Revista UIS Ingenierías
Página de la revista: https://revistas.uis.edu.co/index.php/revistauisingenierias

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

Exploratory study on the implementation of Lean Manufacturing techniques in Software development (Lean Software Development).

Rubio Ardila Jhon Edward.

[1]Grupo de investigación OPALO, Facultad de Ingenierías Físico Mecánicas. Escuela de Estudios Industriales y empresariales. Universidad Industrial de Santander. Colombia.

Resumen

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.

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

Abstract

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.

Keywords: Lean Software Development, Lean Manufacturing, Software Design.

ISSN impreso: 1657 - 4583. ISSN en línea: 2145 – 8456,

1. 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). 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 (Gómez L. V., 2019). Para llevarlo a cabo, se pretende realizar una completa revisión de literatura que recapitule y analice 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.

2. 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 1 resume las fases de la revisión de literatura realizada.

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.

Como objetivos planteados y como fases para cumplir cada uno de estos se plantea:

- **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.

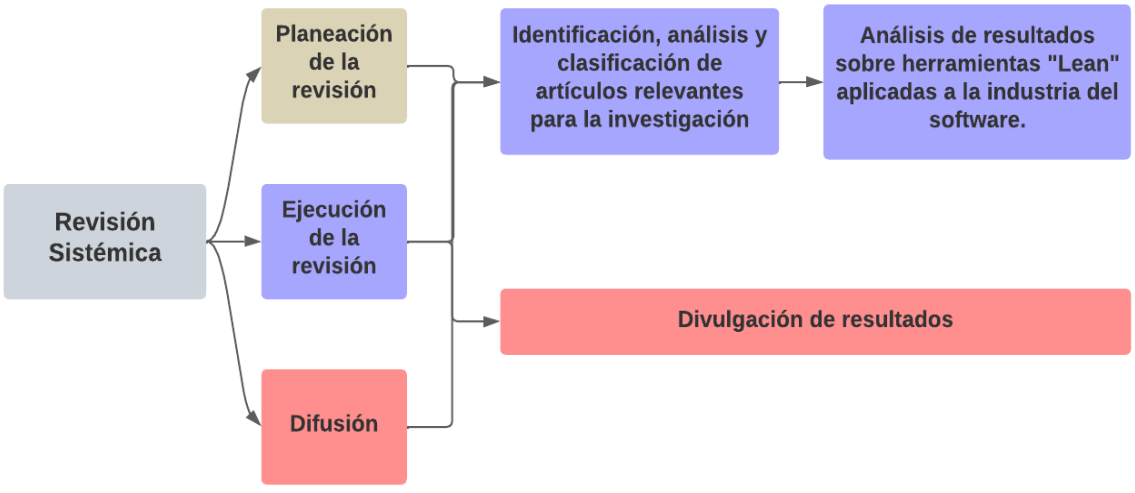

Figura 1.*Metodología de la investigación.*
Nota. Adaptado de: “Towards a Methodology for Developing Evidence-Informed Management Knowledge by Means of Systematic Review”; Tranfield, Denyer and Smart (2003).

- **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.

- **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.

3. Resultados

3.1. 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.

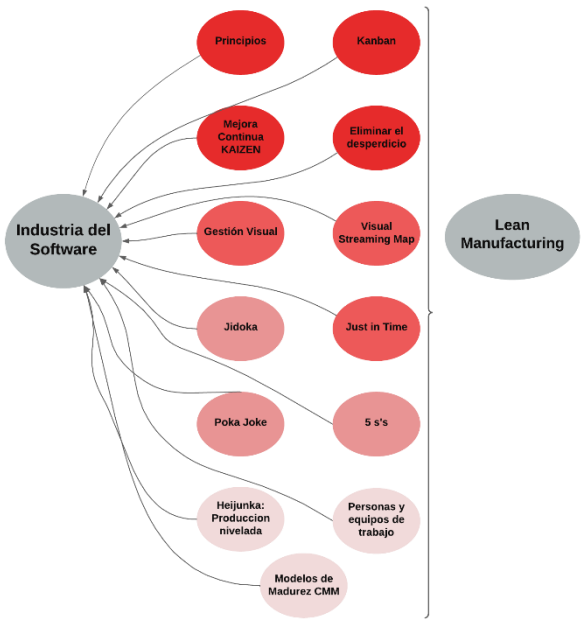

Figura 2. *Componentes "Lean" Adoptados por la Industria del Software*

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 con 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 1
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, encontramos 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 con los resultados de la revisión. Además, es difícil determinar con exactitud cuántas veces se ha utilizado cada herramienta, ya que se basa en una revisión de literatura. 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 presenta 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.

3.2. 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.

3.2.1. 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.

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.

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 con 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.

3.2.2. 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.

3.2.2.1. 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.

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.

3.2.2.2. 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 (2004) 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.

3.2.3. Adopción de Herramientas del Lean Clásico en la Industria del Software

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

Cómo 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:

3.2.3.1. 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).

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).

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 3. *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.

3.2.3.2. 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:

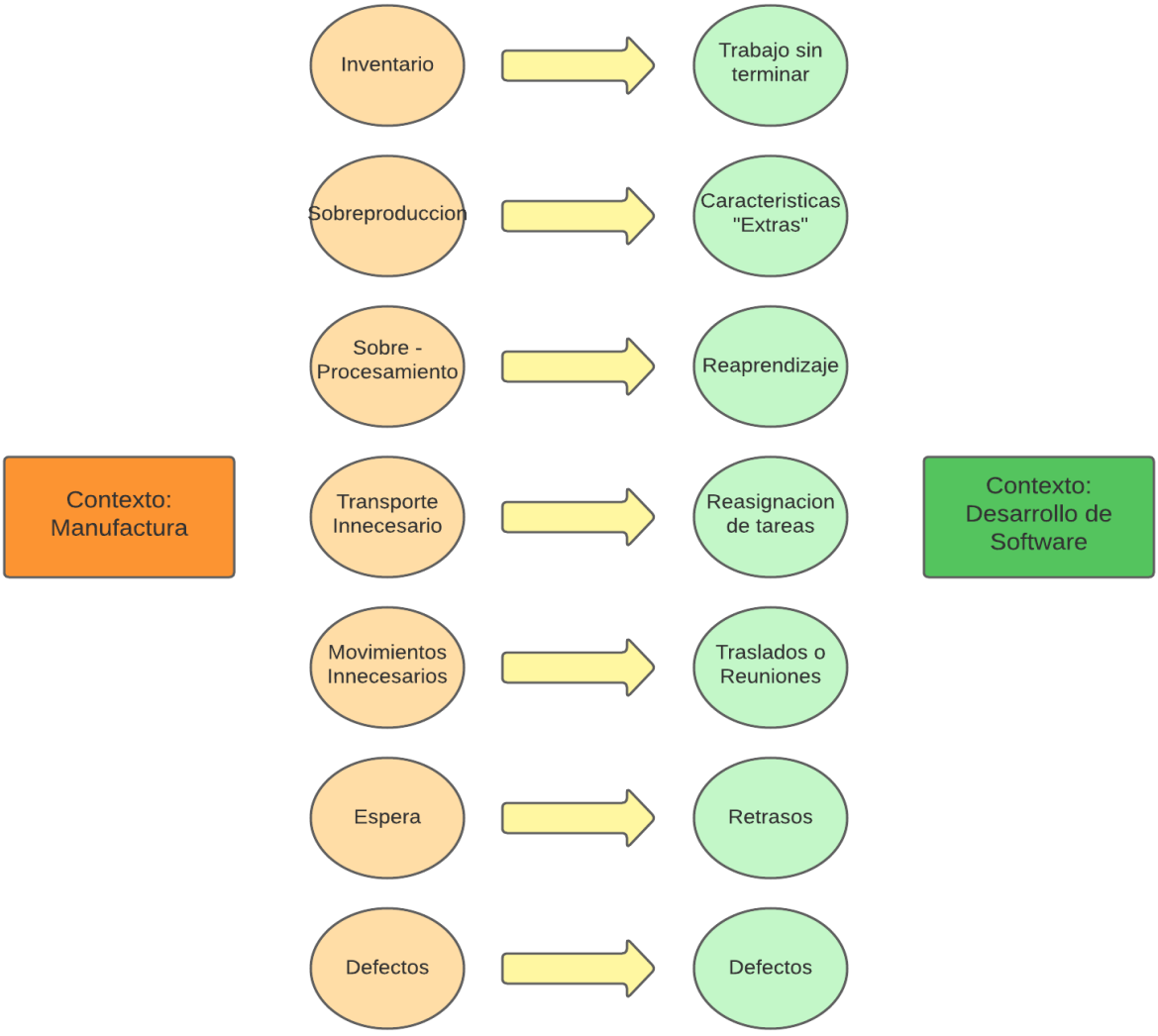

Figura 4.*Desperdicios del Lean en desarrollo de Software.*

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).

• **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

• **Trabajo sin Terminar**. En el contexto de Lean Manufacturing, el trabajo sin terminar es una fuente de desperdicio especialmente preocupante, ya que un trabajo no se considera completo hasta que se encuentra en producción. Si un trabajo no se completa, puede resultar en la pérdida de la inversión realizada. En esta industria puede presentarse en documentación sin

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).

- **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).

- **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).

- **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).

- **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).

- **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.

3.2.3.3. 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).

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).

3.2.3.4. 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)

3.2.3.5. 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).

3.2.3.6. 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

3.2.3.7. 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.

3.2.3.8. 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).

3.2.3.9. 5S's. Lean 5s es también 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).

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.

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.

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).

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).

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 con los resultados obtenidos, y Shitsuke es fundamental para comprender de qué manera se debe realizar esta transición en cada iteración (Thomas, 2018).

3.2.4. 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 duda 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.

3.3. 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.

4. 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.

5. Referencias

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). Integración continua y Entrega continua. Aplyca Tecnología SAS. 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