

**Elementos de prototipado como apoyo al proceso de desarrollo software:
revisión sistemática de literatura**



Karen Viviana Bravo Rivera

Directora:

PhD. Vanessa Agredo Delgado

**Facultad de Ingenierías
Corporación Universitaria de Comfacaucá - Unicomfacaucá
Programa de Ingeniería de Sistemas
Popayán, Cauca
Julio 2024**

Dedicatoria

Quiero dedicar esta tesis a mis padres, quienes me han acompañado y apoyado incondicionalmente en este arduo pero gratificante proceso. También quiero dedicar esta tesis a mi amado gato Silver, cuya amorosa compañía ha sido una fuente constante de alegría para mí.

Contenido

1. Resumen.....	6
2. Abstract.....	7
3. CAPÍTULO 1	9
3.1. Formulación del problema	9
3.2. Justificación.....	11
3.3. Objetivos	13
3.3.1. Objetivo general	13
3.3.2. Objetivos específicos	13
3.4. Organización del documento	14
3.5. Marco teórico.....	15
3.5.1. Prototipo.....	15
3.5.2. Ingeniería de software.....	15
3.5.3. Desarrollo de software	17
3.5.4. Partes interesadas (Involucrados).....	18
3.5.5. Ingeniería de requisitos	18
3.6. Trabajos relacionados	18
3.6.1. Estrategias de creación de prototipos: revisión de la literatura e identificación de variables críticas	18
3.6.2. Análisis de metodologías de desarrollo de software	19
3.6.3. La ingeniería de requisitos y su importancia en el desarrollo de proyectos de software.....	19
3.6.4. Requisitos de software: prototipado, software heredado y análisis de documentos.....	20
3.6.5. Enfoques de creación de prototipos y desarrollo de software	20
3.6.6. Realidad mixta en la creación de prototipos de diseño: una revisión sistemática	21
3.6.7. Diseño mediante prototipos: apoyo estratégico al desarrollo de productos basado en prototipos.....	21
3.6.8. Implementación de métodos ágiles para la simulación de casos de uso y prototipado en el proceso de desarrollo de software.....	22
3.6.9. Ingeniería de requisitos ágil con creación de prototipos: un caso de estudio	22
3.6.10. Creación de prototipos de usabilidad y experiencia de usuario: una técnica sencilla para equipos ágiles	23

3.7. Métodos de revisión de literatura disponibles.....	24
4. CAPÍTULO 2	30
4.1. Identificación de la necesidad de la revisión	30
4.2. Bases de datos para la aplicación de la búsqueda de la información	32
4.3. Estrategia de búsqueda.....	32
4.4. Evaluación de la calidad	34
4.5. Proceso para la selección de estudios primarios.....	35
5. CAPÍTULO 3	37
5.1. Resultados y análisis de los resultados	37
5.2. Presentación del informe de revisión.....	38
6. CAPÍTULO 4	61
7. CAPÍTULO 5	73
7.1. Conclusiones	73
7.2. Trabajos futuros.....	75
8. Referencias	76

Lista de tablas

Tabla 1. Ventajas y desventajas de los diferentes marcos y metodologías para llevar a cabo revisiones sistemáticas de la literatura.....	28
Tabla 2. Preguntas de investigación utilizadas para la revisión sistemática.	30
Tabla 3. Palabras clave, sinónimos y abreviaturas.	33
Tabla 4. Criterios PICOC.	33
Tabla 5. Cadena de búsqueda.....	33
Tabla 6. Cadena de búsqueda aplicada a cada una de las bases de datos	34
Tabla 7. Preguntas de calidad para la selección de estudios primarios.....	34
Tabla 8. Valoración de las preguntas de calidad.	35
Tabla 9. Criterios de inclusión de la investigación.	35
Tabla 10. Criterios de exclusión de la investigación.	36

Lista de figuras

Figura 1. Aportes a la investigación de cada una de las bases de datos.	38
Figura 2. Ventajas del uso de prototipos en los procesos de desarrollo de software.....	39
Figura 3. Distribución de investigaciones de la utilización de prototipos en el proceso de desarrollo de software.	42
Figura 4. Desventajas del uso de prototipos en los procesos de desarrollo de software.....	44
Figura 5. Tipos de prototipado que existen en los procesos de desarrollo de software.....	45

Figura 6. Estrategias o técnicas de prototipado que existen en los procesos de desarrollo de software.	49
Figura 7. Fases del proceso de desarrollo de software donde se utiliza el prototipado.	55
Figura 8. Herramientas de prototipado utilizadas en el proceso de desarrollo de software.	57
Figura 9. Edad de los encuestados	62
Figura 10. Género de los encuestados.	63
Figura 11. Años de experiencia en desarrollo de software.	64
Figura 12. Participación en procesos de desarrollo de software que involucren el prototipado.	65
Figura 13. Herramientas utilizadas para la creación de prototipos.	66
Figura 14. Resultados obtenidos en la revisión de la literatura sobre el prototipado son útiles para el desarrollo de software.	67
Figura 15. Resultados para seleccionar un tipo de prototipo para ser aplicado en un contexto de desarrollo de software específico.	69
Figura 16. Resultados sobre la claridad de los datos presentados para su aplicación en el desarrollo de software.	70
Figura 17. Resultados sobre si sería beneficioso realizar más investigaciones sobre prototipado en el contexto del desarrollo de software.	71

1. Resumen

El propósito fundamental de los procesos de desarrollo de software es generar de forma eficiente y eficaz productos de software que cumplan y satisfagan los requisitos y expectativas de los usuarios. Estos procesos transforman ideas y necesidades en productos de software funcionales y de alta calidad. Sin embargo, los desarrolladores de software suelen necesitar más claridad, comprensión y detalle en la especificación de requisitos. Estos problemas dificultan el logro de la eficiencia en el proceso de desarrollo y, en última instancia, la capacidad de satisfacer plenamente las necesidades de los usuarios al final del proyecto. En este sentido, la definición de prototipos busca proporcionar un acercamiento que permita comprender mejor el software y aclarar dichos requisitos, se basan en la premisa de que es más fácil entender, evaluar y refinarlos cuando se tiene una representación visual en lugar de depender únicamente de descripciones escritas. Es por esto que este proyecto buscaba realizar una revisión sistemática de la literatura que permitiera analizar cuáles son los elementos de prototipado que pueden apoyar los procesos de desarrollo de software, reconociendo que la falta de claridad y detalle en la especificación de requisitos puede obstaculizar la eficiencia del proceso de desarrollo y la satisfacción de las necesidades de los usuarios, esto debido a que se puede identificar que en la literatura existe mucho material disperso sobre esta temática de creación de prototipos y es necesario poder categorizar, organizarlo con el fin de utilizarlo de forma guiada y sistemática cuando se requiera. La revisión sistemática de la literatura tuvo como objetivo caracterizar y analizar los distintos elementos y enfoques de creación de prototipos utilizados en la actualidad para apoyar y mejorar los procesos de desarrollo de software. Por lo tanto, la idea principal de este proyecto fue obtener una caracterización que definiera qué elementos de creación de prototipos pueden apoyar los procesos de desarrollo de software, buscando así reducir y mitigar las lagunas, la ambigüedad o la falta de información durante las fases de concepción del proyecto, garantizando así una comprensión más clara y precisa de los requisitos del producto. Para ello, se diseñó una revisión sistemática de la literatura, con el propósito de establecer una estrategia de búsqueda efectiva que facilitara la identificación exhaustiva y precisa de estudios o documentos relevantes al tema de investigación. El diseño incluyó la formulación de preguntas de investigación, la identificación de fuentes de información relevantes, la selección de términos y palabras clave, así como el desarrollo de estrategias de búsqueda efectivas. Tras ejecutar este diseño, los resultados obtenidos incluyeron la identificación y clasificación de diversas técnicas de prototipado, así como de las herramientas más utilizadas para la creación de prototipos. Además, se mencionaron las ventajas y desventajas del prototipado, así como la importancia de incluir prototipos en las fases iniciales y críticas del desarrollo, considerándolo como una valiosa estrategia para mejorar la comprensión y la comunicación, dando lugar a productos de software más eficaces y satisfactorios.

La validación de los resultados se realizó a través de la presentación y análisis de la utilidad percibida por los desarrolladores de software sobre el compendio de información relacionada con el uso de prototipos en el desarrollo de software,

obtenida a partir de una revisión exhaustiva de la literatura existente. Este análisis pretendía validar y conocer la utilidad de los prototipos y si contribuyen a mejorar los procesos de desarrollo. Finalmente, se identificaron resultados a partir de las respuestas de una encuesta realizada a 13 estudiantes de octavo semestre de ingeniería de sistemas de la Corporación Universitaria de ComfacaUCA-UnicomfacaUCA. Con esto, se puede afirmar que la revisión de literatura sobre prototipado en el desarrollo de software fue útil y ha aportado información valiosa a los encuestados. Finalmente, se presentan las conclusiones obtenidas de la investigación realizada, consolidando y analizando los hallazgos obtenidos a lo largo del estudio, proporcionando una visión global de su relevancia en el contexto de la investigación, además de señalar direcciones para trabajos futuros identificando posibles áreas de mejora o nuevas líneas de investigación.

2. Abstract

The fundamental purpose of software development processes is to efficiently and effectively generate software products that meet and satisfy user requirements and expectations. These processes transform ideas and needs into high-quality, functional software products. However, software developers often need more clarity, understanding and detail in requirements specification. These problems hinder the achievement of efficiency in the development process and, ultimately, the ability to fully satisfy user needs at the end of the project. In this sense, the definition of prototypes seeks to provide an approach to better understand the software and clarify those requirements, based on the premise that it is easier to understand, evaluate and refine them when there is a visual representation rather than relying solely on written descriptions. This is why this project sought to conduct a systematic review of the literature to analyze what are the elements of prototyping that can support software development processes, recognizing that the lack of clarity and detail in the specification of requirements can hinder the efficiency of the development process and the satisfaction of user needs, because it can be identified that in the literature there is much scattered material on this topic of prototyping and it is necessary to categorize, organize it in order to use it in a guided and systematic way when required. The systematic literature review aimed to characterize and analyze the different elements and approaches to prototyping currently used to support and improve software development processes. Therefore, the main idea of this project was to obtain a characterization that would define which prototyping elements can support software development processes, thus seeking to reduce and mitigate gaps, ambiguity, or lack of information during the project conception phases, ensuring a clearer and more accurate understanding of product requirements. To this end, a systematic literature review was designed, with the purpose of establishing an effective search strategy that would facilitate the exhaustive and accurate identification of studies or documents relevant to the research topic. The design included the formulation of research questions, the identification of relevant sources of information, the selection of terms and keywords, as well as the development of effective search strategies. After executing this design, the results obtained included the identification and classification of various prototyping

techniques, as well as the most commonly used tools for prototyping. In addition, the advantages and disadvantages of prototyping were mentioned, as well as the importance of including prototypes in the initial and critical phases of development, considering it as a valuable strategy to improve understanding and communication, resulting in more effective and satisfactory software products.

The validation of the results was performed through the presentation and analysis of the usefulness perceived by software developers on the compendium of information related to the use of prototypes in software development, obtained from an exhaustive review of the existing literature. This analysis aimed to validate and learn about the usefulness of prototypes and whether they contribute to improve development processes. Finally, results were identified from the answers of a survey made to 13 eighth semester students of systems engineering of the Corporación Universitaria de Comfacaucá- Unicomfacaucá. With this, it can be affirmed that the literature review on prototyping in software development was useful and has provided valuable information to the respondents. Finally, the conclusions obtained from the research conducted are presented, consolidating, and analyzing the findings obtained throughout the study, providing an overview of its relevance in the context of the research, as well as pointing out directions for future work by identifying possible areas for improvement or new lines of research.

3. CAPÍTULO 1

Este capítulo comienza con la formulación del problema, cuyo propósito es definir con precisión el problema que se va a investigar y proporcionar una orientación para la investigación. A continuación, viene la justificación de la investigación, que es esencial para demostrar por qué el problema es relevante y merece la pena investigarlo. Esta justificación destaca la importancia práctica o teórica del problema y su impacto en el campo correspondiente. Posteriormente, se identifican los objetivos, que establecen claramente lo que se espera lograr abordando el problema. Estos objetivos sirven de metas concretas y mensurables para el estudio, proporcionando una guía precisa para la investigación. La organización del documento se menciona para destacar la importancia de contar con una estructura coherente y bien definida, que facilite la comprensión a lo largo del trabajo. Le sigue el marco teórico, sección fundamental que ayuda a identificar conceptos y términos clave relevantes para la investigación. Por último, se realiza una revisión exhaustiva de los trabajos relacionados, que comprende un análisis crítico de las investigaciones anteriores relevantes para el área de estudio. Y se describen los métodos de revisión de la literatura existentes con el fin de determinar cuál era el más adecuado para el propósito de la investigación.

3.1. Formulación del problema

El desarrollo de software es una actividad creativa y crítica que ha tenido un alto grado de complejidad, ya que requiere del dominio de técnicas y tecnologías que permitan acelerar el despliegue, calidad y eficiencia del software, es por ello que es llevado a cabo principalmente por programadores, ingenieros y desarrolladores de software, estos roles interactúan y la dinámica entre ellos es la formalización de actividades relacionadas con el desarrollo de software y aplicaciones interactivas para los usuarios [1], que incluyen aspectos como la producción de software, desde las primeras etapas de especificación del sistema hasta el mantenimiento [2], la obtención de requisitos, el modelado cuidadoso de los requisitos junto con la validación sistemática que ayuda a reducir la incertidumbre sobre los objetivos del sistema, el objetivo de la validación de requisitos es confirmar si la especificación de requisitos se ajusta a las necesidades reales de los clientes, entre otros. Sin embargo, este proceso es complicado y resulta difícil producir una especificación de requisitos coherente, entendible por todos los interesados y completa [3].

El desarrollo de software es esencial en nuestra vida cotidiana y en los negocios por varias razones importantes. En primer lugar, los productos obtenidos de este desarrollo, buscan automatizar tareas repetitivas, lo que ahorra tiempo y recursos [4]. Además, impulsa la innovación al crear nuevas aplicaciones y servicios. Un buen software también mejora la experiencia del usuario, lo que se traduce en una mayor satisfacción del cliente. En resumen, el desarrollo de software busca hacer la vida más fácil y mejorar, además, el funcionamiento y productividad de las empresas [4].

Aunque definir los requisitos de un proyecto puede resultar complejo, la creación de prototipos se presenta como una herramienta invaluable para este proceso. Permite la creación de versiones preliminares del producto, facilitando la identificación y clarificación de requisitos clave a través de retroalimentación continua y validación temprana. Mejora la comunicación visualizando el producto, reduciendo malentendidos y detectando problemas de diseño antes de su desarrollo completo. En resumen, la creación de prototipos optimiza el proceso de desarrollo y garantiza que el producto final cumpla con las expectativas y necesidades de los usuarios [5]. Es por ello que es una forma de mejorar este proceso. Sin embargo, en el ámbito del desarrollo de software existe un problema en relación con la adopción de prototipos, técnica que puede brindar soporte al proceso de desarrollo [6]. Los factores que influyen en esta situación son diversos y complejos, ya que la calidad de los resultados obtenidos mediante la creación de prototipos depende de varios factores, entre ellos: Forma: El diseño y el aspecto del prototipo pueden influir en la percepción de la calidad y la facilidad de uso. Función: La capacidad del prototipo para cumplir los objetivos y funciones previstos es un factor determinante de su calidad. Usabilidad: La facilidad de uso y la experiencia del usuario al interactuar con el prototipo son factores cruciales para evaluar su calidad, y los aspectos contextuales, como el enfoque de desarrollo, el entorno organizativo y la infraestructura disponible, también pueden afectar a la calidad de los resultados obtenidos [7]. Además, algunas empresas aún prefieren utilizar métodos de diseño tradicionales, ya que consideran que la creación de prototipos puede implicar ciertos riesgos como por ejemplo costos adicionales y retrasos en el desarrollo [8]. Además de lo mencionado anteriormente, otra problemática es la existencia de mucha literatura dispersa sobre prototipado en el desarrollo de software, lo que crea un desafío significativo para los desarrolladores, ya que se ven a la tarea de discernir cuáles son los recursos más adecuados para sus necesidades específicas [9]. Esta sobrecarga de información puede llevar a los equipos a no saber por dónde empezar o cómo avanzar eficazmente en el proceso de prototipado lo que deja a los desarrolladores sin una orientación clara sobre cómo abordar este aspecto crucial del proceso de desarrollo de software [8], esta falta de orientación al no saber cómo usarla, no saber cómo empezar, cual es la mejor técnica, cuando se debería de usar, que fases, actividades o tareas conlleva el hacer prototipos, hace que se dificulte su implementación efectiva o que muchas veces ni siquiera se utilice, sabiendo todos los beneficios que puede traer en el momento de desarrollar software [9]. Es por esto, que una ausencia en la orientación clara puede obstaculizar la planificación y ejecución eficaz del proceso de prototipado, lo que puede traducirse en una pérdida de tiempo y recursos, y a su vez la falta de comprensión de las mejores prácticas puede conducir a la realización de prototipos que no cumplan los objetivos del proyecto ni satisfagan las necesidades de los usuarios finales [10].

Por eso, el principal problema radica en la falta de orientaciones claras y estructuradas sobre cómo aplicar el prototipado en distintos contextos de desarrollo de software, generando que los desarrolladores puedan sentirse abrumados por la variedad de técnicas, herramientas y enfoques disponibles, lo que dificulta la toma de decisiones informadas sobre qué método utilizar en cada fase del proceso de

desarrollo, considerando la diversidad de técnicas de prototipado, las cuales pueden generar confusión. Además, cada técnica tiene sus propias ventajas y desventajas, y seleccionar la adecuada para un proyecto específico puede ser todo un reto sin una orientación clara que describa cuándo y cómo aplicarla de forma eficaz [10]. Otro reto es la falta de una estructura clara para el proceso de prototipado, al no tener una guía que describa las distintas fases y actividades de la creación de prototipos, los desarrolladores pueden tener dificultades para planificar y ejecutar el proceso con eficacia. Pueden no estar seguros de qué pasos seguir primero, qué actividades realizar en cada fase o cómo integrar el prototipado en el flujo de trabajo general de desarrollo de software [9].

Por ello, esta revisión de la literatura planteada para este proyecto buscaba abordar esta problemática proporcionando un consolidado más organizado y estructurado para que los desarrolladores puedan utilizarlo con eficacia. Este consolidado de información pretendía proporcionar información detallada sobre las distintas técnicas, herramientas y enfoques de prototipado. Además, de describir claramente las diferentes fases y actividades del proceso de prototipado, proporcionando a los desarrolladores una ruta clara para planificar y ejecutar el proceso con eficacia. De este modo, los desarrolladores podrán obtener los beneficios del prototipado en el proceso de desarrollo de software, ya que los prototipos tienen diversos usos en función de los objetivos de cada proyecto [11]. Además, la selección adecuada de las herramientas y técnicas de prototipado puede influir significativamente en la eficacia y eficiencia del proceso de diseño, así como en la calidad del producto final. Además, es posible ofrecer recomendaciones prácticas a los diseñadores sobre cómo aprovechar al máximo estas herramientas y técnicas en su trabajo diario [12].

Por lo tanto, la pregunta de investigación para esta propuesta es: *¿Cuáles son los elementos de prototipado propuestos en la literatura que apoyan el proceso de desarrollo de software?* De esta manera, con el desarrollo de este trabajo se pretendía definir una caracterización a partir de la literatura existente, que permita apoyar y mejorar los procesos de desarrollo de software, integrando la creación de prototipos para que los equipos involucrados puedan eliminar cualquier ambigüedad o falta de información respecto a los requisitos del sistema a construir, compartiendo sus ideas, aportando sus conocimientos y diferentes habilidades, y así obtener una base de referencia común, de manera que trabajando colaborativamente se cumpla con el objetivo de determinar las necesidades que el usuario final requiere del producto de software a construir.

3.2. Justificación

La ingeniería de requisitos es de suma importancia para los procesos de desarrollo de software, ya que contribuye a la creación de productos de éxito. La ingeniería de requisitos desempeña un papel crucial a la hora de garantizar la satisfacción del cliente y la consecución de los objetivos del sistema [13]; como afirma Sommerville [14], la ingeniería de requisitos es el proceso de comunicación entre clientes, usuarios del software y desarrolladores, que garantiza una alineación eficaz de las expectativas. El proceso de ingeniería de requisitos reduce significativamente la

probabilidad de fracaso del proyecto. Esto se debe a que unos requisitos bien definidos proporcionan una visión concisa de lo que el software debe ser capaz de hacer, lo que permite orientar eficazmente las actividades, recursos y esfuerzos, con la consiguiente reducción de costes y retrasos [15]. Además, esta práctica contribuye a mejorar la calidad del software, ya que unos requisitos bien especificados pueden probarse eficazmente y, en consecuencia, cumplirse, lo que garantiza que el software final satisfaga las expectativas y necesidades de los usuarios [16].

Dentro de la ingeniería de requisitos existen tres actividades, la primera es la captura de requisitos es una actividad crucial en el desarrollo de software, que consiste en una intensa comunicación con el cliente/usuario intentando establecer claramente el problema que requiere una solución. Se deben utilizar diferentes técnicas combinadas para obtener la mayor cantidad de información posible. Algunas técnicas: entrevistas, cuestionarios, prototipos (en papel) y sesiones de lluvia de ideas, entre otras [17]. La segunda actividad corresponde a la definición de requisitos, que consiste en establecer de forma precisa y comprensible las necesidades y expectativas del sistema, asegurando una base sólida para el diseño y desarrollo del producto software [17]. Y, por último, la validación sistemática de los requisitos que es importante para reducir la incertidumbre sobre los objetivos del software y confirmar si la especificación de requisitos se ajusta a las necesidades reales de los clientes. Sin embargo, producir una especificación coherente y completa es una tarea complicada [3].

En este contexto, la creación de prototipos juega un papel fundamental en la obtención y validación de requisitos, en la experimentación y exploración de ideas, y en la comunicación tanto del equipo de desarrollo como con el usuario para apoyar la toma de decisiones [7]. El proceso de ingeniería de requisitos es una parte crucial de la ingeniería de software, y cada vez más se incluyen estrategias cuyo objetivo es descubrir, adquirir y detallar los requisitos [18]. Sin embargo, las empresas de desarrollo de software tienen problemas para comunicar sus ideas a diferentes audiencias, porque a menudo carecen de una estrategia de comunicación adecuada y no dedican tiempo y recursos suficientes a desarrollar habilidades de comunicación eficaces en sus equipos, ya que pueden estar más centrados en los aspectos técnicos y menos en la capacidad de transmitir ideas de forma clara y comprensible. Y en este sentido, se hace evidente la necesidad de utilizar estrategias y herramientas que mejoren, apoyen y faciliten la comunicación y el entendimiento de estos requisitos [7].

Una de estas herramientas es el prototipado, que puede definirse como esencial en el diseño y desarrollo de productos, ya que permite explorar conceptos, comunicar ideas, obtener opiniones de los usuarios y mejorar el diseño en las primeras fases del proceso de desarrollo [19], el cual tiene un sin número de beneficios en estos procesos, como por ejemplo, evitar el retrabajo en el desarrollo y permitir persuadir y colaborar con las partes interesadas [7]. Asimismo, permite encontrar fácilmente fallos de comprensión e incertidumbre sobre los requisitos y de esta forma, asegurar el éxito del proyecto [3], los prototipos se caracterizan por sus interacciones

colaborativas, convirtiéndose en una herramienta de apoyo con flexibilidad para adaptarse a diferentes situaciones de diseño, es decir diferentes actividades y personas de diferentes ámbitos que utilizan distintos medios y dispositivos [20].

En el ámbito del desarrollo de software en una organización, es crucial tener una comprensión clara de los objetivos que deben alcanzarse y fomentar la colaboración entre todos los involucrados para garantizar un desarrollo eficaz. En este contexto, la integración de prototipos en el proceso de desarrollo de software ofrece ventajas clave para la organización como [21]:

- Mejora en la comprensión de los requisitos: Los prototipos proporcionan una representación visual y funcional del software en desarrollo, lo que facilita a las partes interesadas la identificación y validación de los requisitos. Esto conduce a una mayor claridad y precisión en las especificaciones del proyecto.
- Reducción de riesgos: Los prototipos ayudan a reducir el riesgo de fracaso del proyecto al permitir la detección temprana de posibles problemas de usabilidad, funcionalidad o rendimiento. Esto permite realizar ajustes y mejoras antes de dedicar recursos importantes al desarrollo completo del software.
- Mejora de la comunicación y la coordinación: Los prototipos son una herramienta eficaz para mejorar la comunicación y la coordinación entre los desarrolladores y los clientes, ya que ofrecen una representación visual del producto en desarrollo que es más fácil de entender y evaluar.

Por lo anterior este proyecto busca mejorar y apoyar los procesos de desarrollo de software mediante la caracterización de toda la información relevante y necesaria que puede requerir un grupo de desarrollo para integrar la definición de prototipos en sus actividades, y de esta manera aprovechar los beneficios que trae consigo la creación de estos prototipos como elemento fundamental para crear un entendimiento común de los objetivos y necesidades del proyecto y eliminar cualquier ambigüedad o discordancia en cuanto a la dirección del desarrollo. Esto permitirá una comunicación fluida y una coordinación eficaz entre todos los involucrados en las actividades posteriores del proceso de desarrollo de software.

3.3. Objetivos

3.3.1. Objetivo general

Caracterizar la literatura científica existente sobre los elementos de prototipado que soportan el proceso de desarrollo de software.

3.3.2. Objetivos específicos

- Seleccionar un método adecuado para realizar la revisión bibliográfica de los elementos de prototipado que soportan el proceso de desarrollo de software.
- Diseñar el protocolo para la revisión de la literatura utilizando el método seleccionado.

- Ejecutar la revisión de la literatura en el dominio de los elementos de prototipado que soportan el proceso de desarrollo de software.

3.4. Organización del documento

Este documento final se estructura de manera coherente y detallada para abordar de forma exhaustiva el tema de investigación propuesto. A continuación, se describe la organización final del documento:

CAPÍTULO 1: Este capítulo inicia con la formulación del problema, donde se identifica y delinea claramente la cuestión central a investigar. Seguidamente, se expone la justificación, destacando la importancia y la relevancia del estudio tanto para el ámbito académico como para el sector específico relacionado con el tema. Los objetivos, tanto generales como específicos, se definen para guiar el enfoque de la investigación. Además, se presenta un marco teórico robusto que ofrece una comprensión profunda de los conceptos clave y las teorías fundamentales relacionadas con el tema de estudio. Finalmente, se revisan trabajos relacionados, proporcionando un análisis crítico de investigaciones previas y los métodos de revisión de literatura existentes para saber cuál es el más utilizado, además, se busca definir por qué se seleccionó el método de revisión de literatura específico utilizado en el estudio, destacando sus ventajas y adecuación para abordar el problema de investigación planteado. En resumen, va a tener la formulación del problema, la justificación, objetivos, marco teórico, los trabajos relacionados y los métodos de revisión de la literatura existentes.

CAPÍTULO 2: En este capítulo se detalla la estrategia metodológica diseñada para llevar a cabo una revisión de literatura exhaustiva y sistemática. Comienza con la formulación de preguntas de investigación claras y precisas, que guiarán todo el proceso de búsqueda y análisis de información. Se procede con la identificación de fuentes de información pertinentes, seleccionando bases de datos académicas, revistas especializadas, la selección de términos y palabras clave se realiza con especial cuidado, optimizando la estrategia de búsqueda para abarcar la mayor cantidad posible de literatura relevante. así como el desarrollo de estrategias de búsqueda efectivas.

CAPÍTULO 3: Este capítulo abarca la ejecución de la revisión de la literatura, presentando los resultados obtenidos a partir de la estrategia de búsqueda definida previamente y cada uno de los elementos diseñados para obtener la información necesaria. Se realiza un análisis detallado de los resultados, evaluando la calidad, relevancia, y contribuciones de los estudios seleccionados. Además, se presentan las respuestas a las preguntas de investigación planteadas, con apoyo de cada uno de los artículos primarios que quedaron como base de información y resultado de la revisión de la literatura.

CAPÍTULO 4: Este capítulo analiza la utilidad del prototipado en el desarrollo de software, fundamentándose en los datos obtenidos de una encuesta realizada a un grupo de estudiantes de ingeniería de sistemas en una de las aulas de la

universidad. En primer lugar, se socializa con los estudiantes el contenido obtenido tras la revisión de la literatura destacando los enfoques y resultados más relevantes, luego se discuten los resultados con mayor detalle para entender cómo el prototipado contribuye a la eficiencia y efectividad de los procesos de desarrollo. Posteriormente, se llevó a cabo la encuesta mencionada donde se evalúa la percepción de los estudiantes sobre la utilidad del prototipado en este campo. El capítulo concluye con unas gráficas que resumen visualmente estos hallazgos, facilitando la comprensión del impacto del prototipado en las prácticas de desarrollo de software y permitiendo extraer conclusiones significativas sobre su valor.

CAPÍTULO 5: En este capítulo se presentan las conclusiones principales de la investigación y se destacan las áreas para futuras investigaciones. Su objetivo es consolidar y analizar los hallazgos del estudio, proporcionando una visión completa de su relevancia en el contexto de la investigación. Esta sección marca el cierre del ciclo de investigación, proporcionando una síntesis significativa que impulsa el avance del conocimiento en el campo estudiado.

3.5. Marco teórico

3.5.1. Prototipo

El término prototipo se refiere a un modelo inicial y experimental de algo en fase de desarrollo, como un producto, un sistema o un proceso. Su finalidad es permitir a diseñadores, ingenieros o desarrolladores probar y evaluar conceptos, funcionalidades y diseños antes de la producción final [22]. Los prototipos también facilitan una base común para la discusión y el entendimiento entre desarrolladores, usuarios y otras partes interesadas, lo que minimiza la brecha de comunicación y proporciona una retroalimentación más rápida al cliente [22], [23]. Los prototipos también permiten la evaluación y aprobación de los conceptos de diseño del sistema por parte de los diseñadores de software, así como la identificación de posibles problemas y defectos en el diseño del sistema, minimizando así los riesgos de desarrollo desde el principio [24]. De este modo, los prototipos permiten a los desarrolladores del proyecto disponer de un prototipo que puede modelarse de forma rápida, interactiva y disponible para referencia y discusión en el trabajo preliminar de combinación de requisitos [24]. La complejidad de los prototipos puede variar desde simples bocetos hasta versiones funcionales casi completas, y se utilizan para obtener retroalimentación, realizar ajustes y mejorar la calidad del producto final [24].

3.5.2. Ingeniería de software

El término “*Ingeniería de Software*” es una disciplina que se centra en la aplicación de principios de ingeniería al desarrollo de software. Implica el uso de métodos, herramientas y procesos sistemáticos para diseñar, desarrollar, probar y mantener sistemas de software de forma eficiente y fiable [13]. Busca principalmente desarrollar software de alta calidad que satisfaga las necesidades y expectativas de usuarios y clientes, de forma eficiente y eficaz [13].

Por otro lado, el término ingeniería de software es una nueva área de la ingeniería y se considera como una disciplina que se encarga de crear y mantener aplicaciones de software utilizando tecnologías, prácticas, métodos y técnicas para el desarrollo de software de calidad, apoyándose en las herramientas y procedimientos que proporciona la informática para su aplicación [13].

Algunos de los objetivos de la Ingeniería del Software son [25]:

- Mejorar la calidad en el desarrollo de aplicaciones (software), proporcionando a los desarrolladores las bases necesarias para construir software de alta calidad y eficiencia.
- Aumentar la productividad del equipo del proyecto y especialmente de los ingenieros de software.
- Facilitar el control del proceso de desarrollo de software.
- Definir un modelo y una metodología que garanticen la producción y el mantenimiento de los productos de software desarrollados en el plazo fijado y dentro del costo estimado.
- La gestión de proyectos es una disciplina encargada de organizar y gestionar los recursos asignados de forma que el proyecto pueda llevarse a cabo con dichos recursos (tiempo, presupuesto, personas, herramientas, etc.).
- En la planificación del proyecto, se establecen los objetivos a corto, mediano y largo plazo, y como se van alcanzar.
- La organización en Ingeniería de Software es donde los gestores determinan en detalle el procedimiento para alcanzar los objetivos. En esta fase se crea el esquema, las relaciones de trabajo y la persona que dirigirá la actividad.
- El control consiste en comprobar si se está llevando a cabo la planificación realizada al principio del proyecto y también si se han alcanzado los objetivos propuestos. En esta fase, el jefe del proyecto debe ser capaz de hacer correcciones.
- El plan del proyecto debe adaptarse y actualizarse a medida que este avanza para tener una visión clara del estado del proyecto.
- Las metodologías de desarrollo de software se han sido construido o generado a través de la mejora continua, no solo exigida por la construcción del software sino también por los avances tecnológicos y las necesidades de las áreas de negocio de las empresas. El control de calidad guiado a través de métricas es muy importante para el desarrollo de software, ya que permite tener elementos objetivos para determinar si el trabajo se realiza o no de manera adecuada

La Ingeniería de Software abarca una serie de actividades a lo largo del ciclo de vida del desarrollo del software y estas actividades de ingeniería de software son [26], [15]:

- Análisis de requisitos: Comprensión y documentación de las necesidades y expectativas del usuario para el software a desarrollar.
- Diseño de software: Creación de una arquitectura y estructura adecuadas para el software, que cumpla los requisitos y objetivos establecidos.

- Implementación: Escribir código de programación siguiendo las especificaciones y el diseño previamente establecidos.
- Pruebas del software: Verificar y validar el software para garantizar que funciona correctamente y cumple los requisitos definidos.
- Gestión de la configuración: Controlar los cambios en el software y sus componentes, garantizando la integridad y trazabilidad del software.
- Gestión de proyectos: Planificar, coordinar y controlar todas las actividades relacionadas con el desarrollo de software, incluida la asignación de recursos y la gestión de riesgos.
- Documentación: Crear y mantener documentación detallada del software, incluidos manuales de usuario, guías de instalación y especificaciones técnicas.
- Mantenimiento del software: Realizar actualizaciones, correcciones de errores y mejoras del software existente para garantizar su funcionalidad continua y su adaptación a los nuevos requisitos.
- Gestión de la calidad: Implementar procesos y técnicas para garantizar la calidad del software, incluida la revisión por pares, las pruebas de unitarias, las pruebas de integración y la garantía de calidad.
- Despliegue y puesta en marcha: Instalar y configurar software en entornos de producción, garantizando una transición fluida del desarrollo al uso operativo.

3.5.3. Desarrollo de software

El desarrollo de software es una actividad esencial y compleja que implica la aplicación de técnicas y tecnologías para acelerar la implementación, mejorar la calidad y aumentar la eficiencia del software. Este proceso es realizado principalmente por programadores, ingenieros y desarrolladores de software, quienes interactúan para formalizar actividades relacionadas con la creación de software y aplicaciones interactivas para los usuarios [1].

El ciclo de vida del desarrollo de software abarca desde la especificación inicial del sistema hasta su mantenimiento continuo [2]. Este proceso pasa por varias fases, incluida la recopilación exhaustiva de requisitos, el modelado detallado de requisitos y la validación sistemática para reducir la incertidumbre con respecto a los objetivos del sistema. La validación de requisitos es crucial para confirmar si la especificación satisface las necesidades reales de los clientes y usuarios finales, entre otros aspectos. Estas fases para crear un producto de software funcional y de calidad se detallan más a continuación [2]:

- Recopilación de Requisitos: En esta fase, se identifican y documentan los requisitos del software. Esto implica comprender las necesidades del cliente, los objetivos del proyecto y las expectativas del usuario final. La comunicación clara y continua con todas las partes interesadas es esencial para garantizar una comprensión precisa de los requisitos.
- Análisis: Una vez recopilados los requisitos, se realiza un análisis detallado para entender su alcance y complejidad.

- **Diseño:** En esta fase se crea la arquitectura del software y se diseñan las interfaces de usuario. El diseño también puede incluir la creación de prototipos para obtener retroalimentación temprana de los usuarios.
- **Implementación:** Una vez finalizado el diseño, se implementa el software. Durante esta fase, el diseño se supervisa de cerca para garantizar que el software se desarrolle según lo planeado.
- **Pruebas:** Después de la implementación, el software se somete a pruebas exhaustivas para identificar y corregir errores.

3.5.4. Partes interesadas (Involucrados)

El término partes interesadas hace referencia a: clientes, usuarios, expertos del dominio, además de grupos formales e informales. Incluye a todos los interesados e impactados con la implantación de una solución basada en software [27].

3.5.5. Ingeniería de requisitos

La ingeniería de requisitos es un proceso utilizado para identificar, analizar, documentar y gestionar los requisitos del sistema durante el desarrollo de software, estos requisitos son ideas formales de las necesidades del cliente y de los usuarios finales, así como de las limitaciones que debe cumplir el sistema de software que se está desarrollando [13].

Dentro de la ingeniería de requisitos existen ciertas actividades que son fundamentales, a continuación, te presentamos algunas de las actividades más importantes en la ingeniería de requisitos [16]:

- **Captura de requisitos:** Es la identificación y recopilación de información sobre las necesidades del cliente y usuarios finales. Ya que es crucial obtener una comprensión clara de lo que se espera del sistema de software.
- **Análisis de requisitos:** Es importante comprender a fondo los requisitos recopilados, y de esta forma identificar posibles inconsistencias o ambigüedades y organizarlos de manera coherente para facilitar su especificación.
- **Validación de requisitos:** Verificar y validar que los requisitos especificados reflejen con precisión las necesidades y expectativas del cliente y los usuarios finales. Es decir, garantizar que los requisitos sean completos, consistentes, relevantes y verificables.

3.6. Trabajos relacionados

3.6.1. Estrategias de creación de prototipos: revisión de la literatura e identificación de variables críticas

Esta es una revisión sistemática realizada por E. Christie et al. [28], donde se documentó un amplio análisis en el área general de la creación de prototipos, donde se pudo identificar cuáles son las estrategias de creación de prototipos, utilizadas actualmente por diversos desarrolladores de productos. Este trabajo respalda la idea de que no sólo es beneficioso para el producto realizar esta práctica, sino

también para el equipo de diseño, sin embargo, encuentran que en el ámbito de la ingeniería hay pocos trabajos publicados sobre estrategias de creación de prototipos, ya que existen tan solo algunos pocos textos que tocan conceptos importantes, pero no abordan la estrategia de creación de prototipos a un nivel exhaustivo. Se menciona que recientemente se ha empezado a trabajar en la evaluación de los efectos de la creación de prototipos en los equipos de diseño, donde se ha demostrado un impacto positivo en los equipos y en las tasas de éxito de los proyectos, es por ello que determinan una hipótesis en la que animan a los ingenieros a enfocar la creación de prototipos de forma estratégica, para aumentar las posibilidades de éxito y eficacia en sus proyectos, además de promover una comunicación satisfactoria entre los equipos, sin embargo resaltan que es necesario que la comunidad siga examinando los factores y el método de creación de prototipos para asegurarse de que esta investigación se complete, con un experimento diseñado en torno a la obtención de una verificación cualitativa y cuantitativa de dicha hipótesis

3.6.2. Análisis de metodologías de desarrollo de software

El trabajo mostrado por S. Soobia et al. [29], no es una revisión sistemática ni un mapeo sistemático, sino un análisis comparativo de las diversas metodologías de desarrollo de software más comunes, considerando diferentes factores como el tipo de proyecto, tamaño, entorno de desarrollo y recursos disponibles. Proporciona una breve descripción de las etapas más comunes y utilizadas en el desarrollo de software. El trabajo concluye que los enfoques metodológicos de prototipado son principalmente adecuados para grandes proyectos de software, así como para nuevas innovaciones, ya que permiten una estrecha comunicación con las partes interesadas en el desarrollo, mejorando los resultados y la retroalimentación. Finalmente, en la revisión bibliográfica realizada se encontró que las revisiones o mapeos sistemáticos no realizan un proceso completo sobre la creación de prototipos, lo cual genera un vacío en el conocimiento para el uso de esta metodología.

3.6.3. La ingeniería de requisitos y su importancia en el desarrollo de proyectos de software

La investigación presentada en [30] realizó un estudio el cual dio un mejor panorama acerca de cómo a través de los años se ha podido constatar que los requisitos son la pieza fundamental en un proyecto de desarrollo de software, ya que marcan el punto de partida para actividades como la planeación, y las estimaciones de tiempos y costos; Además de enfocarse en que la especificación de requisitos es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto, ya que estos son un reflejo detallado de las necesidades de los clientes o usuarios del sistema. Se presenta que un gran número de los proyectos de software fracasan por no realizar una adecuada definición, especificación, y administración de los requisitos y dentro de esa mala administración se encuentran factores, como la falta de comunicación o participación del usuario y requisitos incompletos, es por ello que este estudio busca resaltar la importancia que tienen los requisitos, dentro del ciclo

de desarrollo de proyectos de software, y a su vez dar a conocer las diferentes alternativas o técnicas que existen para identificarlos; ya que su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, las necesidades de los usuarios o clientes; Por lo que finalmente este estudio se concentró en la búsqueda de técnicas, métodos y herramientas que pudieran ser aplicados durante el proceso de definición de requisitos para arribar a una etapa de diseño exitosa, y de las cuales resaltan una de ellas: El prototipado, ya que permite que durante la actividad de extracción de requisitos, los desarrolladores y usuarios finales puedan tener una importante retroalimentación en cuanto a si el sistema diseñado con base a los requisitos recolectados permite al usuario realizar su trabajo de manera eficiente y efectiva ya que debido a una buena comunicación definen los intereses globales del software.

3.6.4. Requisitos de software: prototipado, software heredado y análisis de documentos

El proyecto descrito en [31] presenta un rastreo bibliográfico sobre alternativas poco usuales para el levantamiento de requisitos en proyectos de desarrollo de software. En concordancia con lo antes expresado, en este artículo se expone un rastreo bibliográfico de algunas técnicas de Ingeniería de Requisitos para llevar a cabo el levantamiento de la información. Las técnicas consideradas en este artículo son: el uso de prototipos, los cuales se describen como un medio de comunicación entre analistas, clientes o usuarios, demostrando que permiten solucionar grandes problemas de comprensión, que usualmente se presentan en fases tempranas del proceso de especificación. Indican que el uso de técnicas de prototipado, en las fases de análisis y diseño facilita la validación previa de los requisitos hallados y permite reducir la cantidad de retrabajo requerido ayudando a controlar el riesgo de errores incompletos. Por tales motivos se evidencia que el propósito del uso de la técnica del prototipado es ayudar a los usuarios, ya que se ha observado que esta técnica de creación de prototipos es fácil de manejar por lo que permite crear una idea rápida de la interfaz que el usuario requiere.

3.6.5. Enfoques de creación de prototipos y desarrollo de software

El estudio presentado en [11] encontró necesario proporcionar un marco dentro del cual se clasifican los enfoques de desarrollo de software usando la creación de prototipos que se emplean para una variedad de propósitos diferentes. Esencialmente, los prototipos se utilizan para tres principales propósitos en el desarrollo de software: exploración, experimentación y evolución. Un enfoque exploratorio de creación de prototipos se lleva a cabo principalmente para obtener o clarificar al usuario sobre sus requisitos, la creación de prototipos exploratorios ayuda a los desarrolladores a obtener información sobre los usuarios, tareas y problemas de trabajo, y ayuda a cristalizar percepciones y necesidades confusas de los usuarios en los requisitos para un sistema inicial, la creación de prototipos experimentales proporciona a los usuarios posibilidades alternativas, ayuda a evaluar la viabilidad del sistema, en términos de rendimiento y otros aspectos

técnicos, cuando se conocen los requisitos. La creación de prototipos evolutivos es un enfoque que permite flexibilidad en el software ya que el proceso de desarrollo de software evoluciona continuamente. Finalmente, esta revisión encuentra relevante hacer esta clasificación, debido a que tiende a haber cierta confusión en el uso de los términos de creación de prototipos, con la intención de darle un contexto más claro y un buen uso a las estrategias de creación de prototipos.

3.6.6. Realidad mixta en la creación de prototipos de diseño: una revisión sistemática

La propuesta de Kent et al. [32], define la gestión integrada de ingeniería y diseño de productos que emplea prototipos virtuales y físicos para mejorar la optimización de procesos, reducir costos y mejorar el rendimiento del producto de software. La revisión mencionada abarcó un total de 108 artículos y su principal contribución consistió en explorar e identificar la creación de prototipos, señalando los retos y las áreas en las que es necesario trabajar. Los retos definidos en esta revisión consisten en analizar en detalle la creación de prototipos para extraer las ventajas que pueden aportar a los procesos de desarrollo de software; en esta sección se explora la oportunidad de integrar la creación de prototipos físicos y virtuales, considerando los distintos ámbitos y sus respectivas ventajas y retos. Por otra parte, se centra en las áreas de trabajo relacionadas con los procesos de diseño y las estrategias de creación de prototipos, que son fundamentales para la realización de productos de software. Estas áreas resaltan la importancia de construir prototipos con un propósito específico, ya que demuestran que su creación sistemática y decidida proporcionan un valor significativo en la clarificación de tareas.

3.6.7. Diseño mediante prototipos: apoyo estratégico al desarrollo de productos basado en prototipos

El estudio mencionado en [33] se centra en la investigación sobre el desarrollo de nuevos productos, destacando la naturaleza riesgosa de este proceso. Antes de lanzar un producto al mercado se requieren importantes inversiones en tiempo y recursos, y cada decisión que se tome durante el proceso de diseño puede determinar el éxito o el fracaso del producto. Los prototipos emergen como una herramienta fundamental para probar suposiciones y evaluar decisiones durante el diseño de productos. Si se utilizan correctamente, los prototipos pueden mejorar la calidad del producto, su adaptación al mercado, la comunicación con el cliente y acortar el tiempo de comercialización. Por lo tanto, este estudio busca enfatizar la importancia de enseñar habilidades estratégicas de creación de prototipos para aumentar su valor en el diseño de productos. Aprovechando la metodología de investigación de diseño, utilizando estudios en el aula y casos de la industria para comprender mejor el proceso de creación de prototipos y ayudar a los diseñadores a administrarlo de manera intencional. A si mismo la investigación propone un modelo conceptual para el proceso de creación de prototipos, junto con herramientas como ProtoMapping, Prototyping Planner, Prototyping Target y ProtoMap, que ayudan a estructurar y gestionar este proceso de forma sistemática. Se presenta el marco "Design by Prototyping", que describe cómo gestionar un

proceso de creación de prototipos estratégico utilizando sprints de creación de prototipos. Se demostró que el uso de estas herramientas ayuda a los diseñadores a definir el propósito de sus prototipos, identificar resultados y tomar decisiones más deliberadas para la creación de prototipos eficaces. Y finalmente se obtiene que este enfoque puede integrarse en la educación formal de estudiantes de diseño de ingeniería y aplicarse en la industria para implementar prácticas de diseño basadas en prototipos. En última instancia, el estudio contribuye a una teoría mejorada del proceso de creación de prototipos, reconociendo la creación de prototipos como una parte esencial del proceso de diseño.

3.6.8. Implementación de métodos ágiles para la simulación de casos de uso y prototipado en el proceso de desarrollo de software

El estudio presentado en [34] aborda la gestión de la simulación de casos de uso, la creación de prototipos y la generación de escenarios como fase inicial del ciclo de vida del desarrollo de software, aplicando el concepto de diseño ágil. Se destaca la importancia de lograr un acercamiento temprano a la dinámica del negocio a través de la interacción activa con el cliente desde las primeras etapas del proyecto. Se propone el uso de simulaciones de casos de uso para diseñar un modelo de sistema real que permita presentar diversos escenarios que satisfagan las necesidades del cliente. Estas simulaciones conducen a etapas de modelado de la realidad, facilitando la comprensión de procesos y la identificación de objetos y estados importantes. El Lenguaje Unificado de Modelado (UML) se utiliza para construir modelos de simulación, permitiendo una evolución ágil y el refinamiento del diseño a través de una retroalimentación efectiva con el usuario. Los prototipos resultantes se desarrollan de manera eficiente y rápida, permitiendo la exploración de múltiples escenarios con recursos limitados. La simulación se revela como un medio indispensable en numerosos proyectos de desarrollo de software, permitiendo proyectar los resultados esperados y facilitando una rápida adaptación a los cambios. Finalmente, este enfoque proporciona aproximaciones que ayudan a establecer límites, identificar objetos relevantes y relaciones entre actividades, proyectos o procesos. Involucrar al cliente en las primeras etapas del proceso mediante la presentación de modelos de solución, mejora significativamente la calidad del diseño y desarrollo en etapas posteriores del ciclo de vida del proyecto. La metodología propuesta garantiza un desarrollo consistente con las necesidades del cliente, permitiendo ajustes ágiles y oportunos durante el proceso, y contribuyendo a la creación de aplicaciones simples, funcionales y estandarizadas que cumplan con el objetivo de ser evolutivas.

3.6.9. Ingeniería de requisitos ágil con creación de prototipos: un caso de estudio

El trabajo mencionado en [35] habla de cómo el auge de los métodos ágiles de desarrollo de software ha llevado a un cambio en las prácticas tradicionales, especialmente en la ingeniería de requisitos. A medida que estas metodologías continúan ganando popularidad, surge la necesidad de encontrar prácticas más formales para proyectos ágiles. Este trabajo describe las experiencias adquiridas a

partir de un caso de estudio realizado en un gran proyecto ágil, con el objetivo de explorar cómo la creación de prototipos puede abordar los desafíos de la metodología ágil en la ingeniería de requisitos. Los hallazgos indican que, si bien la creación de prototipos puede ayudar con algunos desafíos de la ingeniería de requisitos ágil, como la falta de documentación, la motivación laboral y la mala comunicación, también se requieren prácticas complementarias para alcanzar su máximo potencial. Combinar las prácticas tradicionales de ingeniería de requisitos con métodos ágiles puede resultar beneficioso. La ingeniería de requisitos tradicional enfatiza la determinación de un conjunto de requisitos antes de comenzar el desarrollo de software, mientras que los métodos ágiles adoptan un enfoque iterativo y menos formal. Las investigaciones sugieren que adaptar las prácticas tradicionales para que se ajusten mejor al flujo de trabajo ágil es cada vez más común, y la creación de prototipos se considera como una forma eficaz de mejorar la comunicación entre las partes interesadas. Sin embargo, todavía existen desafíos, como requisitos parcialmente optimizados y volátiles en proyectos ágiles. La creación de prototipos puede centrarse demasiado en objetivos a corto plazo y descuidar la calidad y la arquitectura de requisitos. Para abordar estos desafíos, se sugiere revisar periódicamente el panorama completo del proyecto, mantener un seguimiento de la calidad de los requisitos y realizar pruebas de aceptación basadas en prototipos. Concluyendo finalmente que la creación de prototipos puede ser beneficiosa en proyectos ágiles de ingeniería de requisitos, pero se necesita un enfoque integral que incluya prácticas complementarias para maximizar su efectividad. Además, se sugieren más casos de estudio para explorar cómo la creación de prototipos se puede integrar de manera óptima en proyectos de desarrollo de software.

3.6.10. Creación de prototipos de usabilidad y experiencia de usuario: una técnica sencilla para equipos ágiles

La investigación realizada en [36] aborda la integración de metodologías ágiles con conceptos de usabilidad y experiencia de usuario (UX) en el desarrollo de software. Se reconoce que la asociación entre metodologías ágiles y usabilidad-experiencia de usuario puede ser compleja y costosa en términos de conocimiento, tiempo y recursos. Por ello, se propone una técnica denominada "Guía de Experiencia de Usuario y Usabilidad para Proyectos Ágiles" (UXUG-AP) que integra la técnica de prototipado ágil con criterios de calidad del software enfocados a la usabilidad y experiencia de usuario. Esta técnica fue diseñada para ser utilizada por equipos ágiles, independientemente de los recursos disponibles, el tamaño del proyecto y la metodología ágil adoptada. Esta investigación también aborda la evolución del desarrollo de software desde la aparición del Manifiesto Ágil en 2001, destacando el creciente enfoque en la satisfacción del usuario y la entrega de software utilizable en un contexto altamente competitivo. La usabilidad se define como la capacidad de un software para ser utilizado por un usuario específico para lograr objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso específico, mientras que la experiencia de usuario se define como la percepción y las respuestas de una persona resultantes del uso o uso anticipado del software. La pregunta de investigación del trabajo se centra en cómo diseñar requisitos de

usabilidad y experiencia de usuario en proyectos ágiles, anticipando problemas de usabilidad-experiencia de usuario. Se realizó una revisión de la literatura para comprender el estado actual de la integración de metodologías ágiles con conceptos de usabilidad y experiencia de usuario, con el objetivo de identificar cómo contribuir al campo. Además, se presenta un mapeo sistemático de la literatura para respaldar la propuesta y se discuten los estudios de viabilidad que se aplicaron para investigar la utilidad de la técnica propuesta. Se destaca la importancia de definir requisitos bien escritos para mejorar los resultados de la técnica (UXUG-AP) y se menciona la necesidad de apoyar la definición de necesidades claras del usuario para mejorar la productividad y precisión en prototipado. El trabajo también menciona que se está realizando un análisis de los datos proporcionados por los participantes para mejorar la técnica, seguido de un segundo estudio de viabilidad con profesionales en un ciclo de vida ágil para evaluar nuevamente la técnica.

Según los estudios revisados, muestran un interés compartido por la aplicación del prototipado en el desarrollo de software, destacando su utilidad para mejorar la comunicación, validar los requisitos en una fase temprana y reducir los errores y retrabajos. Se exploran diversos aspectos como las estrategias de creación de prototipos, las metodologías de desarrollo ágil, la ingeniería de requisitos, la usabilidad y la experiencia del usuario. Sin embargo, se identifican varias brechas y áreas que requieren mayor investigación, como la integración efectiva del prototipado en metodologías ágiles, la gestión de la complejidad en proyectos a gran escala y la medición precisa de los beneficios del prototipado en términos de calidad y eficiencia del desarrollo de software. Además, se necesitan más estudios que exploren cómo mejorar la práctica del prototipado en contextos específicos y adaptar estas técnicas a las cambiantes necesidades de los usuarios y el mercado.

3.7. Métodos de revisión de literatura disponibles

A continuación, se presentan algunos de los métodos disponibles para realizar revisiones de la literatura. Estos métodos varían en cuanto a enfoque, rigor metodológico y objetivos, y se seleccionan en función de las necesidades específicas de la investigación. Cada método tiene sus propias características distintivas que influyen en la forma de realizar la revisión y en la interpretación de los resultados obtenidos. Al comprender estas diferencias, los investigadores pueden elegir el método más adecuado para abordar sus preguntas de investigación y alcanzar sus objetivos de revisión con eficacia y son los siguientes [37]:

Revisión rápida: Los métodos de revisión rápida, son considerados inicialmente como una estrategia utilizada para obtener una comprensión ágil y rigurosa de la información disponible sobre un tema específico. Este enfoque se utiliza principalmente en contextos donde se requiere tomar decisiones rápidas y basadas en evidencia, como en el ámbito político, empresarial o de políticas públicas. En la actualidad, el sitio web de investigación social del Gobierno propone este método como un medio para proporcionar una "evaluación de lo que ya se sabe sobre una cuestión política o práctica, mediante el uso de métodos de revisión sistemática para buscar y evaluar críticamente la investigación existente".

Revisión Narrativa: Este método implica una revisión exhaustiva de la literatura existente sobre un tema en particular, seguido de una síntesis narrativa de los hallazgos. No sigue un protocolo predefinido y puede ser más subjetivo que otros métodos.

Revisión sistemática: La revisión sistemática, es el tipo de revisión más conocido, ya que busca, evalúa y sintetiza sistemáticamente las pruebas de investigación. Una revisión sistemática es definida como un resumen de evidencias, habitualmente realizada por un experto o panel de expertos en un tema determinado, que utiliza un riguroso proceso (para minimizar los sesgos) que identifica, evalúa y sintetiza estudios para contestar a una pregunta específica y extraer conclusiones sobre los datos recopilados. La diferencia más importante entre la revisión sistemática y otro tipo de revisiones es que fundamentalmente la metodología utilizada es explícita y precisa, y además se sigue un protocolo claramente delineado, estandarizado y replicable que asegura la calidad, consistencia y transparencia del proceso de revisión.

Las revisiones paraguas o revisión de revisiones: Son un nuevo tipo de revisiones que están ganando un creciente interés ya que estas revisiones se centran fundamentalmente en resumir la evidencia disponible. Pueden ser utilizadas para evaluar las similitudes y diferencias en las revisiones publicadas, para resumir lo que se sabe sobre un tema y normalmente implican un amplio número de diferentes tipos de revisiones. La revisión paraguas permite a los investigadores, profesionales y responsables de la toma de decisiones obtener una comprensión más completa y profunda de un área de estudio específica.

Revisión del alcance: Este tipo de revisión proporciona una evaluación preliminar del tamaño y alcance potenciales de la literatura de investigación disponible. Su objetivo es identificar la naturaleza y el alcance de las pruebas de investigación (normalmente incluyendo la investigación en curso). Puntos fuertes percibidos. Las revisiones exploratorias pueden informar a los responsables políticos sobre si es necesaria una revisión sistemática completa. Comparten varias características de la revisión sistemática al intentar ser sistemáticas, transparentes y reproducibles.

Revisión sistematizada: Las revisiones sistematizadas o revisiones estructuradas intentan incluir uno o más elementos del proceso de revisión sistemática, sin llegar a afirmar que el producto resultante es una revisión sistemática. Podrían catalogarse como una "revisión sistemática" pero en muchas ocasiones carecen de algún elemento que no permite etiquetarlas como tales. Un ejemplo puede ser los trabajos de revisión que pueden realizar los estudiantes de postgrado que, aun siguiendo toda la metodología de una revisión sistemática, carecen de recursos suficientes para realizarla (por ejemplo, disponer de dos revisores que evalúen los documentos sometidos a análisis). La fortaleza de este tipo de revisiones radica fundamentalmente en que posee un alto grado de sistematización de la búsqueda, aunque el autor no realice una búsqueda en todas las bases de datos disponibles. El proceso de revisión sistematizada permite al autor demostrar el conocimiento de

todo el proceso y la competencia técnica en las etapas que lo componen. Sin embargo, esta revisión carece de la amplitud para ser identificada como una revisión sistemática. Esta es quizás la mayor debilidad de la revisión sistematizada: la amplitud y la profundidad del análisis de la bibliografía.

Revisión Scoping (o Exploratoria): Este enfoque se utiliza para mapear la evidencia disponible sobre un tema amplio, especialmente cuando el área es compleja o no ha sido revisada comprehensivamente antes. Ayuda a identificar el tipo, el volumen y las características de la literatura existente, así como las lagunas en la investigación. Es útil para aclarar conceptos o definiciones y para informar futuras revisiones sistemáticas.

El método elegido para la investigación fue el de la de revisión sistemática, debido a varias razones fundamentales que lo hacen especialmente adecuado para el propósito de este estudio. Este método se caracteriza por su enfoque riguroso y estructurado para identificar, evaluar y sintetizar las investigaciones existentes sobre un tema específico. A continuación, se exponen las principales razones por las que se seleccionó este método: El rigor metodológico es una de ellas, ya que la revisión sistemática se basa en un protocolo predefinido que especifica los criterios de inclusión y exclusión de los estudios, las bases de datos y las palabras clave que se utilizarán en la búsqueda, y cómo se evaluará la calidad de los estudios incluidos. Este rigor metodológico reduce el riesgo de sesgo y garantiza la fiabilidad de los resultados. Exhaustividad porque, a diferencia de otras formas de revisión, como las narrativas, la revisión sistemática pretende ser exhaustiva en la identificación de estudios relevantes, empleando estrategias de búsqueda amplias y detalladas. Esto garantiza que se tengan en cuenta todas las pruebas disponibles, lo cual es esencial para comprender plenamente el estado actual de los conocimientos sobre el tema en cuestión. Además, la síntesis de la evidencia porque la revisión sistemática permite no solo recopilar sino también sintetizar los resultados de los estudios incluidos otra razón es el apoyo a la toma de decisiones la consolidación de la evidencia que ofrecen las revisiones sistemáticas es de gran valor para los profesionales y responsables de la toma de decisiones, ya que proporciona una base de conocimientos sólida y actualizada para orientar las prácticas y las políticas. Por estas razones, se eligió el método de revisión sistemática como el más adecuado para alcanzar los objetivos de la investigación. Este enfoque no solo garantiza una cobertura exhaustiva y objetiva del tema, sino que también proporciona un marco para evaluar la calidad de las pruebas y extraer conclusiones fiables basadas en un conjunto de conocimientos existentes. Además, dado que, para el proyecto de investigación sobre prototipos, el método de revisión sistemática resulta ser una opción ideal por varias razones clave. En primer lugar, su rigor metodológico garantiza el establecimiento de criterios claros para la inclusión y exclusión de estudios relevantes sobre prototipos, lo que resulta crucial para mantener la coherencia y la objetividad en la recopilación de evidencia. Este enfoque también permite realizar una búsqueda exhaustiva y detallada en diversas bases de datos, lo que garantiza que se recojan todos los estudios pertinentes sobre el desarrollo y la evaluación de prototipos en diferentes contextos. Además, la capacidad de sintetizar la evidencia de forma sistemática proporciona una visión

completa y actualizada de los conocimientos existentes sobre prototipos, lo que permite identificar tendencias, patrones y áreas en las que existe consenso o discrepancia en la literatura. Esto es crucial para evaluar críticamente el estado actual del campo y proporcionar recomendaciones fundamentadas para futuras investigaciones o prácticas en el desarrollo de prototipos. Por último, la elección de la revisión sistemática como método de investigación para este proyecto garantiza no solo la exhaustividad y la objetividad, sino también la utilidad práctica y la relevancia de los resultados para los profesionales y los responsables de la toma de decisiones en el campo de la creación de prototipos.

Estos son algunos ejemplos de marcos y metodologías utilizados para llevar a cabo revisiones sistemáticas de la literatura en distintas disciplinas. Cada uno tiene sus propias características y enfoques específicos, pero todos comparten el objetivo común de sintetizar rigurosamente las pruebas disponibles sobre un tema concreto. A continuación, se presentan:

PRISMA: Es un marco ampliamente utilizado para la elaboración y presentación de revisiones sistemáticas y metaanálisis. Proporciona una lista de verificación de elementos esenciales que deben incluirse en la planificación, ejecución y reporte de una revisión sistemática [38].

COCHRANE: Es una red internacional dedicada a la producción de revisiones sistemáticas de intervenciones en salud. Utiliza un enfoque riguroso y transparente para la síntesis de la evidencia científica, con el objetivo de informar la toma de decisiones en el ámbito de la atención médica [39].

JB: Desarrolla y promueve métodos y herramientas para la síntesis de la evidencia en el ámbito de la atención sanitaria. Su enfoque se centra en la elaboración de revisiones sistemáticas y otros tipos de síntesis de la evidencia basadas en la práctica clínica [40].

ROBIS: Es un marco diseñado para evaluar el riesgo de sesgo en revisiones sistemáticas. Proporciona pautas y herramientas para evaluar críticamente la calidad y la confiabilidad de las revisiones sistemáticas en función de diferentes dominios de riesgo de sesgo [41].

ARKSE: Es un marco que combina principios de la ciencia de la computación, la inteligencia artificial y la ciencia de la información para mejorar la eficiencia y la calidad de las revisiones sistemáticas de la literatura [42].

EL MARCO SALSA: Se compone de cuatro componentes para llevar a cabo una revisión sistemática: (1) una búsqueda bien definida, en la que se identifican los artículos relevantes; (2) una evaluación rigurosa de los artículos, rechazando aquellos que no cumplan con los criterios de calidad preestablecidos; (3) un análisis detallado de la descripción y valoración global de los resultados encontrados, y (4) una síntesis de los resultados analizados para crear un nuevo producto que integre todas las partes [43].

A continuación, se presenta una tabla comparativa de los marcos mencionados, destacando sus ventajas y desventajas:

Tabla 1. Ventajas y desventajas de los diferentes marcos y metodologías para llevar a cabo revisiones sistemáticas de la literatura.

Marco	Ventajas	Desventajas
PRISMA	- Proporciona una lista de verificación detallada para garantizar la inclusión de elementos esenciales en la planificación, ejecución y reporte de la revisión.	- Puede resultar muy detallado para algunos usuarios, lo que podría aumentar la complejidad de la revisión.
COCHRANE	- Utiliza un enfoque riguroso y transparente, lo que garantiza la calidad y la objetividad en la síntesis de la evidencia.	- Se centra principalmente en intervenciones en salud, por lo que puede no ser adecuado para todas las áreas de investigación.
JBÍ	- Se enfoca en la práctica clínica, lo que lo hace especialmente relevante para profesionales de la salud.	- Su aplicación puede ser limitada en áreas fuera del ámbito de la atención sanitaria.
ROBIS	- Proporciona pautas y herramientas específicas para evaluar el riesgo de sesgo en revisiones sistemáticas, lo que mejora la calidad de la evaluación crítica.	- Requiere cierto nivel de experiencia en la evaluación de riesgo de sesgo, lo que podría representar una barrera para algunos usuarios.
ARKSE	- Incorpora principios de ciencia de la computación e inteligencia artificial para mejorar la eficiencia y calidad de las revisiones sistemáticas.	- Puede requerir habilidades técnicas avanzadas en ciencia de la computación para su implementación.
SALSA	- Proporciona un marco claro y estructurado con cuatro componentes esenciales para realizar una revisión sistemática de manera efectiva.	- La efectividad puede depender en gran medida de la adecuada aplicación de cada uno de los componentes del marco.

El marco o metodología elegido para la investigación fue el Marco SALSA, por varias razones: presenta una estructura completa y bien definida; el Marco SALSA proporciona un marco claro y detallado con cuatro componentes esenciales para llevar a cabo una revisión sistemática: búsqueda, evaluación, análisis y síntesis. Esta estructura exhaustiva garantiza que todas las etapas importantes de la revisión se cubran adecuadamente. Además, el Marco SALSA está alineado con los

objetivos del estudio; si los objetivos de la revisión sistemática incluyen la identificación, valoración y síntesis crítica de la literatura existente en un área específica, el Marco SALSA proporciona una guía completa para alcanzar estos objetivos. En el contexto de este proyecto sobre prototipos, esta estructura permite una identificación precisa de la literatura relevante, una evaluación rigurosa de la calidad de los estudios existentes, un análisis profundo de los datos recopilados y una síntesis crítica de los hallazgos.

Este marco no solo garantiza una revisión sistemática exhaustiva y objetiva, sino que también proporciona resultados prácticos y relevantes para los profesionales y los responsables de la toma de decisiones en el ámbito de los prototipos. La facilidad de aplicación del Marco SALSA es otra de las razones de su elección. Proporciona una guía paso a paso para cada etapa del proceso de revisión, lo que facilita su aplicación incluso a investigadores sin experiencia previa en la realización de revisiones sistemáticas. Además, aunque SALSA tiene un marco definido, permite cierta flexibilidad para adaptarse a diferentes contextos, aunque SALSA tiene un marco definido, también permite cierta flexibilidad para adaptarse a las necesidades específicas del estudio o a la naturaleza del tema de investigación, garantizando así que pueda ajustarse según los requisitos del proyecto. En resumen, la elección del marco SALSA puede justificarse por su estructura completa, su adecuación a los objetivos del estudio, su facilidad de aplicación y su adaptabilidad a diferentes contextos de investigación.

4. CAPÍTULO 2

En esta sección, se presenta el diseño de la revisión sistemática de la literatura, con el propósito de establecer una estrategia de búsqueda eficaz que facilite la identificación exhaustiva y precisa de estudios o documentos relevantes para el tema de investigación, siendo fundamental para recopilar información de manera rigurosa y completa. En este contexto, el diseño contendrá las siguientes secciones: la formulación de preguntas de investigación, la identificación de fuentes de información pertinentes, la selección de términos y palabras clave relevantes, así como el desarrollo de estrategias de búsqueda efectivas. Todas estas actividades son fundamentales para establecer una base sólida en la búsqueda de información y garantizar la calidad y relevancia de los hallazgos obtenidos.

4.1. Identificación de la necesidad de la revisión

La idea de esta revisión sistemática de la literatura es diseñar el protocolo para la revisión utilizando el método seleccionado. Para esto, se formularon preguntas de investigación claras, estructuradas y precisas para responder el objetivo principal que era poder identificar qué elementos de creación de prototipos apoyan eficazmente los procesos de desarrollo de software y permiten un análisis riguroso del estudio propuesto. Estas preguntas se basaron en la selección, el análisis y la categorización de la información encontrada en el ámbito del estudio. La Tabla 2 presenta las preguntas de investigación junto con su correspondiente motivación y posibles respuestas a obtener, de manera que luego se pudiera obtener datos más específicos.

Tabla 2. Preguntas de investigación utilizadas para la revisión sistemática.

Preguntas de investigación	Motivación	Respuestas
P1. ¿Cuál es la frecuencia de publicación de investigaciones que utilicen los prototipos de software en el proceso de desarrollo de software?	Comprobar con qué frecuencia se están realizando estudios que impliquen el uso de prototipos en los procesos de desarrollo de software, además de observar si la línea de tendencia es positiva y por lo tanto, si es un tema actual.	Año de creación del artículo o documento
P2: ¿Cuáles son las ventajas de aplicar la creación de prototipos en el proceso de desarrollo de software?	Determinar las ventajas del uso de prototipos en los procesos de desarrollo de software.	Se desarrollará al momento de responder la pregunta de investigación

P3: ¿Cuáles son las desventajas de aplicar la creación de prototipos en el proceso de desarrollo de software?	Determinar las desventajas del uso de prototipos en los procesos de desarrollo de software.	Se desarrollará al momento de responder la pregunta de investigación
P4: ¿Qué tipos de prototipado se utilizan en el proceso de desarrollo de software?	Establecer qué tipos de prototipado se están utilizando en los procesos de desarrollo de software.	• Prototipos desechables • Prototipos rápidos • Prototipos exploratorios • Prototipos evolutivos • Prototipo incremental • Prototipo de sistema piloto • Prototipos de presentación • Otro
P5: ¿Qué estrategias o técnicas de creación de prototipos existen en el proceso de desarrollo de software?	Determinar cuáles son las estrategias o técnicas de creación de prototipos que existen en el proceso de desarrollo de software.	• Bocetos • Maquetas • Prototipado en papel • Prototipado en video • Prototipado virtual • Prototipado incremental • Prototipado rápido o desarrollo rápido de aplicaciones web • Otro
P6: ¿Cómo se aplican las estrategias o técnicas de creación de prototipos en el desarrollo de software?	Determinar cómo se están implementando las estrategias o técnicas de creación de prototipos en el desarrollo de software.	Se desarrollará al momento de responder la pregunta de investigación

P7: ¿Qué metodologías o procesos de desarrollo de software han incluido el uso del prototipado?	Encontrar qué metodologías o procesos de desarrollo de software han incluido el uso de prototipos, para de esta forma observar cómo ha sido el impacto y en donde se realizan las mayores contribuciones.	• Modelo en espiral • Prototipado • Incremental • Modelos ágiles • Otro
P8: ¿En qué fase del proceso de desarrollo de software se utiliza el prototipado?	Encontrar qué fases del proceso de desarrollo de software han incluido el uso de prototipos, para de esta forma conocer cómo las aplican los desarrolladores en su marco de trabajo.	Se desarrollará al momento de responder la pregunta de investigación
P9: ¿Qué herramientas de creación de prototipos se utilizan en el desarrollo de software?	Establecer qué tipo de herramientas (framework, DSL, aplicaciones, etcétera) se están utilizando para integrar la creación de prototipos en los procesos de desarrollo de software.	• TAP • Matlab • RM2PT • SNAP • SpeAR • Axure RP • Cacao • Justinmind • Pencil • Otro

4.2. Bases de datos para la aplicación de la búsqueda de la información

Para llevar a cabo la búsqueda de información, se utilizaron diversas bases de datos académicas, tales como Scopus, IEEEExplore, ACM Digital Library. Estas fuentes son relevantes en el campo de la ingeniería de software, ya que previas revisiones sistemáticas de la literatura han empleado estas bases de datos, tal como se indica en la revisión [44]. Los motores de búsqueda proporcionan opciones avanzadas, tales como criterios de búsqueda, formatos de cadena y operadores booleanos, lo que permite la agrupación lógica de términos o palabras clave para ampliar o definir búsquedas avanzadas para bases de datos específicas, adecuándose así a los requerimientos de la revisión.

4.3. Estrategia de búsqueda

Las estrategias de búsqueda son los métodos utilizados para localizar y recuperar información relevante para una revisión sistemática. Son importantes porque minimizan el sesgo de selección y garantizan la exhaustividad y reproducibilidad de la búsqueda [45]. Es importante destacar que en la estrategia de búsqueda se

establecieron palabras clave para abarcar un amplio espectro del tema de interés como se muestra en la Tabla 3.

Tabla 3. Palabras clave, sinónimos y abreviaturas.

PALABRAS CLAVE:	RELACIONADO CON:
Ingeniería de software	Población
Proceso de desarrollo de software	Población
Prototipado	Intervención

Además, se aplicaron los criterios PICOC (Población, Intervención, Resultado, Comparación) para ajustar los conceptos y términos necesarios durante el proceso de búsqueda de artículos, tal como se presenta en la Tabla 4.

Tabla 4. Criterios PICOC.

PICOC:	
POBLACIÓN	Software engineering, software development
INTERVENCIÓN	Prototyping
COMPARACIÓN	--
OUTCOME (RESULTADO)	Characterize
CONTEXTO	Academy, Software industry, Computer industry

Combinando las palabras clave con los operadores AND & OR, se definió la cadena de búsqueda que se presenta en la tabla 5:

Tabla 5. Cadena de búsqueda.

CADENA DE BÚSQUEDA:
("Software development" OR "Software engineering" OR "Software requirements engineering" OR "RE" OR "SRS" OR "collaboration engineering") AND ("shared understanding" OR "common understanding")

Con base en la cadena anterior, se realizaron modificaciones correspondientes a cada una de las bases de datos seleccionadas. A continuación, se muestra la cadena resultante:

Tabla 6. Cadena de búsqueda aplicada a cada una de las bases de datos

Cadena de búsqueda	Base de datos	Resultados
((("Abstract": "Software development process" OR "Software engineering" OR "Publication Title": "Software development process" OR "Software engineering") AND ("Abstract": "Prototyping" OR "Publication Title": "Prototyping"))	IEEE 2015-2022	116
ContentGroupTitle:((("Software development process" OR "Software engineering")) AND ContentGroupTitle:("Prototyping")) OR Abstract:((("Software development process" OR "Software engineering")) AND Abstract:("Prototyping"))	ACM 2015-2022	16
ABS (("Software development process" OR "Software engineering") AND ("Prototyping")) AND (LIMIT-TO (PUBYEAR , 2022) OR LIMIT-TO (PUBYEAR , 2021) OR LIMIT-TO (PUBYEAR , 2020) OR LIMIT-TO (PUBYEAR , 2019) OR LIMIT-TO (PUBYEAR , 2018) OR LIMIT-TO (PUBYEAR , 2017) OR LIMIT-TO (PUBYEAR , 2016) OR LIMIT-TO (PUBYEAR , 2015)) AND (LIMIT-TO (LANGUAGE , "English")) AND (LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "ENGI"))	SCOPUS	85

4.4. Evaluación de la calidad

Se empleó un mecanismo de puntuación cuantitativa para evaluar la calidad de los posibles estudios que debían cumplir como fuentes primarias. Este mecanismo consiste en un conjunto de preguntas que se definieron a las cuales se asignó una puntuación como se muestra en la Tabla 8. Mediante la evaluación final de cada artículo utilizando este conjunto de preguntas de calidad, se determinó si se consideraban o no fuentes primarias. La Tabla 7 presenta la lista completa de preguntas utilizadas en esta evaluación:

Tabla 7. Preguntas de calidad para la selección de estudios primarios.

PREGUNTAS DE CALIDAD
P.C1 ¿El artículo específico los objetivos de la investigación?
P.C2 ¿El artículo describe claramente las técnicas o métodos utilizados?
P.C3 ¿El artículo muestra claramente el enfoque propuesto?
P.C4 ¿El artículo muestra una validación o evaluación de la propuesta definida en él?

Las preguntas anteriores se evaluaron considerando las siguientes posibles respuestas que se encuentran en la Tabla 7, con sus respectivas puntuaciones:

Tabla 8. Valoración de las preguntas de calidad.

DESCRIPCIÓN	PUNTUACIÓN
Si	1.0
Parcialmente	0,5
No	0,0

La puntuación de cada pregunta se calculó en función del número de preguntas y de la respuesta con mayor peso, donde 4,0 es la puntuación máxima que puede tener un artículo en su evaluación y 2,2 es la puntuación de corte, es decir, los artículos cuya evaluación sea inferior a 2,2 no se considerarán estudios primarios, ya que su evaluación no supera el límite de evaluación.

4.5. Proceso para la selección de estudios primarios

Siguiendo las directrices definidas por Petersen et al. en [46], se utilizaron criterios de inclusión (Ver Tabla 9) y exclusión (Ver Tabla 10) para determinar qué estudios son más relevantes para la investigación y responder así a las preguntas planteadas [47]. Con la definición de cada criterio, se evaluaron cada uno de los estudios, considerando que un estudio se consideró pertinente si cumplía todos los criterios de inclusión; por otro lado, si un estudio cumplía al menos un criterio de exclusión, no se tendría en cuenta.

Tabla 9. Criterios de inclusión de la investigación.

CRITERIOS DE INCLUSIÓN:

Acceso al texto completo.
Artículos en inglés.
El estudio aborda el prototipado en el contexto de la ingeniería de software, la informática o el desarrollo de software.
El estudio se publica en actas de conferencias, revistas, capítulos de libros, libros o talleres.

Tabla 10. Criterios de exclusión de la investigación.

CRITERIOS DE EXCLUSIÓN:
Documentos repetidos.
No se puede acceder al documento.
El estudio es una revisión sistemática de la literatura.
El estudio está en un idioma distinto al inglés.
El estudio no es un documento científico (editoriales, prólogos, resúmenes de artículos, entrevistas, noticias, revisiones, correspondencia, debates, comentarios, cartas de lectores y resúmenes de tutorías, paneles)

5. CAPÍTULO 3

En esta sección, se presenta la ejecución de la revisión de literatura, la cual permite recopilar, examinar y sintetizar los hallazgos relevantes de las investigaciones previas relacionadas con nuestro tema de estudio. En este contexto, la ejecución contendrá las siguientes secciones: la ejecución de la revisión de la literatura, presentando los resultados obtenidos a partir de la estrategia de búsqueda definida previamente. Un análisis detallado de los resultados, evaluando la calidad, relevancia, y contribuciones de los estudios seleccionados. Además, se presentan las respuestas a las preguntas de investigación planteadas.

El proceso de ejecución del estudio, tal como se diseñó en el capítulo anterior, se llevó a cabo en tres fases principales, abarcando la búsqueda, selección y evaluación de artículos relevantes para la revisión sistemática de la literatura (RSL).

Para el proceso de ejecución se llevó a cabo la fase 1, donde se aplicó la cadena de búsqueda definida anteriormente a cada una de las tres bases de datos bibliográficas, IEEEExplore, ACM Digital Library y Scopus, y se obtuvieron un total de 116 artículos de IEEEExplore, 16 artículos de ACM Digital Library y 85 de Scopus, para un total de 217 artículos. Posteriormente, se eliminaron las investigaciones duplicadas (36) y se aplicaron los criterios de inclusión y exclusión a cada uno de los artículos, considerando solamente la lectura del resumen, palabras clave y título, obteniéndose un total de 107 artículos de IEEEExplore, 12 de ACM Digital Library y 62 de Scopus, obteniéndose 181 resultados, denominados "candidatos a estudios primarios".

En la fase 2, se aplicaron los criterios de inclusión y exclusión a la lectura del resumen, palabras clave, título y conclusiones, obteniendo un total de 33 artículos de IEEEExplore, un total de 5 artículos de ACM Digital Library y 16 artículos de Scopus obteniendo 54 artículos denominados "artículos primarios".

Por último, en la fase 3, los artículos seleccionados como artículos primarios fueron sometidos a una evaluación de calidad utilizando el mecanismo de puntuación definido anteriormente. Tras esta evaluación, se obtuvieron un total de 28 artículos de IEEEExplore, 3 artículos de ACM Digital Library, y 12 artículos de Scopus obteniendo un total de 43 artículos de alta calidad, que fueron considerados para su inclusión en la revisión final, y con los que se buscó recopilar la mayor cantidad de información posible, con el fin de responder a cada una de las preguntas de investigación especificadas para esta RSL.

5.1. Resultados y análisis de los resultados

La Figura 1 presenta los resultados y el análisis de las contribuciones a la investigación de cada una de las bases de datos. En cuanto a los artículos primarios, duplicados y rechazados, se observa que aproximadamente la mitad de los resultados proceden de la base de datos "Scopus". Por otro lado, se observa que la base de datos "IEEE Xplore Digital Library" ha sido la principal fuente de artículos

primarios, lo que concuerda con investigaciones anteriores [48] [44]. Es importante destacar que, en el contexto del prototipado, esta base de datos ha proporcionado la mayor cantidad de resultados relevantes.

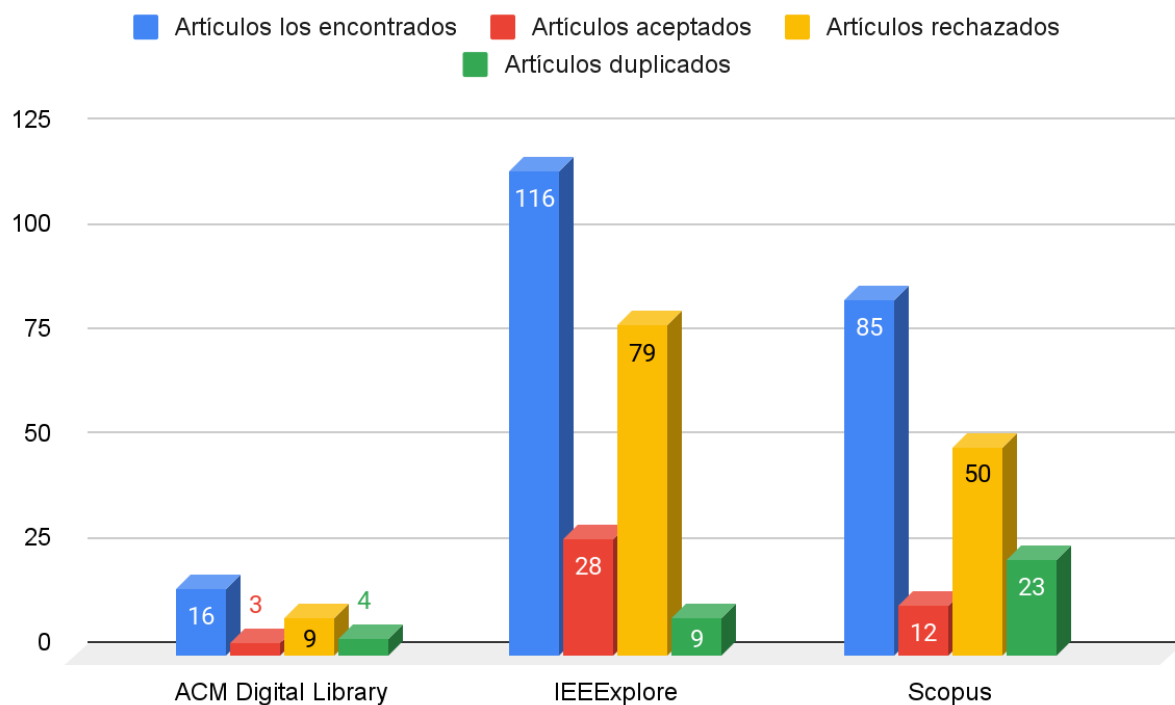


Figura 1. Aportes a la investigación de cada una de las bases de datos.

5.2. Presentación del informe de revisión

El objetivo principal de esta fase de la presentación del informe de revisión es comunicar de forma clara y eficaz los hallazgos, resultados y conclusiones obtenidos durante este proceso.

En este sentido, a continuación, se presentan las respuestas a las preguntas de investigación, derivadas del análisis de los 43 estudios recopilados en la revisión sistemática de literatura, que quedaron como artículos primarios finales.

P1: ¿Qué ventajas tiene la aplicación del prototipado en el proceso de desarrollo de software?

La revisión de los artículos primarios finales revela que el uso de prototipos ofrece ventajas en varios aspectos, específicamente en el desarrollo de software. La Figura 2 presenta los resultados obtenidos de esta revisión, donde se destaca que un porcentaje significativo de los artículos enfatiza y detalla las principales ventajas de los prototipos. Se observa que el 20,0% de los artículos mencionan que una de las ventajas de los prototipos es que apoyan y enriquecen la coordinación, la

comunicación y la interacción entre los desarrolladores y los clientes; el 23,3% de los artículos indican que el uso de prototipos permite identificar fácilmente los requisitos. El 26,7% destaca que los prototipos mejoran la usabilidad del sistema. Y, por último, el 30,0% de los artículos concluye que la mayor ventaja de los prototipos es que pueden evitar el riesgo de fracaso del proyecto.

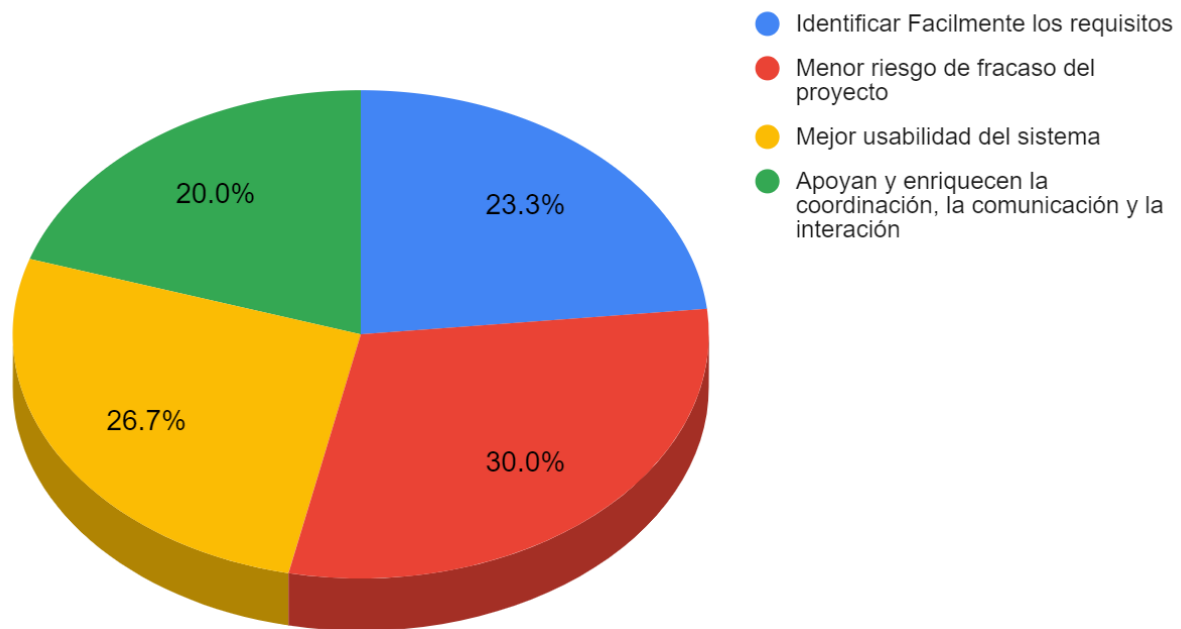


Figura 2. Ventajas del uso de prototipos en los procesos de desarrollo de software.

Teniendo esto en cuenta, la creación de prototipos se considera una herramienta importante para garantizar el éxito de los proyectos de desarrollo de software [48], esto debido a que una de las principales ventajas de incluirlos durante el proceso de desarrollo, es la reducción de posibles conflictos entre los usuarios finales y los profesionales de software, lo que minimiza los esfuerzos de reingeniería y proporciona retroalimentación valiosa en las primeras fases del desarrollo [49]. Los prototipos también facilitan una base común para la discusión y el entendimiento entre desarrolladores, usuarios y otras partes interesadas, lo que minimiza la brecha de comunicación y proporciona una retroalimentación más rápida por parte del cliente [50] [29]. Por otro lado, la creación de prototipos permite la participación activa de los usuarios en el proceso de desarrollo, lo que facilita la detección temprana de errores [3] ya que según [51], las tecnologías de prototipado rápido son muy deseables en la industria y muchas empresas han implementado estas tecnologías debido a su capacidad para reducir significativamente el tiempo y el costo de los trabajos de ingeniería en la producción de equipos tecnológicos, así como para aumentar la calidad de los productos y reducir el desperdicio de recursos a través del desarrollo de requisitos erróneos [24]; Permite identificar funcionalidades faltantes y reducir el riesgo de fracaso del proyecto [52], ya que los

prototipos también permiten la evaluación y aprobación de los conceptos de diseño del sistema por parte de los diseñadores de software, así como la identificación de posibles problemas y defectos en el diseño del sistema, minimizando así los riesgos de desarrollo al principio [24]. De este modo, los prototipos permiten a los desarrolladores del proyecto disponer de un prototipo que puede modelarse de forma rápida, interactiva y disponible para referencia y discusión en el trabajo preliminar de combinación de requisitos [24].

Es por eso que la creación de prototipos puede considerarse como una actividad preventiva. También mejora la comunicación entre los miembros del equipo [21], ya que pueden expresar sus opiniones y hacer comentarios con mayor claridad ya que el sistema de prototipado ayuda a los desarrolladores y diseñadores de software a obtener una comprensión más intuitiva del software [24], Aumenta la satisfacción del cliente, ya que le permite hacerse una primera idea del producto. Y reduce las posibilidades de rechazo del software, teniendo en cuenta que los prototipos proporcionan una representación visual del producto en desarrollo, lo que permite a los consumidores hacerse una idea clara de cómo será el producto final, además de interactuar con el prototipo, los consumidores pueden explorar sus características, funcionalidades y diseño, lo que les ayuda a formarse expectativas realistas sobre el producto.

Esta claridad de expectativas reduce la posibilidad de insatisfacción o rechazo por parte del consumidor. Proporciona información rápida al cliente, lo que permite a los responsables de la toma de decisiones obtener mejores enfoques del desarrollo. Permite al cliente evaluar si el software cumple las especificaciones porque, al interactuar con el prototipo, puede ver y probar cómo se implementan las distintas funciones y si se ajustan a las especificaciones establecidas. Esto le da la oportunidad de evaluar visualmente si el software se ajusta a lo acordado [21]. Ayuda a los responsables de la toma de decisiones a identificar las características que faltan en el sistema, ya que los prototipos son una herramienta muy utilizada para recopilar y validar requisitos, permitiendo la comunicación, discusión y definición de ideas entre los desarrolladores y los clientes [53].

Los prototipos se consideran herramientas de descubrimiento que permiten explorar cómo afectan los artefactos tecnológicos la cognición y la colaboración [54]. De este modo, el prototipado durante el proceso de desarrollo de software facilita la comprensión de las necesidades de los usuarios [55], lo que resulta especialmente útil para los desarrolladores que tienen dificultades para abordar todos los requisitos por completo. Fomenta la flexibilidad y la innovación en el diseño, porque es un modelo claro y fácil de entender [21].

El prototipado puede servir de base para derivar las especificaciones de un dispositivo, porque permite definir características y requisitos, evaluar la viabilidad técnica, obtener opiniones de los usuarios, facilitar la comunicación y orientar para el desarrollo posterior [21]. Por último, los prototipos proporcionan una formación temprana a los consumidores de futuros sistemas de software, ya que les permiten familiarizarse con el sistema de software antes de que esté completamente

desarrollado. Al interactuar con el prototipo, los consumidores pueden explorar las funcionalidades, interfaces y flujos de trabajo del sistema. Esto les da la oportunidad de entender cómo se espera que funcione el software y qué beneficios puede proporcionarles, además de mejorar la experiencia del consumidor y aumentar la probabilidad de que adopte con éxito el software una vez finalizado [21].

P2: ¿Cuál es la frecuencia de publicación de investigaciones que utilicen los prototipos de software en el proceso de desarrollo de software?

En el campo del desarrollo de software, el uso de prototipos ha sido objeto de numerosas investigaciones y análisis a lo largo de los años. La figura 3 ofrece una visión general de la distribución anual de publicaciones relacionadas con este tema específico. Un análisis detenido de los datos presentados revela un patrón en el flujo de publicaciones a lo largo del tiempo, en este sentido, es importante señalar que en 2015 hubo un flujo relativamente bajo de publicaciones, con un total de 4 investigaciones, equivalentes al 9,3% del número total de estudios. En 2016, hubo un flujo relativamente alto de publicaciones, con un total de 9 investigaciones, lo que equivale al 20,9% del total. Entre 2017 y 2018, hubo 20 estudios, que representan el 23,3% restante. En los últimos dos años (2019-2020), se publicaron 14,0% y 9,3% de las investigaciones. Es importante señalar que 2017 y 2018 fueron los años con mayor número de publicaciones sobre este tema. En consecuencia, se puede inferir que a pesar de encontrarse algo desactualizado, el campo del prototipado en la ingeniería de software continúa generando un gran interés debido a su pasado auge notable e importante. Además, persisten brechas por trabajar, lo cual requiere un continuo esfuerzo de investigación y de otorgarle la importancia necesaria. Asimismo, como se ha mencionado anteriormente, existen numerosas ventajas que deben ser aprovechadas, lo que destaca la relevancia de investigar en este ámbito.

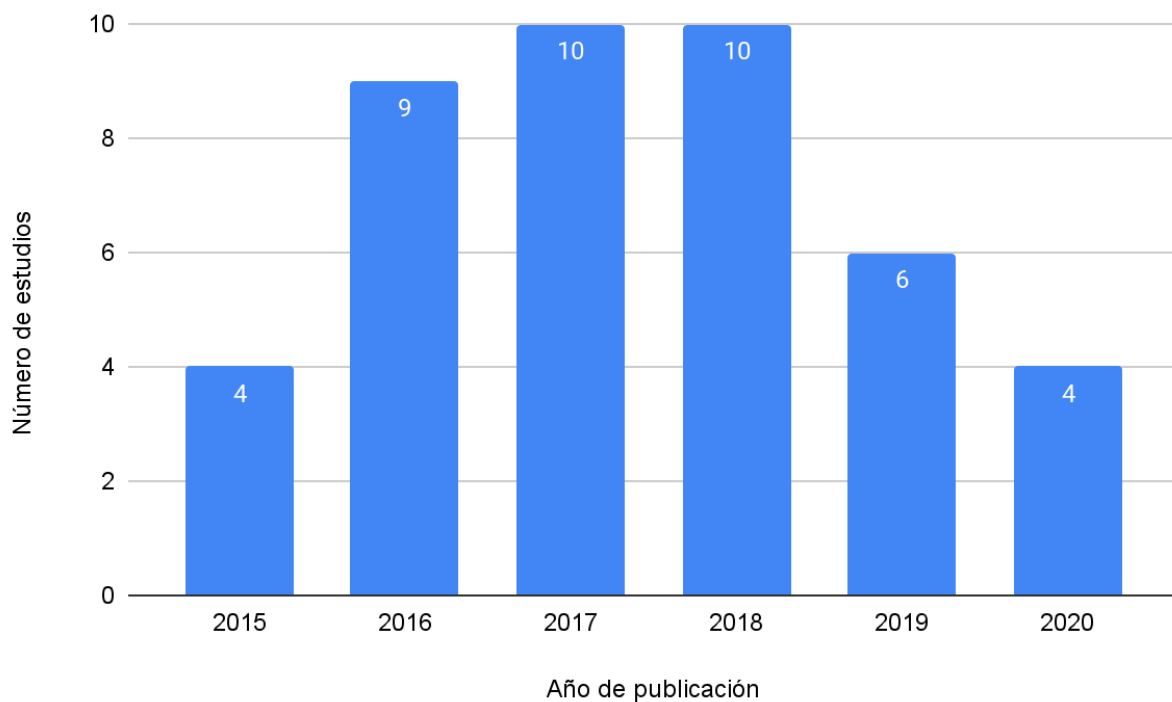


Figura 3. Distribución de investigaciones de la utilización de prototipos en el proceso de desarrollo de software.

P3: ¿Qué desventajas tiene la aplicación del prototipado en el proceso de desarrollo de software?

En el campo del desarrollo de software, el uso de prototipos ha sido ampliamente debatido y analizado, y este análisis se centra en examinar las principales desventajas identificadas por estudios anteriores en relación con el uso de prototipos en este contexto (Ver Figura 4). Estudiando en detalle los artículos aceptados, se revela que existe un consenso general sobre las limitaciones y desafíos asociados a esta estrategia. El análisis de los artículos primarios muestra que el 33,3% de ellos identifican como principal desventaja del uso de prototipos la necesidad de recursos adicionales, como software y hardware específicos cuya adquisición puede ser costosa por, lo que puede aumentar el presupuesto del proyecto; Otro motivo es el tiempo para adquirir los recursos adicionales necesarios para el prototipado lo que puede retrasar el inicio del desarrollo del software o prolongar los plazos del proyecto en general. Otra desventaja identificada por el 22,2% de los artículos son los largos plazos de desarrollo, debido a que el proceso de iteración, comunicación, coordinación adicional, sumado a los cambios de requisitos y la complejidad del software pueden extender significativamente el tiempo necesario para terminar el proyecto. Además, el 16,7% de los artículos indican que los prototipos presentan dificultades para mantener el sistema en grandes proyectos y para responder a los cambios en los requisitos de los usuarios, ya que en los grandes proyectos de desarrollo de software la complejidad del sistema puede ser elevada y, a medida que el sistema crece en tamaño y

complejidad, los prototipos pueden no ser capaces de abordar todas las interacciones y funcionalidades necesarias, lo que dificulta su mantenimiento a largo plazo; Además, los requisitos de los usuarios pueden evolucionar en el transcurso del desarrollo del software debido a nuevas demandas del mercado o a una mejor comprensión de las necesidades de los usuarios, lo que puede provocar retrasos en la aplicación de los cambios requeridos.

Por último, el 27,8% de los artículos encuentra otras desventajas, como menor flexibilidad y adaptabilidad al cambio, una menor preparación ante fallos inesperados y la dependencia de la disponibilidad de representantes adecuados del cliente para participar en el proceso de creación de prototipos; En cuanto a la desventaja de una menor flexibilidad y adaptabilidad al cambio, se debe a que los prototipos suelen diseñarse para capturar una versión inicial de los requisitos y la funcionalidad del sistema. Sin embargo, a medida que se introducen cambios y ajustes durante el proceso de desarrollo, los prototipos pueden perder flexibilidad y tener dificultades para adaptarse a los nuevos requisitos del proyecto, lo que puede limitar la capacidad de los prototipos para evolucionar y satisfacer las necesidades de los usuarios. En cuanto a la desventaja de estar menos preparados para fallos inesperados, se debe a que los prototipos se centran principalmente en la funcionalidad y la interacción del sistema, y prestan menos atención a aspectos como la estabilidad, la seguridad y la tolerancia a fallos. La desventaja de depender de la disponibilidad de representantes adecuados de los clientes se debe a que la creación de prototipos implica la participación activa de los usuarios o representantes de los clientes para proporcionar información y validar los requisitos y la funcionalidad. Sin representantes adecuados de los clientes disponibles y comprometidos en el proceso de creación de prototipos, puede haber dificultades para obtener la retroalimentación necesaria y tomar decisiones informadas sobre el sistema en desarrollo, lo que puede retrasar el proceso de creación de prototipos y afectar a la calidad del producto final.

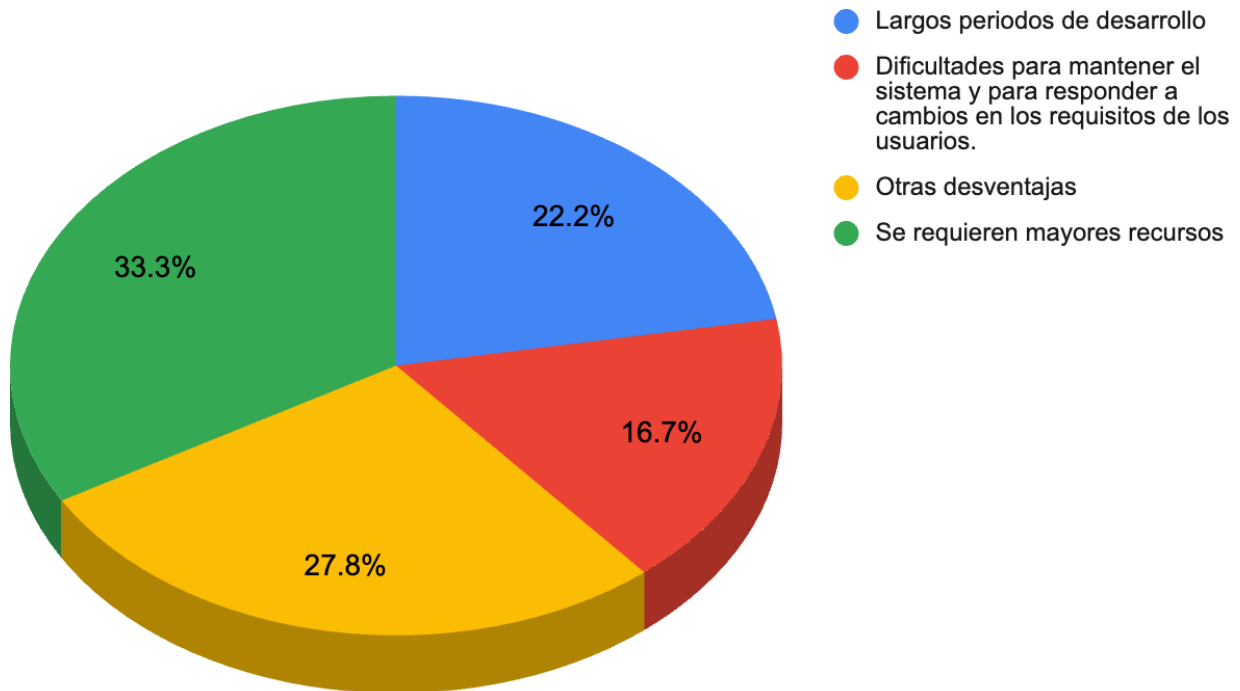


Figura 4. Desventajas del uso de prototipos en los procesos de desarrollo de software.

Otras desventajas adicionales son por ejemplo: a) no todos los métodos de tecnología de creación rápida de prototipos cumplen con los estándares de precisión, estos se refieren a un conjunto de criterios o medidas establecidas para evaluar la calidad y precisión de los resultados obtenidos en el proceso de creación de prototipos, es importante mencionar que el incumplimiento de los estándares de precisión puede resultar en prototipos defectuosos o que no satisfagan las necesidades del usuario final [56], esto puede dar lugar a retrabajo, aumentando así el tiempo y el coste de producción. Además, el usuario puede confundirse al ver un prototipo en funcionamiento y pensar que se trata del producto final [53], lo que puede generar retrasos en el proceso de desarrollo y la necesidad de tener representantes del cliente adecuados que participen en el proceso de creación de los prototipos [54] [55]. b) El proceso de creación de prototipos puede ser extenso y laborioso. c) Los costos asociados con la creación de prototipos pueden ser considerados como un desperdicio, ya que el prototipo finalmente se descarta. d) La creación de prototipos puede generar un número excesivo de solicitudes de cambio. e) Los consumidores pueden tener dificultades para participar en el proceso de iteración por períodos prolongados. Las especificaciones del software pueden variar significativamente cuando el cliente prueba el prototipo. f) La documentación puede ser insuficiente a medida que cambian los requisitos del cliente. g) Los desarrolladores de software suelen tener dificultades para gestionar todas las mejoras solicitadas por los clientes. Cuando los clientes ven un primer prototipo, pueden tener expectativas poco realistas sobre el plazo de entrega y las características del producto final. Si no están satisfechos con el prototipo inicial, el

cliente puede perder interés en el producto final. h) Los desarrolladores que buscan crear un prototipo rápidamente pueden comprometer la calidad del diseño por una producción deficiente [21].

P4: ¿Qué tipos de prototipado se utilizan en el proceso de desarrollo de software?

Luego de analizar los artículos primarios, se obtuvo que, el proceso de desarrollo de software utiliza diversos tipos de prototipado para apoyar la creación y realización de proyectos. La Figura 5 ilustra los diferentes tipos de prototipado, donde el 8,8% de los casos corresponden a prototipos desechables, experimentales y rápidos. Asimismo, el 5,9% corresponde a prototipos de sistemas piloto, mientras que el 20,6% son prototipos evolutivos y exploratorios, y el 26,5% son prototipos de presentación, siendo estos últimos los más comunes en el desarrollo de software.

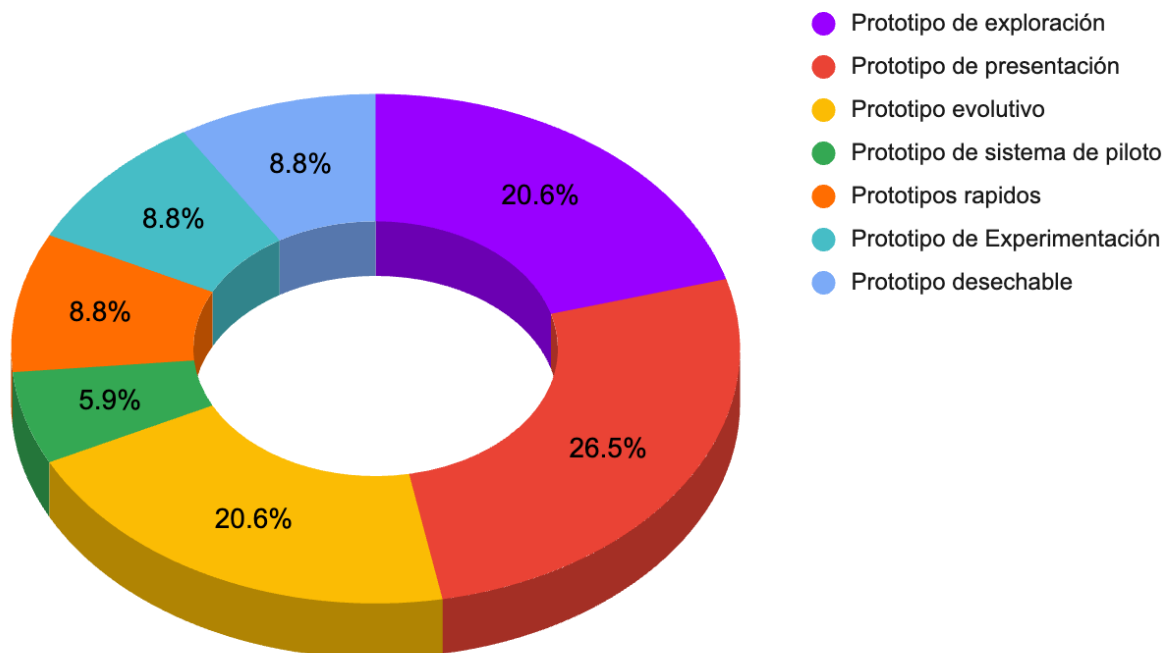


Figura 5. Tipos de prototipado que existen en los procesos de desarrollo de software.

En el ámbito del desarrollo de software, existen diversos tipos de creación de prototipos que se utilizan en función del producto o de los objetivos del proyecto. Entre los tipos de prototipado más comunes se encuentran:

1) Los prototipos desechables, los cuales se emplean para desarrollar aplicaciones de forma rápida y comprobar la viabilidad técnica del proyecto. Estos prototipos pueden ser desechados después de la validación de los requisitos, aunque también pueden evolucionar hasta convertirse en sistemas de software completos [3]. Son

particularmente útiles para evaluar tecnologías y atributos de calidad como el rendimiento y la escalabilidad [57].

2) Los prototipos de presentación proporcionan una vista previa concreta de una idea abstracta que ilustra cómo el sistema previsto puede resolver determinados requisitos y se centran principalmente en la interfaz de usuario [7]. Además, se utilizan como herramienta de persuasión para convencer a un cliente de la viabilidad de un proyecto antes de comenzar a trabajar en él [24].

3) El Prototipo evolutivo es un tipo de prototipo en el que el desarrollador define un prototipo y, una vez que ha recibido los comentarios del usuario, continúa desarrollando los siguientes prototipos, cada uno con mejoras hasta que surge el producto final. La constante interacción de los usuarios permite que el prototipo evolucione y se adapte a los requerimientos del proyecto [51]. Dentro de este proceso de creación de prototipos se encuentra el desarrollo exploratorio, que tiene como objetivo explorar con el usuario todos los requerimientos para llegar al sistema final y así el diseño irá evolucionando a medida que se agreguen nuevos requerimientos [58]. Los prototipos evolutivos pueden reducir el riesgo si se implantan como una infraestructura en la que puedan integrarse componentes y tecnologías, lo que ayuda a minimizar los riesgos de integración y respalda las primeras demostraciones de valor para el usuario [57]. Además, el software se va haciendo gradualmente más sofisticado y completo a través de una serie de versiones, en las que el equipo de mantenimiento corrige los errores, realiza adaptaciones y añade nuevas funcionalidades según las peticiones del usuario [59]. La nueva versión suele incluir nuevas funcionalidades y cambios en función de los avances actuales y las necesidades de los usuarios [53].

4) El prototipo de sistema piloto es un tipo de prototipo que constituye el núcleo de un sistema de aplicación, es decir, es una versión inicial y funcional de un producto o sistema que se desarrolla para probar y evaluar su rendimiento antes de la producción [11]. Experimentar con prototipos prepara a la organización para adoptar el sistema de aplicación al ofrecer a los usuarios una vista previa. Esto es especialmente cierto en el caso de los sistemas piloto que permiten aumentar el sistema de aplicación paso a paso [60], ya que este prototipo piloto se utiliza para verificar y validar conceptos clave, características y funcionalidades del producto o sistema, y también para identificar posibles problemas o mejoras necesarias; su principal objetivo es obtener retroalimentación y aprendizaje mediante pruebas en situaciones reales. Su desarrollo pretende simular las condiciones de funcionamiento, permitiendo a todas las partes interesadas en el desarrollo interactuar con el prototipo y evaluar su rendimiento [7].

5) El Prototipo rápido, se enfoca en diseñar rápidamente prototipos tangibles y funcionales en un breve periodo de tiempo para demostrar visualmente cómo quedaría el requisito que se ha establecido. En este sentido, los comentarios de los clientes son importantes para impulsar las modificaciones en los requisitos. Los prototipos se vuelven a crear en función de los comentarios recibidos antes de

establecer definitivamente el requisito; Este enfoque es útil para probar soluciones y obtener los requisitos del cliente [61].

6) Los prototipos exploratorios, son útiles durante las primeras fases del desarrollo de software, estos se enfocan en la comunicación entre los posibles usuarios y los desarrolladores de software, y actúan como un catalizador para fomentar la participación creativa y estimular sugerencias de todas las partes implicadas [59]. El objetivo central de este tipo de prototipos es ayudar a detallar y tener claridad en los objetivos del proyecto a realizar, permitiendo comparar las ideas conceptuales iniciales con las expectativas y requerimientos del diseño [62].

7) El prototipado experimental consiste en construir un prototipo de una solución propuesta para un problema concreto. El prototipo se evalúa mediante el uso experimental para determinar la idoneidad de la solución propuesta antes de iniciar el costoso trabajo de implementación [59].

Existen otros tipos de prototipos que también se mencionan en menor porcentaje en los artículos primarios, y que se detallarán a continuación. Estos prototipos amplían aún más el campo de investigación.

1) Los prototipos de alta fidelidad se refieren a modelos interactivos que permiten a los usuarios experimentar efectos visuales en tiempo real [58]. Estos prototipos son ejecutables y ofrecen una interacción completa, con una clara estructura de navegación y una apariencia similar a la del sistema final [63]. El equipo de desarrollo utiliza estos prototipos para probar el diseño de interacción de cada función y realizar múltiples experiencias de sitio web [64]. Esto permite un mejor diseño y procesos de negocio en el desarrollo posterior, mejorando la experiencia del usuario [65].

2) Los prototipos de baja fidelidad son representaciones aproximadas de las ideas de diseño, y generalmente se utilizan bocetos dibujados a mano y otros enfoques basados en papel [63]. Son útiles para identificar los requisitos de diseño en las fases iniciales del proyecto y para ello se emplean bocetos y wireframes [20] [7]. Estos prototipos pueden crearse rápidamente y tienen bajos costos de desarrollo, sin embargo, su utilidad para las pruebas de usabilidad es limitada y son difíciles de utilizar en fases posteriores del desarrollo [63].

3) La creación de prototipos horizontales y verticales son un tipo de prototipo muy común en el diseño de sistemas y programas de software. Los prototipos horizontales son útiles para confirmar el alcance del dispositivo y la especificación de la interfaz de usuario, así como para crear una versión demostrativa del programa que pueda ser presentada al cliente para su aceptación. En la literatura casi siempre se denomina prototipado de la interfaz persona-ordenador [60]. En el prototipo horizontal, se construyen capas individuales específicas del sistema, como la capa de interfaz de usuario con sus formularios y menús, o capas funcionales, como las transacciones de la base de datos. Estos prototipos también pueden ser utilizados para realizar estimaciones preliminares de tiempo, coste y esfuerzo para

el desarrollo. Por otro lado, los prototipos verticales suponen una mejora completa de un sistema o función y son útiles para obtener especificaciones detalladas para un fin determinado [21]. Estos prototipos pueden ayudar en la creación de una arquitectura de servidor adecuada, la obtención de especificaciones del volumen de datos y detalles de la interfaz del dispositivo para el dimensionamiento de la red y la ingeniería de rendimiento, y clarificar las demandas específicas profundizando en las características esenciales del dispositivo.

4) El Prototipo operativo [21] combina las ventajas del prototipo desechable y evolutivo con los sistemas tradicionales para proporcionar un desarrollo rápido y de calidad. Se utiliza el prototipo desechable para obtener claridad sobre requisitos poco entendidos y el prototipo evolutivo para implementar requisitos definidos.

5) Los prototipos estratégicos se utilizan para identificar y analizar los riesgos en la creación de una arquitectura y son críticos para el desarrollo de sistemas de big data que implican un alto riesgo, alta complejidad y tecnologías inmaduras [57].

P5: ¿Qué estrategias o técnicas de prototipado existen en el proceso de desarrollo de software?

Las estrategias o técnicas de prototipado son los enfoques y métodos utilizados para diseñar, crear y validar prototipos en el proceso de desarrollo de software; estas estrategias o técnicas desempeñan un papel clave en el éxito y la eficacia del desarrollo de software; a diferencia de los tipos de prototipos, que son las diferentes formas o representaciones en las que se presentan los prototipos. La Figura 6 ofrece una visión general de las diferentes estrategias o técnicas utilizadas con mayor frecuencia en los artículos primarios analizados, mostrando que el 20,6% corresponde a bocetos, el 14,7% a maquetas, el 17,6% a prototipado en papel y el 8,8% a prototipado en video. Además, el 2,9% corresponde al prototipado virtual y al prototipado incremental y, por último, el 35,3% al prototipado rápido o desarrollo rápido de aplicaciones web, siendo este último el más utilizado.

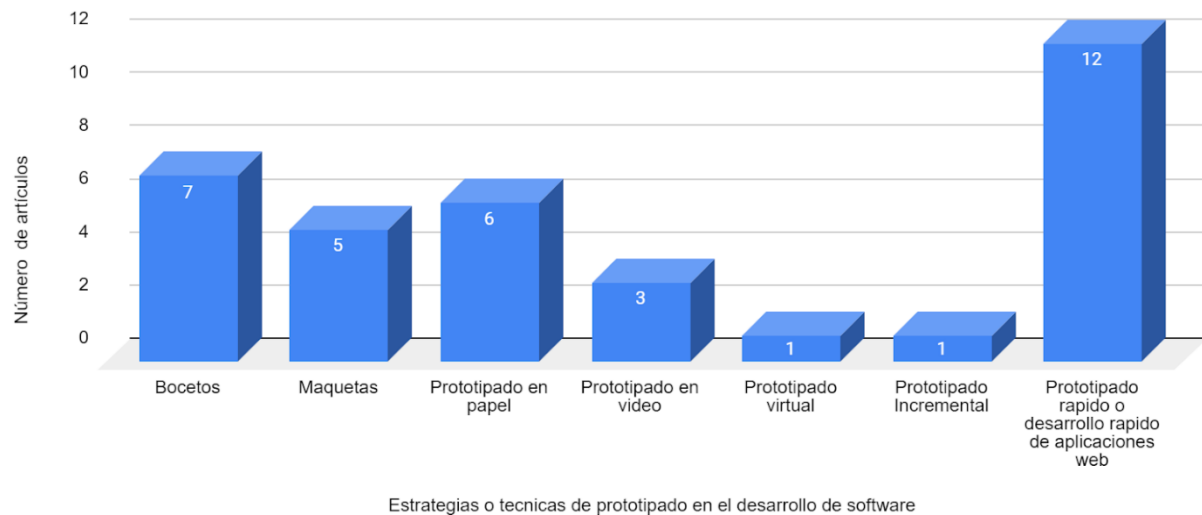


Figura 6. Estrategias o técnicas de prototipado que existen en los procesos de desarrollo de software.

En la actualidad, las técnicas de prototipado más utilizadas en los procesos de desarrollo de software son los bocetos, que son dibujos rápidos que las partes interesadas y los usuarios finales pueden realizar durante sesiones colaborativas de intercambio de ideas sobre los requisitos [8], [66]. Los bocetos son útiles como herramienta de diseño, ya que estimulan un ciclo de reinterpretación en el proceso de generación de ideas del diseñador [67]. Dado que los recursos necesarios para crear bocetos son relativamente "baratos", se clasifican como prototipos de baja fidelidad, ya que pueden crearse utilizando únicamente lápiz y papel, lo que los hace muy fáciles de elaborar [20]. De hecho, los bocetos se diferencian de otras técnicas en que es una técnica rápida e intuitiva para representar el flujo oportunista de ideas, permitiendo una exploración eficaz y amplia del problema/solución con un contenido mínimo, estimulando el descubrimiento y la exploración [68], [69]. Además, los bocetos también pueden ser utilizados por cualquier interesado, ya que es una forma de expresión que no requiere conocimientos avanzados de modelado. Por lo tanto, los bocetos se perfeccionan para lograr la validación del usuario, ya que el objetivo es proporcionar apoyo para el modelado colaborativo y flexible en las primeras fases del desarrollo [20].

Por otro lado, las maquetas o mockups son otra técnica de prototipado que se clasifica como prototipos de baja fidelidad utilizados para establecer ideas y requisitos de la interfaz de usuario con el cliente en la fase inicial del desarrollo. Los mockups complementan las técnicas de prototipado y se utilizan para realizar pruebas de usabilidad y detectar problemas antes de crear prototipos [48]. Los mockups son adecuados para las pruebas respectivas, proporcionan información importante sobre la experiencia del usuario y ayudan a optimizar el diseño en una fase temprana [62]. Por ello presenta una ventaja de usabilidad, ya que los mockups son adecuados para las pruebas respectivas, proporcionan información importante sobre la experiencia del usuario y ayudan a optimizar el diseño en una fase

temprana. Por eso, los términos maqueta e interfaz de usuario se utilizan a menudo en relación con el término prototipo, ya que mientras que los prototipos se crean para validar requisitos funcionales, las maquetas de interfaz de usuario se utilizan para explorar y validar requisitos no funcionales, como la experiencia del usuario, la usabilidad y el modelo de interacción con el usuario previsto [29]. Las maquetas interactivas son representaciones digitales de un producto o una interfaz que permiten simular cómo funciona y cómo interactúan los usuarios con él. La ventaja de utilizar maquetas interactivas es que permiten a los diseñadores y a las partes interesadas en el proyecto hacerse una idea más clara del aspecto y el funcionamiento del producto final [70], y pueden ser muy útiles en la ingeniería de requisitos iniciales y suelen utilizarse al principio del proceso de desarrollo, sobre todo cuando se dispone de poco tiempo y dinero [63].

Otra técnica de prototipado que se clasifica como prototipos de baja fidelidad es el prototipado en papel, que permite visualizar rápidamente una idea mediante un dibujo que facilita la comunicación de conceptos e ideas en una fase temprana del proceso de diseño [67]. Una de las ventajas de esta técnica es que permite a los diseñadores interactuar con los usuarios finales para obtener los requisitos y especificaciones del sistema. Además, el prototipado en papel es ideal para pruebas de usabilidad en las primeras fases del diseño, ya que es flexible y los cambios y ajustes pueden hacerse rápidamente con un lápiz y una hoja de papel. Por lo que resultan muy eficaces para el diseño de sitios web [69]. Reducción de costos, ya que el prototipado en papel es demasiado barato en comparación con otras técnicas de prototipado, lo que permite a los diseñadores disponer de todos los materiales necesarios sin caer en grandes gastos; y, por último, también hay una buena disposición por parte de los diseñadores para realizar modificaciones rápidas en el diseño, ya que han invertido un mínimo de tiempo y esfuerzo [69].

La técnica de prototipado en video, puede variar en términos de fidelidad ya que puede ser un prototipo de baja o alta fidelidad dependiendo de las necesidades del proyecto y de los recursos disponibles; esta técnica se refiere a la creación de ilustraciones en vídeo, generalmente utilizando métodos de baja tecnología, el prototipado en vídeo captura la interacción del usuario para comunicar ideas o para reflexionar sobre el diseño. Estos vídeos de bajo presupuesto son adecuados para las primeras fases del diseño, pero también pueden ser de alto presupuesto cuando se necesita un mayor nivel de fidelidad [66]. El prototipado virtual, por su parte, es una técnica que utiliza herramientas y tecnologías digitales para crear representaciones virtuales y simuladas de productos y sistemas y su nivel de fidelidad también dependerá de los objetivos del proyecto y las necesidades del equipo de diseño. Esta técnica ofrece varias ventajas como la integración y prueba continuas de los componentes del sistema durante el ciclo de desarrollo y permite realizar pruebas, es decir, probar el sistema por adelantado para detectar problemas conceptuales que no pueden preverse en la fase de planificación [71].

El prototipado incremental es una técnica utilizada para reducir el tiempo de retroalimentación entre los usuarios y el equipo de desarrollo de una aplicación. En este enfoque, el producto final se divide en varios prototipos pequeños que se crean

individualmente, lo que permite combinar posteriormente los distintos diseños en un producto cohesionado [21]. Por último, el prototipado rápido o desarrollo rápido de aplicaciones web se puede clasificar como una estrategia o técnica de prototipos de baja fidelidad y es una técnica utilizada para crear prototipos funcionales e iterativos de aplicaciones web se utilizan para obtener retroalimentación temprana y validar conceptos antes de pasar al desarrollo completo, esto ayuda a los diseñadores a probar de forma rápida y rentable los componentes del producto más propensos a presentar problemas, resolver problemas específicos y luego construir el diseño completo [21].

P6: ¿Cómo se aplican las estrategias o técnicas de prototipado en el proceso de desarrollo de software?

Las estrategias o técnicas de prototipado se aplican en el proceso de desarrollo de software de diversas formas, en función de los objetivos, las necesidades y las características del proyecto. A continuación, se describen algunos enfoques comunes para aplicar estas estrategias de prototipado en el proceso de desarrollo de software, considerando lo establecido en los artículos primarios analizados: 1) Identificación de requisitos: Como ya se ha mencionado, las técnicas de prototipado pueden utilizarse para explorar y definir los requisitos del software. Dado que la opinión del usuario es un elemento crítico en el proceso de prototipado en el campo de la interacción persona-ordenador, pueden crear prototipos de baja fidelidad, como bocetos o maquetas, para visualizar y comunicar conceptos y funcionalidades [65]. Estos prototipos iniciales ayudan a los equipos de desarrollo y a las partes interesadas a tener una comprensión compartida de los requisitos del sistema. Este proceso implica definir y desarrollar los requisitos del software, utilizando un modelo de interacción que facilite la comunicación y el intercambio de información entre el usuario y el prototipo, con el objetivo de comprender las necesidades y deseos del usuario y crear un software adaptado a sus requisitos [64].

2) Evaluación del diseño: Las técnicas de prototipado permite también, realizar evaluaciones tempranas y rápidas del diseño. Los prototipos de alta fidelidad pueden utilizarse para simular interacciones de trabajo de los usuarios, lo que facilita la identificación de problemas de usabilidad y diseño. En consonancia con lo anterior, el proceso de prototipado facilita la toma de decisiones de diseño [69]. De hecho, se ha afirmado que los prototipos son fáciles de usar y que la creación de prototipos permite la aportación del usuario, lo que resulta en un proceso de desarrollo de software más cómodo [29]. La técnica de prototipado se aplica principalmente en la fase previa a la codificación, en la que la funcionalidad se valida mediante la creación de prototipos y la evaluación basada en la interacción del usuario. En este sentido, el proceso de prototipado permite a los desarrolladores perfeccionar sus ciclos de pruebas de aceptación a lo largo de una iteración. En última instancia, los prototipos son elementos cruciales que permiten a los clientes validar la funcionalidad antes y después de la generación del código fuente [72]. Por eso, para evaluar las decisiones de diseño y el uso de prototipos con los usuarios, se ha desarrollado un modelo holístico que pone a prueba la fiabilidad del software en las primeras fases del proceso del ciclo de vida del software. El equipo de

desarrollo puede utilizar esta técnica para probar el diseño de interacción de cada función y realizar múltiples experiencias en el sitio web [50] [24].

3) Validación de conceptos: Los prototipos también se utilizan para validar conceptos o ideas antes de invertir demasiados recursos en el desarrollo completo del software. Los prototipos rápidos o experimentales pueden ayudar a probar la viabilidad o la aceptación por parte de los usuarios de un enfoque o una característica concreta. Esta validación temprana ayuda a tomar decisiones con conocimiento de causa y a reducir riesgos en fases posteriores del proyecto [3]. En este proceso de validación, la interfaz gráfica de usuario (GUI) es una parte esencial del prototipo, ya que permite a clientes y expertos del área evaluar y validar los requisitos de forma visual e interactiva [3]. Además, los desarrolladores pueden utilizar datos de muestra generados automáticamente para rellenar la interfaz y editar, añadir o eliminar entidades de la base de datos para probar el prototipo en condiciones más realistas [73].

P7: ¿Qué metodologías o procesos de desarrollo de software han incluido el uso del prototipado?

A continuación, se detallan las metodologías o procesos de desarrollo de software que han integrado el uso de prototipos, considerando los artículos primarios resultados de la RSL. La investigación y la industria han confirmado que una metodología centrada en la creación de prototipos puede mitigar algunos de los puntos débiles del enfoque de desarrollo basado en el ciclo de vida. Sin embargo, los expertos no han llegado a un consenso sobre los métodos y herramientas necesarios para respaldar el desarrollo de software orientado a la creación de prototipos [74].

En este sentido, se ha encontrado que el Modelo Espiral Avanzado y el Modelo Espiral Win-Win, creados por Boehm en 1998 [48], son modelos que proporcionan una metodología iterativa que garantiza un desempeño efectivo en cada una de las fases del Ciclo de Vida del Desarrollo de Software (SDLC) [53]. Estos modelos incluyen el uso de prototipos como parte de su enfoque y se siguen los siguientes pasos para cada proceso de desarrollo: 1) Determinar los objetivos para identificar las limitaciones del proceso teniendo en cuenta los requisitos del cliente, con la intención de proporcionar una guía clara para el desarrollo del prototipo. 2) Analizar y planificar los riesgos relacionados con el desarrollo del software. Además, se considera la planificación y búsqueda de alternativas en caso de que surjan problemas durante la creación del prototipo. 3) Desarrollar un prototipo inicial que incluya todas las funcionalidades y características clave identificadas en los requisitos. 4) Evaluar y proporcionar información al cliente para obtener una retroalimentación temprana, en la que el cliente tenga la oportunidad de interactuar con el prototipo y hacer valiosos comentarios. Estos comentarios se utilizan para mejorar el prototipo en iteraciones posteriores. 5) Iteraciones adicionales, basadas en la retroalimentación del cliente y en los requisitos de cambio identificados, se realizan iteraciones adicionales del desarrollo, incluida la construcción de un nuevo

prototipo que se mejora continuamente de acuerdo con las necesidades del cliente y los objetivos establecidos [53].

El uso de prototipos en las metodologías del Modelo Espiral Avanzado y del Modelo Espiral Win-Win contribuye y refuerza el enfoque iterativo, permitiendo obtener una mejor comprensión de los requisitos del cliente, así como disminuir riesgos, identificar problemas y, en consecuencia, proporcionar una mayor satisfacción al cliente al permitirle participar activamente en el proceso de desarrollo [53].

La utilización de prototipos puede ser beneficiosa en una metodología de diseño que se centra en la creatividad, la experimentación, el aprendizaje y la evolución [75]. Diferentes metodologías de desarrollo de software han incluido el uso de prototipos, como el desarrollo dirigido por modelos, el desarrollo de aplicaciones web y móviles, y una de las más comunes, la metodología de prototipado. Esta última se basa en un modelo iterativo de ensayo y error para identificar las especificaciones del producto y desarrollar rápidamente un prototipo de software, que los usuarios pueden probar y modificar en caso de encontrar errores o agregar nuevos requisitos [76], [21].

Otra metodología de desarrollo de software son los modelos incrementales, que comparten el concepto de entrega de proyectos de software en partes con los modelos Agile. Según esta metodología, cada incremento implementa solo una parte de los requisitos especificados del sistema de software, y el tamaño de cada incremento varía en función de los objetivos seleccionados [53]. Cada incremento incluye varias fases, como el análisis, diseño, codificación, pruebas del sistema, instalación, conversión, puesta en marcha y mantenimiento, y se integra con los incrementos anteriores para crear una nueva versión del software en desarrollo. En el contexto de la metodología incremental, los prototipos pueden desempeñar un papel importante durante la fase de análisis y diseño de cada incremento, ya que pueden utilizarse para explorar y validar ideas, así como para obtener retroalimentación temprana de los usuarios y del equipo de desarrollo; los prototipos permiten una comprensión más clara de los requisitos y facilitan la comunicación entre clientes y desarrolladores. Tanto los clientes como los desarrolladores prefieren la entrega incremental porque les permite satisfacer los objetivos más prioritarios del proyecto en el menor tiempo posible, poner en práctica los objetivos claramente definidos en el primer incremento, lo que a su vez permite explorar y perfeccionar los requisitos para los incrementos posteriores, lo que es posible gracias a la comprensión de las necesidades del cliente y los ajustes mediante la retroalimentación. La entrega incremental reduce el riesgo de fracaso del proyecto al implementar objetivos limitados en cada incremento, minimizando la posibilidad de encontrar problemas en el desarrollo del software, lo que permite una mayor flexibilidad a la hora de realizar cambios en incrementos posteriores sin tener que dañar el trabajo ya terminado [48].

En esta metodología de desarrollo de software, el producto final se construye progresivamente, añadiendo nuevas características en cada etapa incremental. Esto permite obtener resultados más rápidamente y es considerada una de las

metodologías más flexibles utilizadas en proyectos ágiles, como Scrum [62], [20] y desarrollo ágil de software [7], dentro de esta metodología, el uso de prototipos juega un papel importante, ya que se construyen y presentan prototipos en cada incremento para obtener retroalimentación temprana del cliente y del equipo de desarrollo estos prototipos permiten visualizar y probar funcionalidades específicas, lo que ayuda a mejorar los requisitos y ajustar el desarrollo en cada iteración. Además, en metodologías ágiles como Scrum, se llevan a cabo iteraciones más cortas, denominadas "Sprints", que incluyen etapas como la planificación, la ejecución y la revisión del sprint. Durante cada sprint, se pueden desarrollar y probar prototipos para validar las funcionalidades implementadas antes de pasar a la siguiente iteración. El uso de prototipos en estas metodologías incrementales permite una mayor interacción con los clientes, una mejor comprensión de los requisitos y la posibilidad de realizar ajustes y mejoras continuas en el desarrollo de software. Además, los prototipos ayudan a reducir el riesgo al permitir la detección temprana de posibles problemas y la validación de las soluciones propuestas.

P8: ¿En qué fase del proceso de desarrollo de software se utiliza el prototipado?

En el desarrollo de software, es esencial contar con un enfoque que garantice la eficacia y la calidad del proceso. El uso de prototipos se ha convertido en una práctica ampliamente reconocida para mejorar la eficacia del desarrollo de software. La Figura 7 presenta las distintas fases del proceso de desarrollo de software que incorporan el uso de prototipos, a partir de la revisión de los estudios primarios aceptados de la RSL, donde se define que el mayor porcentaje de uso de prototipos se encuentra en la fase de ingeniería de requisitos, con un 42,9% seguido de la fase de análisis y diseño, donde el 30,4% de los artículos utilizan prototipos. Además, el 25,0% de los proyectos utilizan prototipos en la fase de pruebas, mientras que solo el 1,8% los incorporan en la fase de implementación.

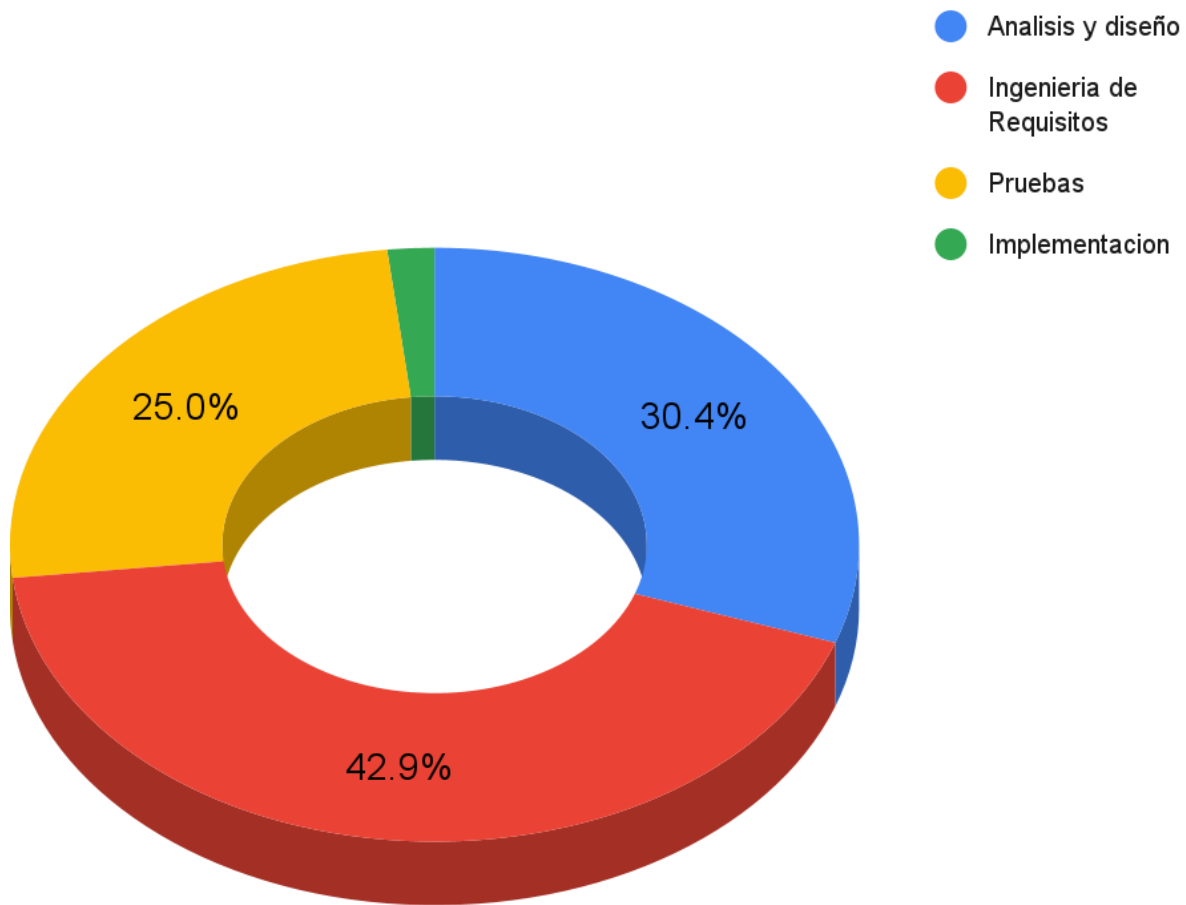


Figura 7. Fases del proceso de desarrollo de software donde se utiliza el prototipado.

Dado que el modelo de creación de prototipos comienza con una descripción general de los requisitos [21], la fase de ingeniería de requisitos se destaca como la fase principal en la que se incorpora el uso de prototipos. Esta fase se centra en la obtención y validación de requisitos, en la que el sistema de creación de prototipos complementa la falta de descripción de requisitos mediante la obtención de un esquema general sobre el diseño de la interfaz de usuario, lo que ahorra mucho tiempo de desarrollo [65]. La validación de requisitos verificará si el diseño cumple las especificaciones y el rendimiento esperado del producto. Para ello, se llevarán a cabo pruebas funcionales básicas y procesos de fabricación que hacen referencia a las etapas y métodos utilizados para convertir una idea de diseño en un prototipo funcional. Estos procesos son esenciales porque permiten realizar y evaluar el diseño en fases tempranas, lo que ayuda a detectar problemas y llevar a cabo mejoras antes de pasar a la producción, garantizando que el prototipo cumple los

requisitos establecidos y se obtenga una buena base para una producción satisfactoria del producto final. La validación de requisitos es un proceso crítico ya que se enfoca en determinar si los requisitos satisfacen las necesidades reales del cliente, es de vital importancia porque los errores en los requisitos llevarán a una revisión exhaustiva cuando se descubran esos problemas durante el desarrollo del software. Se trata principalmente de encontrar problemas en la definición de estos requisitos, ya que hay diferentes tipos de verificación en el proceso de validación [3], que son la comprobación de validez, que se centra en si los requisitos reflejan las necesidades reales de las partes interesadas y, por otra parte, la comprobación de coherencia, que se centra en si un modelo de requisitos contiene conflictos y contradicciones, especialmente para las restricciones contradictorias en el mismo contrato de la función del sistema [3].

La siguiente fase es la de análisis y diseño, la cual busca determinar o identificar con mayor precisión los requisitos del sistema, esto se logra a través de la comunicación con el usuario final [72], durante esta fase el esfuerzo principal es analizar las implicaciones de los requisitos para formar la definición inicial del sistema. Es fundamental destacar que el usuario participa en toda la fase de análisis y diseño [73]. Por otro lado, la fase de diseño corresponde a un diseño conceptual o diseño rápido, en este nivel se genera un diseño simple del sistema [21]. Esta fase implica una definición detallada de las salidas, entradas y procedimientos, incluyendo las estructuras de datos y la estructura del software, etc [53].

Por último, se sabe que la intención de la creación de un prototipo es poder evaluarlo, por lo que la siguiente fase corresponde a las pruebas, es decir, la implementación y las pruebas [57]. Esta fase de implementación se realiza para comenzar a formar a los usuarios antes de que el sistema sea totalmente funcional y para los componentes del sistema comparando los resultados de las pruebas con los obtenidos con los prototipos correspondientes [73]. Esto se hace con el fin de poner a prueba el prototipo con el propósito de mejorarlo antes de elaborar un nuevo producto. De este modo, los desarrolladores pueden estar seguros de que el producto cumplirá las expectativas y, si no las cumple, se puede seguir modificando [53]. Un prototipo demuestra la viabilidad de la funcionalidad de un sistema al tiempo que evalúa los posibles riesgos; proporciona una base común para el debate entre desarrolladores, usuarios y otras partes interesadas y mejorará la comunicación entre desarrolladores y clientes, además de ser una ventaja para cualquier proyecto de software [29]. Esta fase de pruebas consiste en poner a prueba las ideas para mejorarlas, eliminarlas o rechazarlas antes de que se implanten, en ese sentido el prototipo no tiene valor si no se prueba o si no se prueba con el usuario. Esta fase está dedicada íntegramente a las actividades de aseguramiento de la calidad, es decir, a aquellas pruebas que evalúan el sistema software en su conjunto [77]. El objetivo principal de las pruebas del sistema es descubrir el mayor número posible de errores del software, para lograr un nivel aceptable de calidad una vez que se han completado las correcciones [53].

P9: ¿Qué herramientas tecnológicas de prototipado se utilizan en el desarrollo de software?

Considerando los artículos primarios analizados, existen varias herramientas tecnológicas de prototipado ampliamente utilizadas en los procesos de desarrollo de software, estas herramientas permiten la creación de prototipos interactivos y visuales que representan la funcionalidad y apariencia del software en desarrollo. La Figura 8 muestra que las herramientas más utilizadas son Axure RP con un 16,7%, seguida de RM2PT, IFMLEdit y balsamiq con un 11,1%, luego se encuentran TAP, matlab, cacao, justinmind, pencil, spear, con un 5,6% siendo estas las menos utilizadas en este ámbito.

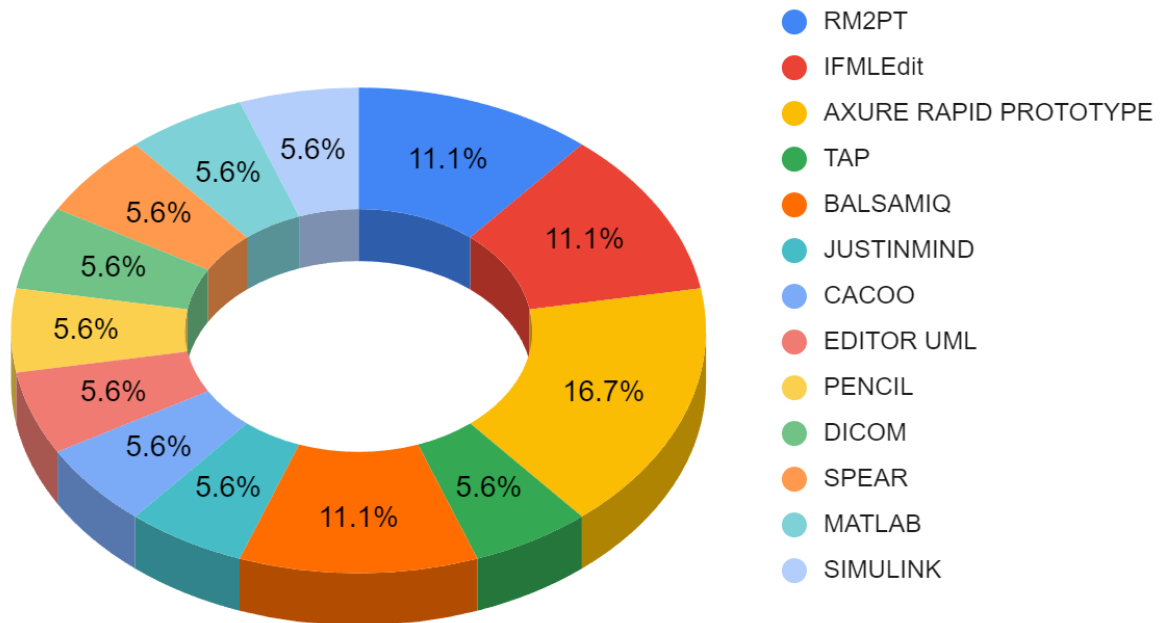


Figura 8. Herramientas de prototipado utilizadas en el proceso de desarrollo de software.

Es importante destacar que las herramientas para crear prototipos son objeto de constante revisión y comparación, lo que se evidencia en sitios web dedicados a su seguimiento y ranking. Sin embargo, estos sitios no realizan evaluaciones basadas en parámetros e indicadores, ya que es importante tener en cuenta que las evaluaciones realizadas por estos sitios suelen basarse en criterios subjetivos y dado que el uso de estas herramientas está en constante aumento, es necesario identificarlas y evaluarlas de forma rigurosa, ya que cada proyecto y equipo tiene unos requisitos y objetivos específicos, es esencial realizar una evaluación más exhaustiva que implique considerar los parámetros e indicadores específicos más relevantes para el proyecto en cuestión, lo que permitirá determinar en detalle qué herramienta de creación de prototipos se ajusta mejor a las necesidades y requisitos del proyecto, proporcionando una solución precisa y adecuada [78].

Es por ello que, en el 2010, PérezMontoro y Codina publicaron en [78], donde evaluaron diferentes herramientas para la generación de prototipos. Para dicha evaluación, consideraron cuatro categorías: operatividad, funcionalidad, plantillas y soporte. Para cada una de estas categorías, definieron indicadores ponderados para obtener un ranking de herramientas. Los indicadores considerados fueron los siguientes: 1) Operatividad: sistemas operativos, idioma y tipo de licencia. 2) Funcionalidad: formatos para importar y exportar, y tipos de representación gráfica. 3) Plantillas: inclusión de plantillas de wireframes y blueprints. 4) Soporte: tutoriales, blogs, espacios de ayuda y chats.

Por otro lado, es importante considerar que, las herramientas utilizadas para el prototipado deben ser altamente flexibles y permitir a los desarrolladores realizar cambios rápidamente mientras ven la interfaz en acción. Es fundamental que estas herramientas ofrezcan un amplio control sobre las características del prototipo y que permitan probar nuevas ideas en una variedad de diseños de interfaz. Además, deben ser capaces de soportar a grupos de personas que trabajen juntas de forma simultánea o asincrónica y quizás remotamente [79]; y una forma de familiarizar a los profesionales con el uso de este tipo de herramientas, es a través de su uso en el ámbito académico (universidades y centros de formación) ya que en el mercado existen herramientas con licencia académica, con un precio preferencial para las instituciones educativas; También existen herramientas de código abierto, que permiten a los estudiantes o profesionales crear, modificar e intercambiar programas e información, el software libre es una alternativa a las soluciones costosas, ya que proporciona una ventaja de costo y representa un ahorro potencial para las instituciones educativas de cualquier nivel, especialmente para aquellas del sector público que están sujetas a un presupuesto ajustado [78]. A medida que la creación de prototipos se ha popularizado, se han propuesto diferentes métodos y herramientas en este ámbito, como lenguajes de alto nivel, sistemas de gestión de bases de datos, sistemas de definición de diálogos, lenguajes de especificación interpretativos y sistemas de ejecución simbólica. Estas herramientas permiten construir rápidamente versiones y demostrables de los prototipos [75].

Por otra parte, se conoce que la creación de prototipos apoyaría una metodología de diseño que hace hincapié en la creatividad, la experimentación, el aprendizaje y la evolución pero desgraciadamente, en la actualidad, este tipo de sistemas aún están en el ámbito de la investigación [75], pero como alternativa más realista a corto plazo, pueden utilizarse kits de herramientas de creación de prototipos, que pueden adaptarse a aplicaciones específicas (por ejemplo, sistemas de aplicaciones interactivas). Un kit de herramientas debería incluir un editor y un sistema de procesamiento de texto, un lenguaje de alto nivel, un sistema de base de datos, herramientas de simulación, un paquete estadístico, un sistema de gestión de ventanas/pantallas y un sistema de gestión de la información para poder utilizar el software existente, todas las herramientas deben estar integradas para que el trabajo sea eficaz, estas herramientas del diálogo hombre-máquina debe ser interactivo y autoguiado para permitir cambios necesarios durante la evaluación del prototipo, dichas herramientas deben recopilar automáticamente la información

rutinaria y almacenarla en una base de datos para que pueda consultarse fácilmente [75].

Simultáneamente, los siguientes autores nos permiten identificar algunas de las herramientas de creación de prototipos, más utilizadas en el desarrollo de software: En primer lugar, tenemos la herramienta de creación de prototipos más popular y profesional llamada Axure Rapid Prototype (Axure RP), el diseño de interacción es la principal característica que distingue a Axure RP de otras herramientas de creación de prototipos de uso común, ya que crea prototipos de alta fidelidad. La herramienta ofrece muchas funciones que permiten a los usuarios diseñar y crear prototipos de interfaces de usuario, debido a que una de las características clave de la herramienta Axure RP es su capacidad para crear prototipos interactivos sin escribir código, ya que los usuarios pueden arrastrar y soltar elementos de interfaz predefinidos, como botones y campos de texto, para construir el diseño de la interfaz, esta herramienta a su vez permite la colaboración con otros miembros del equipo, permitiendo la revisión en tiempo real y la retroalimentación que facilita la colaboración y la iteración en el proceso de diseño [65].

El autor Yang en [3], nombra a RM2PT como herramienta para la generación automatizada de prototipos a partir de un modelo de requisitos en UML. Luego se tiene una herramienta de creación de prototipos, llamada SNAP, que permite a los diseñadores implementar fácilmente interacciones basadas en sensores; esta combinación entre RM2PT Y SNAP brinda a los diseñadores una forma eficiente de desarrollar prototipos funcionales lo que es de utilidad para la comprensión de las ideas de diseño antes de la implementación [58]. El autor Bernaschina en [76], hace referencia a IFMLEdit como una herramienta MDD online de código abierto, para la especificación y generación automática de prototipos rápidos de aplicaciones web y móviles a partir de especificaciones IFML (Interaction Flow Modeling Language). El autor Banchón en [78], nombra a Balsamiq como una herramienta de creación de prototipos rápida y sencilla para crear maquetas y bocetos de interfaces de usuario. Su enfoque ágil y sencillo permite a los diseñadores comunicar y explorar ideas de diseño de forma útil en las primeras fases del proceso de diseño. Por otro lado Jorgensen et al, en [48] desarrollo TAP, una herramienta para crear prototipos interactivos para dispositivos móviles y Lasecki et al, en [80] utilizo técnicas de crowdsourcing para crear mockups, dando a conocer que existen una gran variedad de herramientas para crear mockups de diferente forma, precisión, interactividad y ejecutabilidad, herramientas como: Illustrator, Sketch, Balsamiq y Marvelapp, que son capaces de crear maquetas interactivas, algunos servicios ofrecen la posibilidad de descargar y ejecutar una maqueta utilizando soluciones propietarias [29], entre otras herramientas que se incluyen en el desarrollo de software, se tiene la herramienta Matlab que sirve para la creación rápida de prototipos, diseño de interfaces e implementación del software final [68].

Por otro lado, entre las herramientas de prototipado que incluyen licencia académica (disponible a través de una aplicación) se tiene Axure RP y Cacao [78]. Cacao es una herramienta donde se pueden desarrollar dibujos que permiten la creación de prototipos creativos, esto facilita el trabajo colaborativo ya que permite que su

desarrollo sea en conjunto con los usuarios, además de que es fácil de usar. También está la herramienta Justinmind la cual tiene una versión gratuita o en algunos de los casos presenta costos por suscripción de acuerdo a rangos de usuarios [78]; esta herramienta de prototipado puede trabajar con Windows, Mac y también en iOS y Android, uno de sus principales beneficios es que permite reducir costos de desarrollo ya que permite corregir errores antes de entregar el producto final al cliente.

En un estudio presentado en [78] se define que aparentemente las mejores opciones serían Axure RP, Cacao y Pencil, ya que las dos primeras ofrecen licenciamiento académico, mientras que la tercera es gratuita, además de tomar en cuenta la parte económica, de acuerdo a la percepción de los estudiantes y en cuanto a funcionalidad, podemos notar que Cacao supera ampliamente a Axure y Pencil. Y si bien se consideran las categorías de usabilidad y aprendizaje, a consideración de los estudiantes Pencil sería una buena opción debido a que es una herramienta gratuita con una aplicación muy fácil de instalar y utilizar, que permite proporcionar una interfaz para la creación de prototipos, esta es una herramienta multiplataforma que está disponible para Linux, Mac OSX y Windows, siendo muy recomendable para el diseño de aplicaciones de escritorio, siendo una forma muy sencilla y rápida de crear un diseño que se puede mostrar al cliente antes de iniciar su desarrollo [78]. El autor Muller en [80], menciona Pocket Code como herramienta para crear y ejecutar programas. Y, por último, Spear, una herramienta de especificación y análisis de requisitos, (SpeAR) es un mecanismo de creación de prototipos y análisis de requisitos expresado en notaciones formales que captura los requisitos de un modo que se apoya en la semántica del lenguaje y permite analizarlos mediante comprobación de modelos para garantizar que son correctos, coherentes y realizables [21].

6. CAPÍTULO 4

Este apartado está dedicado a presentar y analizar los resultados obtenidos sobre la utilidad percibida por los desarrolladores de software sobre el compendio de información relacionada con el uso de prototipos en el desarrollo de software obtenido a partir de una revisión exhaustiva de la literatura existente. Este análisis pretende validar y conocer la utilidad de los prototipos y si contribuyen a mejorar los procesos de desarrollo. Finalmente, los resultados se identifican a partir de las respuestas a una encuesta realizada a 13 estudiantes de octavo semestre de ingeniería de sistemas de la Corporación Universitaria de Comfacaucá–Unicomfacaucá. Estos estudiantes, próximos a graduarse, conocen el uso de prototipos en el proceso de desarrollo de software porque están cursando la asignatura electiva "Interacción Humano-Computadora", donde utilizaron prototipos para materializar los requerimientos de un proyecto de desarrollo de software. Esta encuesta fue diseñada para evaluar la percepción de la utilidad del compendio de información de prototipado en los procesos de desarrollo de software. Los resultados obtenidos se interpretan y discuten en detalle, ofreciendo una visión clara de cómo el prototipado influye en la eficiencia y eficacia del desarrollo de software, y más en detalle al disponer de la información categorizada y organizada, permitiendo a los desarrolladores conocer de primera mano y, de forma más directa, el uso de esta estrategia. El análisis culmina con la presentación de gráficos que ilustran los resultados de la encuesta, proporcionando una representación visual que facilita la comprensión de cómo se valora el resultado obtenido de esta revisión bibliográfica sobre el prototipado en el área de desarrollo. Esta visualización permite analizar e identificar conclusiones significativas sobre el valor del prototipado en las prácticas de desarrollo de software.

A continuación, se presentan los resultados obtenidos de las respuestas proporcionadas por los encuestados.

P1: ¿Cuál es su edad?

El análisis de la Figura 9 muestra que las edades de los encuestados varían considerablemente. Un dato relevante es que el 7,7% de ellos se encuentra en una franja de edad amplia, entre 20 y 45 años. Mas concretamente, el 15,4% de los encuestados tiene 24 años, lo que es significativo dentro de la muestra. Además, el grupo más numeroso, que representa el 23,1%, corresponde a los encuestados de 25 años. Esta información sugiere que la mayoría de los participantes son adultos jóvenes, con una concentración particular en los que rondan la veintena.

¿Cual es edad?

13 respuestas

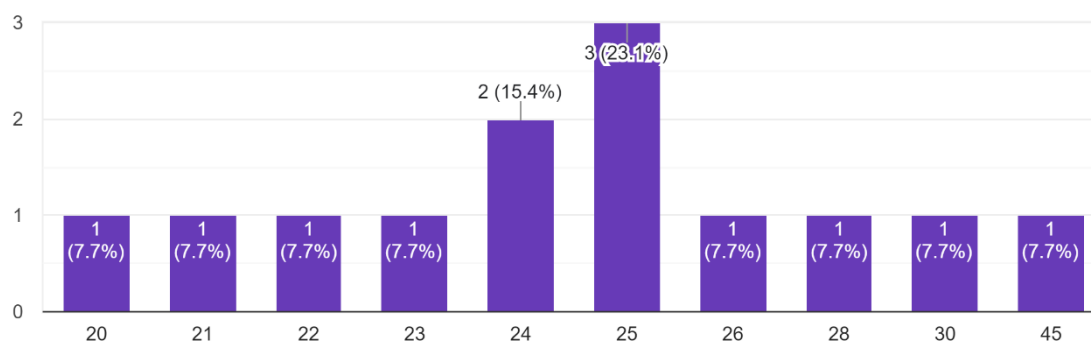


Figura 9. Edad de los encuestados

P2: ¿Cuál es su género?

El análisis de la Figura 10 revela la distribución de géneros entre los encuestados. Es importante señalar que solo el 15,4% de los encuestados se identifica como género femenino, mientras que el 84,6% restante se identifica como género masculino.

¿Cuál es su género?

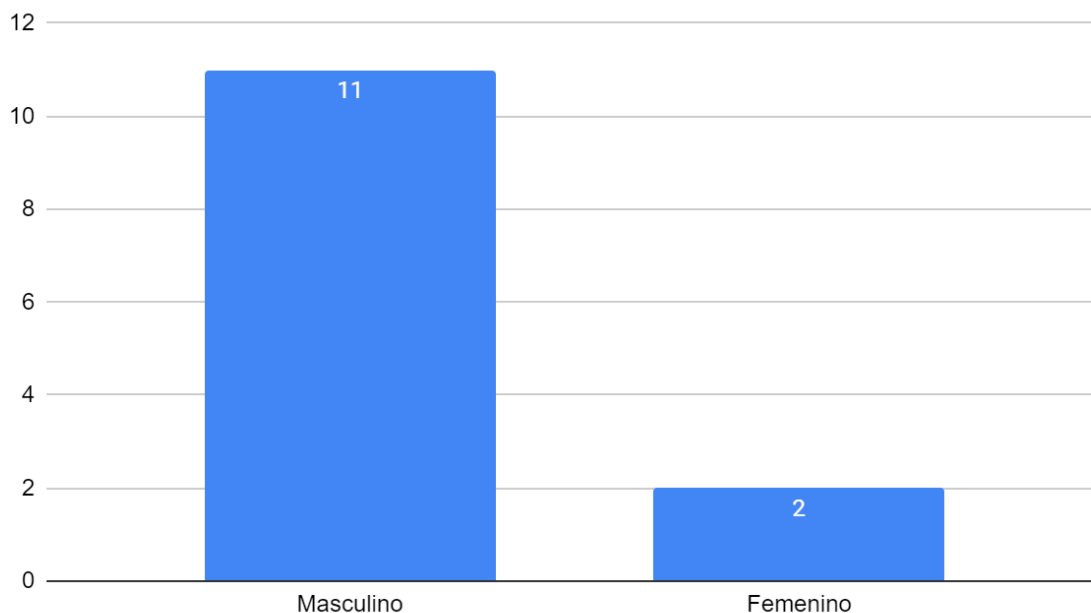


Figura 10. Género de los encuestados.

P3: ¿Cuántos años tiene de experiencia en desarrollo de software?

La figura 11 muestra que 8 de los encuestados tienen 0 experiencia en desarrollo de software equivalente al 53,8% y 7,7%. Asimismo, 3 de los encuestados indican tener 1 año de experiencia equivalente al 23,1% mientras que 2 de los encuestados tienen 2 años de experiencia en esta área equivalente al 15,4%. Estos datos ilustran la distribución de la experiencia en desarrollo de software entre los encuestados, lo que puede influir en nuestras conclusiones y futuras recomendaciones.

¿Cuanto años tiene de experiencia en desarrollo de software?

13 respuestas

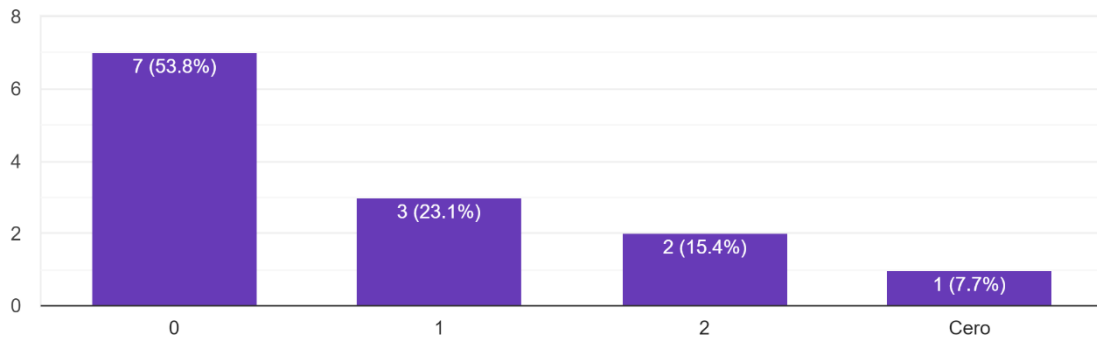


Figura 11. Años de experiencia en desarrollo de software.

P4: ¿Ha participado anteriormente en procesos de desarrollo de software que involucren el prototipado?

La figura 12 indica que el 53,8% de los encuestados no ha participado en procesos de desarrollo de software que incluyan la creación de prototipos, mientras que el 46,2% si lo ha hecho. Esto puede implicar que, los encuestados que respondieron que no, se deben a una falta de conocimiento sobre los beneficios de la creación de prototipos, a limitaciones de recursos o a alguna otra razón. También es crucial considerar cómo puede influir la participación en procesos de desarrollo de prototipos en el éxito general del proyecto, la calidad del producto final y la satisfacción del cliente. Quienes han participado en procesos de creación de prototipos pueden informar de resultados más positivos en términos de identificación temprana de problemas, adaptación a las necesidades del cliente y eficacia del desarrollo.

2. ¿Ha participado anteriormente en procesos de desarrollo de software que involucren el prototipado?

13 respuestas

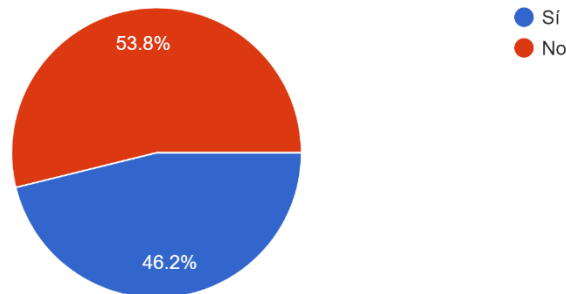


Figura 12. Participación en procesos de desarrollo de software que involucren el prototipado.

P5: Si tu respuesta fue sí. ¿Qué herramienta has utilizado para la creación de prototipos?

La figura 13 ofrece información sobre las herramientas preferidas por los encuestados para la creación de prototipos. Destaca que Adobe XD fue la herramienta más utilizada, con un 28,6% de los encuestados que optaron por ella. Además, el 14,3% de los encuestados utilizaron Balsamiq, Figma y Proto.io respectivamente. Este análisis revela una preferencia significativa por Adobe XD en comparación con otras herramientas. Esta preferencia puede atribuirse a varios factores, como la facilidad de uso, la interfaz intuitiva y la amplia gama de funcionalidades que ofrece. Cabe destacar que Adobe XD se ha ganado una reputación positiva en la comunidad del diseño gracias a su capacidad para agilizar el proceso de creación de prototipos y mejorar la colaboración entre equipos. Sin embargo, aunque Adobe XD puede ser la opción predominante, es interesante observar que otras herramientas como Balsamiq, Figma y Proto.io también tienen su base de usuarios. Estas herramientas pueden ser preferidas por su enfoque específico en determinados aspectos del diseño de prototipos o por sus características únicas que pueden satisfacer mejor las necesidades de determinados proyectos o equipos.

Según la revisión, algunas herramientas mencionadas en la literatura coinciden con las utilizadas actualmente en la academia para crear prototipos de software, como Axure RP y Balsamiq. Sin embargo, también se identifican nuevas herramientas como Adobe XD, Figma y Proto.io, que reflejan una tendencia hacia herramientas que favorecen la creación rápida y colaborativa de prototipos. Se puede concluir que esta diversidad de herramientas sugiere que la elección de la herramienta adecuada puede variar según las necesidades específicas del proyecto o las

preferencias del equipo, lo que refleja una práctica flexible y adaptativa tanto en la academia como en la industria para la creación eficaz de prototipos de software.

3. Si tu respuesta fue si. ¿Qué herramienta haz utilizado para la creación de prototipos?

7 respuestas

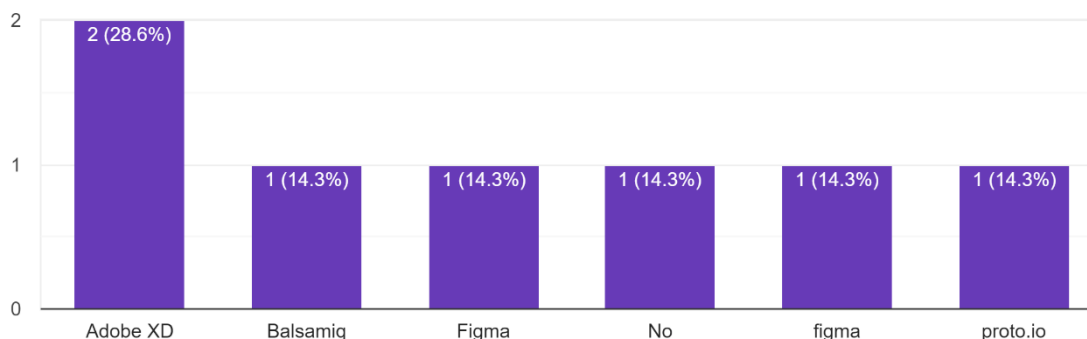


Figura 13. Herramientas utilizadas para la creación de prototipos.

P6: En una escala del 1 al 5, indique en qué medida considera que los resultados obtenidos en la revisión de la literatura sobre el prototipado son útiles para el desarrollo de software, donde 1 es "Nada útil" y 5 es "Muy útil".

La figura 14 proporciona una perspectiva importante sobre la utilidad percibida de los resultados obtenidos en la revisión de la literatura sobre el prototipado en el desarrollo de software. Destaca que el 38,5% de los encuestados considera que los resultados obtenidos son "muy útiles", mientras que el 61,5% restante los clasifica como "útiles". Este análisis revela una división en las opiniones de los encuestados sobre la utilidad de la revisión de la literatura sobre prototipado en el contexto del desarrollo de software. Los que consideran que los resultados son "muy útiles" pueden valorar la profundidad y relevancia de la información encontrada en la literatura, reconociendo su capacidad para proporcionar ideas críticas y perspectivas innovadoras que pueden mejorar significativamente el proceso de desarrollo de software. Por otro lado, el grupo más numeroso de encuestados que considera los resultados como "útiles" también reconoce el valor de la revisión de la literatura sobre prototipado, aunque puede que no se considere tan crucial como el primer grupo. Esto puede deberse a que, aunque consideran beneficiosa la información, no la perciben como esencial para el éxito del desarrollo de software. Es posible que consideren que la información aún no es lo suficientemente clara o que hay áreas que requieren más investigación. Identificar estas nuevas necesidades podría ser un objetivo para futuros trabajos.

4. En una escala del 1 al 5, indique en qué medida considera que los resultados obtenidos en la revisión de la literatura sobre el prototipado son útiles para el desarrollo de software, donde 1 es "Nada útil" y 5 es "Muy útil".

13 respuestas

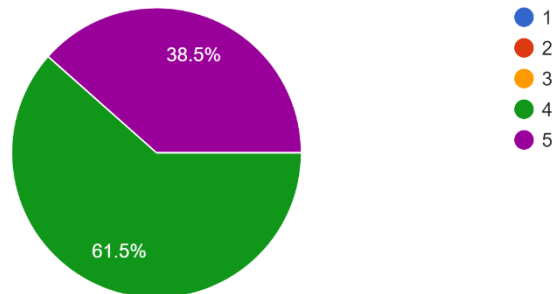


Figura 14. Resultados obtenidos en la revisión de la literatura sobre el prototipado son útiles para el desarrollo de software.

P7: ¿Qué aspectos específicos de los resultados de la revisión de la literatura considera más útiles para el desarrollo de software? (Por favor, enumere los aspectos según su importancia, donde 1 es el más importante y así sucesivamente).

A partir de las respuestas proporcionadas por los encuestados, se puede realizar un análisis detallado. Estas respuestas proporcionan información valiosa sobre diversos aspectos del tema en cuestión:

Los aspectos más útiles de los resultados de la revisión de la literatura para el desarrollo de software según los encuestados fueron 1) Las mejores prácticas y metodologías, ya que proporcionan una orientación para desarrollar software de forma eficiente y eficaz, basándose en el conocimiento acumulado y la experiencia de la comunidad. 2) Tendencias y estado de la técnica ya que permite conocer las tecnologías actuales y de este modo los desarrolladores pueden mantenerse al día y adaptar sus prácticas a medida que evoluciona el panorama tecnológico. 3) Problemas y desafíos comunes ya que es importante identificar y comprender los desafíos típicos en el desarrollo de software, lo que ayuda a anticipar y abordar posibles obstáculos durante el proceso de desarrollo. 4) Herramientas y tecnologías emergentes ya que es importante la información sobre herramientas y tecnologías emergentes proporciona a los desarrolladores recursos adicionales para mejorar su eficiencia y calidad en el desarrollo de software. 5) Casos prácticos y casos de éxito, ya que proporcionan información práctica sobre cómo aplicar las mejores prácticas en situaciones reales, inspirando y guiando el desarrollo de proyectos. 6) La creación y evaluación de prototipos como prototipos ligeros y prototipos piloto son enfoques valiosos para el desarrollo de software, cada uno con sus propias aplicaciones y ventajas específicas. Es crucial conocer bien las herramientas de creación de prototipos más utilizadas para seleccionar la opción más adecuada para

cada proyecto. 7) Aplicar filtros a la literatura, revisiones sistemáticas y evaluaciones comparativas, ya que son prácticas esenciales para garantizar que se utiliza la información más relevante y actualizada en el desarrollo de software.

El análisis de los resultados de la revisión de la literatura según los encuestados revela varios puntos clave que se consideran muy útiles para el desarrollo de software. En primer lugar, los encuestados valoran especialmente las mejores prácticas y metodologías, lo que sugiere la importancia de seguir las directrices establecidas para garantizar la eficacia de los proyectos. La atención a las tendencias tecnológicas y al estado de la técnica pone de manifiesto la necesidad de mantenerse al día y adaptar las prácticas. Identificar problemas y retos comunes ayuda a anticiparse a los obstáculos, mientras que la información sobre herramientas emergentes aumenta la calidad y eficiencia del software. Los casos prácticos y de éxito inspiran y guían a los desarrolladores. La creación de prototipos y la evaluación permiten validar las ideas antes de su implementación completa. Por último, la aplicación de filtros a la literatura y la realización de revisiones sistemáticas garantizan el uso de información pertinente y actualizada. En conjunto, estos aspectos subrayan la necesidad de una base sólida de conocimientos y prácticas actuales para desarrollar software de alta calidad.

P8: ¿Crees que, con la información suministrada, es difícil seleccionar un tipo de prototipo para ser aplicado en un contexto de desarrollo de software específico?

La figura 15 presenta que el 100% de los encuestados considera que no es difícil seleccionar un tipo de prototipo para aplicarlo en un contexto de desarrollo de software. Este resultado sugiere que la información proporcionada ha sido valiosa para proporcionar conocimiento y claridad sobre los tipos de prototipos disponibles y cómo se utilizan en diferentes proyectos según las necesidades específicas. Además, la utilidad de esta información radica en varios aspectos: como la comprensión de los tipos de prototipos, ya que la información proporcionada ha permitido a los encuestados entender claramente los diferentes tipos de prototipos disponibles y cómo se puede aplicar cada uno de ellos en situaciones específicas de desarrollo de software. También la facilidad de selección, ya que a ningún encuestado le resultó difícil seleccionar un tipo de prototipo, indica que la información les ha proporcionado las pautas necesarias para tomar decisiones informadas de manera eficaz.

6. ¿Crees que con la información suministrada, es difícil seleccionar un tipo de prototipo para ser aplicado en un contexto de desarrollo de software específico?

13 respuestas

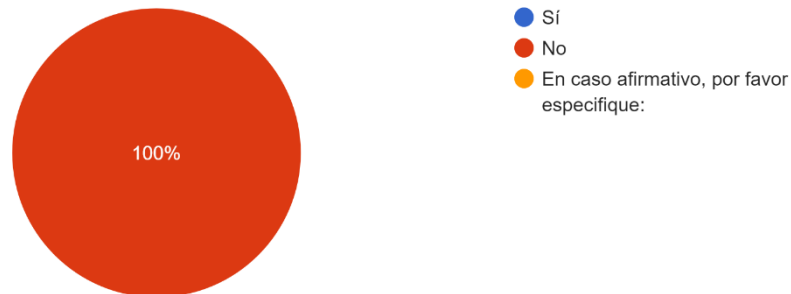


Figura 15. Resultados para seleccionar un tipo de prototipo para ser aplicado en un contexto de desarrollo de software específico.

P9: ¿Qué sugerencias tendría para mejorar la utilidad de los resultados de la revisión de la literatura en el desarrollo de software?

A continuación, se ofrece un análisis basado en las sugerencias aportadas por los encuestados: Las sugerencias para mejorar la utilidad de los resultados de la revisión de la literatura en el desarrollo de software según lo encuestados fueron:

1) Ejemplos reales de revisión de la literatura: Mostrar ejemplos concretos de cómo se ha aplicado la revisión de la literatura en el desarrollo de software mostrando estudios de casos o proyectos específicos en los que se ha utilizado esta información de forma eficaz. 2) Elementos visuales y dinámicos: Incorporar imágenes o ilustraciones que representen los conceptos y procesos tratados en la revisión de la literatura, lo que facilita la comprensión y retención de la información por parte de los lectores. 3) Hoja de ruta en el proceso de desarrollo: Utilizar la revisión de la literatura para proporcionar una guía detallada y secuencial sobre cómo abordar las diferentes etapas del proceso de desarrollo de software, lo que ayudaría a los desarrolladores a tener una dirección clara y estructurada. 4) Ejemplos aplicados al desarrollo: Complementar los conceptos teóricos con ejemplos prácticos aplicados al desarrollo de software, que ayudarían a los lectores a visualizar cómo utilizar la información en situaciones reales. 5) Lenguaje claro y sencillo: Utilizar un lenguaje claro y sencillo en la revisión de la literatura, asegurándose de que sea comprensible para todos los miembros del equipo de desarrollo, independientemente de su nivel técnico. 6) Comentarios de los usuarios: Obtener comentarios de los usuarios sobre la utilidad de la revisión de la literatura en el desarrollo de software, lo que proporciona información valiosa para mejorar y adaptar la información en función de las necesidades del público objetivo.

En resumen, las sugerencias aportadas pretenden mejorar la utilidad de los resultados de la revisión de la literatura en el desarrollo de software proporcionando ejemplos concretos, elementos visuales, guías prácticas y manteniendo la información actualizada y accesible para todos los miembros del equipo.

P10: ¿Cuál es su opinión sobre la claridad de los resultados presentados sobre prototipado para su aplicación en el desarrollo de software?

El análisis de la figura 16 muestra las diferentes opiniones de los encuestados sobre la claridad de los resultados presentados sobre prototipado para su aplicación en el desarrollo de software donde se indica que el 38,5% de los encuestados considera que la información proporcionada es "algo clara", el 15,4% se mantiene neutral en su evaluación de la claridad de los resultados por otro lado el 7,7% opina que la información es "poco clara" y finalmente el 38,5% restante la considera "muy clara". Este análisis indica que existe división de opiniones entre los encuestados, ya que una proporción significativa considera que los resultados son claros o algo claros. Sin embargo, también es importante reconocer que una minoría percibe la información como poco clara. Sin embargo, es importante abordar las preocupaciones de quienes las perciben como poco claras para garantizar que la información sea accesible y útil para todos los usuarios. Para abordar estas preocupaciones sería beneficioso realizar entrevistas más extensas con los encuestados que consideran que los resultados no son claros. Esto nos permitirá identificar con precisión qué aspectos concretos de la información les resultan difíciles de entender. Como investigación posterior se llevarán a cabo revisiones por expertos de la claridad de la información suministrada, con elementos más específicos para analizarlos con más detalle.

8. ¿Cuál es su opinión sobre la claridad de los resultados presentados sobre prototipado para su aplicación en el desarrollo de software?

13 respuestas

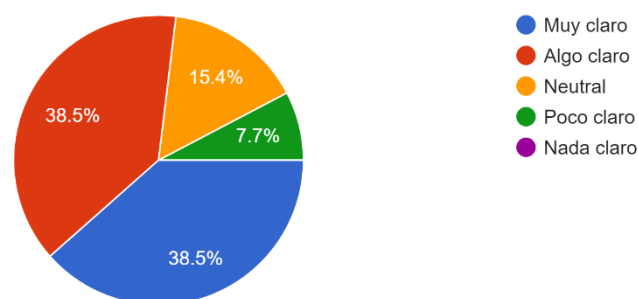


Figura 16. Resultados sobre la claridad de los datos presentados para su aplicación en el desarrollo de software.

P11: ¿Cree que sería beneficioso realizar más investigaciones o ampliar la revisión de la literatura sobre prototipado en el contexto del desarrollo de software?

Según la figura 17, el 84,6% de los encuestados considera beneficioso seguir investigando o ampliar la literatura sobre prototipado en el contexto del desarrollo de software. Este resultado refleja una gran demanda y reconocimiento de la importancia de profundizar en este tema. Por otro lado, un 7,7% no comparte esta opinión, mientras que otro 7,7% se muestra indeciso o poco claro. Sin embargo, en conjunto, estos resultados subrayan la relevancia y el interés generalizado por avanzar en la investigación sobre el prototipado en el desarrollo de software.

9. ¿Cree que sería beneficioso realizar más investigaciones o ampliar la revisión de la literatura sobre prototipado en el contexto del desarrollo de software?

13 respuestas

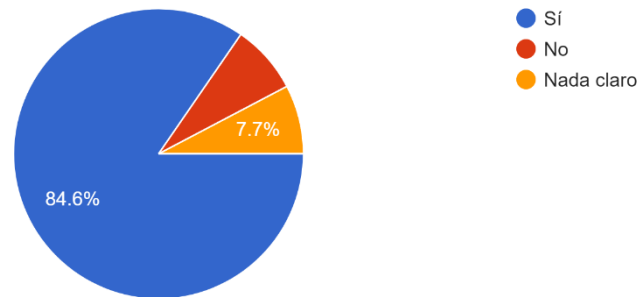


Figura 17. Resultados sobre si es beneficioso realizar más investigaciones sobre prototipado en el contexto del desarrollo de software.

Con todo esto se puede decir que es útil lo mostrado y obtenido de la revisión basándose en los resultados obtenidos y se puede concluir que la revisión de literatura sobre prototipado en el desarrollo de software ha proporcionado una información muy valiosa a los encuestados. La variada distribución de la experiencia en el desarrollo de software subraya la diversidad de perspectivas y niveles de conocimiento entre los participantes, lo cual es crucial para comprender cómo influye la experiencia en las prácticas de desarrollo. Además, las claras preferencias hacia herramientas específicas como Adobe XD y la percepción generalmente positiva de la utilidad de la revisión de literatura reflejan la importancia de seguir las mejores prácticas, mantenerse al día de las tendencias tecnológicas y utilizar herramientas eficaces en el proceso de creación de prototipos. Aunque hay una minoría a la que los resultados le parecen menos claros, la mayoría considera beneficioso ampliar la investigación en este campo, lo que pone de relieve la continua relevancia de mejorar la comprensión y la aplicación del prototipado en el desarrollo de software. En conjunto, estos resultados respaldan la utilidad y relevancia de la revisión de literatura como guía fundamental para optimizar las prácticas y promover la innovación en los proyectos de desarrollo de software.

La validación del trabajo realizado demuestra la utilidad y efectividad del prototipado en el contexto educativo y profesional del desarrollo de software. La encuesta realizada proporciona una confirmación empírica de la hipótesis y los objetivos del estudio, ya que los resultados indican que la mayoría de los estudiantes perciben el prototipado como una herramienta valiosa en el desarrollo de software, validando la premisa de que el prototipado puede mejorar la eficiencia y la calidad del desarrollo de software, que es el objetivo central del trabajo.

También destaca que existe una preferencia de los encuestados por herramientas específicas como Adobe XD lo que valida la elección de estas herramientas como estándar de la industria y sugiere que son eficaces y accesibles para la creación de prototipos. Este hallazgo apoya la recomendación de estas herramientas en el contexto educativo y profesional. Los resultados de la validación también indican que los estudiantes que han participado en procesos de creación de prototipos encuentran valor en esta práctica, lo que puede influir positivamente en su desarrollo profesional. Al incorporar las sugerencias de mejora, el trabajo puede proporcionar una orientación más eficaz y práctica para los futuros profesionales, ayudándoles a aplicar las técnicas de prototipado de forma más eficiente y eficaz en sus proyectos.

7. CAPÍTULO 5

Esta sección tiene como objetivo principal presentar las conclusiones obtenidas a partir de la investigación realizada, así como señalar las orientaciones para futuros trabajos. Pretende consolidar y analizar los hallazgos obtenidos a lo largo del estudio, proporcionando una visión global de su relevancia en el contexto de la investigación. A través de un análisis detallado, se pretende ofrecer recomendaciones prácticas y orientadas a la acción, identificando posibles áreas de mejora o nuevas líneas de investigación. Esta sección sirve como cierre del ciclo de investigación, ofreciendo una síntesis significativa y relevante que contribuya al avance del conocimiento en el campo abordado.

7.1. Conclusiones

Esta revisión sistemática de la literatura ha sido un paso significativo para identificar y clasificar los elementos de prototipado que pueden apoyar y mejorar el proceso de desarrollo de software y el área de ingeniería del software. Los resultados principales de esta revisión de la literatura se obtuvieron mediante un proceso global que constó de tres fases: planificación de la búsqueda, ejecución de la búsqueda y presentación del informe de revisión. Se aplicó la cadena de búsqueda definida anteriormente a cada una de las tres bases de datos bibliográficas, IEEEExplore, ACM Digital Library y Scopus para un total de 217 artículos. Posteriormente, se eliminaron las investigaciones duplicadas (36) y se aplicaron los criterios de inclusión y exclusión a cada uno de los artículos, considerando solamente la lectura del resumen, palabras clave y título, obteniéndose un total de 181 resultados, denominados "candidatos a estudios primarios". En la fase 2, se aplicaron los criterios de inclusión y exclusión a la lectura del resumen, palabras clave, título y conclusiones, obteniendo un total de 54 artículos denominados "artículos primarios". Por último, en la fase 3, los artículos seleccionados como artículos primarios fueron sometidos a una evaluación de calidad utilizando el mecanismo de puntuación definido anteriormente. Tras esta evaluación, se obtuvieron un total de 43 artículos de alta calidad, que fueron considerados para su inclusión en la revisión final, y con los que se buscó recopilar la mayor cantidad de información posible, con el fin de responder a cada una de las preguntas de investigación especificadas para esta RSL. Estos artículos incluyen la identificación de elementos de prototipado y la clasificación de diversas técnicas como bocetos, maquetas y prototipado rápido o desarrollo rápido de aplicaciones web. También se identificaron herramientas utilizadas en el desarrollo de software, como Axure RP, RM2PT e IFMLEdit.

El diseño de esta revisión sistemática de la literatura, tuvo el propósito de establecer una estrategia de búsqueda eficaz que facilitara la identificación exhaustiva y precisa de estudios o documentos relevantes para el tema de investigación. En esta parte del diseño se formuló un conjunto de preguntas de investigación, se identificaron fuentes de información pertinentes, se seleccionaron términos y palabras clave relevantes, y se desarrollaron estrategias de búsqueda efectivas. Todos estos elementos son fundamentales para establecer una base sólida en la búsqueda de información y garantizar la calidad y relevancia de los hallazgos

obtenidos. La ejecución de la revisión de literatura permitió recopilar, examinar y sintetizar los hallazgos relevantes de las investigaciones previas relacionadas con nuestro tema de estudio. En este contexto, se presentaron los resultados obtenidos a partir de la estrategia de búsqueda definida previamente, y se realizó un análisis detallado de los resultados, evaluando la calidad, relevancia, y contribuciones de los estudios seleccionados. Además, se presentan las respuestas a las preguntas de investigación planteadas. Al proporcionar una visión clara y completa de las opciones disponibles, se ofrece a los profesionales de ingeniería del software una base sólida para la mejora de sus prácticas de desarrollo.

El principal problema identificado en el desarrollo de software es la falta de orientaciones claras y estructuradas sobre cómo aplicar el prototipado en distintos contextos. Esta carencia genera confusión y dificultades entre los desarrolladores, quienes pueden sentirse abrumados por la variedad de técnicas, herramientas y enfoques disponibles, dificultando la toma de decisiones informadas. La revisión de literatura realizada en este proyecto ha buscado abordar esta problemática y ha encontrado y proporcionado un consolidado más organizado y estructurado de información, detallando las distintas técnicas, herramientas y enfoques de prototipado, así como describiendo claramente las diferentes fases y actividades del proceso. A pesar de estos avances, aún quedan áreas por explorar, como la necesidad de guías más específicas y adaptadas a distintos tipos de proyectos y equipos de desarrollo. Además, es crucial seguir investigando para mantener actualizada la información sobre las mejores prácticas y nuevas herramientas emergentes en el campo del prototipado. Este proyecto se desencadenó por la evidente demanda de los desarrolladores de contar con recursos más claros y aplicables para mejorar la eficacia y eficiencia del proceso de diseño y, en última instancia, la calidad del producto final. Por lo tanto, el trabajo realizado hasta ahora sienta una base sólida para futuras investigaciones y mejoras continuas en la aplicación del prototipado en el desarrollo de software.

La revisión también enfatiza la importancia del prototipado en las fases críticas del desarrollo, como la ingeniería de requisitos, análisis y diseño. La aplicación adecuada del prototipado en estas etapas tempranas facilita la comunicación efectiva y la comprensión precisa de los requisitos del cliente, lo que conduce a una mayor satisfacción del cliente y una reducción de riesgos. En esta línea, se concluye que, gracias a los análisis de los artículos primarios revisados, que la creación de prototipos ofrece varias ventajas, como la mejora de la comunicación entre desarrolladores y clientes, la identificación más fácil de los requisitos, la mejora de la usabilidad del sistema y la reducción del riesgo de fracaso del proyecto. Sin embargo, también se identificaron retos como la necesidad de recursos, tiempo y costes adicionales, la dificultad de mantener y adaptar el sistema en proyectos grandes y complejos, y la dependencia de la disponibilidad de representantes adecuados de los clientes. Además, se destaca la complejidad de la recopilación de requisitos y la necesidad de una participación activa de todas las partes interesadas. El prototipado se identifica como una herramienta valiosa para mejorar la comprensión y la comunicación en estas fases críticas del desarrollo, lo que resulta en productos de software más efectivos y satisfactorios.

7.2. Trabajos futuros

A partir de los hallazgos de este proyecto, se sugieren varias áreas para futuras investigaciones y desarrollos en el campo de la creación de prototipos en el desarrollo de software. En primer lugar, es recomendable investigar y desarrollar herramientas de creación de prototipos más avanzadas y eficaces, que puedan optimizar aún más los procesos de desarrollo de software. Estas herramientas podrían mejorar la precisión, velocidad y adaptabilidad de los prototipos, facilitando su integración en las distintas fases del ciclo de vida del software. Además, se proponen la elaboración de guías detalladas para el uso eficaz de los prototipos. Aunque la creación de prototipos ha demostrado ser una técnica valiosa, es crucial establecer directrices y mejores prácticas para su aplicación con éxito. Estas guías deben abordar aspectos como el alcance y la profundidad de la creación de prototipos en las distintas fases de desarrollo, así como su integración en metodologías de desarrollo de software ágiles y tradicionales.

Otro aspecto importante para futuras investigaciones es la definición de elementos de creación de prototipos más claros y eficaces. Estos elementos deberían promover una mejor comunicación y entendimiento entre los equipos de desarrollo y las partes interesadas, así como fomentar una comprensión más profunda de los requisitos y la funcionalidad del software. Además, se sugiere investigar la simplificación de las técnicas de creación de prototipos para facilitar su adopción en diferentes contextos y equipos de desarrollo. Esta simplificación permitiría a un mayor número de organizaciones incorporar el prototipado a sus procesos de desarrollo, potenciando sus beneficios. En conclusión, abordar estas áreas puede enriquecer significativamente el campo de la creación de prototipos en el desarrollo de software, mejorando la eficiencia, la calidad y el éxito de los proyectos. Los investigadores y profesionales de este campo pueden avanzar en el uso eficaz del prototipado como una herramienta poderosa para el diseño y desarrollo de software explorando y desarrollando estas áreas propuestas.

8. Referencias

- [1] M. A. O. Esparza, J. M. Arteaga, R. J. Á. Francisco, C. J. R. Rojano, y J. G. García, «A Production Model of Mobile Applications for Children with Learning Problems in Basic Math», en *Proceedings of the XVII International Conference on Human Computer Interaction*, Salamanca Spain: ACM, sep. 2016, pp. 1-2. doi: 10.1145/2998626.2998644.
- [2] A. Herden, P. P. M. Farias, y A. B. Albuquerque, «An Agile Approach to Improve Process-Oriented Software Development», en *Software Engineering Perspectives and Application in Intelligent Systems*, vol. 465, R. Silhavy, R. Senkerik, Z. K. Oplatkova, P. Silhavy, y Z. Prokopova, Eds., en *Advances in Intelligent Systems and Computing*, vol. 465. , Cham: Springer International Publishing, 2016, pp. 413-424. doi: 10.1007/978-3-319-33622-0_37.
- [3] Y. Yang, X. Li, W. Ke, y Z. Liu, «Automated Prototype Generation From Formal Requirements Model», *IEEE Trans. Rel.*, vol. 69, n.º 2, pp. 632-656, jun. 2020, doi: 10.1109/TR.2019.2934348.
- [4] E. Gabriel, «Metodologías de desarrollo de software».
- [5] R. S. Pressman, *Software engineering: a practitioner's approach*, 7th ed. New York: McGraw-Hill Higher Education, 2010.
- [6] X. Zhang, S. Lv, M. Xu, y W. Mu, «Applying evolutionary prototyping model for eliciting system requirement of meat traceability at agribusiness level», *Food Control*, vol. 21, n.º 11, pp. 1556-1562, nov. 2010, doi: 10.1016/j.foodcont.2010.03.020.
- [7] R. F. Ciriello, A. Richter, y G. Schwabe, «When Prototyping Meets Storytelling: Practices and Malpractices in Innovating Software Firms», en *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, Buenos Aires, Argentina: IEEE, may 2017, pp. 163-172. doi: 10.1109/ICSE-SEIP.2017.24.
- [8] B. Camburn et al., «Design prototyping methods: state of the art in strategies, techniques, and guidelines», *Des. Sci.*, vol. 3, p. e13, 2017, doi: 10.1017/dsj.2017.10.
- [9] H. Lichter, M. Schneider-Hufschmidt, y H. Zullighoven, «Prototyping in industrial software projects-bridging the gap between theory and practice», *IEEE Trans. Software Eng.*, vol. 20, n.º 11, pp. 825-832, nov. 1994, doi: 10.1109/32.368126.
- [10] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, y A. Wesslén, *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-29044-2.
- [11] M. Carr, «Prototyping and Software Development Approaches», p. 17.
- [12] M. Beaudouin-Lafon y W. Mackay, «PROTOTYPING TOOLS AND TECHNIQUES».
- [13] R. S. Pressman, *Ingeniería del software: un enfoque práctico*, Séptima edición. México; Bogotá; Buenos Aires; Caracas; Guatemala; Madrid: McGraw-Hill Education, 2013.
- [14] I. Sommerville y M. I. Alfonso Galipienso, *Ingeniería del software*, 7. edición. Madrid: Pearson Educación, 2005.
- [15] I. Sommerville, *Software engineering*, 9th ed. Boston: Pearson, 2011.

- [16] John Doe, «Recommended Practice for Software Requirements Specifications (IEEE)».
- [17] Anabel Fraga Vázquez, «CLASIFICACIONES DE TIPOS DE REQUISITOS PARA LA MEJORA DEL PROCESO DE DESARROLLO DEL SOFTWARE».
- [18] U. Rafiq, S. S. Bajwa, X. Wang, y I. Lunesu, «Requirements Elicitation Techniques Applied in Software Startups», en *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Vienna, Austria: IEEE, ago. 2017, pp. 141-144. doi: 10.1109/SEAA.2017.73.
- [19] S. Houde y C. Hill, «What do Prototypes Prototype?».
- [20] J. L. Pérez-Medina y J. Vanderdonckt, «Sketching by Cross-Surface Collaboration», en *Information Technology and Systems*, vol. 918, Á. Rocha, C. Ferrás, y M. Paredes, Eds., en *Advances in Intelligent Systems and Computing*, vol. 918. , Cham: Springer International Publishing, 2019, pp. 386-397. doi: 10.1007/978-3-030-11890-7_38.
- [21] N. Kunicina, A. Zabasta, A. Patlins, I. Bilic, y J. Peksa, «Prototyping process in education and science», en *2020 IEEE 61th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON)*, Riga, Latvia: IEEE, nov. 2020, pp. 1-6. doi: 10.1109/RTUCON51174.2020.9316550.
- [22] T. McGraw, A. Huang, y S. Wang, «Multilab: Multispectral image analysis in Matlab», en *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, Phoenix, AZ: IEEE, oct. 2017, pp. 223-224. doi: 10.1109/VAST.2017.8585672.
- [23] L. Alperowitz, A. M. Weintraud, S. C. Kofler, y B. Bruegge, «Continuous Prototyping», p. 7.
- [24] B. Hartmann, «GAINING DESIGN INSIGHT THROUGH INTERACTION PROTOTYPING TOOLS».
- [25] J. C. B. Zubietta, O. L. P. Samacá, O. Daniel, y P. Mahecha, «El papel de la Ingeniería de Software en el desarrollo de aplicaciones», vol. 4, n.º 1, 2016.
- [26] P et al., «Fundamentals of software engineering (2. ed.).», abr. 2009.
- [27] Names: Project Management Institute, publisher., *THE STANDARD FOR PROJECT MANAGEMENT and A GUIDE TO THE PROJECT MANAGEMENT BODY OF KNOWLEDGE*.
- [28] E. Christie et al., «Prototyping Strategies: Literature Review and Identification of Critical Variables», en *2012 ASEE Annual Conference & Exposition Proceedings*, San Antonio, Texas: ASEE Conferences, jun. 2012, p. 25.1091.1-25.1091.22. doi: 10.18260/1-2--21848.
- [29] S. Soobia.et.al., «Analysis of Software Development Methodologies», *IJCDS*, vol. 8, n.º 5, pp. 445-460, ene. 2019, doi: 10.12785/ijcds/080502.
- [30] M. Arias, «La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software», *InterSedes: Revista de las Sedes Regionales*, vol. VI, pp. 0-, ene. 2005.
- [31] E. Pineda Ballesteros, F. R. Tellez Acuña, y J. Medina Cruz, «Requerimientos de software: prototipado, software heredado y análisis de documentos», *inde*, vol. 37, n.º 02, pp. 327-345, jun. 2022, doi: 10.14482/inde.37.2.1053.

- [32] L. Kent, C. Snider, J. Gopsill, y B. Hicks, «Mixed reality in design prototyping: A systematic review», *Design Studies*, vol. 77, p. 101046, nov. 2021, doi: 10.1016/j.destud.2021.101046.
- [33] C. Arndt Hansen, *Design by prototyping: strategic support for prototypedriven product development: PhD thesis*. Lyngby: DTU Mechanical Engineering: DCAMM, 2021.
- [34] Mónica Patiño Castro¹, Luis Merchán², César Pardo³, «IMPLEMENTACIÓN DE MÉTODOS ÁGILES PARA LA SIMULACIÓN DE CASOS DE USO Y PROTOTIPADO EN EL PROCESO DE DESARROLLO DE SOFTWARE».
- [35] M. Kapyaho y M. Kauppinen, «Agile requirements engineering with prototyping: A case study», en *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, Ottawa, ON, Canada: IEEE, ago. 2015, pp. 334-343. doi: 10.1109/RE.2015.7320450.
- [36] A. De Oliveira Sousa y N. M. C. Valentim, «Prototyping Usability and User Experience: A Simple Technique to Agile Teams», en *Proceedings of the XVIII Brazilian Symposium on Software Quality*, Fortaleza Brazil: ACM, oct. 2019, pp. 222-227. doi: 10.1145/3364641.3364667.
- [37] M. J. Grant y A. Booth, «A typology of reviews: an analysis of 14 review types and associated methodologies», *Health Info Libraries J*, vol. 26, n.º 2, pp. 91-108, jun. 2009, doi: 10.1111/j.1471-1842.2009.00848.x.
- [38] David Moher^{1,2*}, Alessandro Liberati^{3,4}, Jennifer Tetzlaff¹ y , Douglas G. Altman⁵, «Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement».
- [39] H. Jpt, «Manual Cochrane de revisiones sistemáticas de intervenciones».
- [40] W. M. D. Santos, S. R. Secoli, y V. A. D. A. Püschel, «The Joanna Briggs Institute approach for systematic reviews», *Rev. Latino-Am. Enfermagem*, vol. 26, n.º 0, nov. 2018, doi: 10.1590/1518-8345.2885.3074.
- [41] M. da Saúde, *Robis - risk of bias in systematic reviews: ferramenta para avaliar o risco de viés em revisões sistemáticas: orientações de uso*. Ms, 2016.
- [42] H. Arksey y L. O'Malley, «Scoping studies: towards a methodological framework», *International Journal of Social Research Methodology*, vol. 8, n.º 1, pp. 19-32, feb. 2005, doi: 10.1080/1364557032000119616.
- [43] L. Codina, «Revisiones bibliográficas sistematizadas», p. 87.
- [44] A. Solis y J. Hurtado, «Reutilización de software en la robótica industrial: un mapeo sistemático», *Rev. iberoam. autom. inform. ind.*, vol. 17, n.º 4, p. 354, sep. 2020, doi: 10.4995/riai.2020.13335.
- [45] A. Yoshii, D. A. Plaut, K. A. McGraw, M. J. Anderson, y K. E. Wellik, «Analysis of the reporting of search strategies in Cochrane systematic reviews», *J Med Libr Assoc*, vol. 97, n.º 1, pp. 21-29, ene. 2009, doi: 10.3163/1536-5050.97.1.004.
- [46] K. Petersen, R. Feldt, S. Mujtaba, y M. Mattsson, «Systematic Mapping Studies in Software Engineering», presentado en 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), jun. 2008. doi: 10.14236/ewic/EASE2008.8.
- [47] N. I. R. Yassin, S. Omran, E. M. F. El Houbay, y H. Allam, «Machine learning techniques for breast cancer computer aided diagnosis using different image

- modalities: A systematic review», *Computer Methods and Programs in Biomedicine*, vol. 156, pp. 25-45, mar. 2018, doi: 10.1016/j.cmpb.2017.12.012.
- [48] B. W. Boehm, T. E. Gray, y T. Seewaldt, «Prototyping Versus Specifying: A Multiproject Experiment», *IEEE Trans. Software Eng.*, vol. SE-10, n.º 3, pp. 290-303, may 1984, doi: 10.1109/TSE.1984.5010238.
- [49] K. K. Mohan, A. Srividya, y A. K. Verma, «Prototype dependability model in software: an application using BOCR models», *Int J Syst Assur Eng Manag*, vol. 7, n.º 2, pp. 167-182, jun. 2016, doi: 10.1007/s13198-016-0427-1.
- [50] P. Beynon-Davies, D. Tudhope, y H. Mackay, «Information systems prototyping in practice».
- [51] F. Kordon y Luqi, «An introduction to rapid system prototyping», *IEEE Trans. Software Eng.*, vol. 28, n.º 9, pp. 817-821, sep. 2002, doi: 10.1109/TSE.2002.1033222.
- [52] Y.-K. Lim, E. Stolterman, y J. Tenenberg, «The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas», *ACM Trans. Comput.-Hum. Interact.*, vol. 15, n.º 2, pp. 1-27, jul. 2008, doi: 10.1145/1375761.1375762.
- [53] «From SDLC to Agile - Processes and Quality Assurance Activities», en *Software Quality: Concepts and Practice*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2018, pp. 635-666. doi: 10.1002/9781119134527.app4.
- [54] K. L. Henson y G. A. Knezek, «The Use of Prototyping for Educational Software Development», *Journal of Research on Computing in Education*, vol. 24, n.º 2, pp. 230-239, dic. 1991, doi: 10.1080/08886504.1991.10782004.
- [55] M. A. Larionov, D. V. Gusev, y A. A. Krapostin, «Development of a Computer Program to Predict the Expected Parameters of the Accuracy of the Future Product Manufactured by Rapid Prototyping Technology», p. 4.
- [56] R. M. Kimmond, «Survey into the acceptance of prototyping in software development», en *Proceedings Sixth IEEE International Workshop on Rapid System Prototyping. Shortening the Path from Specification to Prototype*, Chapel Hill, NC, USA: IEEE Comput. Soc. Press, 1995, pp. 147-152. doi: 10.1109/IWRSP.1995.518584.
- [57] H.-M. Chen, R. Kazman, y S. Haziyevev, «Strategic Prototyping for Developing Big Data Systems», *BIG DATA*, p. 8.
- [58] T. S. Kim, S. Hwang, y J. Gim, «SNAP: Sensor Aid Prototyping Tool for Designers», en *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, Seoul Republic of Korea: ACM, abr. 2015, pp. 2217-2222. doi: 10.1145/2702613.2732740.
- [59] P. J. Mayhew y P. A. Dearnley, «An Alternative Prototyping Classification», *The Computer Journal*, vol. 30, n.º 6, pp. 481-484, dic. 1987, doi: 10.1093/comjnl/30.6.481.
- [60] R. Budde, K. Kautz, K. Kuhlenskamp, y H. Züllighoven, «What is prototyping?».
- [61] F. García, «Revisiones y mapeos sistemáticos de literatura». marzo de 2019. [En línea]. Disponible en: <https://goo.gl/yt7wKt>
- [62] A. Abdul, J. M. Bass, H. Ghavimi, y P. Adam, «product innovation with scrum a longitudinal case study», p. 6, 2017.
- [63] K. Kolthoff, «Automatic Generation of Graphical User Interface Prototypes from Unrestricted Natural Language Requirements», en *2019 34th IEEE/ACM*

International Conference on Automated Software Engineering (ASE), San Diego, CA, USA: IEEE, nov. 2019, pp. 1234-1237. doi: 10.1109/ASE.2019.00148.

- [64] B. Suranto, «Software prototypes: Enhancing the quality of requirements engineering process», en *2015 International Symposium on Technology Management and Emerging Technologies (ISTMET)*, Langkawai Island, Kedah, Malaysia: IEEE, ago. 2015, pp. 148-153. doi: 10.1109/ISTMET.2015.7359019.
- [65] D. Wang, «Prototype Design of Landslide Forecast System Based on Axure», en *2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, Shenzhen, China: IEEE, abr. 2020, pp. 1-4. doi: 10.1109/AEMCSE50948.2020.00008.
- [66] G. Leiva, C. Nguyen, R. H. Kazi, y P. Asente, «Pronto: Rapid Augmented Reality Video Prototyping Using Sketches and Enaction», en *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, Honolulu HI USA: ACM, abr. 2020, pp. 1-13. doi: 10.1145/3313831.3376160.
- [67] R. Sefelin, M. Tscheligi, y V. Giller, «Paper Prototyping - What is it good for? A Comparison of Paper- and Computer-based Low-fidelity Prototyping», *NEW HORIZONS*.
- [68] B. Camburn et al., «Design prototyping methods: state of the art in strategies, techniques, and guidelines», *Des. Sci.*, vol. 3, p. e13, 2017, doi: 10.1017/dsj.2017.10.
- [69] H. M. Grady, «Web site design: a case study in usability testing using paper prototypes», en *18th Annual Conference on Computer Documentation. ipcc sigdoc 2000. Technology and Teamwork. Proceedings. IEEE Professional Communication Society International Professional Communication Conference and ACM Special Interest Group on Documentation Conferenc*, Cambridge, MA, USA: IEEE, 2000, pp. 39-45. doi: 10.1109/IPCC.2000.887259.
- [70] S. R. Humayoun, S. Hess, F. Kiefer, y A. Ebert, «i2ME: a framework for building interactive mockups», en *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*, Munich Germany: ACM, ago. 2013, pp. 606-611. doi: 10.1145/2493190.2494437.
- [71] P. O. Antonino, «Enabling Continuous Software Engineering for Embedded Systems Architectures with Virtual Prototypes», p. 16.
- [72] J. F. S. Marion Buchenau, «Experience Prototyping», 2000.
- [73] «HANDS-ON PROTOTYPING IN SYSTEM ANALYSIS AND DESIGN», *IIS*, 2005, doi: 10.48009/1_iis_2005_10-14.
- [74] G. Pomberger, W. Bischofberger, D. Kolb, W. Pree, y H. Schlemm, «Prototyping-Oriented Software Development— Concepts and Tools».
- [75] C. Floyd, «A Systematic Look at Prototyping», en *Approaches to Prototyping*, R. Budde, K. Kuhlenkamp, L. Mathiassen, y H. Züllighoven, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, pp. 1-18. doi: 10.1007/978-3-642-69796-8_1.
- [76] C. Bernaschina, S. Comai, y P. Fraternali, «IFMLEdit.org: Model Driven Rapid Prototyping of Mobile Apps», en *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, Buenos Aires, Argentina: IEEE, may 2017, pp. 207-208. doi: 10.1109/MOBILESoft.2017.15.

- [77] A. Geras, M. Smith, y J. Miller, «A prototype empirical evaluation of test driven development», en *10th International Symposium on Software Metrics, 2004. Proceedings.*, Chicago, IL, USA: IEEE, 2004, pp. 405-416. doi: 10.1109/METRIC.2004.1357925.
- [78] Jimmy Banchon Z, M. Villavicencio, y E. Izquierdo, «Evaluation of prototyping tools based on categories and parameters», en *2016 IEEE Colombian Conference on Communications and Computing (COLCOM)*, Cartagena, Colombia: IEEE, abr. 2016, pp. 1-6. doi: 10.1109/ColComCon.2016.7516408.
- [79] P. Szekely, «User interface prototyping: Tools and techniques», en *Software Engineering and Human-Computer Interaction*, vol. 896, R. N. Taylor y J. Coutaz, Eds., en *Lecture Notes in Computer Science*, vol. 896. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 76-92. doi: 10.1007/BFb0035808.
- [80] M. Muller, C. Schindler, y W. Slany, «Pocket Code - A Mobile Visual Programming Framework for App Development», en *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, Montreal, QC, Canada: IEEE, may 2019, pp. 140-143. doi: 10.1109/MOBILESoft.2019.00027.