuestión.
2. Definición: momento de conversión de ideas donde los actores del proceso se ponen a agregar a lo recogido algunas ideas y así entienden cuál debe ser la necesidad del usuario. En esta etapa, también se piensan formas de interpretar la empatía causada por el proceso de descubrimiento y vinculación con la información sobre los requisitos de las personas.
3. Ideación: a partir de ese momento, las necesidades de las personas están pensadas para resolver su requerimiento. Es un momento de divergencia en el que todos tienen la oportunidad de hacer observaciones y pensar en cómo podrían ser las soluciones a estos problemas planteados.
4. Prototipado: este es un momento para crear un prototipo de la solución y entregarla al usuario para que la pruebe y reciba comentarios sobre sus impresiones de la solución. En este momento no se desarrollan soluciones terminadas, solo prototipos que dan la idea de lo que la gente podría tener como solución a sus problemas en el desafío propuesto. Es tiempo de escuchar e interpretar al usuario, como en la segunda fase.
5. Testeo: La evolución del proceso ocurre al final de un ciclo de empatía, colaboración y experimentación, donde el usuario fue escuchado, pensó en cómo se sentía, observó y habló, interpretó y entregó un prototipo a estos usuarios.

Revista Innovación y Software
Vol. 1, No. 2, Mes Septiembre-Febrero, 2020
ISSN: 2708-0935
Pág. 40-51
https://revistas.ulasalle.edu.pe/innosoft

Después de recopilar comentarios en la fase anterior, los participantes del proceso ahora pueden pensar en cómo evolucionar el prototipo en función de lo que se ha escuchado de los usuarios y lo que se debe rediseñar.

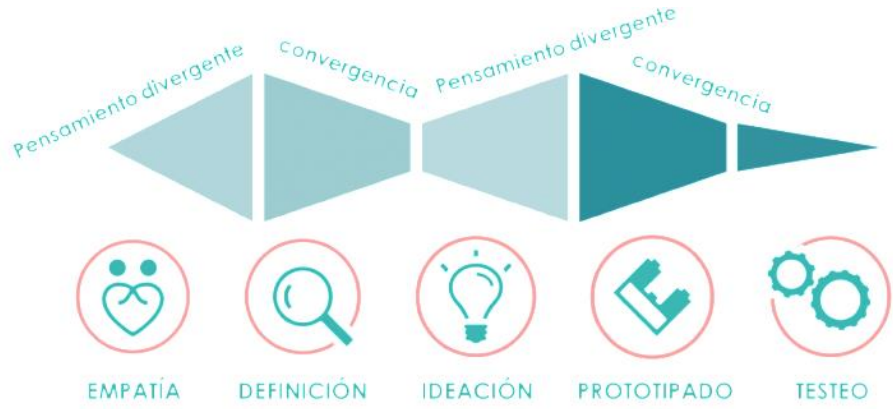

Figura 2. Fases de *Design Thinking* [6].

Todo el proceso debe repetirse hasta que la solución sea satisfactoria para el usuario. Es importante tener en cuenta que el proceso también puede permitir la evolución de un proyecto *ad-eternum* por el principio muy innovador que tiene *Design Thinking*.

Métodos

El método elegido para realizar la investigación es el método análisis-sintético. Por lo tanto, se tratará de investigar cómo se desarrolla un plan de pruebas y los pasos a seguir en un *Design Thinking* para luego unificarlos y lograr un resultado óptimo en las pruebas de software. La investigación se realiza mediante la búsqueda de artículos relacionados sobre cómo realizar un plan de pruebas y las actividades que uno debe realizar para llegar a un resultado favorable, las librerías específicas utilizadas son las siguientes [7]–[11], también informándonos sobre la metodología *Design Thinking* el cual se enfoca en la interacción de los individuos de un equipo, así como también las tareas realizadas por cada uno de ellos que pueden ser resultado de un proceso de *Design Thinking*.

Para el entendimiento de la toma de requerimientos de los usuarios, se pueden seguir actividades de creación de interfaces a bajo nivel. La interacción del usuario a bajo nivel permite retroalimentar rápidamente el proceso. De esta forma, se realiza un ejercicio de prueba/falla hasta que el usuario considere que un resultado cubre con sus expectativas. Esto permite acortar el tiempo de desarrollo de interfaces a alto nivel, eliminando la curva de aprendizaje y el coste de

las herramientas necesarias para realizarlo. De esta forma, el usuario puede identificar rápidamente con qué características de diseño está conforme y cuáles no, usando solo el tiempo necesario para llegar a esta conclusión. Las entrevistas con el equipo en donde los temas a tratar serán sobre identificar las funcionalidades nuevas o ya existentes, la estrategia, los entornos y la metodología se realiza en una habitación, que el equipo puede acondicionar a su criterio, en donde se debe incentivar el pensamiento creativo. En este caso, los miembros del equipo se convierten en los usuarios sobre los que hay que aplicar el proceso de *Design Thinking* en cual nos apoyaremos de algunas herramientas como es el mapa mental, entrevista y manualidades creativas. Para la validación y verificación del plan de pruebas el scrum master debe considerar las fases de observación y evaluación.

Design Thinking es un proceso iterativo en donde cada paso a seguir será utilizado de manera optimizada, dentro del paso entender el objetivo fundamental es comprender por qué se usan las pruebas de software con mayor precisión y enfatizar cada uno para luego verificar si se han aumentado los riesgos y por qué es importante crear un plan de pruebas también nos sirve para darnos cuenta sobre las vulnerabilidades solucionadas y por solucionar.

En el segundo paso, llamado observar, es donde uno aborda directamente el tema de las pruebas, problemas y riesgos, en este paso se sostienen conversaciones y también se escuchan, donde cada persona explica su propio enfoque y cómo resuelven los problemas durante las pruebas dependiendo al plan desarrollado, en el cual todo tiene que estar detalladamente documentado. El tercer paso llamado punto de vista es donde los involucrados analizan que las pruebas de software necesitan soluciones creativas nuevas e innovadoras con respecto a los problemas y riesgos encontrados donde cada uno da su punto de vista para encontrar la solución más adecuada al problema y documentar correctamente.

El cuarto paso, el paso llamado idear es donde se recopila las ideas sobre las soluciones creativas sin evaluarlas y se usan herramientas en forma de lluvia de ideas, mapa mental con el fin de resolver los problemas y riesgos del software. Ejemplos de posibles estrategias creativas utilizando pensamiento visual para mejorar el resultado de las pruebas de software, como se muestra en la Tabla 1.

Tabla 1. Posibles Estrategias

Posibles Estrategias	• Matriz de puntos de contacto • *StoryBoard* • *Customer journey* • Aprendizaje de oráculos de pruebas

El quinto paso llamado prototipo es la posible solución encontrada para resolver el problema que al principio se investiga y se verifica relacionando las clases de pruebas con ideas nuevas, innovadoras, creativas y didácticas como se muestra en la tabla 2.

Tabla 2. Modelo prototipo para cada estrategia

Estrategias	Ejemplos
Matriz de puntos de contacto	*Touchpoint Matrix*
StoryBoard	*Shotbox*
Customer journey	*Customer Journey Map*
Aprendizaje de oráculos de prueba	Migración de software

El último paso, denominado prueba, es donde se usa y se prueba de inmediato en el caso de las pruebas de software, esto podría significar que la solución es exitosa y podría integrarse en las pruebas como una parte esencial fija, o puede verse como una forma de ampliar aún más la idea.

Resultados y discusión

Dentro de los resultados se mostrará como los métodos tradicionales de planificación de las pruebas de software pueden ser mejorados poniendo en práctica *Design Thinking* mediante el uso de metodologías ágiles. En este artículo se trabajará con las 5 etapas del plan de pruebas propuestos en [12], las cuales se detallarán en la Tabla 3 con sus principales objetivos.

Tabla 3. Plan de Pruebas de Software

Plan de Pruebas de Software	
Planeación de pruebas.	- Tipos de Prueba. - Estrategia de Pruebas.
Diseño de pruebas.	- Análisis de la documentación.

Implementación de pruebas.	- Creación de datos de prueba. - Ejecución de los casos de prueba.
Evaluación de criterios de salida.	- Finalización del ciclo. - Criterios de salida.
Cierre del proceso.	- Cerrar las incidencias reportadas.

Para medir el impacto que tendrá el uso de *Design Thinking* en la planificación de pruebas se tomará las métricas según el ISO 9126 que nos permitirán obtener el grado de satisfacción de los usuarios. La Tabla 4 nos muestra las métricas que estamos tomando en cuenta y el aspecto que estamos evaluando.

Tabla 4. Métricas a evaluar

Métrica	**Aspecto a evaluar**
Efectividad de la prueba.	Defectos reportados por las pruebas.
Esfuerzo.	Tiempo promedio de resolución de defectos. Cantidad de defectos reportados.
Sobre esfuerzo.	Cantidad de casos de prueba escritos. Cantidad de defectos reportados.
Cobertura de requisitos.	Requisitos funcionales y no funcionales totales.
Cobertura de riesgos.	Cantidad de riesgos detectados.
Efectividad de la prueba.	Cantidad de defectos detectados.

Para la incorporación de *Design Thinking* en la planificación de las pruebas de software se tomó en cuenta las etapas del plan de software expuestas y las fases *Design Thinking*. En la Tabla 5 se muestra las etapas del plan de pruebas utilizando las fases de este enfoque.

Tabla 5. Plan de Pruebas de Software con *Design Thinking*

Plan de Pruebas de Software con *Design Thinking*		
Planeación de pruebas.	- Tipos de Prueba. - Estrategia de Pruebas.	- Empatía: Análisis de lo que se va a juzgar del producto. Ejemplo: Participación activa con los usuarios finales.
Diseño de pruebas.	- Análisis de la documentación.	- Empatía: Entender cómo se comporta el usuario. Ejemplo: Proponer laboratorios de *Design Thinking* para la discusión de requerimientos hacer probados.
Implementación de pruebas.	- Creación de datos de prueba. - Ejecución de los casos de prueba.	- Definición: Se toma en cuenta las necesidades del usuario. Ejemplo: Los datos de prueba deben reflejar los datos de los usuarios finales. - Ideación: Solución y desarrollo de tareas. Ejemplo: Las soluciones esperadas deben ser el reflejo de las necesidades de los usuarios finales.
Evaluación de criterios de salida.	- Finalización del ciclo. - Criterios de salida.	- Ideación: Resolver problemas. Ejemplo: Las necesidades de los usuarios se vuelven prioridad para ser comparadas con los resultados obtenidos. - Prototipado: Todas las suposiciones deben ser validadas. Ejemplo: Las salidas de las pruebas deben ser validadas con los usuarios finales.
Cierre del proceso.	- Cerrar las incidencias reportadas.	- Testeo: Reunión de todo el proceso. Se decide si es necesario un nuevo ciclo. Ejemplo. El cliente recibe los resultados de las pruebas de acuerdo a las necesidades de los usuarios finales.

Los resultados obtenidos en la aplicación de *Design Thinking* en el plan de pruebas se muestran en la Tabla 6 donde podemos observar que es importante tener una comunicación constante con los usuarios para que los resultados al finalizar el plan sean los ideales.

Tabla 6. Resultados aplicando *Design Thinking.*

Resultados aplicando Design Thinking		
Etapas del Plan de Pruebas	**Métodos tradicionales**	**Design Thinking con métricas de evaluación**
Planeación de pruebas.	Las pruebas son levantadas de acuerdo a las especificaciones obtenidas.	Trato directo con los usuarios finales para la planificación.
Diseño de pruebas.	Análisis de documentación de las especificaciones de los requerimientos.	Reuniones con los usuarios y el cliente para la obtención de las pruebas requeridas.
Implementación de pruebas.	Realizar casos de prueba en base a los datos de la documentación.	Obtención de datos mediante el trato directo con el usuario final.
Evaluación de criterios de salida.	Las salidas reflejan los datos de la documentación.	Las salidas reflejan las necesidades de los usuarios finales.
Cierre del proceso.	No intervienen usuarios finales.	El cliente requiere la opinión de los usuarios finales.

Los resultados demuestran que utilizando el enfoque *Design Thinking* podemos obtener mejores resultados de las pruebas debido a que los usuarios finales son nuestros actores importantes y con los que se interactúa directamente. Según las métricas que se consideran un sobre esfuerzo se reduce debido a que la interacción directa con el usuario permite tener un mejor plan de pruebas, la reducción de riesgos aumenta ya que se cuenta con la opinión del cliente, el empatizar tanto con los clientes y usuarios finales nos permitirá tener cubierto la mayor parte de los casos de uso. La posibilidad de realizar de un nuevo ciclo de plan de pruebas debido a que la primera no cubrió las necesidades del producto queda desestimada debido a que en cada momento se cuenta con la opinión de los usuarios y clientes.

Según [13], donde realiza la aplicación de *Design Thinking* en la ingeniería de requerimientos y la medida de la calidad de software indican que utilizando el enfoque tiene un impacto positivo al igual que en [1] que aplica *Design Thinking* a la gestión y desarrollo de software.

Conclusiones

Los resultados obtenidos en el artículo demuestran que las metodologías tradicionales pueden ser mejoradas para la obtención de mejores resultados, mediante la aplicación de *Design Thinking* en las etapas del plan de pruebas. Este enfoque todavía es nuevo en comparación con las metodologías de desarrollo anteriores, sin embargo, lo hace adecuado para la ingeniería de software con cierto éxito, ya que puede integrarse con los métodos existentes. El uso de *Design Thinking* se usa normalmente cuando se encuentran involucrados individuos el cual ayuda a mejorar su comprensión en los problemas. *Design Thinking* es dinámico y por lo tanto se puede adaptar y aplicar a cualquier disciplina.

Referencias

[1] J. C. Espinoza Vásquez and E. E. Espinoza Zapata, “Marco de trabajo en base a Design Thinking y metodologías ágiles de desarrollo de software,” 2017.

[2] R. Villarroel, H. Spencer, and R. Muñoz, “Aplicación de design thinking de manera interdisciplinaria en la asignatura de ingeniería de software,” 2017.

[3] M. Greene, “Design Thinking Vs. Systems Thinking for Engineering Design: What’s the difference?[WWW Document],” in *DS 87-2 Proceedings of the 21st International Conference on Engineering Design (ICED 17)*, 2017, vol. 2, pp. 21–25.

[4] Y. Tomita, K. Watanabe, S. Shirasaka, and T. Maeno, “Applying Design Thinking in Systems Engineering Process as an Extended Version of DIKW Model,” *INCOSE International Symposium*, vol. 27, no. 1, pp. 858–870, 2017, doi: 10.1002/j.2334-5837.2017.00398.x.

[5] K. Beck *et al.*, *The agile manifesto*. Feb, 2001.

[6] T. Brown and B. Katz, *Change by design: how design thinking transforms organizations and inspires innovation*, vol. 20091. HarperBusiness, 2019.

[7] IEEE, “IEEE Xplore Digital Library,” 2019. http://ieeexplore.ieee.org/Xplore/home.jsp (accessed Mar. 20, 2017).

[8] Springer, “Home - Springer,” 2019. http://link.springer.com/ (accessed Mar. 20, 2017).

[9] SciELO, “SciELO - Scientific Electronic Library Online,” 2019. http://www.scielo.org/php/index.php (accessed Mar. 20, 2017).

[10] ScienceDirect, “ScienceDirect.” http://www.sciencedirect.com/ (accessed Jun. 08, 2016).

[11] Elsevier, “Elsevier Home,” 2019. https://www.elsevier.com/ (accessed Mar. 20, 2017).

[12] D. E. Soto Durán, A. X. Reyes Gamboa, and J. Jiménez Builes, “Aplicación de la Gestión de Conocimiento al proceso de pruebas de software,” *Ing.USBMed*, vol. 8, no. 2, pp. 6–13, Jul. 2017, doi: 10.21500/20275846.2836.

[13] W. Santos, C. Quarto, and L. Fonseca, “Study about software project management with Design Thinking,” in *Proceedings of the Euro American Conference on Telematics and Information Systems*, 2018, pp. 1–4.