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

Huarcaya Zapana, Gerson Italo; Herencia Castro, Nicolas; Sarmiento Tico, Miguel Angel; Chalco Choquehuanca, Elber Diego; Ticona Bejarano, Alex Daniel
Aplicación del método Design Thinking en el área de requerimientos de software
Innovación y Software, vol. 2, núm. 1, 2021, Marzo-Agosto, pp. 43-52
Universidad La Salle
Perú

Revista Innovación y Software
Vol. 2, No. 1, Mes Marzo-Agosto, 2021
ISSN: 2708-0935
Pág. 43-52
https://revistas.ulasalle.edu.pe/innosoft

Tipo de artículo: Artículos originales
Temática: Ingeniería de software
Recibido: 28/11/2020 | Aceptado: 20/02/2021 | Publicado: 30/03/2021

Aplicación del método *Design Thinking* en el área de requerimientos de software

Application of the Design Thinking method in the area of software requirements

Gerson Italo Huarcaya Zapana [1*], Nicolas Herencia Castro [2], Miguel Angel Sarmiento Tico [3], Elber Diego Chalco Choquehuanca [4], Alex Daniel Ticona Bejarano [5]

[1] Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú. ghuarcayaz@unsa.edu.pe
[2] Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú. nherencia@unsa.edu.pe
[3] Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú. msarmientoti@unsa.edu.pe
[4] Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú. echalcoc@unsa.edu.pe
[5] Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú. aticonabe@unsa.edu.pe

[*] Autor para correspondencia: ghuarcayaz@unsa.edu.pe

Resumen
El presente artículo presenta la aplicación de *Design Thinking* en el área de requerimientos de software para ayudar y solucionar algunos problemas que existen en el desarrollo de software. Se pretende integrar las herramientas, características y fases de *Design Thinking* dentro de las actividades del análisis de requerimientos de software. El principal objetivo es integrar el enfoque de *Design Thinking* de manera que ayude en la definición de los requerimientos de software y finalmente ver y analizar los resultados obtenidos.

Palabras clave: *Design Thinking*, Requerimientos de software.

Abstract
This article presents the application of Design Thinking in the area of software requirements to help and solve some problems that exist in software development. It is intended to integrate the tools, features and phases of Design Thinking into the activities of the analysis of software requirements. The main objective is to integrate the Design Thinking approach in a way that helps in the software requirements and finally see and analyze the results obtained .

Keywords: *Design Thinking, Software requirements.*

Revista Innovación y Software
Vol. 2, No. 1, Mes Marzo-Agosto, 2021
ISSN: 2708-0935
Pág. 43-52
https://revistas.ulasalle.edu.pe/innosoft

Introducción

En los diferentes proyectos de desarrollo de software, es común que surjan situaciones que comprometan el éxito o la continuación de los proyectos debido a diferentes factores. Sin embargo, ¿cómo explicamos la alta incidencia de fallos en los proyectos de software? ¿Por qué existen tantos proyectos de software víctimas de retrasos, presupuestos sobregirados y con problemas de calidad? ¿Cómo podemos tener una producción o una economía de calidad, cuando nuestras actividades diarias dependen de la calidad del sistema? 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. Tal vez suene ilógico, pero a pesar de los avances que ha dado la tecnología, aún existen procesos de producción informales, parciales y en algunos casos no confiables.

Los Requerimientos Software cumple un papel primordial en el proceso de producción de software, ya que enfoca un área fundamental: la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema; de esta manera, se pretende minimizar los problemas relacionados al desarrollo de sistemas.

En este trabajo, se propone una solución y ayuda a algunos problemas que se enfocan en el área desarrollo de software tal como son los requerimientos de software, diseñando y aplicando un marco de trabajo para desarrollo de software basado en *Design Thinking*, de tal forma que se puedan integrar las herramientas, características y fases de *Design Thinking* dentro de las actividades del análisis de requerimientos de software, como marco de trabajo. Finalmente, integrar el marco de trabajo a los requerimientos de software y ver algunos resultados obtenidos.

Marco Teórico

Ingeniería de Requerimientos

Desde la perspectiva del proceso del software, la ingeniería de requerimientos es una de las acciones importantes de la ingeniería de software que tiende un puente para el diseño y la construcción y proporciona el mecanismo apropiado para entender lo que desea el cliente, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida de que se transforman en un sistema funcional [1]. Un requisito del software es una característica que se debe exhibir para solucionar un cierto problema en el del mundo real [2]. Algunos de los componentes de la ingeniería de requerimientos son:

Revista Innovación y Software
Vol. 2, No. 1, Mes Marzo-Agosto, 2021
ISSN: 2708-0935
Pág. 43-52
https://revistas.ulasalle.edu.pe/innosoft

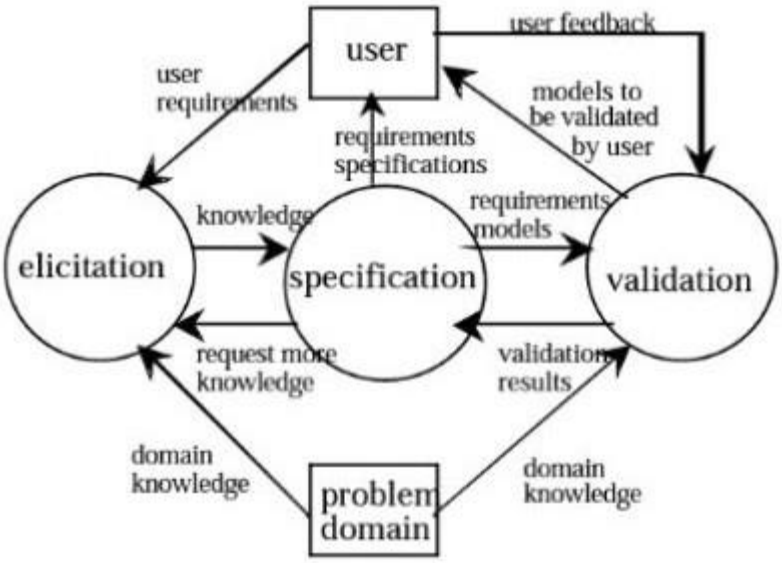

Figura 1. Componentes de la Ingeniería de Requerimientos.

Problemas del Usuario

Introduce los papeles de la gente que participa en el proceso de los requisitos. Este proceso es fundamental interdisciplinario, y el especialista de los requisitos necesita mediar entre el dominio del tenedor de apuestas y el de la tecnología de dotación lógica. Hay mucha gente implicada además del especialista de los requisitos, cada uno de ellos tiene una función en el software. Los tenedores de apuestas variarán según los proyectos, pero incluyen siempre usuarios/operadores y clientes (quiénes no necesitan ser iguales) [2].

Elicitación de Requerimientos

Uso de técnicas para conseguir que los stakeholders articulen sus requisitos. Es un área muy difícil en ingenieros de software, es particularmente importante entender que la captura no es una actividad pasiva, y que, ingenieros de software tienen que trabajar difícilmente para sacar la información adecuada. Existe un número de técnicas para hacer esto, como Entrevistas, Escenarios, Prototipos, Reuniones, Observaciones [2].

Especificación de Requerimientos

Se refiere típicamente a la producción de un documento, o a su equivalente electrónico, que puede estar sistemáticamente repasado, evaluado, y aprobado. Para los sistemas complejos, particularmente ésos que implican

componentes no-software, se elaboran tres tipos de documentos: definición de sistema, sistema requisitos, y requisitos del software. Para sistemas simples, solamente el tercero de éstos es requerido [2].

Validación de Requisitos

Los documentos de los requisitos pueden estar conformes a la validación y procedimientos de verificación. Los requisitos pueden ser validados para asegurarse de que el ingeniero del software entiende los requisitos, y es también importante para verificar que un documento de requisitos se conforma con la compañía de los estándares, y éste es comprensible, constante, y finito [2].

Design Thinking

El estudio de los procesos cognitivos que se manifiestan en la acción de diseñar y como la serie de actividades cognitivas específicas de diseño que los diseñadores aplican durante el proceso de diseñar, pues se centra en el proceso en lugar de en el producto o, dicho de otra manera, se enfoca en la resolución de problemas, pero no comienza con ninguna solución previa [3].

Pasos del *Design Thinking*

Según [4] los pasos del *Design Thinking* son:

- EMPATÍA: Empatía es la base del proceso de diseño que está centrado en las personas y los usuarios.
- DEFINIR: Traer claridad y enfoque al espacio de diseño en que se definen y redefinen los conceptos.
- IDEAR: Esta etapa se entrega los conceptos y los recursos para hacer prototipos y crear soluciones innovadoras
- PROTOTIPAR: Es la generación de elementos informativos como dibujos, artefactos y objetos con la intención de responder preguntas que nos acerquen a la solución final. O sea, no necesariamente debe ser un objeto sino cualquier cosa con que se pueda interactuar.
- EVALUAR: Consiste en solicitar *feedback* y opiniones sobre los prototipos que se han creado de los mismos usuarios y colegas además de ser otra oportunidad para ganar empatía por las personas de las cuales estas diseñando de otra manera.

Materiales y métodos o Metodología computacional

The Role of Design Thinking and Physical Prototyping in Social Software Engineering

Este artículo defiende el papel de *Design Thinking* en Ingeniería de software social y destaca sus implicaciones para la ingeniería de software en general. Lo hace al informar sobre las contribuciones que el pensamiento de diseño, y en particular el diseño físico, ha llevado a la definición del espacio del problema, la captura de requisitos del usuario y el diseño de características del sistema de un sistema de pronóstico de energía renovable [5].

In Two Minds: How Reflections Influence Software Design Thinking

Este artículo define que la calidad del diseño del software depende en gran medida de *Design Thinking* y los procesos cognitivos de los diseñadores, como se muestra en muchos estudios de diseño. Uno de estos procesos cognitivos es la capacidad de reflexionar sobre el propio diseño durante un discurso de diseño. Sin embargo, las metodologías de diseño de ingeniería de software han ignorado en gran medida dicho proceso de pensamiento de diseño [6].

IBM Design Thinking Software Development Framework

Este trabajo propone un enfoque diferente centrado en satisfacer las necesidades del usuario final empleando el desarrollo de software iterativo *Design Thinking*. Esta metodología se aplicó en cinco proyectos reales de desarrollo de software que se analizaron como parte de este trabajo [7].

Designerly thinking: what software methodology can learn from design theory

Este artículo define que *Design Thinking* es un enfoque que promueve la comprensión de las necesidades del cliente teniendo en cuenta lo que es factible técnica y económicamente. Para evaluar cómo se utiliza el enfoque *Design Thinking* integrado con las metodologías de desarrollo ágil de software [8].

Problemas al momento de la Indagación de Requisitos

Problemas que se encuentran cuando ocurre la indagación según [1]:

- Problemas de alcance. La frontera de los sistemas está mal definida o los clientes o usuarios finales especifican detalles técnicos innecesarios que confunden, más que clarifican, los objetivos generales del sistema.
- Problemas de entendimiento. Los clientes o usuarios no están completamente seguros de lo que se necesita, comprenden mal las capacidades y limitaciones de su ambiente de computación, no entienden todo el dominio del problema, tienen problemas para comunicar las necesidades al ingeniero de sistemas, omiten información que creen que es “obvia”, especifican requerimientos que están en conflicto con las necesidades de otros clientes o usuarios, o solicitan requerimientos ambiguos o que no pueden someterse a prueba.
- Problemas de volatilidad. Los requerimientos cambian con el tiempo.

El Design Thinking puede fomentar el desarrollo de innovación orientada a la sostenibilidad

El enfoque de *Design Thinking* para la formulación de problemas parece útil para establecer el alcance de innovación apropiado para Innovación Orientada a la Sostenibilidad. Su fuerte enfoque en los usuarios (por ejemplo, la elección y el uso del producto) y las partes interesadas fomenta el desarrollo de Innovación Orientada a la Sostenibilidad que satisfaga las necesidades reales de los usuarios. Con su enfoque en la experimentación iterativa, *Design Thinking* permite garantizar efectos positivos de sostenibilidad al tiempo que reduce el riesgo de fracaso de la innovación. En este caso, los efectos positivos de sostenibilidad previstos se neutralizan o incluso se invierten en efectos negativos. Además, nuestro marco conceptual proporciona una base diferenciada que respalda los supuestos de varios investigadores de que un enfoque basado en *Design Thinking* es un activo esencial, pero a menudo ignorado al abordar los desafíos de sostenibilidad [9].

Visual Design Thinking: A Collaborative Dimensions framework to profile visualisations

Desde un punto de vista teórico, las Dimensiones Colaborativas identificadas impulsan aún más el debate sobre el *Design Thinking* visual al sistematizar nuestra comprensión de la variedad de características visuales disponibles para los diseñadores. En particular, los aspectos de las representaciones visuales previamente identificadas en los estudios de diseño, como la capacidad de las visualizaciones para guiar el trabajo del diseñador se agregan en un marco inclusivo. Permite unir el conocimiento académico y la práctica del diseño, haciendo que los diseñadores y gerentes sean más alfabetizados visualmente y, por lo tanto, más competentes en la selección y uso de visualizaciones para apoyar el proceso de diseño en las organizaciones [10].

Resultados y discusión

Utilizar la técnica de *Design Thinking* en el proceso de definición de requerimientos como parte del ciclo de vida de software nos permite identificar las etapas en las cuales dicha técnica es aplicable y cómo su utilización favorece una mejora considerable en la identificación de problemas, identificación de necesidades de usuario, recopilación de nuevas perspectivas de solución, generación de opciones de solución y prototipos que permitan representar dichas ideas [11]. Aquí se muestra, cómo se inicia desde la etapa del diseño del modelo de negocio donde se analiza la viabilidad del proyecto de software, seguida de la identificación de requerimientos donde se seleccionan y definen los requerimientos que van a ser parte de las características funcionales y no funcionales que definen al producto de software, seguidamente dicha técnica es aplicable al análisis y diseño de la propuesta, la implementación y pruebas del producto de software.

Revista Innovación y Software
Vol. 2, No. 1, Mes Marzo-Agosto, 2021
ISSN: 2708-0935
Pág. 43-52
https://revistas.ulasalle.edu.pe/innosoft

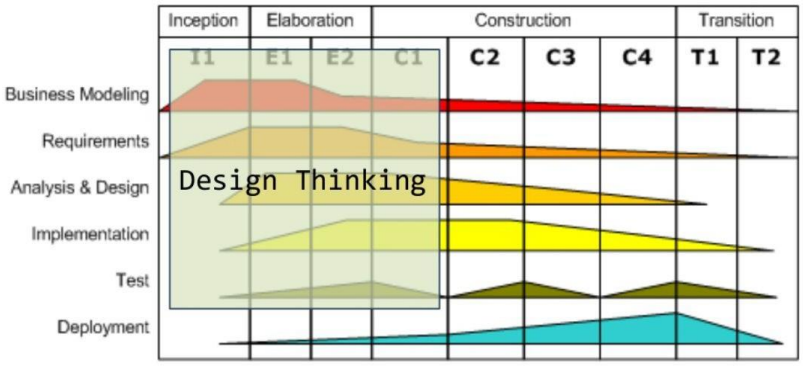

Figura 2. Utilización de *Design Thinking* dentro del ciclo de vida del software [11].

El siguiente cuadro resume cómo se aplica la técnica de *Design Thinking* al área de requerimientos de software:

Tabla 1. Aplicación de *Design Thinking* a la Ingeniería de Requerimientos.

Ingeniería de Requerimientos	***Design Thinking***		
Elicitación	Empatizar	Observación	Un conjunto de actividades para ayudar a comprender a los usuarios, en el contexto de sus problemas. Observar: A través de la Técnica de Observación para la elicitación de Requerimientos.
		Entrevista	Involúcrate: Generar una conversación e interacción con los Stakeholders a través de Reuniones y Entrevistas, preguntando "¿Por qué?" ya que eso descubre nuevos significados, preguntar una y dos veces si es necesario.
		Reuniones	Mira y Escucha: Lo mejor siempre es combinar estas dos, la conversación y el *engagement*. Pídele también que te explique como hace algunas cosas y que vaya vocalizando lo que pasa por su mente cuando esté en su trabajo. Ten una conversación mientras trabaja y esté en su contexto.
	Definir	Reuniones	Enmarcar un problema con un enfoque directo, inspirando al equipo, evaluando ideas, capture emocionalmente a las personas que has estudiado y que ayude a resolver el problema

			imposible de desarrollar conceptos que sirven para todo y para todos.
	Idear	Escenarios	Creación de soluciones a través de casos de Uso, aprovechando de mejor manera las distintas visiones de cada equipo de trabajo y el trabajo colectivo. Descubrir áreas inesperadas de exploración creando mayor volumen y mayores opciones para innovar.
	Prototipar	Prototipo	Para inventar y construir para pensar en resolver el problema. Para cometer errores antes y de manera barata. Para controlar el proceso de la creación de soluciones. Ayuda a identificar distintas variables para poder descomponer grandes problemas que se puedan evaluar y arreglar de mejor forma.
Especificar	Idear	Documento de Definición del Sistema y especificación de Requisitos de Software	Define los requisitos del sistema de alto nivel desde la perspectiva del dominio. La especificación de requisitos software proporciona una base informada para transferir un producto de software a los nuevos usuarios o a las máquinas nuevas.
	Prototipar	Prototipado	Medio para validar la interpretación del ingeniero del software de los requisitos del software, así como para sacar nuevos requisitos. La ventaja de usar prototipos es que pueden hacer más fácil la interpretación del ingeniero del software y, donde lo necesite, dan la explicación útil de por qué son incorrectas
Validar	Testear	Pruebas de Aceptación.	Solicita a los usuarios comentarios sobre el prototipo creado. Las técnicas de evaluación de la experiencia del usuario (UX) podrían usarse para probar el prototipo. Las micro-pruebas son un enfoque común para evaluar prototipos en línea. El tiempo disponible para reclutar usuarios finales, realizar las pruebas y analizar e informar los resultados de las pruebas suele ser muy corto, pero proporciona respuestas rápidas para los equipos de desarrollo.

Conclusiones

El diseño de software es complejo ya que involucra muchas decisiones humanas que por lo mismo no son siempre correctas, por lo que las metodologías de diseño de Ingeniería de Software no han considerado seriamente cómo estos factores de decisión influyen en el razonamiento del diseño resultante además ofrece un análisis inicial y comentarios de los usuarios, ofreciendo una mejor comprensión de qué problemas se deben resolver y cuáles son las mejores soluciones para satisfacer las necesidades del usuario por ende mejora la recopilación y redacción de los requerimientos. El uso de *Design Thinking* en el proceso de definición de requerimientos como parte del ciclo de vida de software nos permite identificar las etapas en las cuales dicha técnica es aplicable y cómo su utilización favorece una mejora considerable en la identificación de problemas, identificación de necesidades de usuario, recopilación de nuevas perspectivas de solución, generación de opciones de solución y prototipos que permitan representar dichas ideas.

Referencias

[1] R. S. Pressman, Software engineering: a practitioner's approach, Eighth edition. New York, NY: McGraw-Hill Education, 2015.

[2] P. Bourque and R. Fairley, "Swebok," Nd: IEEE Computer society, 2004.

[3] R. Pelta Resano, "Design Thinking. Tendencias en la teoría y la metodología del diseño, septiembre 2013," 2018.

[4] H. Plattner, Guía del proceso creativo. Mini guía: una introducción al Design Thinking+ Bootcamp bootleg. 2018.

[5] P. Newman, M. A. Ferrario, W. Simm, S. Forshaw, A. Friday, and J. Whittle, "The role of design thinking and physical prototyping in social software engineering," in 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, 2015, vol. 2, pp. 487–496.

[6] M. Razavian, A. Tang, R. Capilla, and P. Lago, "In two minds: how reflections influence software design thinking," Journal of Software: Evolution and Process, vol. 28, no. 6, pp. 394–426, 2016.

[7] P. Lucena, A. Braz, A. Chicoria, and L. Tizzei, "IBM design thinking software development framework," in Brazilian Workshop on Agile Methods, 2016, pp. 98–109.

[8] J. C. Pereira and R. de FSM Russo, “Design thinking integrated in agile software development: A systematic literature review,” Procedia computer science, vol. 138, pp. 775–782, 2018.

[9] A. Buhl et al., “Design thinking for sustainability: Why and how design thinking can foster sustainability-oriented innovation development,” Journal of cleaner production, vol. 231, pp. 1248–1257, 2019.

[10] S. Bresciani, “Visual design thinking: A collaborative dimensions framework to profile visualisations,” Design Studies, vol. 63, pp. 92–124, 2019.

[11] R. Villarroel, H. Spencer, and R. Muñoz, “Aplicación de design thinking de manera interdisciplinaria en la asignatura de ingeniería de software,” in Memorias XXX Congreso SOCHEDI, 2017, pp. 1–9.