I + T + C Investigación, Tecnología y Ciencia **ISSN: 1909-5775** **e-ISSN: 2805-7201**

Artículo de Investigación

Agile prototyping strategy for building shared understanding in requirements engineering

Estrategia de prototipado ágil para construir entendimiento compartido en la ingeniera de requisitos

Iván Darío Londoño [1 *], Pablo H. Ruiz [2], Vanessa Agredo-Delgado[2]

1 Corporación Universitaria Comfacauca - Unicomfacauca; ivanlondono@unicomfacauca.edu.co
2 Corporación Universitaria Comfacauca - Unicomfacauca; pruiz@unicomfacauca.edu.co
2 Corporación Universitaria Comfacauca - Unicomfacauca; vagredo@unicomfacauca.edu.co
* Correspondencia: ivanlondono@unicomfacauca.edu.co

Citación: Londoño, I .; Ruiz, P .; Agredo-delgado, V. Estrategia de prototipado ágil para construir entendimiento compartido en la ingeniera de requisitos. I + T + C Investigación, Tecnología y Ciencia. Vol 1. Num. 17. 2023.

Derechos de autor:© 2023 por los autores. Publicación en acceso abierto bajo los términos y condiciones de la licencia Creative Commons Attribution (CC BY NC SA)https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es_ES)

Nota del editor: El Sello editorial Unicomfacauca se mantiene neutral con respecto a los reclamos jurisdiccionales en mapas publicados y afiliaciones institucionales.

Resumen: La estrategia propuesta en este artículo aborda un desafío crítico en la ingeniería de requisitos: la construcción de un entendimiento compartido entre todas las partes interesadas en el desarrollo de software. El entendimiento compartido se refiere a la capacidad de las personas de un equipo para tener una comprensión común y coherente de los requisitos que tendrá el sistema, lo cual es esencial para una colaboración efectiva. La contrucción de esta estrategia, se realizó utilizando la Ingeniería de Métodos Situacionales (Situational Method Engineering - SME), la cual permitió obtener, al finalizar la realización de cada una de sus actividades, un conjuto de fases, con sus respectivas actividades y tareas, que buscan la construcción del entendimiento compartido a traves de la realización de la creación de prototipos ágiles. El uso de prototipos permite proporcionar una representación visual y tangible de los requisitos del sistema, lo que facilita la comunicación y la retroalimentación entre las partes interesadas. En este sentido, este articulo muestra el uso de SME para la construccion final de la estrategia, la cual consta de varias fases, que incluyen la planificación, identificación y análisis de requisitos, creación de prototipos no funcionales, gestión de requisitos, construcción de prototipos funcionales y socialización. Estas fases están diseñadas para fortalecer la comunicación, promover la colaboración y garantizar un entendimiento compartido sólido a lo largo del proceso de desarrollo de software.

Palabras clave: Prototipo ágil; Entendimiento compartido; Ingeniería de requisitos; Metodología ágil; Ingeniería de métodos situacionales.

Abstract: The strategy proposed in this paper addresses a critical challenge in requirements engineering: building a shared understanding among all stakeholders in software development. Shared understanding refers to the ability of people in a team to have a common and consistent understanding of the requirements that the system will have, which is essential for effective collaboration. The construction of this strategy was done using Situational Method Engineering (SME), which allowed to obtain, at the end of the realization of each of its activities, a set of phases, with their respective activities and tasks, which seek the construction of shared understanding through the realization of agile prototyping. The use of prototypes allows to provide a visual and tangible representation of the system requirements, which facilitates communication and feedback among stakeholders. In this sense, this article shows the use of SME for the final construction of the strategy, which consists of several phases, including planning, requirements identification and analysis, non-functional prototyping, requirements management, functional prototyping and socialization. These phases are designed to strengthen communication, promote collaboration and ensure a solid shared understanding throughout the software development process.

Keywords: Agile prototyping; Shared understanding; Requirements engineering; Agile methodology; Situational methods engineering.

1. Introducción

La ingeniería de requisitos (IR) es esencial en el desarrollo de software porque facilita la creación de un producto que satisface al cliente, al implicar la interacción continua entre desarrolladores y partes interesadas para analizar, validar y perfeccionar los requisitos, definiendo así lo que el sistema debe hacer y las restricciones que debe cumplir [1]. Por otro lado, el entendimiento compartido hace referencia a "la capacidad de múltiples agentes para coordinar sus comportamientos entre sí para apoyar la consecución de metas u objetivos comunes" [2]. En este sentido, es necesario un entendimiento común de la gestión de requisitos para mejorar la comunicación y la colaboración entre desarrolladores y clientes, lo que a su vez mejora la calidad del software entregado [3]. Sin embargo, puede ser difícil llegar a un entendimiento común en la ingeniería de requisitos debido a desafíos como la diversidad de fuentes de requerimientos, la ambigüedad del lenguaje, la dificultad para verbalizar necesidades, y las diferencias en conocimiento y experiencia entre usuarios y desarrolladores [4]. Generando así que, las especificaciones inadecuadas son el principal motivo de insatisfacción de los clientes con el sistema entregado [5]. A ello se suman problemas como especificaciones de requisitos incompletas y cambios constantes en el proceso de desarrollo [6].

Cuando las empresas de desarrollo de software no tienen una hoja de ruta común para llegar a un entendimiento compartido en la IR, puede enfrentarse a una serie de problemas como: una mala comprensión de los requisitos, una validación ineficaz que da lugar a soluciones insatisfactorias, un aumento de los costes, retrasos, falta de coordinación y comunicación. La creación de prototipos puede ser una solución eficaz para superar estos problemas [7]. En este sentido, la creación de prototipos se percibe como una forma simple y efectiva de generar un entendimiento compartido de los requisitos de un proyecto. Además, promueve una retroalimentación más rápida y fortalece la visión del producto por parte del cliente [8]. Por ello, los proyectos de ingeniería de software requieren metodologías y procesos dinámicos para desarrollar productos y servicios de forma rápida y fiable [9].

La construcción de la estrategia para lograr un entendimiento compartido, se basa en la disciplina de la Ingeniería de Métodos Situacionales (SME), que consiste en adaptar y personalizar los métodos y procesos de ingeniería de software para satisfacer las necesidades específicas de un proyecto o situación concretos [10]. A partir de la ejecución de esta SME, se identificaron actividades importantes como: la planificación, que facilita la comprensión de los requisitos del proyecto, establece plazos y fomenta la colaboración. La identificación y el análisis de requisitos, que ayudan a establecer un consenso, evitar cambios innecesarios, guiar a un diseño inicial del prototipo, tomar decisiones informadas y facilitar la comunicación [8]. El entendimiento compartido, es un objetivo constante a lo largo de todas las fases de la estrategia, no solo para asegurar que todos los involucrados comprendan y estén de acuerdo con los requisitos del proyecto, sino que, también facilita la retroalimentación temprana, la identificación de requisitos faltantes o incorrectos, y promueve una colaboración efectiva.

Este artículo se divide de la siguiente manera: Sección 2 Conceptos y trabajos relacionados; Sección 3 Método de investigación y hallazgos; Sección 4 Ingeniería de Método Situacional (SME) para la construcción de la estrategia; Sección 5 Construcción de la estrategia; Sección 6 Discusión; Sección 7 Conclusiones.

2. Estado del arte

2.1. Conceptos

A continuación, se explican algunos de los conceptos utilizados y necesarios a lo largo del artículo para comprender el tema.

2.1.1. Prototipo

Un prototipo es un modelo ejecutable de un sistema que refleja con precisión un subconjunto elegido de sus propiedades, como los formatos de visualización, los

resultados calculados o los tiempos de respuesta. Los prototipos son útiles para formular y validar requisitos, resolver problemas técnicos de diseño y apoyar el diseño asistido por ordenador de los componentes de software y hardware de los sistemas propuestos [11].

También, se define como un modelo de un sistema de software previsto que proporciona una base para el debate, la aclaración, la toma de decisiones, la experimentación, y el aprendizaje entre usuarios y desarrolladores [12].

2.1.2. Entendimiento compartido

El entendimiento compartido se define como: La capacidad de múltiples agentes dentro de un grupo para coordinar comportamientos hacia metas u objetivos comunes basados en conocimientos compartidos, creencias e hipótesis sobre la tarea, así como el grupo, el proceso o las herramientas y tecnologías utilizadas, que pueden cambiar durante el proceso de trabajo en grupo y repercutir en los procesos y resultados [13].

2.1.3. Ingeniería de requisitos

La ingeniería de requisitos (IR) es una rama de la ingeniería de software que se ocupa de la realización de actividades en un intento de comprender las necesidades exactas de los usuarios del sistema y, a continuación, traducir estas necesidades en funciones y/o acciones precisas que se utilizarán en el desarrollo del sistema o que éste realizará una vez finalizado con éxito [14]. La ingeniería de requisitos se basa en cuatro etapas, que son [6]:

- Elicitación de requisitos: Etapa donde se identifica y recopila información sobre los requisitos del sistema de los usuarios, clientes y otras partes interesadas.
- Análisis de requisitos: Se estudian los requisitos obtenidos en la etapa anterior, a fin de poder identificar áreas no especificadas o algunos requerimientos que sean contradictorios, peticiones irrelevantes, falencias o inconsistencias generadas y que deberán ser eliminadas.
- Especificación de requisitos: Etapa de documentar los requisitos del sistema en un formato claro y conciso que pueda ser utilizado por los desarrolladores para construir el sistema.
- Validación de requisitos: Es el proceso de confirmar que los requisitos del sistema son correctos, completos y coherentes con las necesidades del usuario y las restricciones del sistema.

2.1.4. Metodología

La metodología es una de las etapas específicas de un trabajo o proyecto que parte de una posición teórica y conlleva a una selección de técnicas concretas o métodos acerca del procedimiento para el cumplimiento de los objetivos. Es el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y optimizarla. Determina los pasos a seguir y cómo realizarlos para finalizar una tarea [15].

2.1.5. Manifiesto por el desarrollo Ágil

Las metodologías ágiles se describen como un proceso incremental, cooperativo, sencillo y adaptativo. Se conocen por proporcionar pautas y principios pragmáticos que simplifican la entrega de proyectos, promoviendo una comunicación constante y cercana entre clientes y desarrolladores. Estas metodologías evitan la burocracia de los enfoques tradicionales, minimizando la documentación y renuncian a los métodos formales [15]. La flexibilidad de las metodologías ágiles se puede ver en la subdivisión de proyectos en unidades más pequeñas, la colaboración, la adaptación a cambios y la anticipación de cambios en los requerimientos. La entrega continua al cliente y la retroalimentación constante son clave para mejorar el producto y el proceso de la IR [16].

El manifiesto ágil se basa en principios fundamentales como: la satisfacción del cliente a través de entregas tempranas y continuas de software funcional, la capacidad de adaptarse a cambios en los requisitos en cualquier momento del proyecto, la participación activa del cliente, la búsqueda de la simplicidad en los procesos, equipos de desarrollo motivados y autoorganizados, una comunicación efectiva, la autorreflexión y la capacidad de adaptarse a cambios en los requisitos [16].

2.2. Trabajos relacionados

2.2.1 Prototipado y metodologías ágiles en la ingeniería de requisitos:

En [17], la creación de prototipos en ingeniería de requisitos es esencial para capturar y validar eficazmente los requisitos del software. Ofrece ventajas como la identificación temprana de requisitos, la mejora de la comunicación con las partes interesadas, la gestión de riesgos y la evaluación de la usabilidad. De igual forma, en [11], hace hincapié en la creación rápida de prototipos por su capacidad para obtener retroalimentación temprana e identificar problemas antes de la implementación. Así mismo [6], subraya la importancia de implicar a los usuarios finales y a los expertos en la materia durante el desarrollo de prototipos para mejorar la usabilidad y la eficacia de la aplicación mediante la retroalimentación y la validación continua.

Por su parte [7], destaca que la creación de prototipos es una herramienta importante para visualizar y validar los requisitos en una fase temprana del proceso de desarrollo de software. Esto facilita la comprensión de los requisitos y la comunicación eficaz entre el equipo de desarrollo y las partes interesadas. El enfoque ágil proporciona flexibilidad para adaptarse a los cambios y priorizar las necesidades del cliente de forma eficaz. De manera similar en [18], examina las cuestiones y consideraciones clave asociadas al uso de métodos ágiles en el modelado de requisitos. Destaca la importancia de una comunicación eficaz, la capacidad de adaptarse al cambio, fomentando la flexibilidad y la capacidad de respuesta ante nuevas necesidades o prioridades. Estos enfoques ágiles ofrecen ventajas como: una mejor colaboración, una clara comprensión de los requisitos y una mayor satisfacción del cliente. Finalmente, en [19], subraya que las metodologías ágiles son cruciales para el proceso de ingeniería de requisitos porque fomentan la colaboración estrecha con el cliente, la traducción efectiva de ideas en requisitos concretos, la flexibilidad y la simplificación del proceso.

2.2.2 Entendimiento compartido en el desarrollo de software y la ingeniería de requisitos:

En el estudio [20], se centra en encontrar las supuestas barreras que pueden interferir en la capacidad de los equipos de desarrollo ágil de software para compartir conocimientos de forma eficaz. Al garantizar que los equipos tengan un conocimiento compartido y preciso, se hace hincapié en la importancia del entendimiento compartido como método para fomentar una cooperación eficaz. De manera similar [21], investiga cómo los equipos de desarrollo ágil de software crean y mantienen un entendimiento compartido satisfactorio utilizando la noción de modelos mentales compartidos. La idea sostiene que cuando los miembros del equipo crean modelos mentales comunes de circunstancias, actividades y procesos, se facilita la comunicación, la coordinación y la toma de decisiones en equipo.

En [22], demuestra el valor de un entendimiento común a la hora de desarrollar requisitos de software para evitar malas interpretaciones, errores y problemas de comunicación. El establecimiento de una comunicación clara y el fomento de un entendimiento común entre todas las partes interesadas garantizan un proceso de desarrollo más preciso y satisfactorio. De igual forma [23], destaca la importancia del entendimiento compartido en el desarrollo de una arquitectura empresarial efectiva y cómo la falta de este entendimiento puede llevar a problemas de comunicación y errores en la definición de los requisitos. Luego, introduce el método situacional, que se centra en abordar los desafíos específicos relacionados con la creación de un entendimiento compartido en el contexto de la arquitectura empresarial. Por último, en [24], subraya la necesidad de un entendimiento compartido para el éxito de los proyectos de software. Se señala cómo las diferencias en la interpretación de los requisitos pueden provocar fallos de comunicación y errores en el desarrollo de software.

Existe un consenso generalizado sobre la importancia de un entendimiento compartido en ingeniería de requisitos y desarrollo de software en general después de examinar los trabajos relacionados previamente mencionados. En estos estudios, se destaca de manera consistente la importancia de la comunicación efectiva, la colaboración

temprana y la adaptabilidad a los cambios. Sin embargo, a pesar de los avances significativos, aún se observa una falta de un enfoque más profundo en la mitigación de barreras específicas que impiden un entendimiento compartido en los equipos de desarrollo ágil, así como en la exploración de estrategias y herramientas concretas para abordar estas barreras.

3. Método de investigación:

Inicialmente, el proceso para la construcción de la estrategia comenzó con la planeación y ejecución de dos mapeos sistemáticos, donde se aplicó la metodología propuesta por Barbara Kitchenham y Brereton [25], la cual consta de 3 fases: 1) Planeación de la búsqueda sistemática, 2) Ejecución y 3) Presentación de los resultados. Este estilo de revisión bibliográfica, utiliza un procedimiento para descubrir, analizar e interpretar información relevante para un tema de investigación determinado.

3.1 Hallazgos de los mapeos sistemáticos:

3.1.1 Entendimiento compartido en los procesos de desarrollo de software:

Para llevar a cabo el primer mapeo sistemático de la literatura, se realizó una exhaustiva búsqueda en diversas bibliotecas digitales, como IEEExplore, ACM Digital Library, ScienceDirect, ISI Web Of Science, Scopus y Google Scholar. Inicialmente, se identificaron un total de 354 artículos relevantes. Luego, se aplicaron criterios de inclusión y exclusión, eliminando duplicados, y se obtuvieron 107 artículos. En la segunda fase, se evaluaron los artículos completos y se seleccionaron 27 artículos primarios. Finalmente, estos artículos primarios fueron sometidos a un análisis detallado para responder a las preguntas de investigación planteadas, para luego, a partir de los hallazgos obtenidos tener una base fundamental para la construcción de la estrategia propuesta.

Para el primer mapeo, se descubrió que hay una serie de barreras que pueden dificultar que los equipos de desarrollo de software adquieran un entendimiento común. entre los cuales están: la subcontratación, las limitaciones normativas y el tamaño y la diversidad de los equipos [26], con frecuencia, los miembros del equipo utilizan el mismo lenguaje para conceptos distintos o palabras diferentes para las mismas nociones porque pueden proceder de campos diferentes [13]. Además, los conflictos interpersonales y la falta de contexto e historia comunes pueden dificultar que equipos distantes lleguen a un acuerdo sobre cómo resolver un problema [27].

Por otro lado, los requisitos incompletos, la falta de detalles, el intercambio insuficiente de información entre los miembros del equipo, las incompatibilidades internas y, en algunos casos, los errores que provocan incoherencias con la realidad real se han identificado como obstáculos para alcanzar un entendimiento común de los requisitos [28]. También hay una serie de dificultades relacionadas con las barreras de comunicación [29], la construcción de confianza, el conocimiento del equipo y su progreso en el trabajo, pueden verse afectados por el rápido cambio, la falta de conocimiento del dominio y las barreras geográficas, temporales, organizacionales y culturales, dificultando el entendimiento compartido en la ingeniería de requisitos [26]. Por su parte, debido a su naturaleza transversal, los requisitos no funcionales plantean dificultades adicionales, sobre todo en aplicaciones de ingeniería de software continuas [29]. En contextos dispersos, establecer un entendimiento común puede resultar complicado porque los miembros del equipo no suelen tener la oportunidad de debatir sus preocupaciones cara a cara y porque las dificultades lingüísticas dificultan una comprensión común [3]. Dado que deben transmitir con éxito sus ideas a diversos públicos, las empresas creativas de software se enfrentan a importantes problemas de ambigüedad, especificación inadecuada de requisitos y coherencia [30]. Debido a los problemas lingüísticos y de interpretación, la diversidad cultural puede provocar malentendidos e incertidumbre, lo que dificulta la creación de un entendimiento mutuo entre los equipos y dentro de ellos [3].

En conclusión, el entendimiento compartido es crucial en la fase de ingeniería de requisitos. El desarrollo de un producto que satisfaga las expectativas del usuario es menos probable si se minimizan los malentendidos, se evitan las interpretaciones erróneas y se logra un entendimiento compartido. Además, anima al equipo a colaborar eficazmente, comunicarse con claridad y tomar decisiones bien informadas. Es crucial recordar que el concepto de entendimiento compartido aún está en desarrollo y no cuenta con una metodología específica, pasos concretos o estrategia establecida en la ingeniería de sistemas.

3.1.2 Prototipado en los procesos de desarrollo de software:

En esta investigación se realizó un segundo mapeo sistemático de literatura enfocado ahora, en obtener información sobre el uso de los prototipos en el desarrollo de productos software. Este mapeo se dividió en tres fases, en la primera se aplicó una cadena de búsqueda a tres bases de datos bibliográficas (IEEExplore, ACM Digital Library y Scopus), donde se obtuvo como resultado 217 artículos. Luego, se aplicaron criterios de inclusión y exclusión, eliminando duplicados, y se obtuvieron 181 artículos. En la segunda fase, se evaluaron los artículos completos y se seleccionaron 43 artículos primarios. Finalmente, en la fase 3, se llevó a cabo la extracción de datos de los estudios primarios y dar respuesta a las preguntas de investigación.

Para este segundo mapeo, los principales hallazgos que se encontraron fueron: en el contexto de las empresas de software, surge la necesidad de utilizar estrategias y herramientas de comunicación efectivas como el prototipado, para evitar retrabajo en el desarrollo y colaborar con las partes interesadas [12]. Esto es importante para identificar fácilmente malentendidos e incertidumbre sobre los requisitos, lo que asegura el éxito del proyecto [31]. Los prototipos se caracterizan por su capacidad de permitir interacciones colaborativas y se adaptan a diferentes situaciones de diseño, involucrando a personas de diferentes disciplinas que utilizan diversos medios y dispositivos. En este sentido, se reconoce el valor del prototipado como una herramienta flexible y de apoyo en el proceso de diseño y la IR [28].

Sin embargo, el prototipado es una actividad compleja que depende de varios factores, como el tipo y la función del prototipo, su forma de uso, quién lo utiliza y el contexto [12]. A pesar de esto, existe una creciente tendencia a utilizar el prototipado en los procesos de desarrollo de software, ya que se considera una estrategia eficaz por parte de los desarrolladores de productos software.

La creación de prototipos en el desarrollo de software tiene beneficios significativos, mejora la comprensión de los usuarios y equipo sobre cómo funcionará y se verá el sistema final [26]. Además, facilita la comunicación de los requisitos y promueve un entendimiento compartido al permitir a los interesados ver una representación tangible del sistema [32]. Por otro lado, es crucial considerar que los prototipos pueden plantear ciertos desafíos, entre ellos: dar lugar a demasiadas solicitudes de cambio, confundir a los usuarios al creer que el prototipo es el producto finalizado y surgir diferencias en las especificaciones del software cuando los clientes prueban los prototipos [32]. Estos aspectos deben abordarse adecuadamente para maximizar los beneficios del prototipado en el desarrollo de software.

Como conclusión, en el desarrollo de software, en particular en la etapa de ingeniería de requisitos, es esencial contar con un entendimiento compartido respaldado por prototipos. Los prototipos son herramientas efectivas que materializan las ideas y conceptos, ofreciendo una representación visual y tangible del producto final. Esto mejora la comunicación y la retroalimentación entre las partes interesadas, minimizando malentendidos e interpretaciones erróneas. Y, además, fomenta la colaboración y la toma de decisiones informadas, ya que todos pueden interactuar con el prototipo y comprender cómo se traducen los requisitos en funcionalidades concretas [12].

De este modo, se busca que la estrategia definida en esta investigación permita lograr un entendimiento compartido en la ingeniería de requisitos, la cual se base en el enfoque

de prototipado ágil. Esta estrategia fomentaría la colaboración entre los miembros del equipo y facilitaría la toma de decisiones, mediante la interacción con el prototipo, donde todas las partes involucradas puedan comprender de manera clara cómo los requisitos se materializan en funcionalidades concretas del software. Esto impulsaría una participación activa y agilizaría el proceso de refinamiento de requisitos, permitiendo detectar rápidamente posibles mejoras o ajustes necesarios. Esto debido a que, el prototipado ágil es una herramienta valiosa para establecer una comunicación efectiva y alinear a los stakeholders en torno a una visión común del producto. Considerando que, al experimentar y evaluar el prototipo, se promueve una comprensión más precisa de las necesidades y expectativas de los usuarios finales, lo que resultaría en una mayor calidad y satisfacción del software final.

4. Ingeniería de Método situacional (SME)s para la construcción de la estrategia

4.1. Ingeniería de métodos situacionales:

La Ingeniería de Método Situacional (SME), es la disciplina de ingeniería del diseño, construcción y adaptación de métodos, técnicas y herramientas de desarrollo de sistemas de software. Un método puede definirse como un enfoque para llevar a cabo un proyecto de desarrollo de sistemas de software, basado en una forma específica de pensar, que consiste en directrices, reglas y heurística, estructurada sistemáticamente en términos de actividades de desarrollo, con el correspondiente desarrollo, productos de trabajo y roles de los desarrolladores [10].

Considerando esto, la estrategia se construyó siguiendo el soporte metodológico de SME, utilizando sus tres actividades generales: especificación de los requisitos de la estrategia, selección de los componentes del método, y ensamblaje de los componentes seleccionados [10]. Estas actividades se ejecutaron siguiendo un ciclo de vida iterativo e incremental. Cada iteración corresponde a un ciclo completo, lo que implica una progresión, que aborda la misma secuencia de actividades, pero genera una nueva versión de la estrategia propuesta.

4.1.1 Elementos básicos de la estrategia:

Inicialmente para poder realizar las actividades de SME se realizó el siguiente análisis considerando la ejecución de los mapeos sistemáticos de literatura. Con los respectivos resultados obtenidos y un análisis realizado a lo largo de la ejecución de la investigación, se identificaron un conjunto de oportunidades de mejora, para determinar aquellas necesidades existentes que aún no han sido resueltas y que buscan mejorar el entendimiento compartido en los equipos de desarrollo de software, con el fin de poder ser analizadas en la construcción de la estrategia y de esta forma incluir elementos que pudieran solventar dicha deficiencia y darle un apoyo a la ingeniería de requisitos, mediante la construcción de entendimiento compartido haciendo uso de prototipado ágil, lo cual mejore el desempeño de los participantes y el cumplimiento de los objetivos con respecto a la obtención del producto software de acuerdo con las expectativas del cliente, de forma que todos puedan establecer un lenguaje común de forma ágil, tangible y visual, y que aporte en la consecución de las necesidades del usuario, donde éste participe y pueda estar de acuerdo con lo que se va a construir en el producto final.

4.1.2 Actividad 1 de SME: Especificación de los requisitos de la estrategia:

En la actividad de especificación de requisitos de la ingeniería de métodos situacionales (SME), se identifican, documentan y comunican los requisitos de la estrategia. Esta debe contribuir a la formación de un equipo multidisciplinario, estimular la colaboración, fomentar las iteraciones y la retroalimentación, facilitar el desarrollo y la evaluación del conocimiento común. También debe hacer hincapié en la documentación precisa, la validación continua, la adaptabilidad al cambio, la experiencia del usuario y el apoyo a la creación rápida de prototipos.

Los requisitos son las características y funciones que debe tener la estrategia para satisfacer las necesidades de los usuarios, cada requisito especificado abordará los problemas o carencias detectados en el proceso de construir un entendimiento compartido en la ingeniería de requisitos. Estos requisitos son una hoja de ruta que ayudaran a guiar y planificar el enfoque de la estrategia hacia la mejora del entendimiento compartido de la ingeniería de requisitos.

4.1.3 Actividad 2 de SME: Selección de los componentes del método

Para llevar a cabo esta actividad, se realizaron las tareas de caracterización del entorno y comparación de los elementos de contenido del método, tareas que se detallan a continuación.

4.1.3.1 Caracterización del entorno:

El propósito de esta tarea, fue identificar aspectos generales y elementos de contenido del método que hayan sido previamente utilizados, investigados o analizados por quienes han diseñado, ejecutado y validado actividades para crear entendimiento compartido en los procesos de desarrollo de software o ingeniería de requisitos que deberá tener la estrategia. Inicialmente, como resultado del análisis de las fuentes de información, se identificaron los siguientes aspectos, elementos y características que se deberán considerar y que deben ser parte de la estrategia:

- Identificar a los stakeholders.
- Formación de grupos heterogéneos.
- Definir los objetivos del proyecto.
- Seleccionar un enfoque metodológico.
- Definir roles tanto para la aplicación del proceso como para la ejecución de las actividades colaborativas.
- Definir actividades colaborativas para la ejecución en la ingeniería de requisitos.
- Definir estrategias de comunicación que permitan la construcción de un entendimiento compartido.
- Desarrollar y medir un entendimiento compartido mediante artefactos, modelados, glosarios, y formalización de requisitos.
- Establecer un proceso de gestión de cambios en los requisitos para detectar y corregir problemas temprano.
- Desarrollo iterativo del prototipo.
- Enfoque orientado a la calidad del prototipo.
- Retroalimentación continua.

4.1.3.2 Comparación de los elementos de contenido de la estrategia

El objetivo fue encontrar elementos comunes utilizados en investigaciones anteriores que pudieran incluirse en la estrategia. Para ello, se compararon datos de estudios primarios e investigaciones afines mediante los mapeos sistemáticos de literatura realizados previamente. En esta tarea se detallan en profundidad las características que se consideran esenciales para el éxito de la estrategia, las cuales se basan; en la creación de equipos multidisciplinarios, el establecimiento de objetivos específicos y la elección de enfoques adecuados. También, hace hincapié en funciones y tareas cooperativas claramente definidas, técnicas de comunicación y una evaluación continua del entendimiento compartido. La aplicación temprana del proceso de gestión de cambios, y un proceso iterativo que mantiene la calidad a lo largo de todo el proceso de desarrollo del prototipo. La adaptabilidad ágil a los requisitos cambiantes del proyecto es posible gracias a la retroalimentación continua, lo que garantiza el éxito.

Por último, se propone una serie de productos de trabajo que los autores han reconocido como componentes cruciales para lograr el éxito en la construcción de un entendimiento común. Esto involucra la creación de equipos y la distribución de funciones. Para obtener los requisitos se realizan entrevistas, reuniones con clientes, usuarios y partes interesadas. Estas necesidades se documentan con todo detalle, se

estudia detenidamente su importancia para asignar un orden de prioridad. Se utiliza un informe para ayudar a validar los requisitos y seguir un proceso gestión de requisitos. Para que esto funcione, se registran los cambios y se insta a las partes interesadas a que aporten sus comentarios.

4.1.4 Actividad 3 de SME: Ensamblaje de los componentes de la estrategia

Una estrategia, en el contexto de la ingeniería de sistemas o en cualquier otro ámbito, es un método deliberado y planificado que se utiliza para afrontar un reto o alcanzar un objetivo determinado. Para alcanzar un objetivo deseado, implica identificar recursos, esbozar procesos o actividades precisas, asignar tareas y evaluar resultados [33].

Para definir la estructura que tendrían los elementos de contenido y características del método previamente identificado, se utilizó como fundamento el ensamblaje de la estrategia para la definición de siete fases (planificación, identificación y análisis de requisitos, creación del prototipo no funcional, gestión de requisitos, construcción del prototipo funcional y de socialización).

Para el ensamblaje de los componentes de la estrategia se adopta el manifiesto ágil, el cual se fundamenta no sólo en los resultados, sino también en los valores, principios y prácticas que la sustentan. Las metodologías ágiles permiten responder ágilmente a los cambios, manteniendo las condiciones del proyecto y logrando una gestión más eficiente de las dificultades, reduciendo costos y aumentando la productividad [34]. Los valores y principios establecidos son esenciales para el éxito del modelo ágil, ya que crean un contexto propicio para la colaboración efectiva entre programadores y clientes. El manifiesto hace énfasis en cuatro valores principales: comunicación, simpleza, retroalimentación y valor [15].

La comunicación es la clave para mantener el proyecto en curso y asegurar la alineación entre el equipo de desarrollo y las partes interesadas. La simplicidad se enfoca en lo esencial, permitiendo una rápida y efectiva utilización del software. La retroalimentación constante del equipo y las partes interesadas facilita la detección temprana de errores y áreas de mejora. Valor se traduce en flexibilidad del equipo ante cambios y una comunicación efectiva para abordar desafíos y lograr resultados óptimos.

5. Construcción de la estrategia

Considerando lo anterior, el éxito de un proyecto de desarrollo de software depende de la construcción de un entendimiento compartido en los requisitos [26]. Una estrategia ágil de creación de prototipos (considerando los beneficios que podría traer en este contexto) podría ayudar a lograr este objetivo, ya que permite a las partes interesadas en el proyecto interactuar con los prototipos y con el cliente, de tal forma que se puedan ofrecer retroalimentación en una fase temprana del proceso de desarrollo, lo que ayuda a lograr un entendimiento compartido y generar mejores requisitos del producto que se va a construir. La siguiente figura (Ver figura 1) muestra las fases de la estrategia sugerida y cómo se relacionan entre sí. Es importante considerar que este artículo se enfoca en mostrar el uso de SME para llegar a la estrategia, sin embargo, cada una de las fases mostradas en la figura están formadas por unas actividades y tareas específicas que permiten lograr los objetivos de cada una de las fases establecidas.

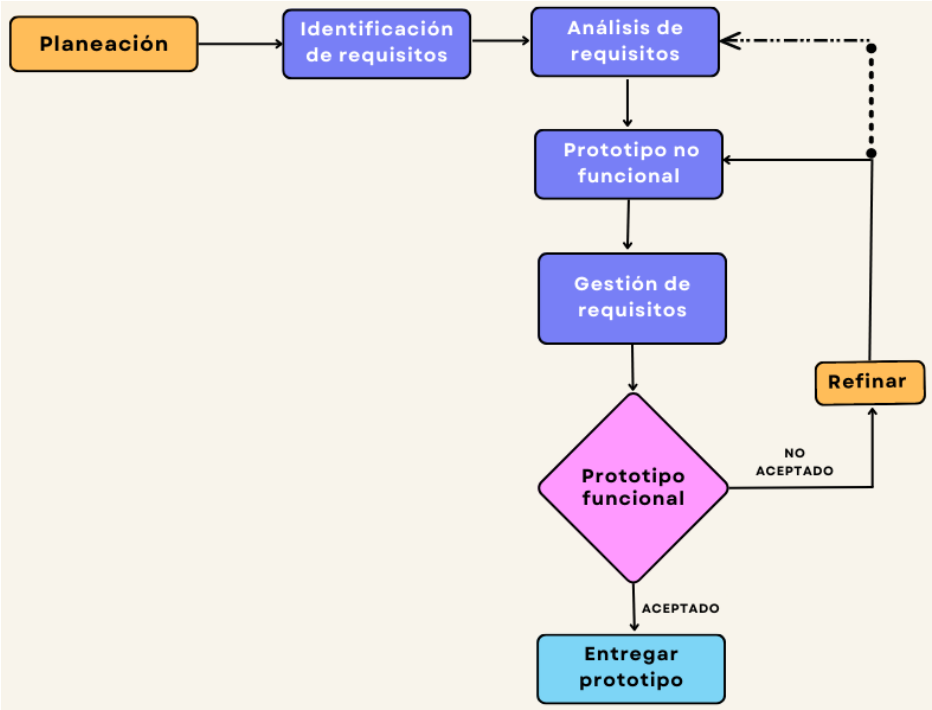

Figura 1: Representación gráfica de las fases de la estrategia

Una estrategia con prototipado ágil que permita construir un entendimiento compartido tiene algunas características importantes, las cuales se logran fomentando reuniones frecuentes y sesiones de trabajo en persona para discutir requisitos, el diseño, prototipos e involucrar a todas las partes interesadas relevantes desde el inicio del proyecto. También se fomenta la comunicación abierta y la participación activa de todos, se realizan pruebas de usabilidad de los prototipos con los usuarios, solicitando retroalimentación de los miembros del equipo en cada iteración, se enfoca en la mejora continua, abierto a cambios y ajustes en los requisitos, diseño del prototipo a medida que se descubren nuevas necesidades o se obtiene nueva información y métricas o indicadores para evaluar el nivel de entendimiento compartido.

A continuación, se abordan algunos atributos claves de la estrategia:

- La comunicación cara a cara: permite que todas las partes involucradas estén alineadas y exista el entendimiento compartido del proyecto.
- Participación oportuna de todas las partes interesadas: lo que implica que estén involucradas desde el inicio del proceso de desarrollo.
- Diseño centrado en el usuario: plasmado a través de la creación de prototipos visuales y pruebas con los usuarios finales para validar las funcionalidades del software y crear entendimiento compartido.
- Prototipado iterativo: implica repetir y mejorar continuamente el prototipo, permitiendo una mayor flexibilidad, adaptabilidad y participación de los interesados a lo largo del proceso.
- Evaluación y mejora continua: mediante la retroalimentación de los usuarios y la implementación de nuevas características y funcionalidades en el software.
- Foco en la colaboración: todas las partes interesadas trabajan juntas para lograr los objetivos del proyecto y crear un entendimiento compartido.
- Flexibilidad y adaptabilidad: sirven para afrontar los cambios y las nuevas situaciones.
- Medición del entendimiento compartido: se realiza para asegurarse de que todos los miembros del equipo están de acuerdo y entienden exactamente lo que se espera de ellos; así mismo ayuda a detectar inmediatamente cualquier problema con la

comprensión de los requisitos y permitir al equipo tomar medidas para solucionarlo antes de que se convierta en un problema grave.

6. Discusión

La estrategia de creación de prototipos ágil en la ingeniería de requisitos puede resultar exitosa para construir un entendimiento compartido. Sin embargo, este éxito requiere la consideración de varios aspectos importantes y la identificación de las brechas que aún deben abordarse. En esta discusión se abordan los componentes necesarios para implementar esta estrategia.

Comunicación: La estrategia se diseñó para garantizar que la comunicación efectiva estuviera en el centro de la ingeniería de requisitos. Esto se consigue mediante una planificación exhaustiva, el fomento de la comunicación bidireccional, la elaboración de documentación clara, la participación en un trabajo de equipo activo y la recepción continua de comentarios. Cada etapa del plan se centra en mejorar la comunicación entre todas las partes interesadas, lo que contribuye al éxito del proyecto y crea una base sólida para el entendimiento.

Entendimiento compartido: La estrategia se centró en la colaboración activa para construir un entendimiento compartido sólido. Desde el comienzo de la estrategia, los equipos multidisciplinarios deben colaborar, compartiendo conocimientos y habilidades. Lo que ayuda a disipar dudas y evitar confusiones. Y así conducir a un proceso de identificación y análisis de requisitos más eficaz. Las fases de creación de prototipos también fueron esenciales, ya que ofrece representaciones visuales que ayudan a la construcción de un entendimiento compartido y retroalimentación eficiente.

Colaboración: La colaboración es un componente fundamental en la estrategia, y se logra mediante la identificación de roles, reuniones regulares, comunicación, trabajo en equipo, documentación compartida y retroalimentación activa. Estos componentes se combinan para promover una colaboración efectiva y constante entre todas las partes interesadas a lo largo de la ingeniería de requisitos.

Prototipos ágiles: La estrategia propone la creación de prototipos ágiles mediante un enfoque iterativo, representación visual clara, retroalimentación activa, alineación con requisitos, mejoras continuas, y retroalimentación. Estas acciones aseguran que los prototipos sean herramientas efectivas para construir un entendimiento compartido y desarrollar requisitos precisos y alineados con las necesidades de los usuarios.

Retroalimentación continua: La estrategia promueve reuniones específicas, prototipado iterativo, canales de comunicación abiertos, documentación, evaluación iterativa, y un análisis de la retroalimentación. Para asegurar que la retroalimentación sea recopilada, analizada y utilizada de manera efectiva para mejorar los requisitos y la solución a lo largo del proyecto de ingeniería de requisitos.

En conclusión, el planteamiento de la estrategia busca demostrar ser un medio eficaz para resolver los problemas de comunicación, entendimiento compartido, colaboración, creación de prototipos ágiles y retroalimentación continua que plantea la ingeniería de requisitos. Estos pasos se propusieron a lo largo de todas las fases, lo que mejorará la precisión de los requisitos y dará lugar a un producto final que satisfaga las demandas y expectativas de los usuarios.

7. Conclusiones

La estrategia de prototipado ágil para construir entendimiento compartido en la ingeniera de requisitos basada en la ingeniería de métodos situacionales, es un enfoque innovador a los retos que plantea la ingeniería de requisitos en el proceso de desarrollo de software. Esta fusión de metodológica no solo permite construir un entendimiento compartido entre las partes interesadas, sino que favorece una comunicación efectiva y una validación temprana, aprovechando la visualización de prototipos tangibles y funcionales. Sin embargo, su diferenciador fundamental radica en la flexibilidad de la

Ingeniería de Métodos Situacionales, que permite adaptar enfoques y herramientas a las necesidades específicas de cada proyecto. Esta estrategia busca eliminar requisitos ambiguos y costosas repeticiones de trabajo, lo que ahorra una cantidad significativa de tiempo y recursos. Además, promueve una colaboración interdisciplinaria más sólida, enriqueciendo la comprensión colectiva de los requisitos del proyecto. La validación continua, respaldada por la retroalimentación constante y la iteración, reduce al mínimo la posibilidad de errores costosos en etapas posteriores a la ingeniería de requisitos.

En resumen, la estrategia que aquí se presenta proporciona un enfoque práctico y dinámico para la creación ágil de prototipos en la ingeniería de requisitos, contribuyendo a crear un sólido entendimiento compartido y al éxito de la estrategia.

Contribuciones de autor: Conceptualización, I.D.L., P.H.R. y V.A.D.; metodología, I.D.L., P.H.R. y V.A.D.; software, I.D.L.; validación, I.D.L., P.H.R. y V.A.D.; análisis formal, I.D.L.; investigación, I.D.L., P.H.R. y V.A.D.; curación de datos, I.D.L., P.H.R. y V.A.D.; redacción — preparación del borrador original, I.D.L.; redacción: revisión y edición, I.D.L., P.H.R. y V.A.D.; visualización, I.D.L.; supervisión, P.H.R., y V.A.D.; administracón de proyectos, I.D.L., P.H.R. y V.A.D.; Todos los autores han leído y aceptado la versión publicada del manuscrito.

Fondos: No aplica.

Conflictos de interés: Los autores declaran no tener ningún conflicto de intereses.

Referencias

1. N. N. B. Abdullah, S. Honiden, H. Sharp, B. Nuseibeh, y D. Notkin, "Communication patterns of agile requirements engineering," *Proc. 1st Agil. Requir. Eng. Work. AREW'11 - Conjunction with ECOOP'11*, pp. 1–4, 2011, doi: 10.1145/2068783.2068784.
2. E. Bittner y J. Leimeister, "Creating shared understanding in heterogeneous work groups: Why it matters and how to achieve it," *J. Manag. Inf. Syst.*, vol. 31, no. 1, pp. 111–144, 2014, doi: 10.2753/MIS0742-1222310106.
3. M. Hummel, C. Rosenkranz, y R. Holten, "The role of shared understanding in distributed scrum development: An empirical analysis," *24th Eur. Conf. Inf. Syst. ECIS 2016*, 2016.
4. M. Arias Chaves, "La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software Michael," *Rev. las Sedes Reg.*, vol. VI, no. 2215–2458, pp. 1–13, 2005, [Online]. Available: https://www.redalyc.org/articulo.oa?id=66612870011
5. D. Carrizo y J. Rojas, "Methodologies, techniques and tools in requirements engineering: A systematic mapping," *Ingeniare*, vol. 26, no. 3, pp. 473–485, 2018, doi: 10.4067/S0718-33052018000300473.
6. N. Fiquitiva Segura y M. A. López Ruiz, "PROTOTIPO DE APLICATIVO PARA ESPECIFICAR REQUERIMIENTOS DE SOFTWARE," vol. 13, 2015.
7. M. Käpyaho y M. Kauppinen, "Agile requirements engineering with prototyping: A case study," *2015 IEEE 23rd Int. Requir. Eng. Conf. RE 2015 - Proc.*, pp. 334–343, 2015, doi: 10.1109/RE.2015.7320450.
8. K. Elghariani y N. Kama, "Review on Agile requirements engineering challenges," *2016 3rd Int. Conf. Comput. Inf. Sci. ICCOINS 2016 - Proc.*, pp. 507–512, 2016, doi: 10.1109/ICCOINS.2016.7783267.
9. L. Zamudio, A. A. B, y C. Tripp, "in Agile Software Development Methods," *Comput. Sci. Its Appl.*, vol. 2, pp. 683–698, 2017, doi: 10.1007/978-3-319-62404-4.
10. B. Henderson-Sellers, J. Ralyté, P. J. Ågerfalk, y M. Rossi, *Situational method engineering*. 2014. doi: 10.1007/978-3-642-41467-1.
11. N. Devadiga, "Tailoring architecture centric design method with rapid prototyping," in *Proceedings of the 2nd International Conference on Communication and Electronics Systems, ICCES 2017*, Mar. 2018, vol. 2018-Janua, pp. 924–930. doi: 10.1109/CESYS.2017.8321218.
12. R. F. Ciriello, A. Richter, y G. Schwabe, "When prototyping meets storytelling: Practices and malpractices in innovating software firms," *Proc. - 2017 IEEE/ACM 39th Int. Conf. Softw. Eng. Softw. Eng. Pract. Track, ICSE-SEIP 2017*, pp. 163–172, 2017, doi: 10.1109/ICSE-SEIP.2017.24.
13. A. Hoffmann, E. Alice, C. Bittner, y J. M. Leimeister, "Emergence of Mutual and Shared Understanding in the System Development Process . In : Requirements Engineering : Foundation for Software Quality , Lecture The Emergence of Mutual and Shared Understanding in the System Development Process," no. 2013, pp. 174–189.
14. J. A. J. Builes, H. J. Hernández-Reinoza, y C. Villota-Ibarra, "Metodología lúdica para la enseñanza de la ingeniería de requisitos basada en esquemas preconceptuales," *Rev. EIA*, vol. 18, no. 35, 2021, doi: 10.24050/reia.v18i35.1394.
15. E. Maida y J. Pacienzia, "Metodologías de desarrollo de software," *Bibl. Digit. la Univ. Católica Argentina*, p. 117, 2018, [Online]. Available: http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf
16. J. M. Cadavid, A. N., Martínez, J. D. F., & Vélez, "Revisión de metodologías ágiles para el desarrollo de software.," *Prospectiva*,

vol. 11, no. 2, pp. 30–39, 2013, [Online]. Available: http://www.redalyc.org/articulo.oa?id=496250736004%0ACómo
17. E. Pineda Ballesteros, F. R. Tellez Acuña, y J. Medina Cruz, "Requerimientos de software: prototipado, software heredado y análisis de documentos," *Ing. y Desarro.*, vol. 37, no. 02, pp. 327–345, 2019, doi: 10.14482/inde.37.2.1053.
18. S. G. Rivadeneira Molina, "Metodologías ágiles enfocadas al modelado de requerimientos," *Inf. Científicos Técnicos - UNPA*, vol. 5, no. 1, pp. 1–29, 2014, doi: 10.22305/ict-unpa.v5i1.66.
19. S. A. Mosquera Moreno, "Requirements Engineering in Agile Development Methods Engenharia de Requisitos , em Métodos Ágeis de Desenvolvimento La Ingeniería de Requisitos en los Métodos de Desarrollo Agiles," *La Ing. Requisitos En Los Métodos Desarro. Agil. Resum.*, no. October, 2016, doi: 10.13140/RG.2.1.3303.7687.
20. S. Ghobadi y L. Mathiassen, "Perceived barriers to effective knowledge sharing in agile software teams," *Inf. Syst. J.*, vol. 26, no. 2, pp. 95–125, 2016, doi: 10.1111/isj.12053.
21. X. Yu y S. Petter, "Understanding agile software development practices using shared mental models theory," *Inf. Softw. Technol.*, vol. 56, no. 8, pp. 911–921, 2014, doi: 10.1016/j.infsof.2014.02.010.
22. M. Corvera Charaf, C. Rosenkranz, y R. Holten, "The emergence of shared understanding: Applying functional pragmatics to study the requirements development process," *Inf. Syst. J.*, vol. 23, no. 2, pp. 115–135, 2013, doi: 10.1111/j.1365-2575.2012.00408.x.
23. A. Nakakawa, P. Van Bommel, E. H. A. Proper, y H. J. B. F. Mulder, "A situational method for creating shared understanding on requirements for an enterprise architecture," *Int. J. Coop. Inf. Syst.*, vol. 27, no. 4, 2018, doi: 10.1142/S0218843018500107.
24. N. W. Varas Cortés, "Una técnica basada en Prototipado rápido para favorecer el entendimiento compartido entre el cliente y el desarrollador," 2021, [Online]. Available: https://repositorio.uchile.cl/handle/2250/181671?show=full
25. B. Kitchenham y P. Brereton, "A systematic review of systematic review process research in software engineering," *Inf. Softw. Technol.*, vol. 55, no. 12, pp. 2049–2075, 2013, doi: 10.1016/j.infsof.2013.07.010.
26. M. Glinz y S. A. Fricker, "On shared understanding in software engineering: an essay," *Comput. Sci. - Res. Dev.*, vol. 30, no. 3–4, pp. 363–376, 2015, doi: 10.1007/s00450-014-0256-x.
27. J. B. Windeler, L. M. Maruping, L. P. Robert, y C. K. Riemenschneider, "E-profiles, conflict, and shared understanding in distributed teams," *J. Assoc. Inf. Syst.*, vol. 16, no. 7, pp. 608–645, 2015, doi: 10.17705/1jais.00401.
28. J. L. Pérez-Medina y J. Vanderdonckt, "Sketching by Cross-Surface Collaboration," *Adv. Intell. Syst. Comput.*, vol. 918, pp. 386–397, 2019, doi: 10.1007/978-3-030-11890-7_38.
29. C. Werner, Z. S. Li, N. Ernst, y D. Damian, "The Lack of Shared Understanding of Non-Functional Requirements in Continuous Software Engineering: Accidental or Essential?," *Proc. IEEE Int. Conf. Requir. Eng.*, vol. 2020-Augus, pp. 90–101, 2020, doi: 10.1109/RE48521.2020.00021.
30. K. A. Mohamed, M. A. Ellatif, y M. S. Farhan, "Using ontology-based concept maps for requirements engineering: A case study," *ICENCO 2017 - 13th Int. Comput. Eng. Conf. Boundless Smart Soc.*, vol. 2018-Janua, pp. 366–371, 2018, doi: 10.1109/ICENCO.2017.8289816.
31. Y. Liu, S. van Nederveen, y M. Hertogh, "Understanding effects of BIM on collaborative design and constructionAn empirical study in China," *Int. J. Proj. Manag.*, vol. 35, no. 4, pp. 686–698, 2017, doi: 10.1016/j.ijproman.2016.06.007.
32. N. Kunicina, A. Zabasta, A. Patlins, I. Bilic, y J. Peksa, "Prototyping process in education and science," *2020 IEEE 61st Annu. Int. Sci. Conf. Power Electr. Eng. Riga Tech. Univ. RTUCON 2020 - Proc.*, 2020, doi: 10.1109/RTUCON51174.2020.9316550.
33. Porter Michael E., "¿Qué es la estrategia?," *Harv. Bus. Rev.*, no. March, pp. 100–117, 2008.
34. A. López Gil, "Estudio comparativo de metodologías tradicionales y ágiles para proyectos de Desarrollo de Software," *Univ. Valladolid*, p. 139, 2018, [Online]. Available: https://uvadoc.uva.es/handle/10324/32875%0Ahttps://uvadoc.uva.es/handle/10324/32875%0Ahttps://agileexperience.es/wp-content/uploads/2020/06/TFG-I-1015.pdf%0Ahttp://uvadoc.uva.es/handle/10324/32875