Estudio exploratorio sobre la implementación de técnicas de Lean Manufacturing en desarrollo de Software (Lean Software Development).

Autor

Jhon Edward Rubio Ardila

Trabajo de Grado para Optar el Título de Ingeniero Industrial

Director

Néstor Raúl Ortiz Pimiento

PhD. En ingeniería

Universidad Industrial de Santander

Facultad de Ingenierías Físico-Mecánicas

Escuela de Estudios Industriales y Empresariales

Ingeniería Industrial

Bucaramanga

2023

Dedicatoria

Al llegar al final de este camino académico y comenzar mi vida profesional, no puedo evitar pensar en ustedes y en el amor inmenso y confianza que han depositado en mí a lo largo de los años; quiero dedicar este logro a dos mujeres extraordinarias que han sido fundamentales en mi vida: mi mami y mi nonita. Vuestra presencia, amor y apoyo han sido mi mayor fortaleza y motivación a lo largo de este camino y han sido fundamentales para que hoy pueda celebrar la culminación de este peldaño.

Ustedes han sido mi inspiración constante. Desde mi niñez, me enseñaron el valor del esfuerzo y la perseverancia y eso ha sido mi motor para alcanzar las metas propuestas. En momentos de dificultad, me han recordado el potencial tengo.

Este trabajo de grado es el resultado, el fruto a la siembra hecha, es preciso recordar que cada éxito alcanzado es por y para ustedes.

Las amo con mi vida.

Agradecimientos

Agradezco a Dios por brindarme oportunidades y recursos para alcanzar mi título profesional. Ha sido mi roca y fortaleza en todo momento, iluminando mi camino y recordándome que nada es imposible cuando tengo fe. Su gracia y poder han sido mi sostén en momentos de dificultad, impulsándome a seguir adelante y superar los desafíos que se presentaban.

Este logro es el comienzo de un camino lleno de bendiciones y oportunidades en mi carrera profesional.

A Dios entrego mis sueños, metas y proyectos, con la certeza que ilumina mi camino y dirige mis pasos.

Quiero expresar mis más sinceros agradecimientos a mi director de proyecto Néstor Ortiz, invaluable guía, agradezco su apoyo y dedicación a lo largo de todo el proceso, el cual ha sido fundamental en mi formación académica y en el logro de este importante hito en mi vida.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

El Apéndice se encuentra en una carpeta adjunta.

Apéndice A. Artículo de carácter publicable.

Resumen

Título: Estudio exploratorio sobre la implementación de técnicas de Lean Manufacturing en desarrollo de Software (Lean Software Development)[*]

Autor: Jhon Edward Rubio Ardila [**]

Palabras Clave: Lean Software Development, Lean Manufacturing, Software Design.

Descripción:

La metodología Lean Manufacturing para la empresa automovilística Toyota, con el objetivo de solucionar problemas detectados en la cadena de producción de dicha empresa. Desde su concepción, su influencia ha sido tan amplia que ha alcanzado a prácticamente todo tipo de organizaciones a nivel global. Un sector donde su influencia ha sido realmente notoria es aquel asociado a la industria de desarrollo de Software. De hecho, el Lean Software Development y las metodologías ágiles que actualmente marcan el rumbo del desarrollo de este sector plantearon su filosofía y principios inspirados en el Lean Clásico. A su vez, hay un sinnúmero de herramientas actualmente empleadas que son herederas de aquellas planteadas en el Lean Manufacturing original.

La presente investigación a partir de una revisión de literatura en bases de datos científicas busca analizar la implementación de técnicas de Lean Manufacturing en el desarrollo de Software. Se identifican aquellas herramientas históricamente adoptadas, se realiza un ranking de aquellas más relevantes o mayormente empleadas, se analiza la forma en que fueron adaptadas y su aplicación en diferentes tipos de servicios ofrecidos en el mercado. Finalmente, a partir de estudios de caso identificados en la literatura se analiza el impacto de la aplicación de las herramientas Lean como estrategia de mejoramiento en el desarrollo de software.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico Mecánicas. Escuela de Estudios Industriales y empresariales. Director: PhD. Nestor Raúl Ortiz Pimiento

.

Abstract

Title: Exploratory study on the implementation of Lean Manufacturing techniques in Software development (Lean Software Development). *.
Author: Jhon Edward Rubio Ardila [1]

Key Words: Lean Software Development, Lean Manufacturing, software Design.

Description:

The Lean Manufacturing methodology was developed by the automotive company Toyota with the aim of solving problems detected in its production chain. Since its conception, its influence has been so extensive that it has reached practically all types of organizations globally. One sector where its influence has been particularly notable is that associated with the Software Development industry. In fact, Lean Software Development and agile methodologies, which currently mark the direction of development in this sector, have proposed their philosophy and principles inspired by Classic Lean. In turn, there are countless tools currently used that are heirs to those proposed in the original Lean Manufacturing.
This research, based on a literature review of scientific databases, seeks to analyze the implementation of Lean Manufacturing techniques in software development. It identifies historically adopted tools, ranks the most relevant or commonly used ones, analyzes the way they were adapted, and their application in different types of services offered in the market. Finally, based on case studies identified in the literature, the impact of applying Lean tools as an improvement strategy in software development is analyzed.

---

[*] Degree Work
[1] Faculty of Physical Mechanical Engineering. School of Industrial and Business Studies. Director: PhD. Nestor Raúl Ortiz Pimiento

Introducción

El término “Lean” ha sido empleado para definir un conjunto de herramientas y técnicas que permiten optimizar los costos productivos a través de la eliminación de los desperdicios, repercutiendo en un aumento de la productividad y mayor rentabilidad para las empresas que lo emplean (Gómez, 2014). Surge de la metodología “Lean Manufacturing”, concebida en Japón, por Taiichi Ohno, para la empresa automovilística Toyota, con el objetivo de solucionar problemas detectados en la cadena de producción de dicha empresa (De Rojas, 2012). Actualmente, el empleo de “Lean Manufacturing” o metodologías derivadas es propia de la industria manufacturera, incluyendo en los últimos años gran desarrollo en áreas de atención médica, educación y organizaciones públicas (Socconini, 2019).

También, se ha implementado metodologías “Lean” en la industria del software. Como es de esperar, las herramientas de gestión de procesos tradicionales no están diseñadas para prosperar en la incertidumbre extrema en que crecen la industria del software y los startups y las metodologías “Lean” aplicadas se implementan de forma diferenciada.

El Dr. Robert Charette propuso por primera vez el “Lean Software Development” (LSD) como una forma de construir organizaciones tolerantes al cambio que se estaban volviendo cada vez más dependientes del software. Posteriormente, se desarrolló “El Manifiesto Agile" que consagró los 12 principios del Desarrollo de Software bajo metodologías “Ágiles”. Ambas metodologías son altamente similares (Tecnologías Información, s.f.).

A partir de allí, son numerosas y variadas las metodologías influenciadas por los postulados fundamentales del “Lean Manufacturing” que hoy marcan el rumbo del desarrollo de software en las empresas de tecnología a nivel mundial. Así, es pertinente evaluar cómo se dio esta influencia,

cuáles y cómo fue la adopción de las herramientas y comprender el impacto que estas tienen actualmente en la industria del software.

Para llevarlo a cabo, se realiza una completa revisión de literatura que recapitula y analiza las principales herramientas y técnicas del “Lean” clásico empleadas actualmente en el desarrollo de software, bien sea bajo la metodología “Lean Software Development” o bajo cualquier filosofía que comparta los principios del “Lean” clásico.

De tal forma, se plantean recomendaciones y crear tendencias que orienten futuras investigaciones respecto a la temática en cuestión, y así aportar en cierta medida al desarrollo de conocimiento asociado y al impulso de la industria nacional de software.

Tabla Cumplimiento de Objetivos

A continuación, se presenta la tabla de cumplimiento de objetivos.

Tabla 1.

Tabla de complimiento de objetivos

Objetivo	**Cumplimiento**
Realizar una revisión documental acerca de los estudios desarrollados de "Lean Software Development" con el fin de determinar qué herramientas clásicas del Lean han sido aplicadas en los desarrollos de software.	6. Resultados.
Identificar qué herramientas "Lean" son empleadas con mayor frecuencia dentro de la cadena de valor en el desarrollo de software.	6.1. Componentes y Herramientas Lean Adoptados en la Industria del Software.
Analizar la forma en que fueron adaptadas las herramientas clásicas de "Lean Manufacturing" en los desarrollos de software y su aplicación en diferentes tipos de servicios ofrecidos en el mercado.	6.2. Adopción de las Herramientas Clásicas del Lean Manufacturing en el Desarrollo del Software y su aplicación en Diferentes Tipos de Servicios Ofrecidos en el Mercado.
Realizar un diagnóstico que permita comprender el impacto de la aplicación de las herramientas Lean como estrategia de mejoramiento en el desarrollo de software.	6.3Impacto de la Aplicación de las Herramientas Lean como Estrategia de Mejoramiento en el Desarrollo de Software.

1. Planteamiento del Problema

El término Lean Manufacturing (Manufactura esbelta) hace alusión “Al proceso continuo y sistemático de identificación y eliminación del desperdicio o exceso, entendiendo como exceso toda aquella actividad que no agrega valor en un proceso, pero sí costo y trabajo” (Socconini, 2019). La filosofía Lean busca optimizar y reducir los desperdicios en cualquier tipo de organización, ya sea de servicios o de producción de bienes. Este enfoque busca fomentar la efectividad y la innovación en las empresas para lograr el éxito. La filosofía Lean es ampliamente conocida por sus contribuciones a la industria en el siglo XX, particularmente en relación con la empresa japonesa Toyota (Socconini, 2019).

Debido a su relevancia se considera un estándar en términos de calidad y gestión organizativa. Tanto así que se podría pensar que todos los sistemas de calidad o gestión en empresas de manufactura actuales tienen en algún porcentaje influencia de la filosofía Lean y la inclusión de una o más herramientas surgidas del Lean Clásico.

De tal forma, vale la pena también mencionar que no es el único sector en el cual sus aplicaciones han generado aportes. La filosofía Lean es aplicable en cualquier tipo de organización, ya que su implementación a través de herramientas específicas permite reducir los desperdicios en los procesos, logrando una mayor eficiencia y, por consiguiente, una mejor satisfacción de las demandas de los clientes, independientemente del tipo de organización (Arias y Sandoval, 2022).

Actualmente, en la industria del software muchos de los aportes de las metodologías “lean” han sido recogidas en metodologías “ágiles” para lograr un rápido crecimiento y una alta adaptabilidad en el desarrollo de software, se buscan entornos de trabajo que reduzcan la

complejidad técnica e interpersonal de los proyectos, con el objetivo de crear procesos de trabajo más eficientes (Hibbs, et al., 2009).

Esta tendencia es propia de empresas del sector tecnológico y se ha incrementado exponencialmente desde el auge del internet, en donde "el futuro es impredecible, los consumidores disponen de una creciente gama de alternativas y el ritmo del cambio se acelera constantemente (Ries y Salbut, 2012)". Sin embargo, muchas empresas de software aún realizan procesos de desarrollo de software bajo metodologías obsoletas o realizadas de forma ineficiente, aun cuando el uso de metodologías ágiles podría ser más eficiente para el desarrollo de sus procesos. A su vez, más allá de procesos propios del desarrollo de software aún se gestionan usando presupuestos anuales, rígidos estudios de mercado, planes de negocio detallados o la planificación estratégica tradicional. Esto puede incidir en baja competitividad de la empresa en el mercado, generación de reprocesos e ineficiencia tanto en el desarrollo de software como en la propia gestión administrativa de la empresa.

Las metodologías ágiles surgen inspiradas del "Lean clásico" y de soportar un largo camino de creación de software bajo metodologías tradicionales y del malestar que estas generaban por su ineficiencia. Antes de su desarrollo, era muy usual que "los proyectos de desarrollo de software se planificaran durante años antes de escribir una sola línea de código, llevando muchas veces a catástrofes, en las que, tras invertir ingentes cantidades de dinero, era luego necesario rehacer todo el software desde cero (DTTL, 2022, párr.6)". Así, el movimiento ágil surge de la publicación del "Agile Manifiesto" por líderes de la industria en el 2001 a partir de reuniones entre varios destacados programadores y líderes de la industria de la época. Este manifiesto resume toda la filosofía de trabajo y presenta los cuatro valores y doce principios que rigen el desarrollo ágil de Software. Estos sin lugar a duda están influenciados por metodologías de gestión "Lean" de otros

sectores económicos y son el pilar actual de Frameworks de trabajo muy empleados como el "Scrum" o "Scrumban".

A su vez, se han desarrollado diversas metodologías derivadas que aportan desde la adopción de los principios "Lean". Por esta línea dentro de las metodologías más destacadas se encuentran las metodologías "ágiles" sus derivadas. En términos generales, son metodologías de gestión empleadas que buscan eliminar todo tipo de "despilfarros" en la creación de software y validar propiamente en el mercado las propuestas de valor generadas, para así validar y tener una retroalimentación rápida, confiable y económica de posibles mejoras de este (Ries y Salbut, 2012).

Como se observa, si bien las filosofías y metodologías mencionadas para el desarrollo de software son muy variadas y diversas (de acuerdo a la particularidad de cada necesidad en el desarrollo del software) parecen tener gran influencia del Lean tradicional en su componente histórico, practico y en la adopción y transformación de las herramientas empleadas. Así, se es pertinente analizar de qué manera se han ido calando y adoptando aquellas herramientas del "Lean" tradicional con el fin de entender y explicar su relación.

Actualmente, una de las principales deficiencias de la investigación y publicaciones sobre la aplicación de Lean es la falta de estudios que sirvan como guía para las organizaciones que deseen adoptar estas prácticas en la Ingeniería del Software. Existe una escasez de investigaciones que aborden el proceso completo, tanto en términos técnicos como de gestión. El enfoque predominante se centra en la reducción del desperdicio y la mejora de la calidad (Sanz, 2015).

De tal forma, se plantea abordar como tópicos en la revisión de literatura las herramientas empleadas en LSD y metodologías de desarrollo de software similares que puedan considerarse derivadas de la filosofía "Lean" clásica. Se explorará cuales herramientas fueron adoptadas, cómo se adoptaron, cuáles son las más empleadas y se podrá, al menos de forma preliminar, comprender

el impacto de la filosofía “Lean” clásica en las filosofías y metodologías actuales en el desarrollo de software.

Ya explorando la industria del software nacional, hay un panorama alentador en empresas dedicadas al desarrollo de software. Para 2020 “Industrias de software y servicios TI” aportaron al crecimiento de las exportaciones nacionales US$164,4 millones, y entre enero y octubre de 2021, este mismo sector reportó a ProColombia US$171,6 millones en ventas internacionales, lo que representó un crecimiento de 87,7% (La República, 2022).

Según un informe de Fedesoft (2022), dentro de las principales líneas de negocio se encontraban las empresas actividades de desarrollo de sistemas informáticos con un 39.1%; consultoría TI con un 29.6%; empresas de procesamiento y hosting de datos con un 5.2%, empresas de edición de software (2.7%) y empresas de diseño y desarrollo web con tan solo un 2.3%. Según dicho informe de Fedesoft, en el año 2020, el 1,9% de las empresas en la industria de software y tecnología de la información eran grandes empresas, las cuales generaron el 66,5% de las ventas totales de la industria. Por otro lado, el 98,1% restante de las empresas estuvo compuesto por pequeñas y medianas empresas (MiPymes) que generaron el 33,5% de las ventas totales en el mismo periodo de tiempo (ACIS, 2021).

Como principal problema en el sector, las empresas identifican una falta de talento humano capacitado para el desarrollo de software, especialmente en las MiPymes por sus componentes informales (El Colombiano, 2022). En una proyección para 2025, Colombia tendrá un déficit de cerca de 200.000 profesionales, un panorama complejo para esta industria en el país (La República, 2022). Esto reafirma la necesidad de explorar alternativas tanto de gestión como de desarrollo de software que sean viables y permitan mitigar esta problemática.

El presente trabajo de grado busca explorar las metodologías de desarrollo de Software actuales y su relación con la filosofía Lean clásica identificando aquellos elementos claves y fundamentales que pueden ser transversales en la generación eficiente de software, especialmente en la industria nacional. Asimismo, se busca comprender el impacto de la aplicación de estas herramientas como estrategia de mejoramiento en el desarrollo de software y permitirá generar posibles oportunidades de mejora en los procesos de gestión de las empresas de software nacionales y apoyar su crecimiento.

2. Objetivos

2.1. Objetivo General

Realizar un estudio sobre la implementación de técnicas de Lean Manufacturing en el desarrollo de Software, con el fin de analizar las diferencias con respecto al enfoque original.

2.2. Objetivos Específicos

Realizar una revisión documental acerca de los estudios desarrollados de “Lean Software Development” con el fin de determinar qué herramientas clásicas del Lean han sido aplicadas en los desarrollos de software.

Identificar qué herramientas “Lean” son empleadas con mayor frecuencia dentro de la cadena de valor en el desarrollo de software.

Analizar la forma en que fueron adaptadas las herramientas clásicas de “Lean Manufacturing” en los desarrollos de software y su aplicación en diferentes tipos de servicios ofrecidos en el mercado.

Realizar un diagnóstico que permita comprender el impacto de la aplicación de las herramientas Lean como estrategia de mejoramiento en el desarrollo de software.

3. Marco de Referencia

El marco de referencia está compuesto por el marco de antecedentes y el marco teórico. En el marco de antecedentes se proporcionan conocimientos de trabajos previos que abordan el tema de investigación de antecedentes, mientras que, en el marco teórico, se realiza una inmersión en la temática de investigación. A continuación, se presentan cada uno de ellos.

3.1. Marco de Antecedentes

El Dr. Robert Charette mencionó por primera vez el "Lean Software Development" (LSD) como una forma de construir organizaciones tolerantes al cambio que se estaban volviendo cada vez más dependientes del software. Posteriormente, se encuentra el trabajo de los hermanos Tom y Mary Poppendieck, (1993) a los que se podría atribuir el término y sobre este desarrollaron sus planteamientos teóricos. La publicación en 1993 de "Lean Software Development: An Agile Toolkit" y su avance en publicaciones posteriores, son los referentes teóricos principales sobre el tema. En este, los hermanos Poppendieck exponen cómo se complementan conceptos de la filosofía Lean con las metodologías ágiles como XP o Scrum. Sin embargo, no es hasta la publicación de "Manifiesto Ágil" que estos se tratan a profundidad. Este manifiesto es el común denominador en la literatura existente y permite asociar claramente la raíz en principios del Lean Clásico. Las aproximaciones ágiles a la gestión de proyectos parecen tener raíces en el pensamiento y prácticas de la manufactura Lean (Moreno, 2010). Sin embargo, donde más cabida han tenido y en donde radica principalmente su diferenciación es su aplicabilidad, siendo el Lean clásico aplicado a la gestión de proyectos, y las metodologías ágiles enfocadas únicamente al proceso de desarrollo de software.

Otra referencia importante debido a su gran difusión y éxito comercial es "El método Lean Start Up" de Eric Ries (2012). Es considerado una lectura obligada sí que quiere hablar de procesos

esbeltos en la industria de software. En este libro narrativo el autor describe su propuesta/estrategia lean startup enfocada en empresas emergentes. Él desarrolla la idea del método "Lean Startup" a partir de sus propias experiencias como asesor de startups, empleado y fundador atribuyendo el fracaso pasado de sus startups a no comprender los deseos de sus clientes objetivo y a dedicar demasiado tiempo y energía a actividades que no aportaban valor y al lanzamiento inicial del producto (Tegegne et al, 2019). El método es totalmente inspirado en el "Lean" tradicional para lograrlo con componentes de iteración constantes.

Ya hablando propiamente de trabajos académicos que aborden la temática expuesta, se encuentran varios con una alta aportación. Un trabajo relevante es el desarrollado por Sanz (2015) titulado "Metodología Lean para el desarrollo de software. Ejemplo práctico de aplicación en empresa de desarrollo de software" en el que desarrolla una revisión histórica y conceptual de la adopción del "Lean" por la industria del software y cómo se dio la transformación de metodologías herederas de esta filosofía dentro de este sector. A su vez, presenta un caso práctico desarrollado para un banco de España con importante presencia internacional, en el cual, se busca explorar cómo la gestión del desarrollo de proyectos de software puede ser transformada en un entorno real al adoptar una metodología ágil como Lean.

Moreno (2010), hace una completa de revisión en su trabajo Filosofía Lean aplicada a la Ingeniería del Software en las que presenta un barrido de las herramientas del Lean clásicas, haciendo un paralelo entre ejemplos de la aplicación de estas herramientas en la industria de la manufactura y ejemplos basados en las variaciones reales de la industria del software. Sin embargo, no revisa ni busca comprender el impacto de estas. A su vez, se hace necesario explorar trabajos completos más recientes, entendiendo la dinámica altamente cambiante de la industria del software.

Por otra parte, cabe destacar algunos trabajos recientes realizados en la Universidad de Santander para intentar comprender la adopción de las herramientas clásicas del Lean Manufacturing en empresas ajenas a la industria manufacturera. Uno de estos es el trabajo desarrollado por Arias y Sandoval (2022) titulado "Estudio exploratorio sobre la implementación de técnicas de Lean Manufacturing en el sector de la salud (Lean Healthcare)." En este exploran la aplicación, adopción e impactos que ha tenido el Lean clásico en el sector salud. Allí, evidencian que el uso de esta filosofía en dicho sector se apoya en el uso de técnicas clásicas con cierta modificación que permite gestionar adecuadamente el servicio brindado al establecer prioridades, tales como la seguridad y la calidad bajo la perspectiva individual de cada paciente, así como mantener los costos en un nivel adecuado, repercutiendo en el continuo mejoramiento del sector.

Cómo este, se encuentran otros trabajos análogos para otros sectores que describen diversos sectores de la economía. El trabajo de Álvarez y Suárez (2022) que aborda la adopción de las estrategias Lean con el sector logístico (Lean Logistics), el de Cardona y Prada (2022) en el sector agropecuario (Lean Farming), y el de Morantes y Delgado en el sector de la construcción (Lean construction). De tal forma, el presente proyecto se encuentra dentro del marco de esta revisión exploratoria al abordar un sector con características tan únicas y propias como lo es la industria del software.

El aporte de cada uno de los trabajos mencionados en la presente investigación es significativo, en términos de que permite un primer acercamiento con respecto a las herramientas más utilizadas en cada uno de los sectores mencionados, en este caso en un sector tan pujante como el de la industria del software.

3.2. Marco Teórico

3.2.1. Lean Manufacturing

Con el transcurso del tiempo, el sistema de manufactura esbelta ha demostrado ser altamente eficiente en la reducción de diferentes tipos de desperdicios dentro de una organización y en la mejora de la eficiencia operativa (Gómez, 2014). La metodología Lean es reconocida por ser un enfoque de trabajo enfocado en la búsqueda constante de la excelencia a través de la mejora continua. Su objetivo principal es eliminar las actividades y procesos que no agregan valor en la fabricación, distribución y comercialización de productos y/o servicios, lo que resulta en la reducción de costos, la mejora de los procesos, la eliminación de desperdicios, el aumento de la satisfacción del cliente y la mantención del margen de utilidad (Alahyari et al, 2017).

En la búsqueda de la eliminación del desperdicio, se busca aumentar la productividad, que se define como la habilidad de la empresa para utilizar eficientemente los recursos humanos, naturales, financieros, científicos y tecnológicos disponibles en la generación de productos y servicios. De esta manera, se busca optimizar la producción y proporcionar una mejor satisfacción al cliente, reduciendo los costos y manteniendo un margen de utilidad adecuado (Hibbs, et al., 2009).

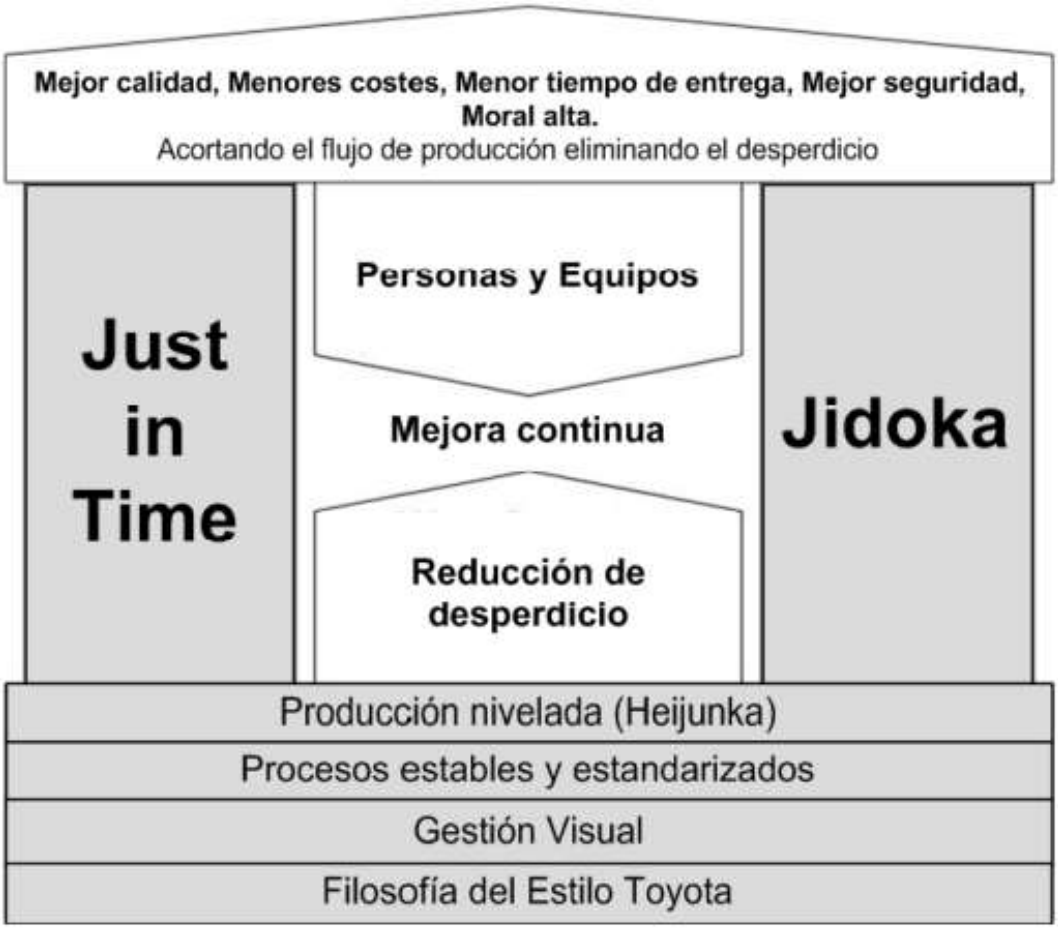
La metodología "Lean" va más allá de simplemente utilizar un conjunto de herramientas y prácticas, ya que está fundamentada en una serie de principios que deben ser adoptados por la cultura organizacional antes de cualquier otra cosa. Su implementación no se limita a la aplicación de herramientas, sino que implica una transformación completa de la organización, iniciando por la adopción de sus principios. Una representación gráfica que incluye los elementos distintivos del sistema Lean es la denominada como "La Casa", la cual integra los principios, la mentalidad, las

herramientas y los resultados esperados en un solo diagrama. La siguiente figura es un extracto de la casa de Lean expuesta por Liker (2021).

.

Figura 1.

La Casa "Lean Manufacturing".



Nota. Tomado de The Toyota way: 14 management principles from the world's greatest manufacturer, por Liker, 2021, Versión Simplificada.

Cada uno de los elementos que se representan en "La Casa Lean" tiene importancia individual, pero su verdadero valor radica en cómo se complementan entre sí. El techo simboliza los objetivos que se buscan alcanzar, tales como mejorar la calidad y reducir los costos. Los pilares

que sostienen el techo son el Just-in-Time (producir solo lo necesario en el momento necesario) y el Jidoka (producir con calidad inherente al proceso).

En el centro de la casa se encuentra la mejora continua, que implica la reducción del desperdicio a través de la participación de los equipos y las personas. La base sobre la que se sostiene todo el sistema es la filosofía de Lean, que incluye conceptos como la gestión visual, la estandarización y el nivelado de la producción (Liker, 2021).

3.2.1.1. Filosofía. Liker (2004) identifica 14 principios fundamentales que permiten resumir lo actualmente denominado “Filosofía Lean”:

1. Basar las decisiones de administración en una filosofía de largo plazo, aún a costo de las metas financieras de corto plazo (Procesos).
2. Crear flujos de procesos continuos para llevar los problemas a la superficie.
3. Usar sistemas "Pull" para evitar la sobreproducción.
4. Nivelar la carga de trabajo.
5. Construir una cultura orientada a la solución de problemas, para obtener calidad a la primera vez.
6. La estandarización de tareas y procesos es la base de la mejora continua y la toma de poder por los empleados.
7. El control visual impide que se oculten los problemas.
8. Utilizar tecnología fiable y testeada que sea de utilidad para las personas y los procesos. Personas y Colaboradores.
9. Desarrollar líderes que entiendan su trabajo en Toyota, vivan su filosofía y se la enseñen al resto.
10. Desarrollar personas y equipos excepcionales que sigan la filosofía de la compañía.

11. Se debe respetar la red de colaboradores y proveedores, dándoles nuevos retos y ayudándolos a mejorar (Resolución de Problemas).

12. Para comprender una situación se debe verificar en primera persona.

13. Tomar decisiones lentas por consenso, considerar profundamente todas las opciones e implementar las decisiones rápidamente.

14. Por medio de la reflexión implacable y la mejora continua (Kaizen) la empresa debe asumir un rol de aprendizaje sistemático.

3.2.1.2. Gestión Visual. La gestión visual es una herramienta complementaria a la labor de las personas, ya que aprovecha la naturaleza sensible de los seres humanos para transmitir información de manera efectiva. Los indicadores visuales son los más eficaces para mostrar el estado de cumplimiento de las tareas y procesos, ya que se valen de los sentidos humanos, como el oído y la vista, para presentar información de manera clara y comprensible. En la actualidad, la gestión visual se está aplicando cada vez más en el ámbito del desarrollo de software. Dado que, consiste en presentar información relevante de manera gráfica y sintética para facilitar el trabajo de los miembros del equipo y reducir tiempos. Al utilizar controles visuales eficaces, se mejora la productividad, se reducen los errores, se fomenta la comunicación y se aumenta el control del entorno laboral. En este aspecto, la filosofía "Lean" enfatiza la importancia de los controles visuales para prevenir que los problemas pasen desapercibidos (Dingsoyr y Moe, 2014).

3.2.1.3. Procesos Estables y Estandarizados. La organización busca estandarizar los procesos de trabajo de sus empleados mediante la formación y supervisión de los líderes de cada área, quienes deben conocer los estándares aplicables y asegurarse de que se siguen correctamente. Esto ayuda a evitar errores en la cadena de producción y a mejorar la eficiencia en general. La estandarización de trabajos se realiza en base a tres conceptos claves: el Tack Time, que determina el ritmo de

entrega de los productos según la demanda del cliente; la secuencia de tareas, que indica las acciones que deben realizarse dentro de un tiempo de ciclo determinado; y el inventario estándar, que incluye las unidades de maquinarias necesarias para evitar problemas de parada en la producción (Sanz, 2015).

La estandarización se basa en estos tres conceptos para optimizar los recursos disponibles y crear la mejor metodología posible. En donde, se recopilan y registran los datos necesarios para que los ingenieros y jefes de equipo diseñen el proceso ideal, mientras que los operarios colaboran proponiendo mejoras para su puesto de trabajo, que son incluidas en la metodología estándar (Sanz, 2015).

Por otro lado, el reto en la implementación de la estandarización es encontrar el equilibrio entre la rigidez de los procedimientos y la participación de los trabajadores en la mejora del proceso de producción. Por lo cual, es fundamental que los trabajadores aporten mejoras a la propuesta estándar para que se sientan motivados en este proceso (Sanz, 2015).

3.2.1.4. Heijunka. Conocida en español como Producción Nivelada, según Hernández (2011), se refiere a la técnica de nivelar la producción, tanto en términos de volumen como de mezcla de productos. En lugar de fabricar según el flujo de pedidos de los clientes, que puede ser variable, se considera el volumen total de pedidos en un período determinado y se equilibra la producción de manera que se fabrique la misma cantidad de cada tipo de producto todos los días para cubrir la demanda global. Con esta técnica se pueden obtener cuatro beneficios fundamentales:

1. Flexibilidad para servir lo que el cliente quiere y cuando lo quiere.

2. Se reduce el inventario reduce el riesgo de producir productos y que éstos no se vendan se produce bajo pedido.
3. Equilibrio en el uso de maquinaria y trabajo de las personas si el trabajo se estandariza teniendo en cuenta que uno de los productos requiere mayor esfuerzo que otros la secuencia de actividades puede contemplar que no se encadenen grandes esfuerzos haciendo el trabajo más llevadero.
4. Se equilibra la demanda de proveedores: esta estabilidad de la producción se tramita a lo largo de la cadena de suministros cuando se utiliza el sistema “Just in time” de forma que los pedidos a proveedores serán siempre similares lo que facilitará que se aprovechen produciendo sus propios inventarios conocida la demanda habitual.

La metodología ágil incorpora el concepto de "Heijunka", que se enfoca en la versatilidad de las personas para desempeñar diversas tareas. Cuanto mayor sea la capacidad de los miembros del equipo para realizar diferentes tareas, más fácil será nivelar la producción de software (Sanz, 2015).

3.2.1.5. Just in Time. La metodología industrial conocida como "Just in Time" o "Justo a Tiempo" busca producir solo los productos necesarios en el momento exacto y en las cantidades requeridas. Su finalidad es reducir la ineficiencia y el tiempo muerto en los procesos de producción, con el propósito de mejorar constantemente dichos procesos y la calidad del producto o servicio correspondiente (Hernández, 2011). En la metodología Lean clásica, un sistema Justo a Tiempo implica el uso de una estrategia de flujo de línea para alcanzar una producción de gran volumen a un bajo costo. El objetivo es mantener un proceso de producción continuo, sin interrupciones, con el fin de minimizar el tiempo total que transcurre desde el inicio de la fabricación hasta la entrega del producto final y su facturación (Hibbs, 2009).

Los 7 pilares de justo a tiempo son los siguientes:

1. Igualar la oferta y la demanda.
2. El peor enemigo: el desperdicio.
3. El proceso debe ser continuo no por lotes.
4. Mejora Continua.
5. Es primero el ser humano.
6. La sobreproducción = ineficiencia.
7. No vender el futuro.

De igual forma, hay algunas técnicas o mecanismos asociados y/o derivados al “Just in time”. Se exponen a continuación.

3.2.1.5.1. Takt Time Planning. El concepto de Takt Time se refiere al “ritmo” o “paso” al que se debe producir un producto para satisfacer la demanda del cliente en un determinado período de tiempo. Se calcula dividiendo el tiempo disponible para la producción entre la demanda del cliente en el mismo período (Hernández & Vizán, 2013). Mantener el ritmo de producción alineado con el Takt Time es crucial para evitar el desperdicio en la producción. Si la velocidad de producción es mayor, se corre el riesgo de generar un exceso de inventario, mientras que, si es menor, se requerirán horas extras o un inventario adicional para satisfacer la demanda del cliente.

3.2.1.5.2. Sistema Pull. En este sistema tipo pull “halar”, la demanda del producto determina la cantidad a producir, se generan órdenes de producción pequeñas, lo que resulta en bajos costos de inventario y un menor riesgo de obsolescencia del producto. Este enfoque es beneficioso en entornos competitivos que requieren innovación y flexibilidad, y su

implementación requiere de una rápida obtención de información desde los puntos de venta y un sistema de producción ágil. Sin embargo, las desventajas incluyen la necesidad de capacidad para períodos de alta demanda, menores economías de escala y transporte en comparación con el enfoque push tradicional (Muñoz, 2009).

3.2.1.5.3. Quick Changeover – SMED. La expresión "Changeover" se refiere a las actividades que se realizan desde el momento en que se interrumpe la operación de una máquina para cambiar de producción hasta que se produce la primera unidad del siguiente lote en las condiciones establecidas de calidad y tiempo (Sanz, 2015). En cuanto al término "Quick Changeover" o SMED (Single Minute Exchange of Die), se refiere a una serie de técnicas utilizadas para disminuir el tiempo necesario para realizar cambios en una máquina. El objetivo de SMED es reducir los tiempos de preparación de la máquina, lo que permite producir lotes más pequeños y alcanzar la producción Justo a Tiempo. El sistema SMED se originó como una necesidad y se desarrolló para reducir el tiempo requerido para la preparación de las máquinas (Hernández, 2011).

3.2.1.5.4. Kanban. El término "kanban" en japonés se refiere a una etiqueta o tarjeta de instrucción utilizada en la metodología Lean Manufacturing. El kanban es un dispositivo de dirección automático que proporciona información sobre qué productos se deben producir, en qué cantidad, qué medios utilizar para la producción y cómo transportarlos (Hernández, 2011). Además, la metodología Kanban se basa en un sistema visual que controla la producción y limita la cantidad de trabajo en curso (Work In Progress - WIP). Es similar a una orden de trabajo, proporcionando información sobre qué producir y en qué cantidad. A través de señales visuales, la demanda se transmite a lo largo de la cadena de valor, permitiendo que cada proceso conozca la cantidad exacta que debe producir en un momento determinado, lo que reduce los inventarios

innecesarios. De esta manera, el trabajo fluye en función de la demanda, guiado por la sucesión de señales proporcionadas por el sistema Kanban (Sanz, 2015).

3.2.1.6. Verificación del Proceso (Jidoka). En el proceso de producción, la palabra "Jidoka" se refiere a la verificación de calidad integrada al proceso. Además, es el segundo pilar de la filosofía "Lean" (Hernández, 2011). La filosofía Jidoka persigue la determinación de parámetros de calidad óptimos en los procesos de producción y se enfoca en integrar la verificación de calidad en el proceso. A través del sistema Jidoka, se comparan los parámetros del proceso con los estándares preestablecidos y, si no coinciden, el proceso se pone en pausa con una alerta de irregularidad, entonces se debe solucionar el problema para evitar la producción de productos defectuosos. Estos procesos son comparativos entre los resultados actuales en producción y lo "ideal" o "estándar". Por otro lado, la implantación de diferentes tipos de sistemas Jidoka depende del tipo de producto y la información "ideal" o "estándar" debe ser el punto óptimo de calidad del producto. Igualmente, Jidoka también puede referirse a equipos que se detienen automáticamente en condiciones anormales o cuando un colaborador encuentra un problema en su estación de trabajo (Caudillo, 2013).

3.2.1.7. Value Stream Mapping (VSM). Michael Porter (2002), propuso el concepto de "cadena de valor" para identificar formas de generar más beneficio para el consumidor y con ello obtener ventaja competitiva. El concepto radica en hacer el mayor esfuerzo en lograr la fluidez de los procesos centrales de la empresa, lo cual implica una interrelación funcional que se basa en la cooperación. El VSM es una herramienta técnica que examina el sistema físico, procesos e interconexiones.

El propósito del mapeo de la cadena de valor es para hacer resaltar la causa del desperdicio y eliminarlos para la implementación de un estado futuro de la cadena de valor que puede convertirse en realidad en un periodo corto de tiempo.

El mapeo de la cadena de valor puede ser una herramienta de la planificación, comunicación y una herramienta para manejar su proceso del cambio. El primer paso es dibujar el estado actual, es recopilar la información sobre el área del piso de ver cómo está trabajando y no como quisiéramos que trabajara. Esta la da la información que necesita para realizar el estado futuro.

El mapa de la cadena de valor representa en un diagrama el conjunto de actividades y procesos y el flujo de materiales e información que rodean el proceso transformación de un producto o prestación de un servicio desde que se recibe la petición de este hasta que se realiza su entrega, pudiendo incluir actividades de clientes y proveedores que intervengan en el proceso de producción (Sanz, 2015).

Típicamente, para realizar mapeo del flujo de valor, se siguen una serie de pasos:

1. Seleccionar una familia de productos, entendida como un conjunto de productos que se producen de forma similar, tanto por los medios utilizados como el propio proceso.
2. Formar el equipo que participará en el análisis.
3. Representar los procesos de producción que se siguen para producir el producto, identificando una serie de valores clave para cada uno de ellos: tiempo de ciclo, número de operarios involucrados, etc.
4. Representar el flujo de material, cómo se mueve el material de un proceso a otro, identificando, si existen, los inventarios que se utilizan y su volumen, así como el flujo de materia prima que llega desde los proveedores y de la entrega del producto al cliente.

5. Representar el flujo de información entre los distintos actores involucrados, empresa (u otras unidades organizativas dentro de la misma si es necesario distinguirlas), proveedores, clientes, etc.
6. Calcular los Lead Time, del producto y del proceso. El mapa realizado permitirá visualizar la situación global del sistema de producción y ayudará a reconocer focos de desperdicio (sobreproducción, tiempos de espera, inventarios, etc.).

3.2.1.8. Despilfarro o Desperdicio. Lean Manufacturing es una filosofía de gestión enfocada a la reducción de los 7 tipos de "desperdicios" (sobreproducción, tiempo de espera, transporte, exceso de procesado, inventario, movimiento y defectos) en productos manufacturados. Eliminando el despilfarro, la calidad mejora, y el tiempo de producción y el costo se reducen. Los despilfarros (o mudas, por su nombre en japonés) a reducir son los siguientes (Sanz, 2015):

1. Defectos y retrabajos: Este es el mayor tipo de derroche, que es la cantidad de trabajo que necesita volverse a hacer, con la consecuente reutilización de recursos para llevarlo a cabo (otra vez). La necesidad de reacondicionar partes en proceso o productos terminados, como así también reciclar o destruir productos que no reúnen las condiciones óptimas de calidad provocan importantes pérdidas.
2. Procesamiento incorrecto: Este tipo de producto no mejora el producto y se trata de pasos innecesarios o procedimientos/elementos de trabajo (trabajo que no agrega valor al producto). Desperdicios generados por fallas en materia de layout, disposición física de la planta y sus maquinarias, errores en los procedimientos de producción, incluyéndose también las fallas en materia de diseño de productos y servicios.
3. Sobreproducción: Este tipo de derroche origina material procesado o producto final que no es requerido.

4. Inventario: Se refiere al material que se acumula en el lugar de trabajo, entre procesos, o como producto final que podría ser entregado al cliente. Tiene muchos motivos, y en él se computan tanto los inventarios de insumos, como de repuestos, productos en proceso e inventario de productos terminados.
5. Movimiento: Movimientos sin valor agregado de gente, materiales, piezas o maquinaria. Se hace referencia con ello a todos los desperdicios y despilfarros motivados en los movimientos físicos que el personal realiza en exceso debido entre otros motivos a una falta de planificación en materia ergonómica.
6. Espera: Tener que esperar a que otro proceso termine antes de empezar el trabajo. Motivado fundamentalmente por: los tiempos de preparación, los tiempos en que una pieza debe esperar a otra para continuar su procesamiento, el tiempo de cola para su procesamiento, pérdida de tiempo por labores de reparaciones o mantenimientos, tiempos de espera de órdenes, tiempos de espera de materias primas o insumos. Los mismos se dan también en las labores administrativas. Todos estos tiempos ocasionan menores niveles de productividad.
7. Transportación: Despilfarro vinculado a los excesos en el transporte interno, directamente relacionados con los errores en la ubicación de máquinas, materiales, herramientas, y las relaciones sistémicas entre los diversos sectores productivos. Ello ocasiona gastos por exceso de manipulación, lo cual lleva a una sobreutilización de mano de obra, transportes y energía, como así también de espacios para los traslados internos.

3.2.1.9. Poka Yoke. El concepto "Poka Yoke" proviene de las palabras japonesas "poka" (error involuntario) y "yoke" (prevención). Un dispositivo Poka Yoke es un mecanismo diseñado para evitar errores antes de que ocurran o hacerlos tan evidentes que el trabajador los note y los

corrija oportunamente (Hernández, 2011). El objetivo principal del Poka Yoke es evitar los defectos en un producto, ya sea mediante la prevención o corrección de errores de manera temprana. Un sistema de Poka Yoke tiene dos funciones clave (Vinod et al., 2017):

- Hacer la inspección del 100% de las partes producidas.
- Si se presentan situaciones anormales, el sistema puede proporcionar retroalimentación y tomar medidas correctivas.

La eficiencia del método Poka Yoke en la reducción de defectos dependerá del tipo de inspección que se realice, ya sea al inicio de la línea, autoinspección o inspección continua **(**Vinod et al.**,** 2017).

3.2.1.10. Mejora Continua (KAIZEN). Kaizen es un término que significa "cambio" y "mejora", y juntos se refieren a un "cambio para mejorar" tanto en los costos como en la cultura organizacional. Este enfoque se compone de tres elementos: la percepción, que implica descubrir los problemas; el desarrollo de ideas para encontrar soluciones creativas; y la toma de decisiones para elegir la mejor propuesta, planificar su realización y comenzar a implementarla.

El término "Kaizen" fue acuñado por Massaki Imai en 1986 y desde entonces su uso se ha estandarizado y popularizado, al menos geográficamente, aunque no todas las empresas lo han adoptado en la misma medida. Este término está relacionado con la mejora continua, la eliminación de desperdicios, la estandarización de procesos y otros pilares del Lean Manufacturing. Así pues, los programas de mejora continua tienen como objetivos principales:

-Aumentar el nivel de calidad.

-Mejorar la satisfacción del cliente (con disminución de las No Conformidades de clientes).

-Optimización de la gestión de la empresa.

-Incrementar en el rendimiento de equipos humanos.

3.2.2. Desarrollo de Software

El desarrollo de software se define como un conjunto de actividades de informática que se enfocan en la creación, diseño, implementación y compatibilidad de programas informáticos. En su forma más genérica, un software es un conjunto de instrucciones o programas entrelazados que indican a un ordenador que realizar. Se pueden diferenciar cuatro tipos básicos (IBM, s.f.):

1. Sistema, programación, aplicación e integrado. El software del sistema incluye funciones básicas como sistemas operativos, administración de discos, servicios y administración de hardware.
2. El software de programación que proporciona herramientas como editores de texto, compiladores, enlazadores y depuradores para los programadores.
3. Las aplicaciones o "Apps" son un tipo de software de aplicación que ayuda a los usuarios a realizar tareas, y se pueden encontrar en la web o en dispositivos móviles.
4. El software integrado que se utiliza para controlar dispositivos que no se consideran computadoras, como redes de telecomunicaciones, automóviles y robots industriales, y se conectan como parte del Internet de las Cosas (IoT por sus siglas en inglés).

3.2.3. Metodologías Tradicionales para el Desarrollo de Software

Una metodología de desarrollo de software se refiere al conjunto de técnicas y prácticas empleadas para crear una solución de software informático. En la actualidad, hay una diversa gama de metodologías y es decisión de cada equipo de desarrollo de software adoptar la que se considere más pertinente (Universitat Carlemany, 2022). Su uso es imprescindible para la organización del trabajo y para generar flujos en el desarrollo de software, minimizando márgenes de error y anticipándose a los mismos.

La selección de una metodología para el desarrollo de software depende de varios factores, incluyendo la prioridad de controlar riesgos y otros aspectos relevantes para el proyecto. Además, la utilización de una metodología adecuada permite optimizar los recursos disponibles y ahorrar tiempo, tanto a corto como a largo plazo. A continuación, se presentan las metodologías de desarrollo de software tradicionales existentes más relevantes:

3.2.3.1. Waterfall (Cascada). En esta metodología la organización del trabajo se realiza de manera secuencial en fases verticales, donde cada actividad se realiza de forma ordenada y no se puede avanzar a la siguiente hasta haber completado la anterior. Además, permite trabajar con mayor seguridad y eficiencia, lo que se traduce en un ahorro de tiempo. Esta metodología fue muy utilizada en los primeros años del desarrollo de software (Universitat Carlemany, 2022).

Sus fases secuenciales están claramente definidas, incluyendo análisis de requisitos, diseño de sistema, diseño de programas, modificación, diseño de pruebas y fase de codificación y mantenimiento. En este caso, es importante controlar cada fase para garantizar un progreso efectivo. Además, se debe tener en cuenta que esta metodología no es adecuada para realizar cambios sobre la marcha, por lo que es una buena opción cuando los objetivos están bien definidos desde el principio, pero en entornos cambiantes como en el de algunos del sector del software hay otras metodologías más pertinentes (Universitat Carlemany, 2022).

3.2.3.2. Prototipo. Esta metodología parte de la generación de un “software rápido” para ser testeado. Esto, con el fin de buscar una contribución retroalimentada por parte de cada uno de los usuarios, iterando, para llegar a una versión enfocada en el usuario de forma rápida y económica validada ya propiamente. La retroalimentación incluye fallos técnicos, mejoras que se pueden introducir para enriquecer el software o componentes que los usuarios no consideren funcionales. Es un modelo completamente iterativo que permite mejorar el producto final y perfeccionarlo.

Dentro de este modelo usualmente se encasillas aquellas metodologías derivadas de la filosofía ágil (Universitat Carlemany, 2022).

3.2.3.3. Incremental. El modelo incremental es una variante del modelo Waterfall que se diferencia en que cada fase incorpora una nueva funcionalidad al software. Esta metodología permite verificar y probar de forma sencilla las mejoras que se van implementando durante el desarrollo, incluso antes de que la herramienta esté completamente terminada. Por lo cual, se ha migrado notoriamente del modelo Waterfall a el modelo incremental. Sin embargo, cabe destacar que, al igual que otras metodologías tradicionales de desarrollo de software, su debilidad es el proceso lento y difícilmente adaptable ante entornos cambiantes y con incertidumbre (Universitat Carlemany, 2022).

3.2.3.4. Espiral. Esta metodología consta de cuatro fases y tiene como objetivo principal satisfacer las necesidades del cliente. La espiral se va acercando al cliente en cada fase, siendo la primera la planificación del proyecto, seguida del análisis de riesgos, el desarrollo de un prototipo y, finalmente, la evaluación por parte del cliente. Al combinar elementos de otros modelos, el resultado final es de alta calidad y la validación del cliente es fundamental para el éxito del proyecto (Universitat Carlemany, 2022).

3.2.3.5. Diseño Rápido de Aplicaciones (RAD). Es una técnica concebida para desarrollar un buen software en poco tiempo. La clave está en tener en cuenta diversos factores. Se basa en determinados ejes: la construcción de un prototipo para recibir retroalimentación de los usuarios; y la velocidad, lo que expone la herramienta a tener más errores de partida, pero realizables de forma rápida y económica (Universitat Carlemany, 2022).

3.2.4. Lean Software Development (LSD)

De acuerdo con lo mencionado por Milić et al. (2017), los principios del LSD se componen de los siguientes puntos:

1. Eliminación del desperdicio: se busca eliminar todo aquello que no aporte valor al cliente durante el proceso de desarrollo del software.
2. Calidad integrada: se enfoca en la calidad desde el inicio del proceso y se implementan acciones correctivas preventivas para evitar errores.
3. Creación de conocimiento: el proceso de desarrollo de software es una creación constante de conocimiento, y se debe centrar en mejorarlo y profundizarlo.
4. Aplazar las decisiones: se recomienda retrasar la toma de decisiones debido a la incertidumbre que rodea los requisitos.
5. Agilidad en la construcción y entrega: se busca entregar el software tan rápido como sea posible sin sacrificar la calidad.
6. Respeto a las personas: se debe reconocer y respetar a todas las personas involucradas en el proceso de desarrollo.
7. Optimización del conjunto: se evita la mejora local en favor de una visión global del proceso de desarrollo del software.

3.2.5. Filosofía Ágil

La introducción inicial a los principios de la filosofía Lean en el ámbito del software se puede atribuir al manifiesto ágil, aunque no se menciona explícitamente. En el mes de febrero del año 2001, un grupo de 17 expertos de la industria del software se reunieron y crearon el término "Ágil" para referirse a los nuevos enfoques metodológicos. Durante esta reunión, se fundó "The

Agile Alliance", una organización dedicada a promover esta nueva forma de trabajo, y se redactó lo que se conoce como el "Manifiesto ágil", el cual estableció los valores y principios de la metodología ágil.

Así pues, los valores de la filosofía ágil son los siguientes:

- Las relaciones interpersonales y la colaboración tienen prioridad sobre los procesos y herramientas.
- El software en funcionamiento es más importante que la documentación excesiva.
- La colaboración con el cliente es preferible a la negociación contractual.
- La capacidad de respuesta al cambio tiene más valor que seguir un plan rígido.

Por otro lado, los principios fundamentales de la filosofía ágil son:

1. La satisfacción del cliente es la máxima prioridad, lograda mediante entregas tempranas y continuas de software con valor.
2. Se acepta que los requisitos pueden cambiar incluso en etapas avanzadas del desarrollo, y los procesos ágiles aprovechan estos cambios para proporcionar ventaja competitiva al cliente.
3. El software funcional se entrega frecuentemente, preferiblemente en períodos cortos.
4. Los responsables de negocio y los desarrolladores trabajan juntos en colaboración constante durante todo el proyecto.
5. Los proyectos se llevan a cabo con individuos motivados, a quienes se les da el entorno y el apoyo que necesitan para lograr el éxito.
6. La comunicación cara a cara es el método más efectivo para transmitir información dentro del equipo de desarrollo.
7. La medida principal de progreso es el software en funcionamiento.

8. Los procesos ágiles promueven el desarrollo sostenible, donde todos los involucrados son capaces de mantener un ritmo constante de trabajo.
9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad del equipo.
10. La simplicidad es esencial, ya que se busca maximizar la cantidad de trabajo no realizado.
11. Los equipos autoorganizados son capaces de producir las mejores arquitecturas, requisitos y diseños.
12. El equipo reflexiona regularmente sobre su propio desempeño para ajustar y mejorar su comportamiento.

A continuación, se presenta el manifiesto original:

"Estamos descubriendo formas mejores de desarrollar
software tanto por nuestra propia experiencia como
ayudando a terceros. A través de este trabajo hemos
aprendido a valorar:

Individuos e interacciones sobre procesos y herramientas
Software funcionando sobre documentación extensiva
Colaboración con el cliente sobre negociación contractual
Respuesta ante el cambio sobre seguir un plan

Esto es, aunque valoramos los elementos de la derecha,
valoramos más los de la izquierda (Los 12 principios del manifiesto ágil, 2003)".

3.2.6. Frameworks de Metodología Ágil

Hay amplia gama de frameworks o espacios de trabajo de tipo "ágil" para las empresas de desarrollo de software, cada uno con sus propias fortalezas y debilidades y actualmente, en constante cambio e innovación (Mendes et al., 2009).

Su elección depende de cada proyecto y de las particularidades de cada empresa dado que se pueden requerir procesos diferentes. Hoy en día, el frameworks de tipo Scrum se considera como uno de los más valorados, por lo que se han desarrollado programas de software específicos para Scrum que pueden facilitar el proceso de desarrollo de cualquier proyecto (Mendes et al., 2009). Algunos de los Frameworks ágiles más importantes son los siguientes:

3.2.6.1. Scrum. Indicado para proyectos con alto ratio de cambio de requerimientos, su principal característica es la definición de "sprints", siendo estos cada una de las iteraciones del proceso con una duración máxima de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. Otra característica importante del Scrum son las reuniones diarias que se llevan a cabo a lo largo del proyecto. Dichas reuniones no requieren más de 15 minutos del equipo de desarrollo y su objetivo son la coordinación e integración del producto a entregar (Mendes et al., 2009).

3.2.6.2. Cristal Methodologies. Las metodologías Cristal son un conjunto de enfoques para el desarrollo de software que se caracterizan por su enfoque en las personas que conforman el equipo de trabajo y en minimizar la cantidad de documentos producidos. Estas metodologías buscan mejorar las habilidades del equipo y establecer políticas de trabajo en equipo. Dependiendo del tamaño del equipo, se utilizan diferentes versiones, por ejemplo, Crystal Clear se enfoca en

equipos de 3 a 8 integrantes, mientras que Crystal Orange se aplica a equipos de 25 a 50 integrantes (Mendes et al., 2009).

3.2.6.3. Dynamic Systems Development Method (DSDM). La metodología DSDM sigue un enfoque iterativo e incremental en su proceso de desarrollo. Su ciclo de vida consta de cinco fases: Estudio de Viabilidad, Estudio del Negocio, Modelado Funcional, Diseño y Construcción, e Implementación. Aunque la iteración se produce en las últimas tres fases, la retroalimentación está prevista en todas las etapas del proceso (Mendes et al., 2009).

3.2.6.4. Adaptive Software Development (ASD). La metodología ASD es un proceso iterativo que se enfoca en la tolerancia a cambios y la creación de componentes de software. Se divide en tres etapas para su ciclo de vida: la primera etapa es la Especulación, en la que se inicia el proyecto y se planifican las características del software; la segunda etapa es la Colaboración, en la que se desarrolla el producto; y la tercera etapa es la de Aprendizaje, en la que se revisa la calidad del producto y se entrega al cliente, con el objetivo de aprender de los errores cometidos y volver a iniciar el ciclo de desarrollo (Mendes et al., 2009).

3.2.6.5. Feature-Driven Development (FDD). El Feature-Driven Development (FDD) es una metodología que establece un proceso iterativo y enfatiza en la entrega de características funcionales. Se caracteriza por tener iteraciones breves, con una duración máxima de dos semanas. El ciclo de vida se divide en cinco pasos: primero, se desarrolla un modelo global; segundo, se construye una lista de funcionalidades; tercero, se planifica por funcionalidad; cuarto, se diseña por funcionalidad; y quinto, se construye por funcionalidad (Mendes et al., 2009).

3.2.6.6. Programación Extrema (XP). La Programación Extrema (XP) se basa en un proceso iterativo e incremental que incluye pruebas unitarias continuas y entregas frecuentes. Además, el cliente o un representante del cliente es parte del equipo de desarrollo. XP sugiere la

programación en pares, donde dos personas trabajan juntas en el mismo puesto, para desarrollar las funciones del producto. Antes de agregar nueva funcionalidad, se deben corregir todos los defectos encontrados y se realizan pruebas de regresión de manera constante para detectar errores potenciales (Mendes et al., 2009).

4. Metodología

Para llevar a cabo la investigación se aplican tres fases que corresponden a una revisión sistemática de estudios que tratan sobre el "Lean Software Development" y filosofías de trabajo herederas del Lean Clásico. La Figura 2 resume las fases de la revisión de literatura realizada.

Figura 2.

Metodología de la investigación.

Nota. Adaptado de Towards a Methodology for Developing Evidence-Informed Management Knowledge by Means of Systematic Review, por Tranfield, Denyer and Smart, 2003.

Para su desarrollo, se emplea la guía metodología propuesta por Kitchenham (2004), con el fin de estructurar una revisión de literatura que permita dar base sólida para el cumplimiento satisfactorio de los objetivos propuestos. Así, se desarrolla una revisión sistémica para identificar, evaluar e interpretar la información disponible respecto a un tema específico.

Se proponen actividades específicas para cada una de las tres etapas del proceso de investigación: planificación, desarrollo e informe. Estas actividades son importantes para establecer la pregunta de investigación, crear los protocolos de búsqueda y revisión, y aplicar los estándares de calidad adecuados para obtener resultados y conclusiones relevantes. La estructura propuesta por Kitchenham (2004) para llevar a cabo una revisión sistemática se presenta a continuación, junto con una descripción detallada de cada fase y etapa necesaria para completarla.

Tabla 2.

Protocolo de Revisión Sistemática

Revisión sistémica	
Etapa 1	**Planificación de la revisión**
	- Identificación de la necesidad de una revisión. - Especificación de la pregunta de investigación. - El desarrollo de un protocolo de revisión.
Etapa 2	**Ejecución de la revisión**
	- Identificación de la investigación. - Selección de los estudios y artículos relevantes. - Extracción, seguimiento y síntesis de los datos. - Cumplimiento de objetivos planteados.
Etapa 3	**Informe de los resultados**

Nota. Adaptado del protocolo de revisión propuesto por Kitchenham (2004).

4.1.1. Etapa 1: Planificación de la revisión

Para esta fase, se desarrollaron las siguientes etapas:

4.1.1.1. Identificación de la Necesidad de una Revisión. Se realiza una inmersión teórica y los antecedentes reflejada en el planteamiento del problema y el marco referencial del plan, se busca evidenciar la necesidad de una revisión que aporte al mundo académico información sobre la temática a abordar.

4.1.1.2. Especificación de una Pregunta de Investigación. Se plantea la pregunta de investigación a resolver y se determina el alcance del proyecto incluyendo qué componentes podrá incluir la revisión de literatura. Sobre la pregunta de investigación girarán los objetivos propuestos y permitirá marcar el horizonte de la investigación. Este componente se realiza en el apartado de planteamiento del problema de la presente investigación.

4.1.1.3. Desarrollo de un Protocolo de Revisión. Se desarrolla el protocolo a implementar la Revisión Sistemática. Para la presente revisión es el propuesto por Barbara Kitchenham (2004), compuesto por tres etapas con sus respectivas actividades y resumido en la figura 2.

4.1.2. Etapa 2: Ejecución de la Revisión

Para esta fase, se desarrollaron las siguientes etapas:

4.1.2.1. Identificación de la Investigación. Se extraen palabras clave de los artículos de investigación que conforman el marco de referencia desarrollado y se analizan las palabras claves empleadas. De tal forma se identifican grupos de palabras clave, grupos conformados por conceptos análogos y relevantes para la investigación a desarrollar.

4.1.2.2. Selección de los Estudios y Artículos Relevantes. Obtenidos los resultados en el numeral anterior, se procede a evaluar la pertinencia y concordancia de los estudios con el tema de investigación, para esto, se implementan criterios de inclusión y exclusión con el fin de identificar aquellos artículos que aporten significativamente a la temática expuesta.

4.1.2.3. Extracción, Seguimiento y Síntesis de los Datos. Para la recopilación de estudios determinantes e influyentes para el desarrollo de la investigación, se aplican criterios para garantizar el respectivo aporte de los artículos de investigación sobre el problema de investigación. De esta forma se pretende realizar la correspondiente extracción y síntesis de los datos capaces de realizar los aportes sustanciales para el proyecto y la correcta elaboración de la revisión de literatura con respecto al cumplimiento de cada objetivo planteado en el documento.

4.1.2.4. Cumplimiento de objetivos planteados. Salvo para el primer y el último objetivo planteado en el desarrollo de la investigación (El primero por corresponder al propio desarrollo de la revisión de literatura planteada y el último por ser el entregable final que sintetiza los resultados) el cumplimiento de los objetivos corresponde a un análisis propio de los resultados de la revisión realizada. En dicho análisis, puntualmente para el cumplimiento de cada objetivo, se propone:

4.1.2.4.1. Hallazgo de Herramientas Lean Empleadas en la Industria del Software. Durante esta fase se revisa el análisis bibliométrico realizado y sus componentes estadísticos, Asimismo, se realiza una selección a partir de la información obtenida del análisis de la web, que contiene artículos, casos de estudio, blogs, revistas, entre otros. Una vez obtenida esta información, se presentan aquellas herramientas identificadas en un esquema.

4.1.2.4.3. Análisis de Adopción de Componentes y Herramientas Clásicas del Lean Manufacturing en el Desarrollo del Software y su Aplicación en Diferentes Tipos de Servicios Ofrecidos en el Mercado. Inicialmente, se realiza un recorrido histórico sobre la adopción de las

herramientas clásicas del Lean, mostrando al lector cómo se adoptaron inicialmente y cómo estas se transformaron hasta formar parte fundamental de las actuales metodologías ágiles. Todo esto a partir de los artículos empleados y otros artículos complementarios usando la técnica de bola de nieve. Una vez realizado esto se realiza un análisis descriptivo a partir de la información encontrada en los artículos de la revisión. Dentro de la descripción se mencionan su aplicación, adopción, ejemplos e importancia de cada uno de ellos.

4.1.2.4.2. Diagnóstico que Permita para Comprender el Impacto de la Aplicación de las Herramientas de Clásicas del "Lean Manufacturing" en los Desarrollos de Software y su Aplicación en Diferentes Tipos de Servicios Ofrecidos en el Mercado.

Para dar cumplimiento a este objetivo, se realiza un diagnóstico basado en una revisión focalizada mostrando estudios de caso encontrados en la revisión que permitan mostrar el impacto de la implantación de las herramientas Lean en cada uno de ellos.

4.1.3. Etapa 3: Informe de los Resultados

Finalmente, la revisión sistemática converge con la realización del estado del arte realizada a partir del análisis de los artículos seleccionados con el fin de aportar soluciones a la pregunta de investigación planteada. El cumplimiento de este objetivo es consecuencia del desarrollo de los objetivos previos y de la síntesis resultante. Para dar cumplimiento se elabora un artículo de carácter publicable en el cual se exponen los principales resultados de la investigación.

5. Revisión de la Literatura

5.1. Análisis Bibliométrico

5.1.1. Planeación de la Revisión de Literatura

5.1.1.1. Identificación de la necesidad de una revisión. Dada la inmersión teórica y los antecedentes expuestos en el planteamiento del problema y el marco referencial de la presente investigación, se puede observar la necesidad de una revisión que aporte al mundo académico información sobre la adopción de herramientas "Lean" en la creación de software y el impacto generado por las mismas.

5.1.1.2. Especificación de una pregunta de Investigación. Teniendo en cuenta el alcance del proyecto se opta por elaborar una revisión de literatura que permita abarcar desde el área científica, el estado actual de la literatura, e identificar las investigaciones y artículos más relevantes respecto al tema expuesto para este proyecto, con el propósito de explorar herramientas clásicas del "Lean" aplicadas en el desarrollo del software, y su adopción e impacto en la industria. Como se mencionó en el planteamiento del problema, la pregunta de la investigación es: ¿Cuáles son las herramientas de "Lean Manufacturing" actualmente aplicadas en la industria del desarrollo del software?, ¿cómo fue su adopción y cuál es su impacto para esta industria?

5.1.1.3. Desarrollo de un Protocolo de Revisión. El protocolo a implementar para la Revisión Sistemática es el propuesto por Barbara Kitchenham (2004), compuesto por tres etapas con sus respectivas actividades. Este se presenta en el capítulo "Metodología" del presente proyecto.

5.1.2. Ejecución de la Revisión de Literatura

5.1.2.1. Identificación de la Investigación. Se extraen palabras clave de los artículos de investigación que conforman el marco de referencia desarrollado y se analizan las palabras claves empleadas. De tal forma se identificaron cuatro grupos de palabras clave, grupos conformados por conceptos análogos y relevantes para la investigación a desarrollar.

Tabla 3.

Identificación de Palabras Clave

Lean Managment	***Herramientas/Componentes***	***Lean Software Development***	***Sector***
- Lean. - Lean Philosophy - Lean Manufacturing - Lean Aproach - Exposure	- Tools - VSM -SMED - Kanban - 5 ´s - Waste - Poka Yoke - Heiyunka - Kaizen - House of Lean	- Lean Software Development. -Lean StartUp. -Agile. - Scrum	- Software. -Scrum. -Frameworks. -Scrum Cristal Methodologies -DSDM -ASD -FDD XP

Una vez identificados los grupos de palabras clave, se procedió a interconectar los conceptos por medio de operadores lógicos, comillas y paréntesis, con el fin de estructurar la primera ecuación de búsqueda que se empleó en las bases de datos “Scopus” y “Web of Science”, bases de datos caracterizadas por facilidad en su acceso y por la confiabilidad que representa su contenido en la comunidad científica.

Tabla 4.

Primera Ecuación de Búsqueda

Bases de Datos	***Ecuación de búsqueda***	***Resultados***
Scopus	(TITLE-ABS-KEY(("Lean" OR "Lean philosophy" OR "Lean practices" OR "Lean methodology" OR "Lean approach" OR "Lean Implementation")) AND TITLE-ABS-KEY(("Lean tools" OR "adoption" OR "influence" OR "Tools" OR "VSM" OR "Heijunka" OR "SMED" OR "waste" OR "Kanban" OR "5s" OR "5 S's" OR "Five S's" OR "Heiyunka" OR "Poka Yoke" OR "Kanban" OR "House of Lean" OR "kaizen" OR "waste")) AND TITLE-ABS-KEY(("Lean Software Development" OR "LSD" OR "Lean Startup" OR "Agile" OR "agile methodology")) AND TITLE-ABS-KEY(("Software" OR "Scrum" OR "Framework" OR "Cristal Methodologies" OR "DSDM" OR "ASD" OR "FDD" OR "XD")))	599

5.1.2.2. Selección de los Estudios y Artículos Relevantes. Obtenidos los resultados en el numeral anterior, se procede a evaluar la pertinencia y concordancia de los estudios con el tema de investigación, para esto, se implementan criterios de inclusión y exclusión con el fin de identificar aquellos artículos que aporten significativamente a la temática expuesta.

Tabla 5.

Criterios de inclusión y exclusión

Criterios de inclusión	**Criterios de exclusión**
- Documentos comprendidos entre los años 2015 – 2022 - Documentos relacionados con los grupos de palabras clave y con el objeto de estudio.	- Documentos escritos en un idioma diferente al español e inglés. - Documentos irrelevantes para el aporte de la pregunta de investigación. - Conference papers.

Con la ecuación de búsqueda de la Tabla 2, se obtienen 599 artículos de la base de datos Scopus. A estos artículos se procede a aplicar los criterios de inclusión y exclusión obteniendo finalmente 26 artículos para la investigación. De esta manera, se obtiene la siguiente ecuación de búsqueda final.

Tabla 6.

Ecuación de búsqueda final

Ecuación de búsqueda final
(TITLE-ABS-KEY (("Lean" OR "Lean philosophy" OR "Lean practices" OR "Lean methodology" OR "Lean approach" OR "Lean Implementation")) AND TITLE-ABS-KEY (("Lean tools" OR "adoption" OR "influence" OR "Tools" OR "VSM" OR "Heijunka" OR "SMED" OR "waste" OR "Kanban" OR "5s" OR "5 S's" OR "Five S's" OR "Heiyunka" OR "Poka Yoke" OR "Kanban" OR "House of Lean" OR "kaizen" OR "waste")) AND TITLE-ABS-KEY (("Lean Software Development" OR "LSD" OR "Lean StartUp" OR "Agile" OR "agile methodology")) AND TITLE-ABS-KEY (("Software" OR "Scrum" OR "Framework" OR "Cristal Methodologies" OR "DSDM" OR "ASD" OR "FDD" OR "XD"))) AND (LIMIT-TO (EXACTKEYWORD , "Software Engineering") OR LIMIT-TO (EXACTKEYWORD , "Lean Manufacturing") OR LIMIT-TO (EXACTKEYWORD , "Agile Software Development") OR LIMIT-TO (EXACTKEYWORD , "Lean Software Development") OR LIMIT-TO (EXACTKEYWORD , "Software Design")) 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)) AND (EXCLUDE (EXACTKEYWORD , "Lean Production")) AND (EXCLUDE (DOCTYPE , "cp"))

De forma complementaria, se realiza una recopilación de artículos con la técnica “Bola de Nieve”, se incluye una revisión web en “Google Scholar” y artículos de conferencias con el fin de ampliar la información relevante y explorar mejor el contexto del LSD y abordar conceptos puntuales encontrados. Finalmente, se analizaron de forma integral 66 documentos referentes al tema central de la investigación o a tópicos de la misma.

5.1.2.3. Extracción, Seguimiento y Síntesis de los Datos. Para la recopilación de estudios determinantes e influyentes para el desarrollo de este proyecto, se aplica como criterio más

importante para la selección de estas publicaciones, el respectivo aporte de los artículos de investigación sobre el problema de investigación. De esta forma se pretende realizar la correspondiente extracción y síntesis de los datos capaces de realizar los aportes sustanciales para el proyecto y la correcta elaboración de la revisión de literatura.

5.1.3. Análisis de los Resultados de la Revisión de la Literatura

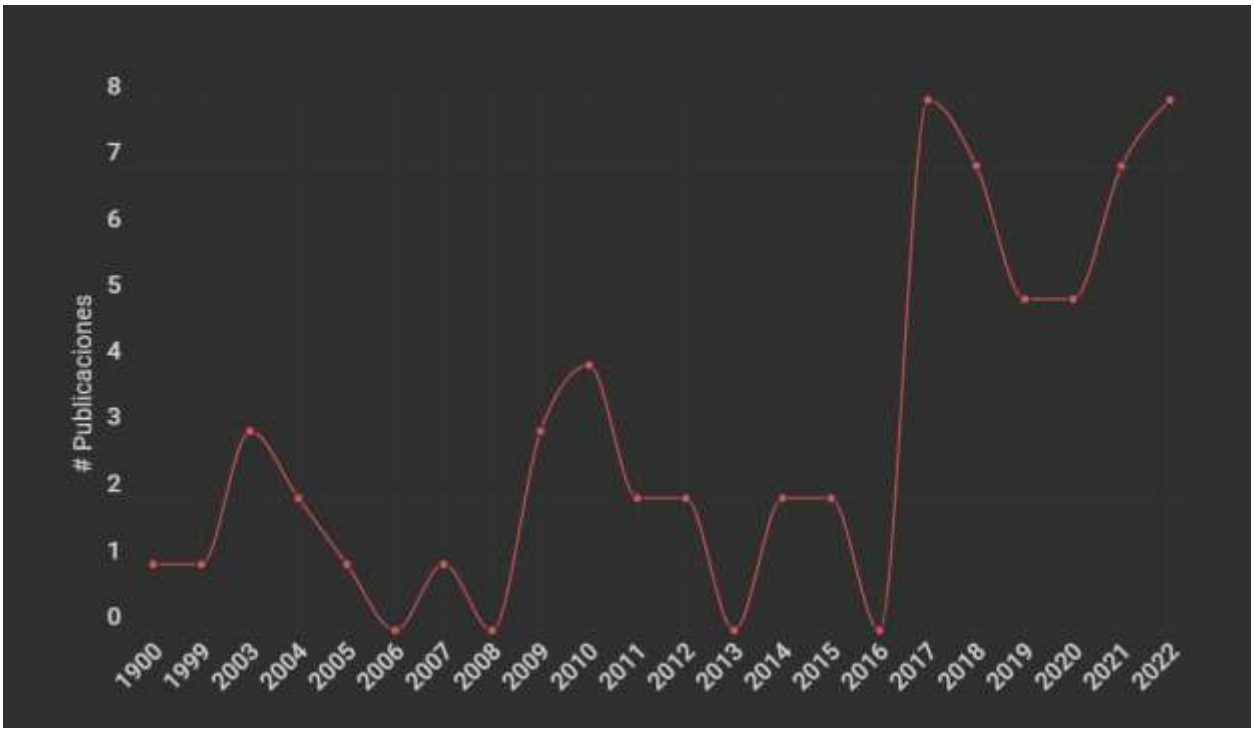
Para dar respuesta a la pregunta de investigación, es necesario realizar un análisis bibliométrico para analizar estadísticamente los documentos de la investigación resultantes. De esta manera se obtiene información sobre el autores, revistas, citaciones y áreas de la temática de investigación. Se emplea la herramienta Vosviewer para el análisis de tendencias y se analizan las tablas.

Para los filtros aplicados en la revisión de literatura se encuentran para el año 2017, 8 publicaciones relacionadas con el problema de investigación, siendo el año más representativo. En los siguientes años disminuye paulatinamente hasta 5 publicaciones por año para el 2019 y 2020. Posteriormente vuelve a incrementar a 7 en el año 2021 y llega a 8 publicaciones nuevamente en 2022.

Vale la pena destacar que se incluyen publicaciones de años anteriores por ser documentos referentes o postulados claves del Lean propiamente, del Lean Software Development o de las metodologías ágiles.

Figura 3.

Artículos Publicados por Año

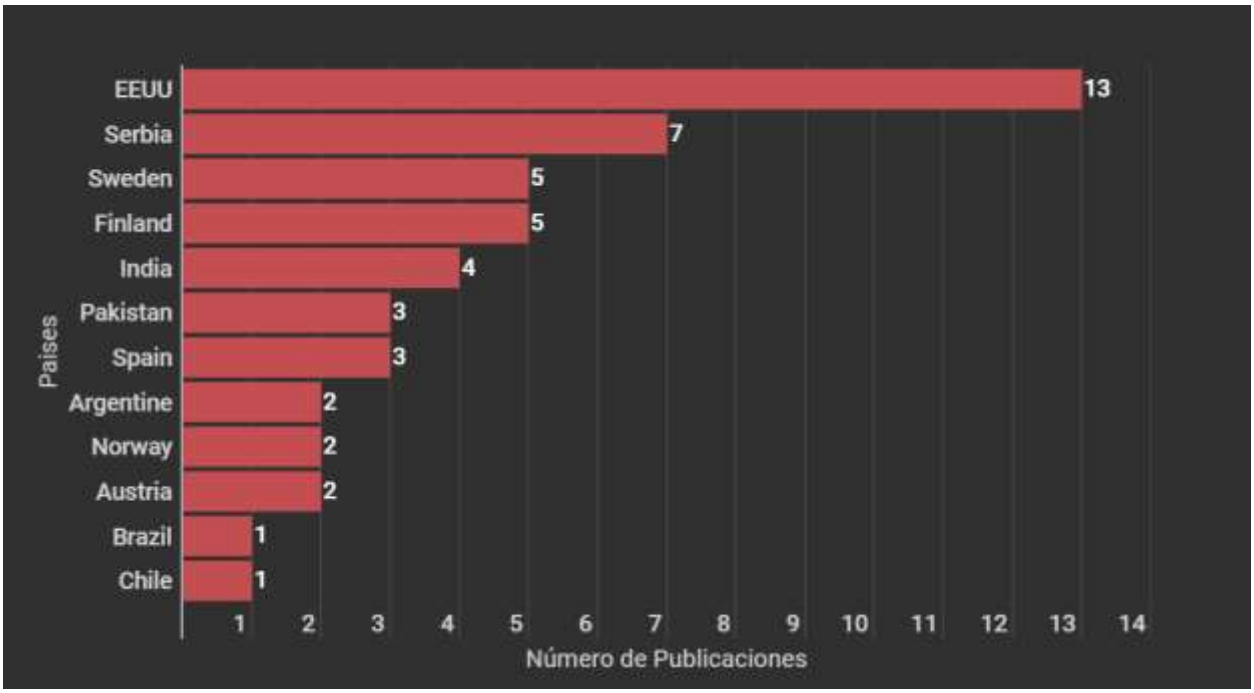


Nota. Elaborado a partir de los resultados de la revisión.

Los países destacados en esta categorización por el mayor número de documentos son: Estados Unidos con 13 documentos, Serbia con 7, Suecia y Finlandia con 5, india con 4, y Pakistán y España con 3. Otros países como Argentina, Noruega y Austria destacan con 2 documentos. De Latinoamérica destacan Brasil y Chile cada uno con 1 artículo.

Figura 4.

Publicaciones por País.

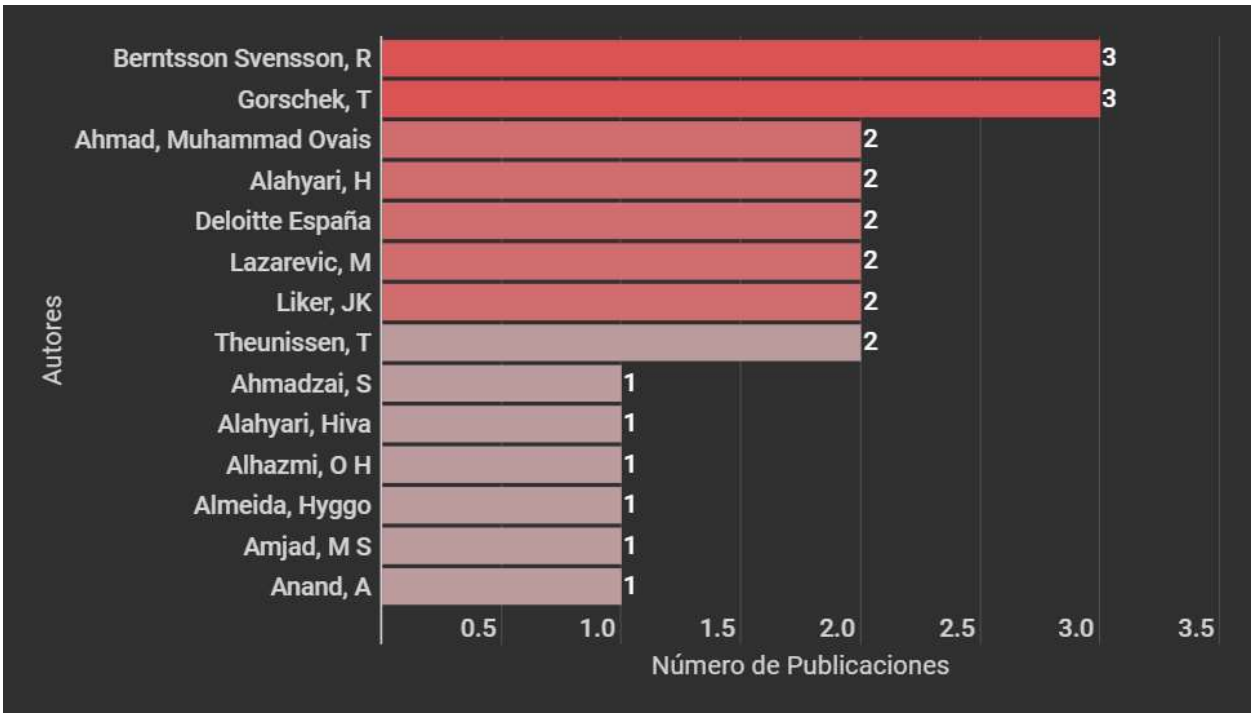


Nota. Elaborado a partir de los resultados de la revisión.

En cuanto a las publicaciones por autor, en la siguiente figura se puede identificar a los once primeros autores que sobresalen por su productividad y su aporte en el tema de investigación. Destacan con 3 publicaciones Berntsson Svensson R y Gorschek, T.

Figura 5.

Artículos Publicados por Autor.

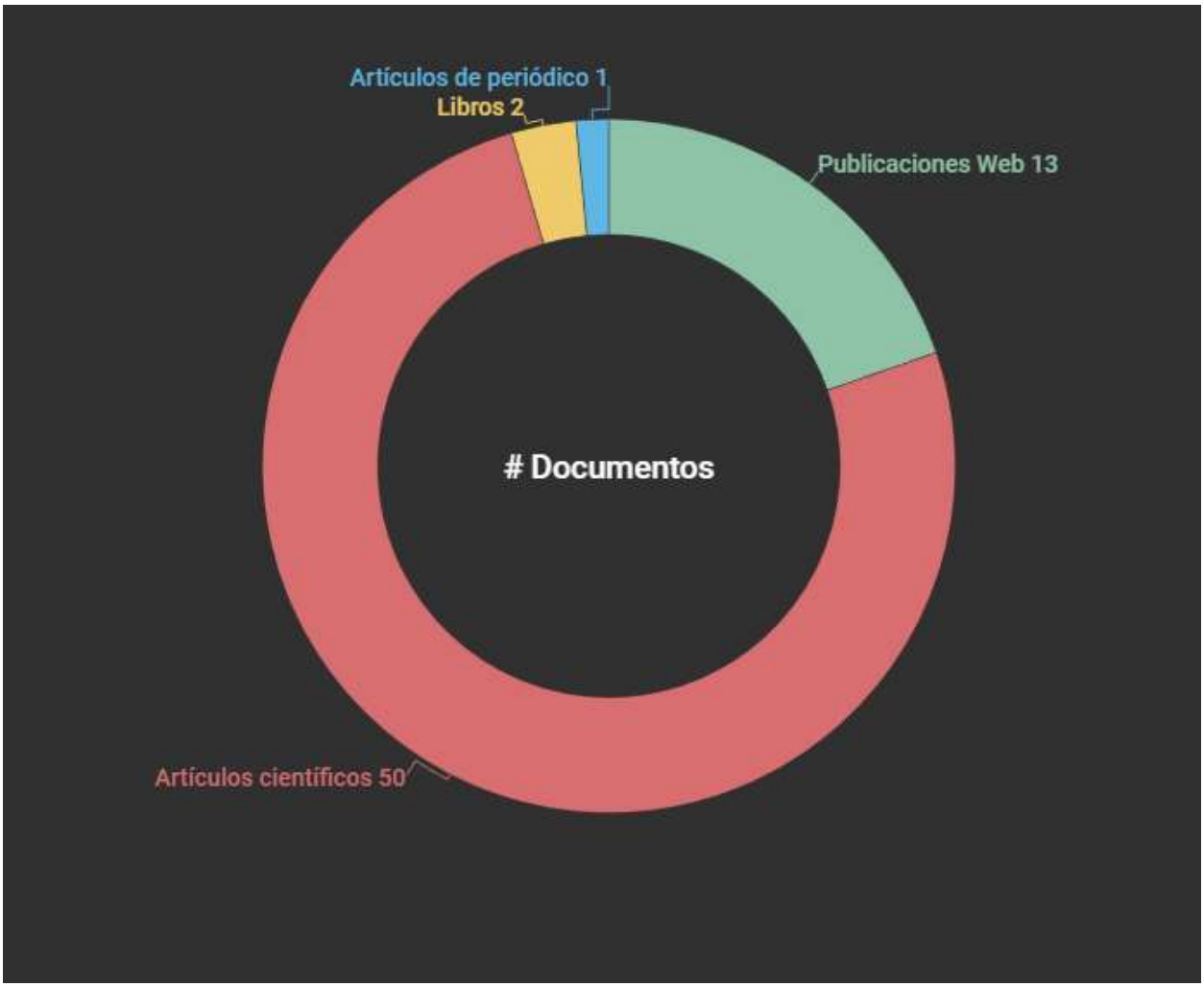


Nota. Elaborado a partir de los resultados de la revisión.

Finalmente, en cuanto al tipo de artículos, el 76% son artículos originales de investigación, el 20% a publicaciones Web, y el 4% faltante a libros y artículos de periódicos. Vale destacar el alto protagonismo de los documentos provenientes de revistas científicas especializadas en temáticas del sector tecnológico.

Figura 6.

Tipo de Documento

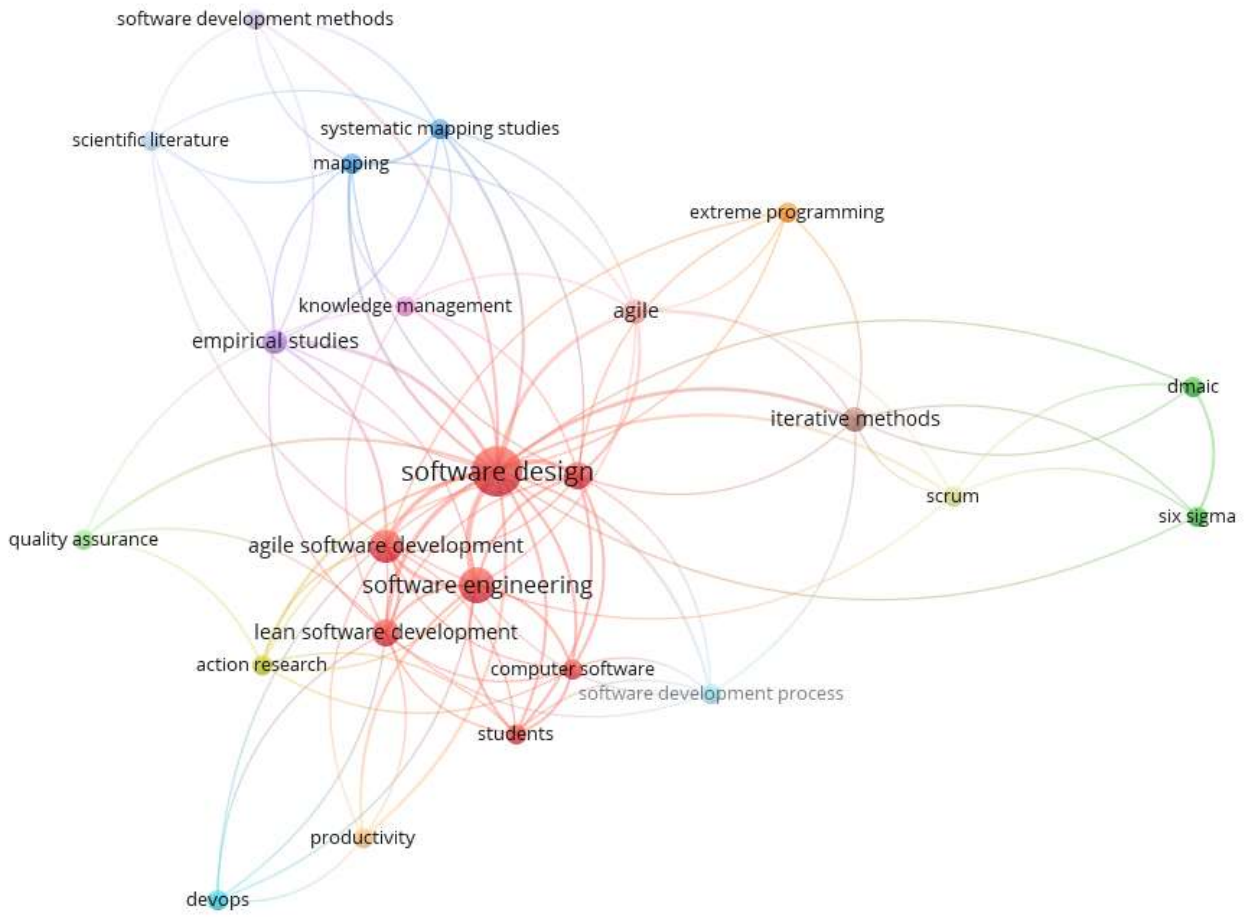


Nota. Tomado de “Scopus” a partir de los resultados.

Tras realizar el análisis de contenido, se identificaron 24 de las palabras claves más frecuentes, representadas en siguiente figura. Como es de esperar, se encuentran términos altamente relacionados con el problema de investigación abordado, tanto desde el componente asociado al “Lean” como con el sector asociado, la industria del software.

Figura 7.

Términos de Mayor Ocurrencia.



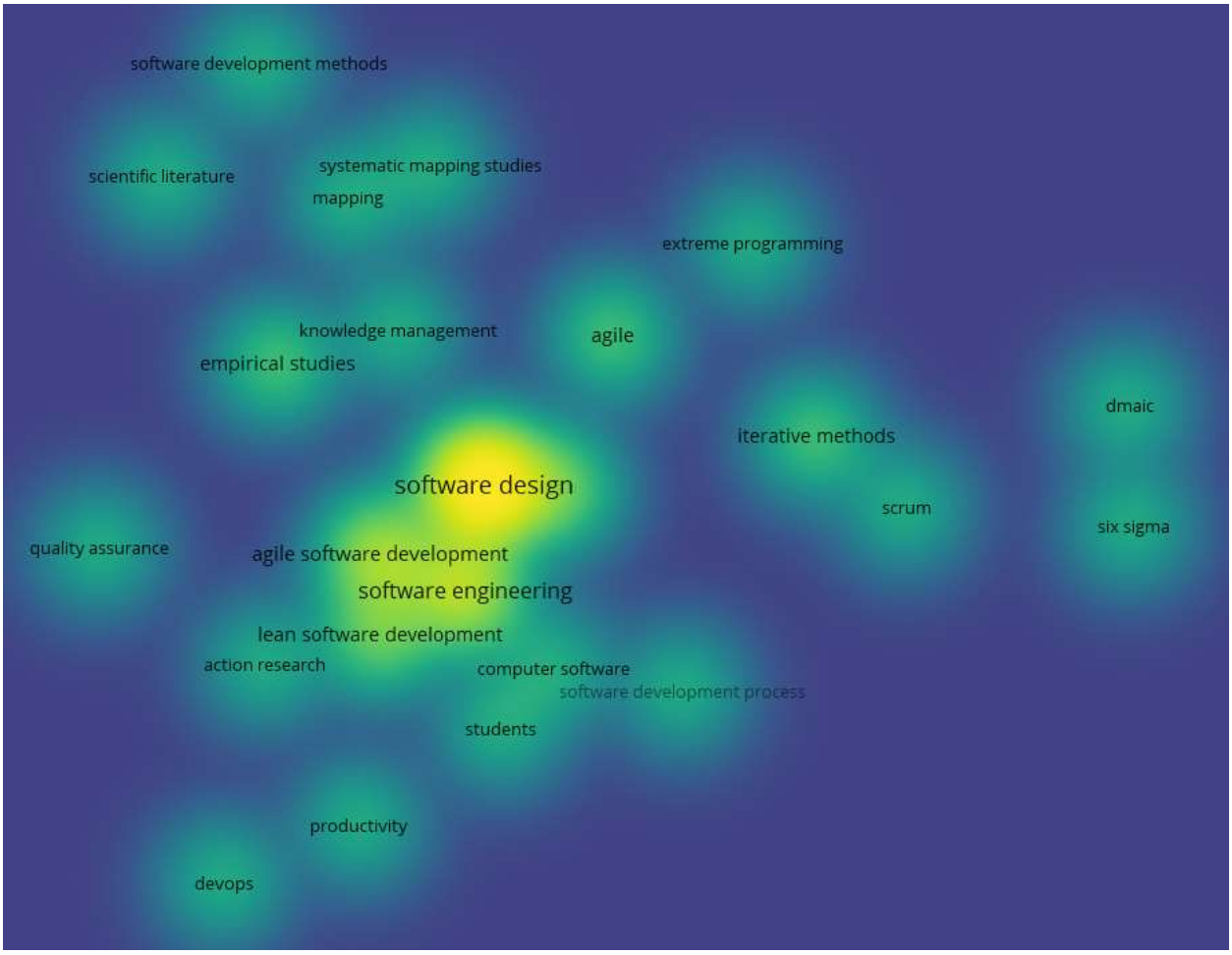
Nota. Figura extraida del software "Vosviewer" con los articulos empleados en la revisión de literatura.

El análisis de co-ocurrencia de palabras claves con el software Vosviewer permite identificar diferentes clústeres para cada palabra clave. Un cluster es un conjunto de nodos estrechamente relacionados según el tipo de vínculo que se analiza; cada nodo es asignado exactamente a un clúster. Se distinguen 5 clústeres principales en los resultados, sin embargo, destaca el clúster rojo por su alta relevancia, englobando el Lean Software Development y el diseño

de software propiamente. Los demás clústeres abordan tópicos de este tópico central: El azul términos asociados al mapeo y gestión de datos, el morado a métodos de desarrollo de software, el naranja a “Extreme Programming” y el verde, aparentemente hace referencia a términos asociados a “Six sixma”.

Figura 8.

Términos de Mayor Ocurrencia por Mapa de Calor.



Nota. Figura extraida del software “Vosviewer” con los articulos empleados en la revisión de literatura.

También, en la anterior figura se puede identificar fácilmente aquellas palabras de mayor ocurrencia. Destacan “Software Design” “Agile Software Development” y “Sofware Enngineering”

Las demás palabras destacadas si bien son relevantes poseen una menor citación. Destacan palabras que hacen referencia a herramientas o frameworks del “Lean” como “Scrum” o “Mapping”, y otros ya propiamente de la industria del software tales como “iterative methods” y “extreme programming”.

6. Resultados

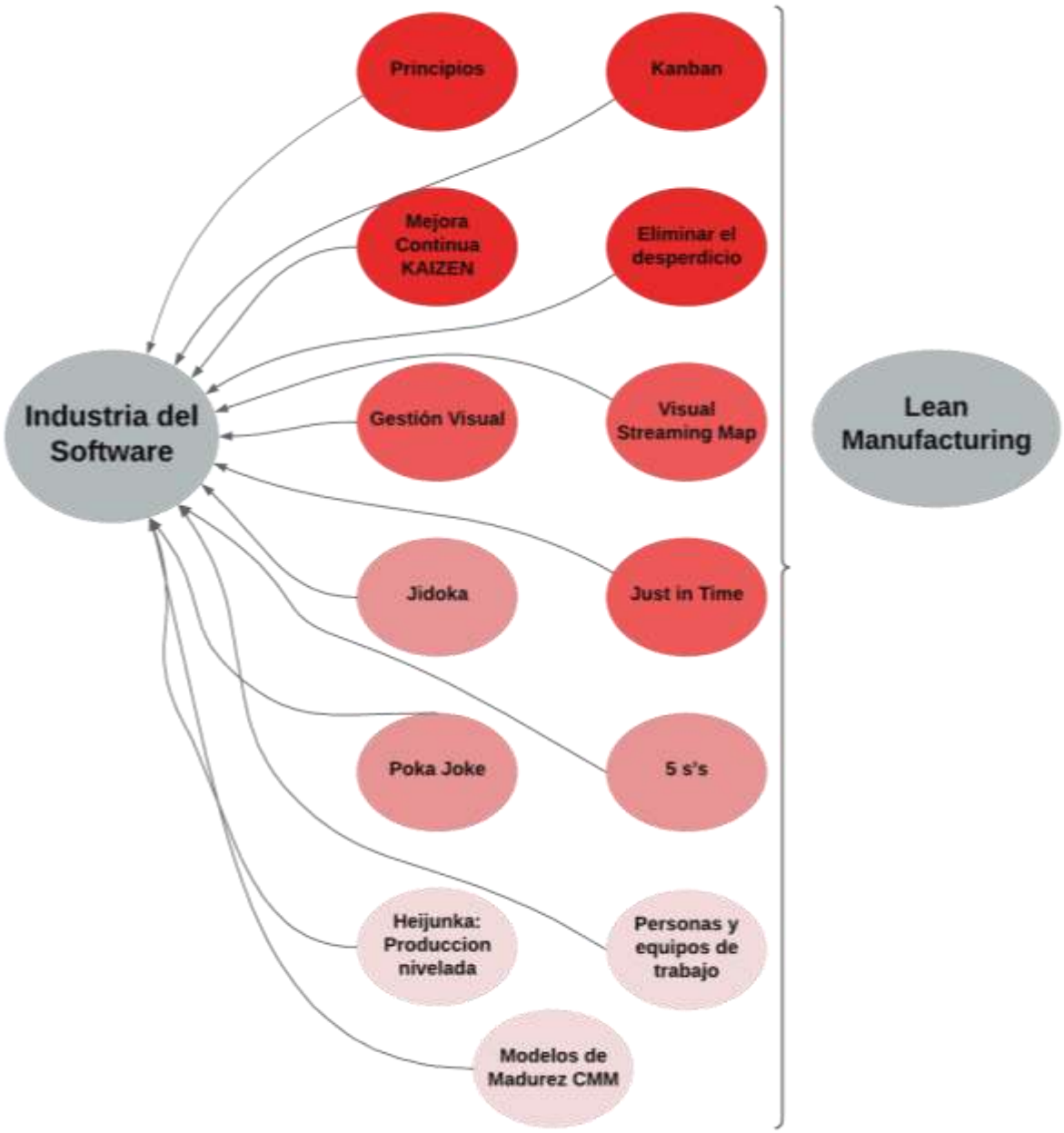
6.1. Componentes y Herramientas Lean Adoptados en la Industria del Software.

Diversos componentes y herramientas de la filosofía Lean clásica fueron adoptados en la industria del Software. Algunos de estos son altamente empleados bajo nuevas metodologías específicas para la industria quienes modificaron y adoptaron a conveniencia estas herramientas y componentes para satisfacer las necesidades del entorno y las complejas y cambiantes condiciones del desarrollo del software. Dentro de estos elementos se incluyen también herramientas que, no siendo propias del Lean Manufacturing, están notoriamente inspiradas por él o son híbridos nacientes de herramientas clásicas.

A continuación, se presenta de manera global aquellos componentes y herramientas del Lean clásico que fueron adaptados por la industria del software.

Figura 9.

Componentes "Lean" Adoptados por la Industria del Software



Es importante entender que la frecuencia de uso de las herramientas Lean en la industria del software no se refleja de forma lineal y directa, debido a que la implementación de estas herramientas no se realiza de manera separada y aislada, sino que se integran en metodologías más

amplias y holísticas que combinan diversos aspectos del Lean, como en el caso de Lean Software Development y las metodologías ágiles anteriormente mencionadas.

Ahora bien, para realizar un ranking que refleje la frecuencia de uso de las herramientas o componentes, se presenta la dificultad de que esta frecuencia no se da de forma normal y homogénea, lo que dificulta la elaboración de un ranking preciso. Sin embargo, se propone una clasificación en categorías A, B, C y D para los componentes adoptados del Lean en la industria del software, donde A corresponde a los componentes más utilizados y D a los menos empleados, sin hacer distinción entre componentes o herramientas dentro de la misma categoría de acuerdo a los resultados de la revisión de literatura realizada.

A continuación, se presenta la tabla de clasificación ABCD de los componentes Lean adoptados en la industria del software:

Tabla 7.

Ranking de Herramientas y Componentes del Lean Clásico más Adoptados en la Industria del Software.

Categoría	Herramientas y componentes
A	Mejora Continua (KAIZEN)
	Reducción de desperdicios.
	Kanban
	Principios
B	Just in time
	Visual Streaming Map (VSM)
	Gestión Visual
C	Jidoka
	Poka Joke
	5S’s
D	Heijunka
	Modelos de madurez: CMMI.
	Personas y equipo de trabajo

Las herramientas en la categoría A son las más utilizadas y adoptadas por la industria, y se relacionan con la mejora continua del proceso, la reducción de desperdicios, Kanban y los principios del Lean. Es lógico que estas herramientas se encuentren en la categoría A, ya que están en la base del Lean y son fundamentales para cualquier organización que quiera implementar esta metodología en su proceso. Además, su inclusión fue la más mencionada en la revisión de literatura encontrada.

Las herramientas de la categoría B, como Just in time, Visual Streaming Map (VSM) y Gestión Visual, también son muy importantes para la industria del software. Sim embargo, su mención o impacto no parece ser tan significativo como las de la categoría A.

En la categoría C, se encuentran herramientas como Jidoka, Poka Joke y 5S's, que están relacionadas con la calidad de los productos o servicios generados y el proceso, pero no son tan utilizadas en la industria del software como las herramientas de las categorías A y B. Estas herramientas son muy útiles para la mejora del desarrollo del software, pero no son ejes centrales de las metodologías ágiles o del LSD.

Finalmente, la categoría D incluye herramientas como Heijunka, Modelos de madurez: CMMI y Personas y equipo de trabajo. Estas herramientas son importantes para la mejora continua del proceso, pero no parecen ser tan mencionadas como las otras herramientas y componentes, o su intervención no es tan significativa como aquellas de las anteriores categorías.

Es importante tener en cuenta que este ranking no pretende ser una lista exhaustiva de todas las herramientas del Lean clásico que se hayan utilizado en la industria del software, sino más bien una selección de las más utilizadas y adoptadas en la transición histórica hacia las metodologías

ágiles y el LSD de acuerdo a los resultados de la revisión. Es importante destacar que la adopción de cada herramienta se dio en conjunto con la inclusión de estas en las metodologías ágiles y el LSD, y no de forma aislada. Como se destaca a lo largo de la presente investigación, el éxito de la implementación del Lean clásico en la industria del software se debe en gran parte a la integración de estas herramientas en las metodologías ágiles y el LSD.

6.2. Adopción de las Herramientas Clásicas del Lean Manufacturing en el Desarrollo del Software y su aplicación en Diferentes Tipos de Servicios Ofrecidos en el Mercado.

En este apartado se presentará la adopción de cada uno de los componentes y herramientas empleados en la industria del software. Sin embargo, primero se vale la pena destacar como se dio la transición histórica que permitió la adopción del Lean clásico en las actuales metodologías de desarrollo de software (metodologías ágiles, LSD y similares). Se dará un contexto que permita comprender cómo se gestó la necesidad de metodologías más flexibles y adaptables, y cómo el Lean clásico y sus principios se adaptaron a este nuevo paradigma. Una vez realizado esto, se describirá detalladamente la adopción de cada herramienta en particular y cómo contribuyen al éxito de las metodologías ágiles y el LSD. Se espera que este capítulo proporcione una visión completa de la evolución del Lean clásico y su integración en las metodologías modernas, así como una comprensión detallada de cómo cada componente y cada herramienta surgida en el Lean clásico se utiliza actualmente para lograr una mayor eficiencia y efectividad en los proyectos de desarrollo de software.

6.2.1. Del Lean Clásico al Lean Software Development y las Metodologías Ágiles

Desde la literatura revisada acerca de la aplicación de la filosofía Lean a la Ingeniería del Software se pone de manifiesto que se está hablando de una disciplina inmersa en un rápido

progreso cuyo interés va en aumento desde la irrupción de la industria del software. La primera referencia obligada al hablar de "Lean" en la industria del software es el trabajo de Tom y Mary Poppendieck, quienes publicaron "Lean Software Development: En Agile Toolkit" en 1993 y los volúmenes posteriores. En su obra, se puede observar cómo se complementan los conceptos de la filosofía Lean con los espacios de trabajo de diseño de software.

También en la misma década surgieron las metodologías ágiles y el Manifiesto Ágil, convirtiéndose en un enfoque popular para el desarrollo de software. Este enfoque se basa no solo en el LSD, sino también en todas las metodologías ágiles actuales, centrándose en la entrega rápida de software funcional, la reducción del desperdicio, la realización de procesos iterativos y la mejora continua de los procesos. Se observa que históricamente, los términos "Lean" y "Ágil" están mutuamente ligados en la industria del software (Theunissen at al, 2022).

Es importante mencionar que, desde la propia concepción de las metodologías ágiles en el Manifiesto Ágil, se puede notar la influencia del Lean Clásico en su filosofía. Además, las aproximaciones ágiles a la gestión de proyectos parecen tener raíces en el pensamiento y prácticas de la manufactura Lean, según afirma R. Morien (2018). A medida que la industria del software ha evolucionado y se ha vuelto más estructurada y orientada a procesos, la adopción de herramientas del Lean ha ido ganando terreno (Anghel et al, 2022).

Algunas de las herramientas transaccionaron a metodologías ágiles de forma particular. La adopción del Kanban en el desarrollo de software se remonta también a principios de la década de 2000, cuando David J. Anderson lo introdujo en su equipo de desarrollo de software en Microsoft (Anderson, 2010). En su libro "Kanban: Successful Evolutionary Change for Your Technology Business", David J. Anderson presentó cómo la herramienta Kanban fue aplicada por primera vez

en el desarrollo de software en el equipo de Microsoft Office en 2004. Desde entonces, ha sido ampliamente adoptada en la industria del software y ha evolucionado para hoy ser una de las metodologías más populares dentro del movimiento ágil (Anghel et al, 2022).

En contraste, las otras herramientas parecen tener una adopción más paulatina y menos transitoria. Entendiendo que en las empresas y en las metodologías ágiles y el LSD (emergentes en su momento) la adopción de cada una de ellas no se realizó de forma aislada, sino que fue realizada de manera conjunta y adaptada de acuerdo a las necesidades particulares de la industria del software y de los propios procesos de desarrollo de software (Theunissen at al, 2022).

Finalmente, debido a la reiterada mención de la filosofía Lean y de los métodos ágiles, vale la pena hacer una diferenciación entre estas dos. Si bien es claro que ambas filosofías pueden tener grandes parentescos y similitudes, y a que ambas conceden gran importancia a entregar rápidamente a los clientes un producto que les genere valor, “Lean” y “Agile” son dos términos distintos en la industria del software y presentan algunas diferencias (BBVA, 2022). Mientras el pensamiento ágil realiza su enfoque en el desarrollo de software y la gestión de dicho desarrollo, Lean es aplicable a todos los ámbitos, desde el mismo desarrollo de software hasta la gestión de la misma empresa donde se produce, pasando por clientes y proveedores. Cuanto mayor sea el alcance del Lean, mayores son los beneficios potenciales (Hibbs, et al. 2009). La mayoría de los esfuerzos Lean comienzan por procesos pequeños y expanden su alcance con el tiempo, obteniendo cada vez más beneficios en el proceso en similitud con el mismo Lean clásico de Toyota. En este sentido, de acuerdo a diversos autores, las metodologías ágiles desde la perspectiva Lean, son prácticas válidas e inmersas dentro de su desarrollo (Sanz, 2015). “Lean” ve todas las metodologías Ágil como prácticas de apoyo válidas (Tegegne et al, 2019).

Hibbs et al. (2009) destacan también como diferencias entre ambas filosofías que el desarrollo de software ágil es la colaboración estrecha con el cliente y la entrega rápida de software con el fin de que este funcione lo antes posible. “Lean” por su parte considera que eso vale la pena, pero se centra principalmente en la eliminación de residuos en el contexto de lo que el cliente valora. Otras diferencias destacadas por diversos autores es que las metodologías Ágiles tienen un buen número de metodologías formales, mientras que Lean no tiene metodologías formales. Lean, en contraparte, tiene un conjunto de prácticas recomendadas entre las que elegir. De hecho, “al implementar el desarrollo de software Lean, es bastante común escoger una metodología ligera Agile como punto de partida y empezar a aplicar otras herramientas Lean (como VSM) desde allí” (Hibbs, 2009, p. 16).

6.2.2. Adopción de Componentes del Lean clásico en la industria del Software

Una vez entendiendo el contexto de adopción del Lean Clásico en la industria del Software, se presentan aquellos componentes que surgen del Lean clásico, se aplican en la industria del software y son mencionados en la revisión de literatura realizada.

6.2.2.1. Principios. Los principios del "Lean Software Development" (LSD) propuestos por Poppendieck en 2003, están directamente inspirados en los pilares del "Lean Manufacturing" originalmente expuesto por Toyota. El LSD, junto con el manifiesto ágil, son los precursores de las actuales metodologías ágiles y, por lo tanto, es importante revisar los principios del LSD para comprender su influencia real.

Estos principios del LSD incluyen la eliminación del desperdicio, la integración de la calidad en el desarrollo desde el primer momento, la creación de conocimiento, la toma de decisiones aplazada, la entrega rápida, el respeto por las personas y la optimización del conjunto.

Si bien estos principios son similares a los del "Lean Manufacturing", hay una diferencia en el ítem número 4, que consiste en aplazar las decisiones. Esto se debe a las dinámicas propias de la industria del software, donde es más pertinente retrasar la toma de decisiones clave del producto para tener una mayor retroalimentación por parte del cliente o usuario final.

Es importante destacar que, aunque se adoptan los principios del Lean Manufacturing, en el caso de la industria del software hay ciertas modificaciones y temas que varían de acuerdo a las características propias de esta industria. Por lo tanto, las acciones correctivas deben emprenderse lo más pronto posible y siempre se debe tener un enfoque preventivo buscando las condiciones que eviten cualquier posibilidad de errores (Theunissen at al, 2022).

Por otro lado, los 12 principios del manifiesto ágil se enfocan directamente en las particularidades de la industria del software. En ellos se destaca la satisfacción del cliente a través de la entrega temprana y continua de software con valor, la aceptación de cambios en los requisitos incluso en etapas tardías del desarrollo, la entrega frecuente de software funcional, el trabajo conjunto entre los responsables de negocio y los desarrolladores, el trabajo en equipo motivado, la comunicación cara a cara como método efectivo, el software funcionando como medida principal de progreso, el desarrollo sostenible, la atención continua a la excelencia técnica y al buen diseño, la simplicidad como esencial, la autoorganización de equipos y la reflexión periódica para mejorar el comportamiento del equipo.

A pesar de la orientación clara hacia las particularidades propias de la industria del software, los principios del manifiesto ágil están influenciados por los pilares del "Lean Manufacturing". La identificación del valor desde la perspectiva de cada cliente es fundamental en los primeros principios del manifiesto ágil, seguido de una orientación clara a la reducción de

desperdicios que no aporten a la propia generación de valor para el cliente, lo que da origen a la naturaleza iterativa de estas metodologías (Zimmermann, 2017).

6.2.2.2. Personas y Equipo de Trabajo. Es importante destacar que el componente humano es fundamental para garantizar su implementación de forma satisfactoria. Según Poppendieck (2003), el Lean Software Development (LSD) se enfoca en la colaboración y la comunicación efectiva entre el equipo de trabajo, lo que se traduce en una mejora en la calidad del software y en una reducción del tiempo de entrega.

Siguiendo esta línea, Liker (2021) menciona que las principales características comunes a los equipos de trabajo Lean incluyen grupos reducidos por tareas, un responsable o líder Lean por cada equipo en el que recaen responsabilidades grupales, el uso de controles visuales de manera estricta, el enfoque en tareas específicas o temas concretos, y la realización de actividades y reuniones planeadas y metódicas para garantizar controles y mejora continua.

Puntualmente, como esencia de personas y equipo de trabajo Lean Poppendieck (2003) plantea una serie de "herramientas" (Sanz, 2015):

Tabla 8.

Herramientas para personas y equipo de trabajo en LSD

Herramientas para equipos de trabajo en LSD	
Autodeterminación	La clave para una transformación exitosa es lograr que las personas involucradas crean en el cambio y se comprometan con él. Si se enfoca la transformación únicamente en transferir prácticas en lugar de principios fundamentales, es probable que el cambio sea efímero y difícil de mantener a largo plazo.
Motivación	Es importante que el propósito del trabajo sea claro, convincente y alcanzable para motivar al equipo. Además, fomentar el sentido de pertenencia, confianza, competencia y progreso, y equilibrar la vida laboral y familiar son claves para mantener la motivación. Es importante evitar el exceso de esfuerzo, moderar el heroísmo y reservarlo solo para emergencias.

Liderazgo	Se buscan líderes más que simples gestores, que hagan frente al cambio, marcando el camino a seguir, alineando y motivando al equipo. Un liderazgo basado en el respeto del equipo hacia el líder, por su profundo conocimiento del cliente y de los aspectos técnicos, más allá de una autoridad concedida.
Experiencia	Facilitar que los equipos adquieran y compartan su experiencia de forma autónoma, tolerando los errores durante el proceso de aprendizaje y fomentando la transmisión del conocimiento, especialmente el tácito. Debido a la especificidad de las tareas técnicas y de conocimiento especializado requerido en el propio desarrollo, es importante facilitar la creación de comunidades de expertos, especialmente en aquellas áreas críticas para el éxito empresarial.

Nota. Adaptado de Sanz (2015).

Estas "herramientas" funcionan como pautas generales en la implementación de equipos de trabajo en LSD y comparten características con la formación de equipos en otras metodologías ágiles (Zorzetti, et al, 2022). Ahora bien, aunque no es explícita su mención la formación de equipos de trabajo en el desarrollo de software bajo metodologías ágiles están claramente influenciadas por los equipos de Lean Manufacturing. La búsqueda de valor en el cliente, la comunicación eliminando despilfarros, los sistemas de motivación orientados a un propósito más que a las simples tareas, la transmisión de experiencia de forma rápida y la realización de reuniones periódicas para garantizar mejoras continuas son clara consecuencia de la aplicación de los principios de Lean, y de la necesidad de poseer dinámicas organizativas en el personal que puedan desarrollar sus funciones bajo este esquema (Lehtinen, 2017).

6.2.3. Adopción de herramientas del Lean clásico en la industria del Sofware.

A continuación, se presentan aquellas herramientas que surgen del Lean clásico más mencionadas en los artículos de la presente revisión de literatura.

Como resultado del proceso investigativo de la revisión realizada, fue posible identificar el empleo de diversas herramientas comprendidas dentro de los 10 componentes del Lean Manufacturing mencionados en la figura 13. Estos son descritos a continuación:

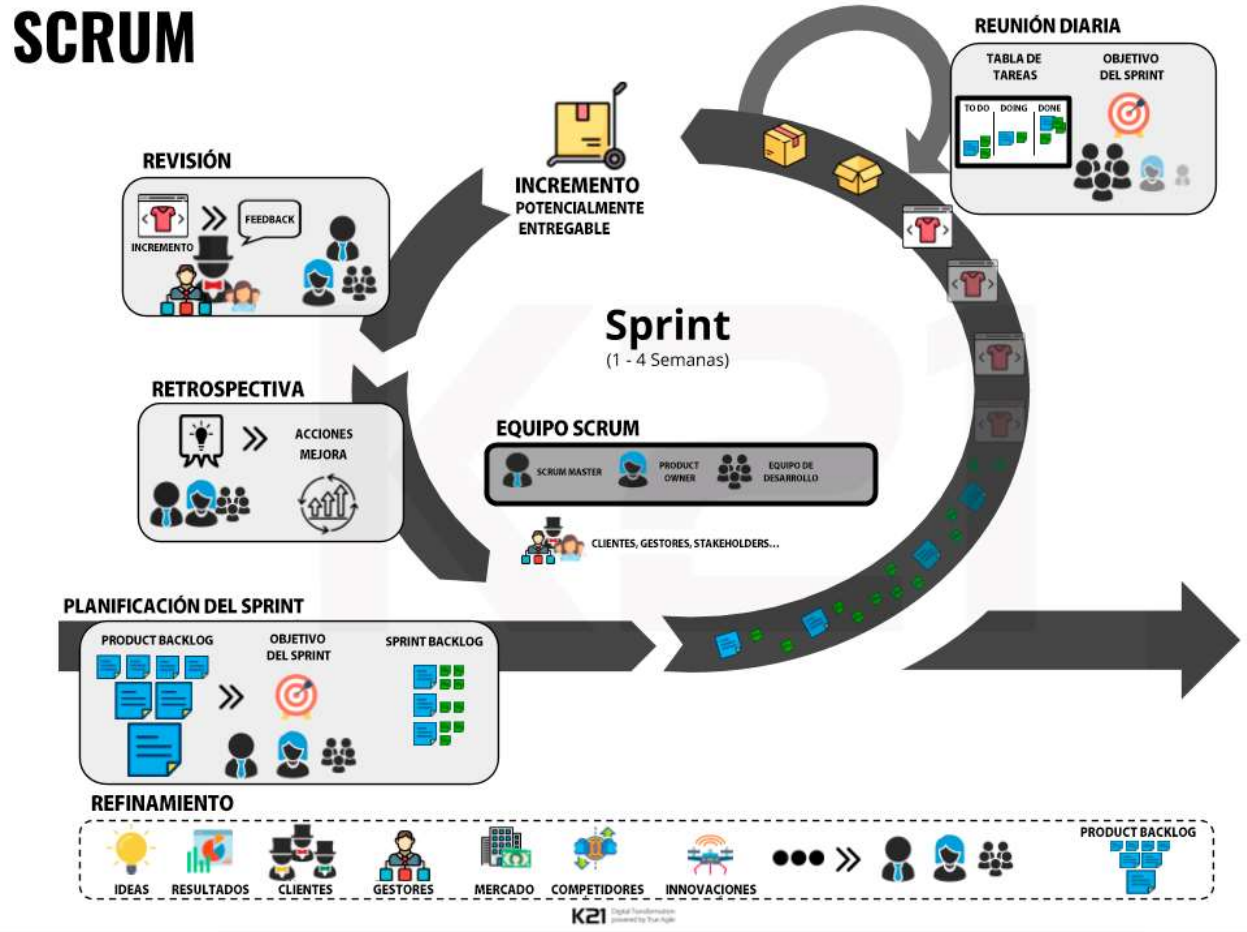
6.2.3.1. Mejora Continua (KAIZEN). La mejora continua es uno de los pilares fundamentales de la filosofía Lean y fundamenta el enfoque sistémico de la misma. Su influencia es tan amplia que prácticamente no hay empresas de ningún sector que no lo implemente de forma directa o indirecta en la actualidad.

En el ámbito del desarrollo de software, en especial dentro del LSD, Poppendieck (2003) propuso la utilización de los eventos Kaizen para la solución de problemas específicos. Estos eventos consisten en reunir a un equipo compuesto por representantes de diversas áreas involucradas en el problema crítico y trabajar intensamente durante no más de una semana para realizar los cambios necesarios en los procesos y resolver el problema en cuestión. Una vez concluido el evento Kaizen, el equipo se disuelve y los miembros regresan a sus actividades habituales, mientras que el proceso modificado queda implementado.

En metodologías ágiles el comportamiento en general no resulta muy diferente, están organizados ciclos de trabajo con periodicidad preestablecida en el cual el cumplimiento de tareas y metas es claro (Gaete et al. 2021). Posterior a este, se dan otros ciclos marcados a partir de los resultados obtenidos. Por ejemplo, en Scrum, uno de los Frameworks más empleados actualmente, estos ciclos son llamados “Sprints” (Gaete et al. 2021), o en Lean Start up, cada ciclo es considerada una iteración (Zorzzeti et al. 2022).

Figura 10.

Ciclo de mejora continua en "Sprint" en Framework "Scrum"



Tomado de: “scrum” (K21, 2019).

Algunas particularidades de la industria brindan espacios positivos para la aplicación efectiva de Kaizen. Estos pueden ser la obtención de feedback en el código o en la propia ejecución del software, facilitando la posibilidad de toma de decisiones o de iteración (Sanz, 2015). Asimismo, es posible proponer métodos para llevar a cabo pruebas de una nueva funcionalidad (o una modificación) en un entorno de producción con el objetivo de evaluar su eficacia, recibiendo una retroalimentación casi de manera instantánea (Anand et al., 2019).

También, para lograr un alto rendimiento dentro del desarrollo de Software, es importante fomentar la experimentación constante y tener una comprensión general del producto y la organización relacionada con él. De tal forma, más allá de las herramientas empleadas en "personas y equipos" es necesaria la proactividad de todos los grupos de interés del propio desarrollo, participando activamente en el proceso de mejora continua. Igualmente, se pueden proponer sistemas que faciliten la realización de pruebas en el ambiente de producción de un conjunto de funcionalidades nuevas o modificadas para evaluar su eficacia y obtener retroalimentación en un corto plazo (Zimmermann, 2017).

En términos generales, se pude decir que toda la naturaleza iterativa en métodos ágiles en el desarrollo de software corresponde a una adopción del principio de mejora continua del Lean Manufacturing. Debido a la naturaleza de las operaciones en el desarrollo de Software cada reunión periódica o cierre de ciclo en el Kaizen tradicional corresponde a una iteración en el flujo de trabajo habitual del LSD o de cualquier metodología ágil (Anand et al., 2019).

Figura 11.

Ciclos iterativos de mejora continua.

Nota. Los procesos iterativos e incrementales que implican pruebas unitarias continuas y entregas frecuentes son una característica del marco de trabajo conocido como Programación Extrema (XP). Tomado de Wells, 2023.

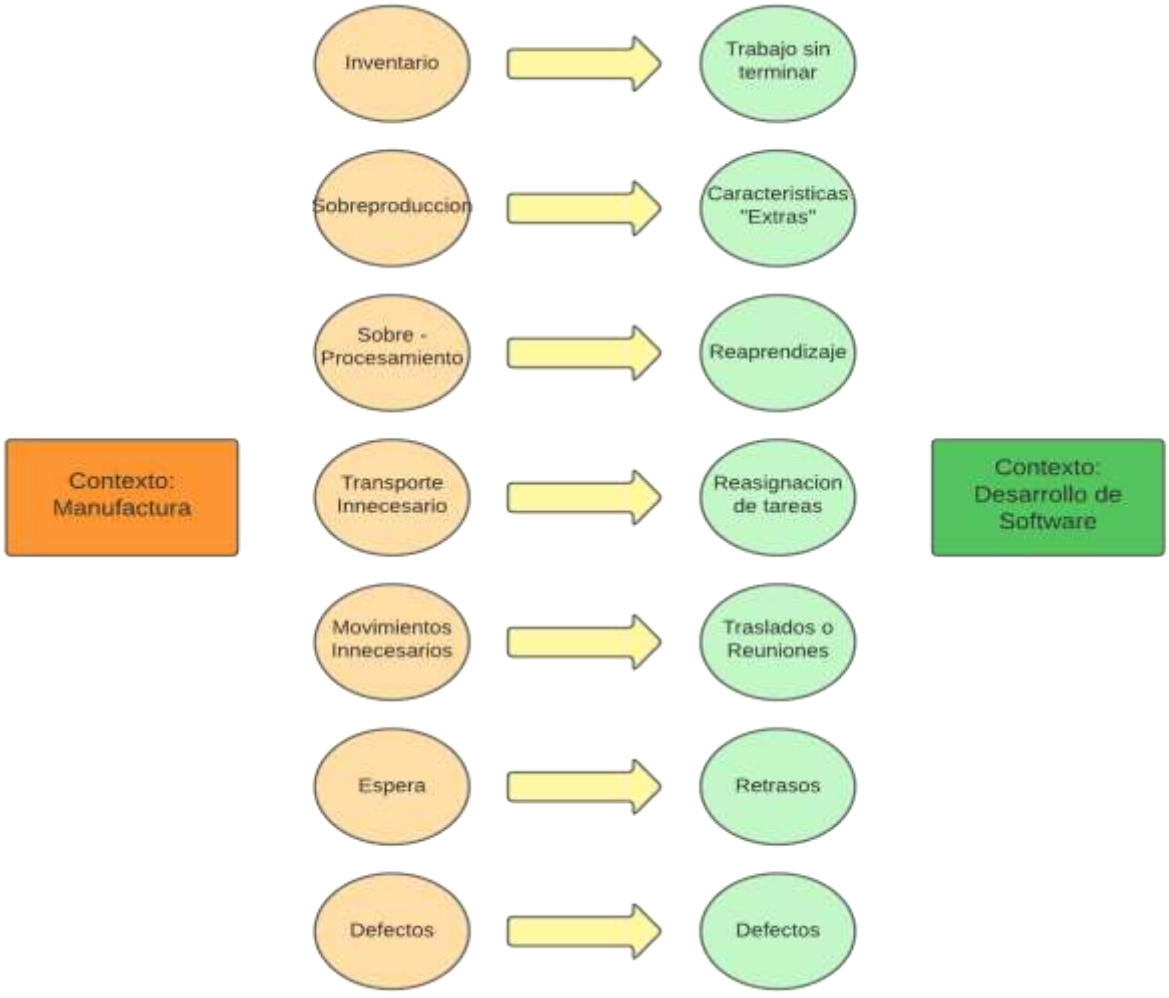
6.2.3.2. Reducción de Desperdicios. Otro de los componentes fundamentales de la filosofía Lean son la reducción de desperdicios. Su propuesta e implementación generó una

disrupción en todo tipo de empresas durante el siglo XX y aún hoy sus principios son fundamentales para la gestión de todo tipo de empresas y organizaciones a nivel global (Hibbs, et al., 2009). Debido a los componentes diferenciadores de cada sector, la reducción de desperdicios expuestas en la filosofía Lean clásica tiene su equivalente en el LSD, y en general, a empresas que trabajen bajo metodologías ágiles (Fátima et al, 2020).

La siguiente figura muestra esta equivalencia:

Figura 12.

Desperdicios del Lean en desarrollo de Software.



Fuente: Elaboración propia a partir de Sanz (2015).

6.2.3.2.1. Trabajo sin Terminar. En el contexto de Lean Manufacturing, el trabajo incompleto es una forma de desperdicio muy preocupante, ya que el trabajo no se puede considerar terminado hasta que se haya implementado en producción. Si el trabajo no se completa, existe el riesgo de perder la inversión realizada. En esta industria puede presentarse en documentación sin codificar, código sin consolidar, código sin probar, código sin documentar, código sin desplegar o entrega parcial de software que no puede considerarse unificada (Sanz, 2015).

6.2.3.2.2. Características "Extras". Cada línea de código que se escribe debe ser necesaria y considerar no solo el costo de su producción, sino también el costo de su mantenimiento, como posibles errores y complejidad general del código. Gastar recursos en frameworks para lograr eficiencias en la producción a gran escala puede resultar en una pérdida de inversión si no se utilizan lo suficiente para compensar el costo invertido (Fátima et al, 2020). Las adquisiciones de terceros, como la compra de librerías para la visualización gráfica, o desarrollos internos, también pueden ser un desperdicio si no se reutilizan en otros proyectos. El modelo de desarrollo en cascada puede aumentar el riesgo de implementar requisitos que nunca se utilizarán, ya que los ingenieros de software deben pensar a largo plazo más allá de las necesidades inmediatas del proyecto (Thomas, 2018).

6.2.3.2.3. Reaprendizaje. Es importante evitar procesos innecesarios en el desarrollo de software, como la creación de documentos que no se utilizarán. Una gestión inadecuada del conocimiento puede llevar a la repetición de trabajo y a la "reinvención de la rueda". Hay varios factores que pueden aumentar el costo de este reaprendizaje, como el código sin documentar, pues es importante crear una documentación precisa y necesaria durante el desarrollo del software para evitar incurrir en costos adicionales de aprendizaje en futuras modificaciones; la planificación deficiente de tareas, si no se tiene en cuenta el conocimiento previo de cada miembro del equipo

al asignar tareas, puede ocurrir que se deba reaprender lo que otro compañero ya ha hecho, lo que supone un costo adicional (Thomas, 2018). Por ejemplo, si se asigna a un programador A una tarea previamente codificada por un programador B, ambos deben aprender qué hizo el otro para poder avanzar en su trabajo. Aunque en ciertas situaciones, como en partes críticas de una aplicación, esta práctica puede ser beneficiosa, es importante realizarla bajo supervisión especial; la calidad deficiente, ya que en caso de que un error en el código llegue al entorno de producción, podría resultar en un desafío para encontrar su origen y podría llevar a tener que reaprender, incluso si el desarrollador que trabaja en la corrección es el mismo que originalmente escribió el código con el error; las tareas en paralelo o el cambio excesivo de tareas conllevan un costo adicional; y la comunicación o gestión deficiente del conocimiento, aunque es cierto que la gestión del conocimiento puede presentar desafíos, cada vez existen más soluciones y herramientas disponibles para facilitar esta tarea. Por ejemplo, hay sistemas para compartir el conocimiento, herramientas de búsqueda de información más avanzadas y wikis que pueden ayudar a los miembros del equipo a colaborar y compartir información de manera más efectiva (Sanz, 2015).

Capturar el conocimiento en una organización puede ser complicado debido a las diferencias culturales en la forma en que se ve el conocimiento. En la cultura occidental, el conocimiento se considera algo que se puede y se debe escribir, mientras que en la cultura oriental se valora más el conocimiento tácito, que se adquiere a través de la experiencia y no del estudio, y que no es fácil de almacenar o procesar por un ordenador. Esto hace que sea difícil formalizar y transmitir este conocimiento a otros, lo que puede llevar a una gestión deficiente del conocimiento y a un mayor costo de reaprendizaje en el desarrollo de software (Theunissen at al, 2022).

6.2.3.2.4. Reasignación de Tareas. Cada vez que el código cambia de manos, existe un riesgo de coste adicional debido a la transferencia de conocimiento. Asignar personas a múltiples proyectos o tareas al mismo tiempo también genera desperdicios, ya que se necesita tiempo para concentrarse en una tarea antes de comenzar a trabajar en ella y puede haber interrupciones en el trabajo (Salleh at al., 2019). Lo más eficiente es abordar las tareas secuencialmente en lugar de en paralelo. Además, es importante evitar reasignaciones de tareas incompletas, ya que esto provoca una pérdida de tiempo y conocimiento. Si no se puede evitar, se debe fomentar la comunicación que facilite la transmisión del conocimiento tácito. También es importante documentar bien los requisitos para evitar llamadas de soporte del cliente. Hay algunas consideraciones que se deben tener en cuenta para reducir el impacto de los cambios de asignación, además de minimizarlos en la medida de lo posible (Sanz, 2015).

6.2.3.2.5. Traslados o reuniones. Se incluyen en la categoría de movimientos de los miembros del equipo acciones como la resolución de dudas y la realización de reuniones. Sin embargo, también se deben tener en cuenta como movimientos la complicación que puede surgir al tratar de ubicar ciertos elementos esenciales para llevar a cabo el trabajo, como documentos, secciones de código y bibliotecas, entre otros (Sanz, 2015).

6.2.3.2.6. Retrasos. El desperdicio de tiempo es uno de los mayores problemas en el desarrollo de software y conduce a una disminución del plazo disponible para entregar el valor al cliente. Para reducir este tipo de pérdidas, es importante posponer las decisiones hasta que se tenga la certeza de que son correctas y aportarán valor al cliente (Sanz, 2015).

6.2.3.2.7. Defectos. El impacto de los desperdicios en el desarrollo de software está relacionado con la gravedad del desperdicio y el tiempo que se tarde en detectarlo. Si se detecta temprano, el impacto será menor. Para evitar estos desperdicios, se recomienda probar

inmediatamente, integrar el código con frecuencia y actualizar el sistema en producción lo antes posible. Además, se hace referencia a la utilización de VSM (Value Stream Mapping), que permite identificar las actividades que atraviesa el producto de software y detectar las causas raíz de los desperdicios. Este enfoque se abordará con más detalle en una sección dedicada a tal efecto (Theunissen at al, 2022).

6.2.3.3. Heijunka. El término Heijunka, que se refiere a la nivelación de la producción, también se aplica en la industria del software en el contexto de las metodologías ágiles. Estas metodologías priorizan la polivalencia de los miembros del equipo para llevar a cabo diversas tareas, lo que facilita la nivelación de la producción de software (Salleh at al., 2019). Además, en un estudio se propone la extrapolación del método de gestión de almacenes para proyectos de soporte y mantenimiento en la industria del software, los cuales no siempre se abordan con las metodologías ágiles habituales. Esta solución permite nivelar la producción y aumentar el rendimiento global. La implementación del sistema pull del almacén para la ejecución del trabajo, la gestión de la carga de trabajo a través de una sola cola, el control autónomo de la carga de trabajo con un enfoque visual y el procesamiento secuencial de las solicitudes mediante una única cola son algunas de las prácticas utilizadas en este proceso (Takagi et al, 2007).

6.2.3.4. Modelos de Madurez. Conocidos como CMM por sus siglas en inglés (Maturity Models), se trata de un conjunto de prácticas recomendadas organizadas por capacidades críticas de negocio, diseñado para mejorar el rendimiento. Estas capacidades críticas se centran en los desafíos principales que enfrentan las organizaciones (Miñana, 2020).

Por otra parte, es vital la estandarización y estabilidad de los procesos involucrados en el desarrollo de software. Análogo a los modelos de madurez de diversas áreas empresariales (logística, calidad, implementación digital, entre otros) dentro de la industria del software se

emplean los “Capability Maturity Model Integration**” o** CMMI. Con estos, se puede mejorar la aproximación ágil mediante la institucionalización, dotando de una mayor disciplina al uso de las prácticas ágiles (Miñana, 2020).

Si bien estos corresponden más a la propia gestión empresarial que al desarrollo de Software vale la pena mencionarlos pues garantiza la estabilidad requerida en los procesos. Estos modelos de madurez están conformados por herramientas que permiten evaluar y mejorar sistemáticamente habilidades, capacidades o competencias para alcanzar un objetivo, y permiten guiar a la organización en la implementación de buenas prácticas con el fin de alcanzar los objetivos deseados. En el desarrollo de software se emplean los “Capability Maturity Model Integration CMMI” (Sanz, 2015).

Las metas habituales dentro de los CMMI para garantizar prácticas eficientes en metodologías ágiles incluyen la creación y mantenimiento de políticas organizacionales, la capacitación del personal y la recopilación de resultados para mejorar de forma continua (Sanz, 2015).

6.2.3.5. Gestión Visual. La gestión visual es cada vez más frecuente en el ámbito de la Ingeniería del Software. Esto corresponde a las características propias de las funciones operativas en del desarrollo de Software, donde la creación de código, diseño de componentes y procesos asociados al propio desarrollo de producto suelen ser altamente especializados y meticulosos (Dingsoyr y Moe, 2014).

Si bien no se puede pensar en una relación únicamente dependiente del Lean clásico en la aplicación de gestión visual en esta industria, vale la pena listarla junto a las otras herramientas por su incidencia diaria y especializada. En todo caso, de una u otra forma la gestión visual y las

herramientas de esta índole del Lean son componente central en la gestión de equipos actual de cualquier metodología ágil y están orientadas a la reducción de desperdicios de tiempo y en la generación de valor final al cliente (Sanz, 2015).

En los últimos años, la aplicación de la gestión visual en el desarrollo de software ha sido posible gracias a la disponibilidad de herramientas digitales que permiten crear tableros virtuales, gráficos y diagramas en tiempo real y compartirlos con todo el equipo. Una de las herramientas más utilizadas en este ámbito es Kanban, una técnica visual que utiliza tarjetas para representar las tareas y procesos, lo que permite visualizar el flujo de trabajo y mejorar la gestión del tiempo y recursos.

Además, la gestión visual se ha extendido a otras áreas del desarrollo de software, como la gestión de proyectos, el seguimiento de errores y el control de calidad. Por ejemplo, las herramientas de seguimiento de errores, como JIRA o Trello, utilizan tableros visuales para mostrar el estado de los errores y su prioridad, lo que permite a los equipos identificar rápidamente los problemas y solucionarlos de manera eficiente (Dingsoyr y Moe, 2014).

La gestión visual también ha demostrado ser útil en el desarrollo de software ágil, ya que permite a los equipos autoorganizados y multidisciplinarios trabajar de manera más eficiente y colaborativa. Por ejemplo, las técnicas de visualización, como el "Daily Stand-up" y el "Sprint Review", permiten a los equipos revisar el estado del proyecto y los objetivos del sprint en tiempo real, lo que mejora la transparencia y la comunicación (Serrador y Pinto, 2015).

6.2.3.6. Visual Streaming Map (VSM). Posiblemente contenida dentro de la “Gestión Visual” el VSM es una herramienta clásica del Lean que permite a las empresas visualizar el flujo de trabajo de un proceso, identificar los cuellos de botella, los desperdicios y las actividades que no agregan valor, lo que facilita la mejora continua. En el desarrollo de software, el VSM ayuda a

visualizar y comprender los procesos complejos que involucran múltiples equipos y recursos, lo que permite identificar rápidamente los cuellos de botella y las actividades que ralentizan el proceso (Zaheer, 2020).

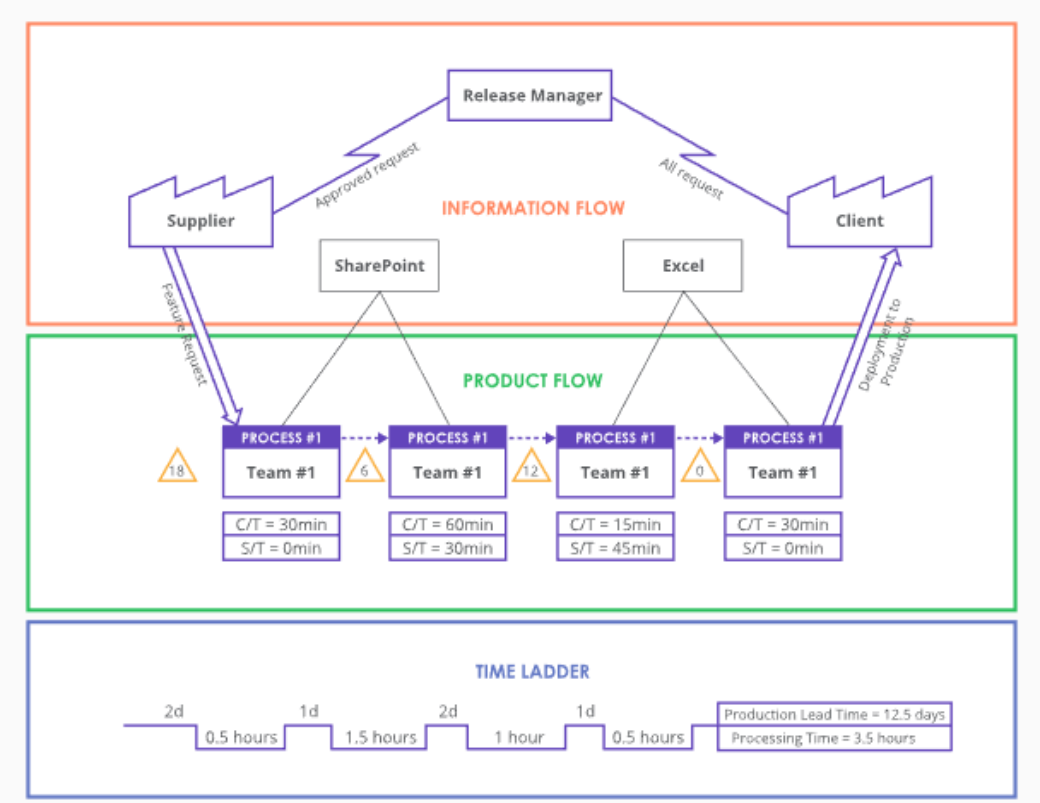
Según Amin et al. (2017), la aplicación del VSM en el desarrollo de software ayuda a identificar las actividades que no agregan valor y a eliminarlas, lo que reduce el tiempo de ciclo y aumenta la eficiencia del proceso. Además, el VSM ayuda a mejorar la calidad del software al permitir a los equipos identificar y corregir los errores de manera temprana en el proceso de desarrollo.

Aunque con variaciones con el original, esta herramienta se adaptó fácilmente al flujo de productos en el desarrollo de software. Esta herramienta estructura el ciclo de vida del proceso de desarrollo comenzando con inputs claros (ideas de los clientes, soporte, análisis de la competencia) y como outputs el servicio o producto final (cliente final). Durante el proceso, las necesidades de valor del cliente son la motivación que guía el proceso (Zaheer, 2020).

La idea central de VSM es optimizar todo el proceso de desarrollo. De modo que utilice la menor cantidad de recursos (costo, tiempo y humanos) para satisfacer las necesidades de valor del cliente. El objetivo es entregar un producto de alta calidad al cliente. De tal forma, mapear los despilfarros son vitales en el VSM y su identificación adecuada permite a la empresa alcanzar este objetivo lo más rápido posible. De tal forma, VSM constituye una excelente herramienta para determinar cuánto tiempo llevará entregar valor y cuál sería la calidad asociada con la creación de valor (Edvantis, 2020).

Figura 13.

Ejemplo de VSM en Software Development



Nota: Ejemplo de VSM en Software Development. Tomado de: VSM definition (Edvantis, 2020).

Utilizando el VSM, se pueden identificar los cuellos de botella y los desperdicios en el proceso de prueba, lo que permitió reducir el tiempo de prueba y mejorar la calidad del software entregado (Fatima et al, 2020).

La adopción del VSM en el desarrollo de software ha demostrado ser una herramienta valiosa para mejorar la eficiencia y eficacia de los procesos de trabajo. La herramienta ayuda a visualizar y comprender los procesos complejos, identificar los cuellos de botella y las actividades que no agregan valor, lo que facilita la mejora continua y mejora la calidad del software entregado.

6.2.3.7. Kanban. El Kanban, si bien es más una herramienta en el Lean Manufacturing que un componente propio de la filosofía, merece una mención especial por su acogida e influencia en la industria del Software. De hecho, un Framework de trabajo bajo metodología ágil posee su nombre y basa su desarrollo en principios en las actividades de la herramienta clásica.

La relevancia del Kanban en el desarrollo de software se basa en su capacidad para visualizar el flujo de trabajo y limitar el trabajo en progreso (WIP, por sus siglas en inglés), lo que ayuda a identificar cuellos de botella y mejorar el tiempo de entrega del proyecto. Además, el Kanban promueve la colaboración y el trabajo en equipo, ya que cada miembro del equipo es responsable de su parte en el proceso y tiene la capacidad de identificar problemas y proponer soluciones (Ahmad et al., 2018). La metodología Kanban en el desarrollo de software se basa en la visualización del flujo de trabajo mediante un tablero Kanban. Este tablero muestra el trabajo que se encuentra en diferentes etapas del proceso de desarrollo de software, como "en espera", "en progreso" o "terminado". Los miembros del equipo pueden mover las tarjetas o notas que representan las tareas a medida que se realizan, lo que proporciona una visión general en tiempo real del progreso del proyecto (Rasmusson, 2010).

La adopción del Kanban en la industria del software ha demostrado ser efectiva para mejorar la eficiencia y la calidad del trabajo, así como para reducir el tiempo de entrega del proyecto. Además, el Kanban fomenta la colaboración y el trabajo en equipo, lo que puede aumentar la motivación y la satisfacción del equipo. Por lo tanto, es una herramienta valiosa para cualquier equipo de desarrollo de software que busque mejorar su proceso de trabajo.

Actualmente, el objetivo principal de la metodología Kanban es limitar el trabajo en progreso para mejorar la calidad del trabajo, reducir el tiempo de ciclo y maximizar la eficiencia

del equipo. El trabajo se visualiza en un tablero Kanban que muestra el estado actual del proceso de desarrollo, y se utiliza un sistema de tarjetas para gestionar y controlar el flujo de trabajo (Kniberg, 2011).

En contraste con el Lean Software Development, que se centra en la eliminación de desperdicios, la metodología Kanban se enfoca en la optimización del flujo de trabajo. La metodología Kanban permite a los equipos adaptarse y evolucionar a medida que avanzan en el proceso de desarrollo, lo que permite una mayor flexibilidad y una mejor capacidad de respuesta a los cambios (Rasmusson, 2010).

Con respecto al framework Scrum, Kanban se diferencia en su enfoque más flexible y menos prescriptivo. Aunque ambos enfoques se centran en la mejora continua y la optimización del flujo de trabajo, Kanban no tiene roles específicos, como el Scrum Master o el Product Owner, y no prescribe un conjunto fijo de eventos como Sprint Planning o Sprint Review (Kniberg, 2009).

6.2.3.8. Just In Time. Just In Time (JIT) Es una estrategia para reducir el desperdicio y los costos en el proceso de desarrollo de software. Esta herramienta implica la entrega de las tareas necesarias en el momento adecuado, evitando la sobrecarga de trabajo y mejorando la calidad del producto (Sanz, 2015).

Esta herramienta es fundamental para todas las metodologías de desarrollo de software con componentes iterativos y, en general, para todos los procesos de desarrollo de software actuales. Permite que en lugar de trabajar en grandes bloques de código y luego probar todo al final del proyecto, los equipos de desarrollo trabajan en pequeñas iteraciones, produciendo pequeñas partes de código en cada iteración (Larman y Basili, 2003). Esto les permite hacer pruebas continuas y hacer ajustes sobre la marcha, lo que significa que se pueden hacer correcciones rápidamente antes de que los problemas se conviertan en problemas más grandes y costosos. De tal forma cumplen

con los requerimientos a tiempo y se vela continuamente por garantizar el mejoramiento continuo durante cada fase del proceso.

Tambien, JIT se aplica en el desarrollo de software mediante el uso de herramientas y metodologías que permiten a los desarrolladores tener acceso rápido y fácil a los recursos y herramientas que necesitan para trabajar en sus tareas, como control de versiones, herramientas de integración continua, automatización de pruebas y sistemas de gestión de proyectos (Popli y Lal, 2016). Esto permite a los desarrolladores trabajar de manera más eficiente, reduciendo los tiempos de espera y mejorando la calidad del software producido (Tegegne et al., 2019).

De hecho, la implicación de estas herramientas es tan elevado que es complicado pensar en el propio desarrollo de la tarea sin el uso de la propia herramienta. Es el caso, por ejemplo, del uso de técnicas de compilación JIT en lenguajes de programación como Java y C# (Kovács, 2018). Esto permite que el código fuente se compile a código de máquina en tiempo de ejecución, lo que mejora el rendimiento del software y reduce los tiempos de carga. Realizar esta tarea sin dicha técnica de compilación es impensable actualmente (Tenev y Kuipers, 2018).

6.2.3.9. Jidoka. La idea de Jidoka hace referencia a la automatización con un toque humano o con inteligencia. En el contexto del desarrollo de software, Jidoka se enfoca en mejorar la calidad del software y disminuir los errores. Para lograr esto, se emplean sistemas y procesos que permiten la detección temprana de errores y su eliminación antes de que se propaguen en el sistema. Para lograr esto se utilizan técnicas de prueba automatizadas y se automatizan procesos, como la integración y la entrega continuas. De esta manera, se pueden detectar y solucionar problemas desde etapas tempranas del proceso de desarrollo, resultando en un software de mayor calidad y más fiable (Cohn, 2017; Aplyca, 2022).

Otra forma en que se aplica Jidoka en la industria del software es a través del monitoreo y la alerta temprana. Los sistemas de monitoreo permiten detectar problemas y errores en el software en tiempo real, lo que permite a los desarrolladores solucionarlos antes de que afecten a los usuarios finales. Las alertas tempranas permiten a los desarrolladores tomar medidas preventivas y corregir problemas antes de que causen mayores daños.

Otros de los muchos empleos de Jidoka en la industria del Software para mejorar la calidad y la eficiencia de los procesos incluyen (Liker, & Hoseus, 2008):

- Automatización de pruebas: La automatización de pruebas es una técnica que se utiliza para detectar errores de software de manera temprana en el ciclo de vida del software. Con Jidoka, las pruebas automatizadas se ejecutan de forma continua y automática, lo que permite detectar y corregir errores de manera oportuna y evitar que se propaguen a otras áreas del sistema.

- Detección de errores en tiempo real: Jidoka también se puede aplicar en la detección de errores en tiempo real durante la ejecución de los procesos de software. Esto implica la configuración de reglas de detección que permiten a los sistemas de software detectar automáticamente errores y alertar al equipo de desarrollo.

- Mejora continua: Jidoka se basa en el principio de mejora continua, lo que significa que se deben identificar y solucionar problemas de manera constante. En la industria del software, esto se puede lograr mediante la recopilación de datos sobre los procesos de desarrollo de software, la identificación de áreas problemáticas y la implementación de mejoras continuas.

- Diseño para la calidad: Jidoka también puede aplicarse en el diseño de software para la calidad. Esto implica diseñar el software con características que minimicen la posibilidad de errores, incluyendo características como el modularidad, la reutilización y la simplicidad.

6.2.3.10. Poka Joke. Poka Yoke se ha convertido en una herramienta valiosa en el desarrollo de software, al ayudar a prevenir errores y mejorar la calidad del producto final. Su aplicación se ha extendido a través de la implementación de pruebas automatizadas, formularios y plantillas estandarizados, y herramientas de análisis estático de código (Lazarevic et al, 2019).

Poka Yoke se refiere a la implementación de dispositivos o sistemas que detectan y corrigen automáticamente errores en el proceso de producción, antes de que se conviertan en defectos en el producto final. En el contexto del desarrollo de software, esto puede implicar la implementación de controles y validaciones automatizadas para evitar errores comunes y mejorar la calidad del software (Vinod, et. al, 2017).

Un caso de cómo se ha aplicado el método Poka Yoke en el desarrollo de software es a través de la implementación de formularios y plantillas estandarizados. Estos documentos guían a los desarrolladores en la creación de código y en la documentación de este, lo que reduce la posibilidad de errores y asegura que se sigan los estándares de calidad establecidos por la organización (Vinod, et. al, 2017).

Otro caso relevante de cómo se ha adoptado el método Poka Yoke en el desarrollo de software es en la implementación de herramientas de análisis estático de código. Estas herramientas analizan el código fuente de una aplicación en busca de errores comunes, como

variables no inicializadas o condiciones de carrera. Los desarrolladores pueden corregir estos errores antes de que se conviertan en defectos y afecten la calidad del software (Lazarevic, 2019).

6.2.3.11. 5S's. Lean 5s es tambien una herramienta tradicional del "Lean" clásico. Permite garantizar condiciones idóneas en los procesos y mantener el puesto de trabajo limpio y ordenado. A continuación, se presenta en resumen como cada una de las 5 fases pueden ser enlazadas en empresas de desarrollo de software (Sanz, 2015).

6.2.3.11.1. Seiri (Separar). La fase de Seiri (Separar) consiste en distinguir entre lo necesario y lo innecesario, establecer prioridades a corto y medio plazo y eliminar lo que no es esencial. En el contexto de la industria del software, hay elementos que se pueden considerar innecesarios, como la documentación física o en línea que no se utiliza, el código fuente en ramas que ya no se necesitan, el código que no es ejecutable o comentado, los comentarios desfasados, los procesos, servidores y scripts que ya no se utilizan, los requisitos que ya no son necesarios y los PBIs obsoletos. Al eliminar estos elementos, se puede reducir el mantenimiento y disminuir los errores en el proceso de desarrollo del software.

6.2.3.11.2. Seiton (Ordenar). Según el Lean clásico, el orden repercute directamente en la disminución del tiempo de alistamientos y de búsqueda. También facilita detectar si falta algún elemento necesario.

Esta fase atañe directamente a puestos de trabajo y a espacios virtuales dentro de ordenadores involucrados en el mismo diseño de software. Esto incluye desde operaciones tan sencillas como poner accesos directos en el escritorio de los ordenadores, hasta gestionar adecuadamente ERP's, softwares de diseño colaborativos y equipos de trabajo interdisciplinares y complejos (Thomas, 2018).

La jerarquización de tareas tambien es fundamental en esta fase. Vale la pena tener en cuenta que el orden puede ser relativo a una posición, donde se estiman las tareas urgentes, o a una cantidad, cuantas tareas pueden estar en ejecución a la vez.

Además, se aconsejan ilustraciones visuales, teniendo como referentes el Work In Progress WIP del Kanban, el Value streaming Map VSM, entre otras herramientas visuales del Lean ya mencionadas.

6.2.3.11.3. Seiso (Limpieza). Una fase importante en el proceso de desarrollo de software es la limpieza del entorno, que busca detectar cualquier problema antes de que cause errores. Para lograr esto, es necesario mantener un entorno limpio en todo momento y aplicar principios como los del "Clean Code". Algunas actividades comunes en esta fase incluyen el uso de nombres significativos, la eliminación de comentarios innecesarios y la creación de clases y métodos pequeños (Thomas, 2018). En cuanto a los sitios donde se aplica Seiso, se incluyen los directorios y archivos temporales, las pruebas excesivas, las capturas de pantalla y las versiones antiguas. Además, se pueden desarrollar pruebas automáticas para prevenir defectos en el futuro (Sanz, 2015).

6.2.3.11.4. Seiketsu (Estandarizar). Es esencial para mantener las fases anteriores. En esta línea, algunos de los controles y ejercicios de estandarización comúnmente utilizados en marcos ágiles incluyen la definición de una cobertura mínima de código, acuerdos sobre el código estático (como líneas máximas y complejidad ciclomática), el formato de la documentación y las capturas de pantalla, y la nomenclatura (Sanz, 2015).

6.2.3.11.5. Shitsuke (Disciplina). Corresponde a las actividades necesarias para garantizar el mantenimiento en el tiempo de las fases previas y de los beneficios obtenidos para dar cumplimiento a cabalidad del desarrollo del software (Sanz, 2015).

Se suelen realizar auditorías, acciones correctivas y sesiones de toma de decisión. Estas últimas corresponden usualmente a las mismas reuniones al finalizar cada sprint o ciclo iterativo. Vale la pena recordar que los Frameworks ágiles basan su naturaleza en esta capacidad de pivotear de acuerdo a los resultados obtenidos, y Shitsuke es fundamental para comprender de qué manera se debe realizar esta transición en cada iteración (Thomas, 2018).).

6.3. Impacto de la Aplicación de las Herramientas Lean como Estrategia de Mejoramiento en el Desarrollo de Software.

La implementación de herramientas Lean en el desarrollo de software ha tenido un impacto significativo en la industria, principalmente mejorando la calidad del software, empujando la competitividad de la industria, reduciendo los tiempos de entrega y disminuyendo los costos. Sin lugar a dudas la implementación de diversas herramientas y de la propia filosofía Lean (tanto de forma directa en el LSD como implícita en todas las metodologías de tipo “ágil”) han generado impactos positivos en las empresas desarrolladoras de software que las empleen (Alahyari, 2019). Dentro de la revisión realizada se encuentran algunos ejemplos notables:

Como se mencionó anteriormente, una de las herramientas Lean más utilizadas en el desarrollo de software es Kanban. Dentro de la literatura revisada se encuentran aportes significativos de la aplicación de esta herramienta Lean:

- Un ejemplo de la adopción del Kanban en la industria del software es el caso de la empresa Spotify. Spotify ha utilizado Kanban para gestionar el flujo de trabajo en su equipo de desarrollo de software, lo que les ha permitido aumentar la eficiencia y la calidad del trabajo, así como reducir el tiempo de entrega del proyecto (Kniberg y Ivarsson, 2010).

- Otro ejemplo es el caso de la empresa de consultoría Agile42, que ha utilizado Kanban para mejorar la eficiencia en el desarrollo de software en varios proyectos. Utilizando Kanban, Agile42 pudo identificar y solucionar cuellos de botella en el proceso de desarrollo de software y mejorar la productividad del equipo (Ahmad et al., 2018).

- En otro caso, se utilizó la herramienta Kanban para visualizar y gestionar el flujo de trabajo en el departamento de desarrollo. Como resultado, se redujo el tiempo de entrega de los proyectos en un 50%, se mejoró la calidad del software y se redujeron los costos de producción en un 30%." (Poppendieck, 2003,).

- Otro ejemplo se encuentra en la empresa de software brasileña “ThoughtWorks”. La empresa decidió implementar Kanban para mejorar su flujo de trabajo y la calidad del software que entregaba a sus clientes. El resultado fue una reducción del tiempo de entrega en un 20%, una disminución del 30% en el número de errores de software y una mayor satisfacción del cliente (Burrows, 2014).

Es importante destacar que la implementación de herramientas y principios Lean en la industria del software es muy amplia y diversa, y que la elección de enfatizar en los casos de Kanban y mejora continua no sugiere que sean las más utilizadas o relevantes, sino más bien que hay una mayor cantidad de literatura disponible en relación con estos casos concretos (Ahmadzai y Bakhsh, 2022; Tomas, 2018). No obstante, es esencial tener en cuenta que la mayoría de los artículos revisados indican que estas herramientas no se utilizan de forma aislada, sino que se integran en el marco de metodologías ágiles donde los principios, herramientas y actividades derivados del Lean forman parte de la gestión estratégica y operativa diaria de las empresas. De

hecho, se observa una clara tendencia en la industria hacia la aplicación de prácticas Lean en la gestión de proyectos de software, donde se enfatiza la importancia de la mejora continua, la eliminación de desperdicios y la eficiencia en la entrega de valor al cliente (Perkusich et al., 2020; Alahyari et al., Thomas, 2018).

Finalmente, vale la pena mencionar otros casos encontrados tanto en la literatura científica como en la búsqueda web realizada:

En un estudio realizado por Klimczak et al. (2020), se encontró que la implementación de herramientas Lean resultó en una reducción del 53% en los errores encontrados en el código y una mejora del 45% en la satisfacción del cliente.

En otro caso, un equipo de desarrollo de software en una empresa de servicios financieros aplicó las prácticas Lean para reducir el tiempo que tardaban en entregar software a sus clientes. “Utilizando una combinación de Kanban, integración continua y pruebas automatizadas, el equipo logró reducir el tiempo de entrega de software de 6 semanas a solo 2 semanas." (Anderson & Anderson, 2010, p. 6).

En una empresa de tecnología de la información, se implementó la herramienta Value Stream Mapping (VSM) para identificar los cuellos de botella y los desperdicios en el proceso de desarrollo de software. Como resultado, se eliminaron los cuellos de botella, se redujeron los tiempos de espera y se mejoró la eficiencia del proceso en un 25%. (Rother, 2009, p. 124).

Vale la pena mencionar algunos casos de empresas actualmente reconocidas:

IBM: La empresa de tecnología IBM utiliza herramientas Lean como el mapeo de flujo de valor y el enfoque en el cliente para mejorar su proceso de desarrollo de software. El mapeo de flujo de valor les permitió a los ingenieros de IBM identificar oportunidades de mejora en el proceso de desarrollo de software y eliminar actividades que no agregaban valor. El enfoque en el cliente les permitió a los equipos de IBM enfocarse en las necesidades del cliente y entregar un producto de alta calidad (Harzl, 2017).

Intel: La empresa de tecnología Intel utiliza herramientas Lean como el enfoque en el cliente y la mejora continua para mejorar su proceso de desarrollo de software. El enfoque en el cliente les permitió a los equipos de Intel enfocarse en las necesidades del cliente y entregar un producto que satisficiera sus necesidades. La mejora continua les permitió a los equipos de Intel identificar oportunidades de mejora en el proceso de desarrollo de software y realizar cambios incrementales para resolver problemas (Shahabuddin y Yalla, 2017).

A la vista de los resultados, se puede afirmar que las herramientas Lean han tenido un impacto altamente positivo en la industria del desarrollo de software. A través de la implementación de prácticas Lean, las empresas han logrado mejorar la eficiencia, la calidad del producto, reducir los tiempos de entrega y aumentar la satisfacción del cliente, así como disparar el desarrollo de la misma industria. Además, se ha demostrado que estas herramientas son aplicables a diferentes áreas de la empresa, desde la gestión de proyectos hasta la misma producción del software. Las empresas que han adoptado estas prácticas se han vuelto más competitivas en el mercado actual y han logrado mantenerse a la vanguardia en la industria. De tal forma, hoy se puede afirmar que la adopción de herramientas Lean es esencial para la

supervivencia y el éxito de las empresas de software en el entorno empresarial actual altamente competitivo.

7. Difusión de los resultados de la investigación

Con el fin de compartir los resultados de la presente investigación, se desarrolla un artículo de carácter publicable en el que se incluyen los principales resultados de la presente investigación. Este artículo se encuentra en el Apéndice A.

8. Conclusiones

Los resultados obtenidos muestran que las herramientas Lean se han adaptado de manera efectiva a las necesidades específicas de la industria del software, permitiendo la creación de procesos más eficientes y la mejora de la calidad del producto final. Se destaca el enfoque del Lean Software Development de Poppendieck y el manifiesto ágil como componentes fundamentales en la adaptación de las herramientas Lean al desarrollo de software y en la creación de procesos ágiles y eficientes en esta industria.

Se pudo identificar 13 componentes y herramientas del Lean clásico que son clave en la industria del software y que se integran en la ejecución de las metodologías ágiles. Si bien existen otras herramientas, estas son las más destacadas y han demostrado ser efectivas en la mejora de la eficiencia y calidad en el desarrollo de software. Entre estas herramientas se encuentran los principios del Lean, la mejora continua, la reducción de desperdicios, Kanban, gestión visual, Visual Streaming Map, Jidoka, Just in time, Poka Yoke, 5S, Heijunka, personas y equipos de trabajo, y modelos de madurez CMM. Estas herramientas han sido adaptadas de manera exitosa a la industria del software y se aplican en diferentes áreas, desde la gestión de proyectos hasta la producción del software.

La frecuencia de uso de las herramientas Lean en la industria del software no se refleja de forma lineal y directa, debido a que la implementación de estas herramientas no se realiza de manera separada y aislada, sino que se integran en metodologías más amplias y holísticas que combinan diversos aspectos del Lean, como en el caso de Lean Software Development y las metodologías ágiles anteriormente mencionadas. Sin embargo, La clasificación ABCD propuesta permite brindar una idea sobre la importancia y el empleo de cada una de estas.

La implementación de estas herramientas y principios Lean en la industria del software es muy amplia y diversa, por tanto, no se puede afirmar cuales de estas son las más empleadas. Sin embargo, es esencial tener en cuenta que la mayoría de los artículos revisados indican que estas herramientas no se utilizan de forma aislada, sino que se integran en el marco de metodologías ágiles donde los principios, herramientas y actividades derivados del Lean forman parte de la gestión estratégica y operativa diaria de las empresas. Mención aparte tiene la herramienta Kanban que ha girado a ser un tipo de metodología de trabajo ágil y que destaca por su evolución, importancia y cantidad de menciones en la literatura científica. Ocurre algo similar con el concepto de mejora continua, que está implícito actualmente en todas las metodologías de trabajo ágil y en general a todo proceso iterativo involucrado en desarrollo de software.

Las herramientas Lean han tenido un impacto altamente positivo en la industria del desarrollo de software. A través de la implementación de prácticas Lean, las empresas han logrado mejorar la eficiencia, la calidad del producto, reducir los tiempos de entrega y aumentar la satisfacción del cliente, así como disparar el desarrollo de la misma industria. Además, se ha demostrado que estas herramientas son aplicables a diferentes áreas de la empresa, desde la gestión de proyectos hasta la misma producción del software. Las empresas que han adoptado estas prácticas se han vuelto más competitivas en el mercado actual y han logrado mantenerse a la vanguardia en la industria. Hoy se puede afirmar que la adopción de herramientas Lean es esencial para la supervivencia y el éxito de las empresas de software en el entorno empresarial actual.

9. Recomendaciones

Es importante que las empresas del sector de desarrollo de software sigan adoptando y aplicando herramientas herederas del Lean Manufacturing en su cadena de valor, ya que estas han demostrado ser altamente efectivas en la mejora de los procesos y la eliminación de desperdicios, lo que puede conducir a una mayor eficiencia y rentabilidad.

Se recomienda el uso de metodologías ágiles en el desarrollo de software en las empresas nacionales del sector, ya que estas se basan en principios Lean y pueden ayudar a las empresas a reducir el tiempo de desarrollo, aumentar la satisfacción del cliente y mejorar la calidad del software entregado. Además, son el estándar en la generación de software actualmente a nivel global.

Es fundamental que las empresas del sector de desarrollo de software sigan investigando y experimentando con la aplicación de herramientas y principios Lean Manufacturing en su cadena de valor, con el fin de adaptarlas a las necesidades específicas del sector y maximizar su efectividad.

Se sugiere que se utilicen técnicas de minería de datos para el análisis de la información obtenida en esta búsqueda, con el fin de identificar patrones y tendencias en el uso de estas herramientas en el tiempo. También, sería positivo desarrollar análisis comparativos entre las metodologías ágiles existentes y Lean Software Development, para determinar sus similitudes y diferencias, y evaluar cuál es más efectiva en términos de calidad y eficiencia en el desarrollo de software en diferentes casos de aplicación.

Referencias Bibliográficas

Agile Manifiesto. (2022). Manifesto for Agile Software Development. Agilemanifesto.org. https://agilemanifesto.org/

Ahmad, M. O., Dennehy, D., Conboy, K., & Oivo, M. (2018). Kanban in software engineering: A systematic mapping study. Journal of Systems and Software, 137, 96-113

Ahmadzai, S., & Bakhsh, M. (2022). An empirical investigation on lean method usage: Issues and challenges in afghanistan doi:10.1007/978-3-030-90618-4_13 Retrieved from www.scopus.com

Alahyari, H., Berntsson Svensson, R., & Gorschek, T. (2017). A study of value in agile software development organizations. Journal of Systems and Software, 125, 271-288. doi:10.1016/j.jss.2016.12.007

Alahyari, H., Gorschek, T., & Berntsson Svensson, R. (2019). An exploratory study of waste in software development organizations using agile or lean approaches: A multiple case study at 14 organizations. Information and Software Technology, 105, 78-94. doi:10.1016/j.infsof.2018.08.006

Álvarez, M. A., & Suárez, G. (2022). Estudio exploratorio sobre la implementación de técnicas de Lean Manufacturing en procesos logísticos [Exploratory study on the implementation of Lean Manufacturing techniques in logistics processes]. (Trabajo de Grado para Optar al Título de Ingeniero Industrial). Universidad Industrial de Santander, Bucaramanga.

Anand, A., Kaur, J., Singh, O., & Alhazmi, O. H. (2021). Optimal sprint length determination for agile-based software development. Computers, Materials and Continua, 68(3), 3693-3712. doi:10.32604/cmc.2021.017461

Anderson, D. J. (2010). Kanban: Successful evolutionary change for your technology business. Blue Hole Press.

Anghel, I. I., Călin, R. Ș., Nedelea, M. L., Stănică, I. C., Tudose, C., & Boiangiu, C. A. (2022). SOFTWARE DEVELOPMENT METHODOLOGIES: A COMPARATIVE ANALYSIS. UPB Scientific Bulletin, Series C: Electrical Engineering and Computer Science, 84(3), 45-58. Retrieved from www.scopus.com

Aplyca (2022). https://www.aplyca.com/blog/integracion-continua-y-entrega-continua-cicd

Arias, K. Y., & Sandoval, C. A. (2022). Estudio exploratorio sobre la implementación de técnicas de Lean Manufacturing en el sector de la salud (Lean Healthcare) [Exploratory study on the implementation of Lean Manufacturing techniques in the healthcare sector (Lean Healthcare)]. (Trabajo de Grado para Optar el título de Ingeniero Industrial). Universidad Industrial de Santander, Bucaramanga.

Asociación Colombiana de Ingenieros de Sistemas ACIS (2021). Disponible en: https://acis.org.co/portal/content/proceso-de-desarrollo-de-software-en-colombia-%E2%80%93-2021

BBVA. (2022). “Agile” vs “Lean”: ¿cuál es la diferencia? BBVA NOTICIAS; BBVA. https://www.bbva.com/es/agile-vs-lean-cual-es-la-diferencia/

Burrows, M. (2014). Kanban from the Inside. Sequim, WA, USA: Blue Hole Press.

Cardona, N. & Prada, J. A. (2022). Estudio exploratorio sobre la implementación de técnicas de Lean Manufacturing en el sector agropecuario [Exploratory study on the implementation of Lean Manufacturing techniques in the agricultural sector]. (Trabajo de grado para optar al título de Ingeniero Industrial). Universidad Industrial de Santander, Bucaramanga.

Caudillo, J. (2006). Diplomado en Seis Sigma: VI (Lean Seis Sigma).

Charette, R. N. (1999). The competitive edge of risk entrepreneurs. IT professional, 1(4), 69-73.

Cohn, M. (2017). Agile estimating and planning. Pearson Education.

De Rojas, A. (2012). Lean Manufacturing aplicado al desarrollo de software: En busca de la eficiencia en el mundo IT. CLAVEI.

DTT(2023). CMMI. Disponible en: https://www2.deloitte.com/es/es/pages/technology/articles/que-es-cmmi-capability-maturity-model-integration.html

DTT. (2017). ¿Qué es Scrum? Deloitte Spain. https://www2.deloitte.com/es/es/pages/technology/articles/que-es-scrum.html

Edvantis(2020). VSM definition. Disponible en: https://www.edvantis.com/blog/vsm-definition/

El colombiano (2022). Desarrollo de software no hay desempleo, faltan profesionales. Disponible en: https://www.elcolombiano.com/negocios/en-desarrollo-de-software-no-hay-desempleo-faltan-profesionales-HM17826166

Fatima, N., Nazir, S., & Chuprat, S. (2020). Knowledge sharing framework for modern code review to diminish software engineering waste. International Journal of Advanced Computer Science and Applications, 11(6), 442-450. doi:10.14569/IJACSA.2020.0110656

Fedesoft (2022). Fedesoft. Disponible en: https://fedesoft.org/

Gaete, J., Villarroel, R., Figueroa, I., Cornide-Reyes, H., & Muñoz, R. (2021). Agile application approach with scrum, lean and kanban. [Enfoque de aplicación ágil con Scrum, Lean y Kanban] Ingeniare, 29(1), 141-157. doi:10.4067/S0718-33052021000100141

Gómez, M. F. (2014). Lean Manufacturing En Español: Cómo eliminar desperdicios e incrementar ganancias. Estados Unidos de América: Editorial Imagen.

Harzl, A. (2017). Can FOSS projects benefit from integrating kanban: A case study. Journal of Internet Services and Applications, 8(1) doi:10.1186/s13174-017-0058-z

Hernández J., & Vizán, A. (2013). Lean Manufacturing Conceptos, técnicas e implantación. Madrid: Fundación EOI.

Hibbs, C., Jewett, S. y Sullivan, M. (2009). The Art of Lean Software Development: A Practical and Incremental Approach Theory in practice. Editor"O'Reilly Media, Inc.".ISBN 0596554435, 9780596554439.

IBM. (2020). ¿Qué es el desarrollo de software? | IBM. Ibm.com. https://www.ibm.com/co-es/topics/software-development

K21 (2019). Qué es Scrum. Disponible en: https://k21.global/es/blog/que-es-el-scrum

Kalenda, M., Hyna, P., & Rossi, B. (2018). Scaling agile in large organizations: Practices, challenges, and success factors. Journal of Software: Evolution and Process, 30(10) doi:10.1002/smr.1954

Kitchenham, B. (2004). Procedures for performing systematic reviews. Keele, UK, Keele University, 33(2004), 1-26.

Klimczak, K., Krawiec, A., & Kowalczyk, R. (2020). The impact of Lean Management on software quality: A case study. Information and Software Technology, 122, 106274. https://doi.org/10.1016/j.infsof.2020.106274

Kniberg, H. (2009). Kanban and Scrum - Making the Most of Both. C4Media Inc.

Kniberg, H. (2011). Lean from the trenches: Managing large-scale projects with Kanban. Lean from the Trenches, 1-178.

Kniberg, H., & Ivarsson, A. (2010). Scaling agile @ Spotify. Retrieved from https://dl.acm.org/doi/pdf/10.1145/2018556.2018558

Kovács, G. (2018). Novel supply chain concepts and optimization of virtual enterprises to reduce cost, increase productivity and boost competitiveness. Bulletin of the Polish Academy of Sciences: Technical Sciences, 66(6), 973-980. doi:10.24425/bpas.2018.125945

La República (2022). La industria del software representa alrededor de US100 Millones en Colombia. Disponible en: https://www.larepublica.co/internet-economy/la-industria-del-software-representa-alrededor-de-us-10-000-millones-en-colombia-3330546

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments: a brief history. IEEE Computer, 36(6), 47-56.

Lazarevic, M., Mandic, J., Sremcev, N., Vukelic, D., & Debevec, M. (2019). A systematic literature review of Poka-Yoke and novel approach to theoretical aspects. Journal of Mechanical Engineering, 65(7-8), 454-467.

Lehtinen, T. O. A., Itkonen, J., & Lassenius, C. (2017). Recurring opinions or productive improvements—what agile teams actually discuss in retrospectives. Empirical Software Engineering, 22(5), 2409-2452. doi:10.1007/s10664-016-9464-2

Liker, J. K. (2021). Toyota way: 14 management principles from the world's greatest manufacturer. McGraw-Hill Education.

Liker, J. K., & Choi, T. Y. (2004). Building deep supplier relationships. Harvard business review, 82(12), 104-113.

Melegati, J., Guerra, E., & Wang, X. (2021). Understanding hypotheses engineering in software startups through a gray literature review. Information and Software Technology, 133 doi:10.1016/j.infsof.2020.106465

Mendes Calo, K., Estévez, E. C., & Fillottrani, P. R. (2009). Un framework para evaluación de metodologías ágiles. In XV Congreso Argentino de Ciencias de la Computación.

Milić, M., Vlajić, S., Antović, I., Savić, D., Stanojević, V., & Lazarević, S. (2017). Software quality standards and lean approach in teaching and learning programming. International Journal of Engineering Education, 33(4), 1345-1360. Retrieved from www.scopus.com

Miñana, R. (2020). *¿Qué es CMMI? | Deloitte España*. Deloitte Spain. https://www2.deloitte.com/es/es/pages/technology/articles/que-es-cmmi-capability-maturity-model-integration.html

Moreno, M. A. (2010). Filosofía Lean aplicada a la Ingeniería del Software. España: Universidad de Sevilla. Recuperado de http://bibing. us. es/proyectos/abreproy/70201/fichero/02+-+ Ingenieria+ del+ Software. pdf.

Morien, R. (2005,). Agile management and the Toyota way for software project management. In INDIN'05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005. (pp. 516-522). IEEE.

Morien, R. I. (2018). Pedagogical agility and agile methodologies in computer system development education. Int. J. Adv. Intell. Paradigms, 11(1/2), 19-32.

Muñoz, D. (2009). Administración de operaciones. Enfoque de administración de procesos de negocios. México: CENGAGE learning.

Perkusich, M., e Silva, L. C., Costa, A., Ramos, F., Saraiva, R., Freire, A., ... & Perkusich, A. (2020). Intelligent software engineering in the context of agile software development: A systematic literature review. Information and Software Technology, 119, 106241.

Poppendieck, M., & Poppendieck, T. (2003). Lean software development: An agile toolkit. Addison-Wesley Professional.

Porter, M. E. (2002). Ventaja Competitiva. Creación y Sostenimiento de un Desempeño Superior. (2da Edición) México. Compañía Editorial Continental, S.A. DE C.V

Rasmusson, J. (2010). The agile samurai: How agile masters deliver great software. The Agile Samurai, 1-264.

Ries, E., & Sałbut, B. (2012). El método Lean Startup.

Rother, M. (2009). Toyota Kata: Managing people for improvement, adaptiveness, and superior results. McGraw Hill Professional.

Salleh, N. M., & Nohuddin, P. N. E. (2019). Comparative study between lean six sigma and lean-agile for quality software requirement. International Journal of Advanced Computer Science and Applications, 10(12), 212-218. doi:10.14569/ijacsa.2019.0101230

Sanz, M. I. (2015). Metodología LEAN para el desarrollo de software. Ejemplo práctico de aplicación en empresa de desarrollo de software.

Scrum Alliance (s.f). Scrum. Disponible en: http://www.scrumaliance.org

Serrador, P., & Pinto, J. K. (2015). Does Agile work?—A quantitative analysis of agile project success. International Journal of Project Management, 33(5), 1040-1051.

Shahabuddin, S. M., & Yalla, P. (2017). Impact of lean software development into agile process model with integration testing prior to unit testing. Journal of Theoretical and Applied Information Technology, 95(22), 6163-6175. Retrieved from www.scopus.com

Socconini, L. (2019). Lean manufacturing. Paso a paso. Marge books.

Gómez, L. V. (2019). Lean Manufacturing paso a paso. Valencia, Barcelona: Marge Books.

Socconini, L. (2019). Lean manufacturing. Paso a paso. Marge books.

Takagi, V. K. F. V. T., Sakata, V. A., & Okayama, V. D. (2007). Innovation in software development process by introducing Toyota Production System. Fujitsu sci. Tech. J, 43(1), 139-150.

Tecnologías de Información (s.f.) Método Lean. Disponible en: https://www.tecnologias-informacion.com/metodo-lean.html

Tegegne, E. W., Seppänen, P., & Ahmad, M. O. (2019). Software development methodologies and practices in start-ups. IET Software, 13(6), 497-509. doi:10.1049/iet-sen.2018.5270

Tenev, T., & Kuipers, B. (2018). Just-in-time compilation in Java: a survey. Journal of Systems and Software, 140, 1-15.

Theunissen, T., van Heesch, U., & Avgeriou, P. (2022). A mapping study on documentation in continuous software development. Information and Software Technology, 142 doi: 10.1016/j.infsof.2021.106733

Thomas, A. (2018). Developing an integrated quality network for lean operations systems. Business Process Management Journal, 24(6), 1367-1380. doi:10.1108/BPMJ-02-2018-0041

Tranfield, D., Denyer, D. and Smart, P. (2003), Towards a Methodology for Developing Evidence-Informed Management Knowledge by Means of Systematic Review. British Journal of Management, 14: 207-222. https://doi.org/10.1111/1467-8551.00375

Universitat Carlemany (2022). Metodologías de desarrollo de software. Disponible en: https://www.universitatcarlemany.com/actualidad/metodologias-de-desarrollo-de-software

Vinod, M., Devadasan, S. R., Sunil, D. T., Thilak, V. M. M., & Murugesh, R. (2017). POYSS: a model for integrating Poka-Yoke technique with Six Sigma concept. International Journal of Productivity and Quality Management, 22(2), 223-242.

Wells, D. (2023). *Introducing Extreme Programming*. Extremeprogramming.org. http://www.extremeprogramming.org/introduction.html

Zaheer, S., Amjad, M. S., Rafique, M. Z., & Khan, M. A. (2020). A K-chart based implementation framework to attain lean & agile manufacturing. International Journal of Production Management and Engineering, 8(2), 123-135. doi:10.4995/ijpme.2020.12935

Zimmermann, O. (2017). Microservices tenets: Agile approach to service development and deployment. Computer Science - Research and Development, 32(3-4), 301-310. doi:10.1007/s00450-016-0337-0

Zorzetti, M., Signoretti, I., Salerno, L., Marczak, S., & Bastos, R. (2022). Improving agile software development using user-centered design and lean startup. Information and Software Technology, 141 Doi: 10.1016/j.infsof.2021.106718