Universidad del Valle de Guatemala

Ingeniería de Software 1

Prof. Erick Marroquín

Tarea Investigativa 3:

Principios de Lean Software Development

Juan Diego Solís Martínez - 23720

Nils Muralles Morales - 23727

Víctor Manuel Pérez Chávez - 23731

Diego Oswaldo Flores Rivas - 23714

Isabella Recinos Rodríguez- 23003

Guatemala, 30 de marzo de 2025

Introducción

En la actualidad, el desarrollo de software requiere metodologías que permitan la entrega rápida y eficiente de productos de alta calidad. En este contexto, el Lean Software Development (LSD) ha emergido como una metodología ágil basada en los principios del Lean Manufacturing, desarrollado por Toyota. Su enfoque se centra en optimizar los procesos de desarrollo, reducir desperdicios y mejorar la colaboración entre equipos para ofrecer valor continuo a los clientes. Este trabajo de investigación explora los fundamentos del Lean Software Development, destacando sus siete principios clave y las herramientas utilizadas para su implementación. Además, se analiza su aplicación en distintos entornos y su impacto en la mejora de la eficiencia del desarrollo de software.

Resumen

El Lean Software Development (LSD) es una metodología ágil que busca optimizar el flujo de trabajo en el desarrollo de software, minimizando desperdicios y asegurando la entrega rápida de productos funcionales. Basado en los principios del Lean Manufacturing, LSD se rige por siete principios fundamentales: eliminar desperdicios, amplificar el aprendizaje, tomar decisiones tardías, entregar lo antes posible, potenciar el equipo, crear la integridad y visualizar todo el conjunto. Para implementar estos principios, se utilizan herramientas como Value Stream Mapping (VSM) y diagramas de Ishikawa para identificar y eliminar desperdicios. El desarrollo basado en pruebas y la integración continua para mejorar el aprendizaje y la calidad del software, y el desarrollo basado en conjuntos para tomar decisiones informadas. Asimismo, se promueve una cultura de mejora continua y colaboración entre equipos para garantizar un producto final eficiente y de alto valor para los clientes. LSD es especialmente útil en proyectos con requerimientos cambiantes, equipos de tamaño reducido y organizaciones que buscan mejorar la eficiencia en el desarrollo de software. Su enfoque flexible y basado en la retroalimentación continua lo convierte en una metodología clave para empresas que desean optimizar sus procesos y ofrecer soluciones de calidad de manera ágil.

Conceptos principales

El Lean Software Development (LSD) es una metodología ágil derivada de los principios del Lean Manufacturing, originalmente desarrollado por Toyota. Se enfoca en optimizar el flujo de trabajo, minimizar el desperdicio y entregar software al cliente de manera continua y eficiente.

Este se basa en siete principios clave, diseñados para mejorar el desarrollo de software. Dentro de ellos, promueve entregar lo más rápido posible, priorizando entregas frecuentes de software funcional para que los clientes puedan evaluar el producto y dar retroalimentación temprana. La calidad es un aspecto central, por lo que se insiste en construir un buen producto desde el inicio, integrando buenas prácticas como pruebas automatizadas y revisiones de código durante todo el proceso (Herranz, 2025).

Para implementar el LSD, se pueden seguir prácticas como el uso de Kanban para visualizar el flujo de trabajo y reducir desperdicios, la implementación de entrega continua para obtener retroalimentación rápida, la adopción de pruebas automatizadas para garantizar la calidad desde el principio, y fomentar una cultura de mejora continua con revisiones frecuentes. También es fundamental involucrar a los clientes en el proceso para asegurarse de que el software entregue valor real.

LSD es ideal en proyectos con requerimientos cambiantes, donde la flexibilidad es clave. También es útil en equipos pequeños o medianos que necesitan eficiencia y agilidad, en empresas emergentes que buscan validar rápidamente sus ideas con entregas incrementales, y en organizaciones que desean reducir costos y mejorar el flujo de desarrollo. Su enfoque lo convierte en una excelente opción para equipos que buscan desarrollar software de alta calidad de manera ágil y eficiente (Herranz, 2025).

Principios de Lean SW Development

1. Eliminar desperdicios: Es identificar y eliminar todo aquello que no le aporta valor al cliente. Para poder aplicarlo se debe evitar la generación de código que ofrezca funciones no deseadas o innecesarias. Reducir los retrasos en el proceso de desarrollo, mejorar la toma de requisitos para evitar malentendidos, optimizar la comunicación interna y evitar la documentación excesiva o mal estructurada.

2. Amplificar el aprendizaje: Es fomentar el aprendizaje continuo entre los miembros del equipo de desarrollo. Para aplicarlo se debe promover la formación continua en nuevas tecnologías, facilitar el intercambio de conocimiento entre compañeros y estar al día con las tendencias tecnológicas para adaptarse a los cambios rápidos.

3. Tomar decisiones tardías: Es retrasar las decisiones hasta que se tenga la mayor cantidad de información posible para adaptarse a cambios en los requisitos o necesidades del cliente. Para aplicarlo se deben utilizar historias de usuario en lugar de requisitos rígidos para reflejar las necesidades reales. Además, se debe esperar a que estas estén claramente definidas antes de comenzar el desarrollo.

4. Entregar lo antes posible: Es realizar entregas frecuentes de software que incluyan funcionalidades alineadas con las necesidades más importantes de los usuarios. Para aplicarlo se debe planificar entregas basadas según su prioridad y valor para el usuario. Además, se deben implementar cortos ciclos de desarrollo en los que se pueda realizar feedback rápidamente.

5. Potenciar el equipo: Es involucrar a los desarrolladores en la toma de decisiones, tales como la estimación del tiempo, priorización de las tareas y otros puntos clave. Se puede aplicar fomentando la participación activa del equipo y creando un ambiente de confianza en donde se valore la opinión de cada miembro.

6. Crear la integridad: Es garantizar que el software sea fácil de mantener, mejorar y reutilizar a través de prácticas de integración continuas y pruebas automatizadas. Para poder aplicarlo se pueden implementar sistemas de integración continua que incluyan pruebas automatizadas, realizas pruebas de usabilidad para asegurar que los usuarios tengan una buena experiencia y evitar la acumulación de código innecesario.

7. Visualizar todo el conjunto: Es adoptar una visión general del proyecto para entender cómo es que cada componente se integra en el conjunto y contribuye al valor final para el cliente. Se puede adoptar analizando las interdependencias entre las diferentes partes del proyecto y fomentar la colaboración entre equipos para lograr una mejor comprensión del objetivo final.

(Rayo, 2016).

Herramientas de cada uno de los principios

1. Eliminar desperdicios: Debido a que en este principio se trata de eliminar todos los procesos que sean innecesarios, surgen algunas herramientas que simplifican esta tarea. Entre ellas tenemos:
    a. Value Stream Mapping: Es un método de diagrama de flujo que se utiliza para ilustrar y analizar un proceso de producción. Una de sus principales ventajas es que elimina todos los procesos innecesarios y que no aportan valor. VSM logra esto a través de 4 pasos que son:
        i. Generar un mapa del proceso actual
        ii. Buscar y eliminar los desperdicios
        iii. Generar un mapa del proceso mejorado
        iv. Implementar el proceso que se utilizará en el futuro

        (Asana, 2025).

    b. Diagramas de Ishikawa: También conocidos como diagramas espina de pescado, se utilizan para determinar las causas de un problema. El diagrama se estructura como una espina de pescado, con el enunciado del problema o efecto en la cabeza del pez y las causas potenciales representadas por ramas o espinas. Las espinas se dividen a su vez en subcausas, lo que permite a los equipos explorar múltiples niveles de causalidad e identificar las causas más probables del problema. (Narvaez, s.f.)

        Con esto se logra observar con mayor facilidad las causas más profundas de un problema, evitando centrarse en las más superficiales, reduciendo así procesos innecesarios. (Narvaez, s.f.)

2. Amplificar el aprendizaje: Ya que en este principio se necesita fomentar el aprendizaje entre todos los miembros del equipo, algunas herramientas que ayudan con esto son:
    a. Desarrollo basado en pruebas: Como su nombre lo indica significa desarrollar software utilizando pruebas o test que validan el correcto funcionamiento y si pasan los test la funcionalidad se aprueba. Esto funciona como una fuente de retroalimentación continua tanto para uno como para varios integrantes, ya que cada que hay un error o una prueba falla, se tiene que volver a analizar para corregir el error. (Hamilton, s.f.).
    b. Integración continua: Es la práctica de automatizar la integración de los cambios de código de varios contribuidores en un único proyecto de software. Permite a los desarrolladores fusionar con frecuencia los cambios de código en un repositorio central donde luego se ejecutan las compilaciones y pruebas. Las herramientas automatizadas sirven para verificar que el nuevo código es correcto antes de la integración. (Atlassian, 2024).
        La parte de las pruebas brinda retroalimentación a los desarrolladores, haciendo que si existe algún problema, el software es el encargado de notificar y mostrar que puede estar fallando.

3. Tomar decisiones tardías: En este principio se procura tomar las decisiones cuando se tiene la mayor cantidad de información posible, por ende se tiende a posponer la mayor cantidad de tiempo, ante esto surge una herramienta que puede ayudar a no perder tiempo y contemplar todas las posibles opciones.
    a. Desarrollo basado en conjuntos: Esta técnica examina una amplia gama de opciones posibles. Los conjuntos de soluciones posibles se reducen gradualmente hasta converger en una solución final. A lo largo del proceso de diseño, algunas opciones se eliminan debido a restricciones, inviabilidad o falta de idoneidad. La ideación también puede servir para generar más opciones que se pueden añadir al flujo de trabajo, tomando cada vez más opciones hasta determinar cuál es la mejor. (Do, 2022)
4. Entregar lo antes posible: Debido a que en este principio se debe entregar a la brevedad posible los resultados, la mejor forma de hacerlo es por medio de la organización, por tal motivo una de las principales herramientas de este principio es Scrum.

    Scrum es un conjunto de buenas prácticas que se hacen con el fin de mejorar el trabajo en equipo para obtener el mejor resultado posible en el menor tiempo posible. Aquí se realizan entregas parciales priorizadas por el beneficio que aportan, por tal motivo es una excelente herramienta para entornos complejos. (proyectosagilesorg, s.f.).

5. Potenciar el equipo: Para crear entornos que involucren más a todo el equipo y que los haga partícipes en la toma de decisiones durante las gestiones del proyecto, resultan útiles herramientas como Kanban.

    Kanban se centra en ayudar a encontrar un equilibrio entre el trabajo que necesitan hacer y la disponibilidad de cada miembro del equipo. Kanban se basa en una filosofía centrada en la mejora continua donde las tareas se extraen de un flujo de trabajo constante. (Martins, 2025).

    Con Kanban, todos los miembros se ven involucrados en el flujo de trabajo, siendo capaces de estar al tanto de cuáles son sus responsabilidades y que deben hacer, marcando cada paso hecho en una categoría o lista diferente y aunque parezca tener similitud con Scrum, son marcos diferentes aunque compatibles.

6. Crear la integridad: Este principio busca garantizar que el software sea fácil de mantener, mejorar y reutilizar. Para ello, se pueden utilizar herramientas como:
    a. Integración continua (CI): Automatiza la integración de cambios en el código, evitando errores acumulativos y asegurando que el software siempre esté en un estado funcional. Ejemplo: Jenkins, GitHub Actions.
    b. Desarrollo basado en pruebas (TDD): Asegura calidad desde el inicio al escribir pruebas antes del código. Herramientas como JUnit o PyTest ayudan a validar funcionalidad de manera continua.

    (ACKstorm, 2024)

7. Visualizar todo el conjunto: Este principio promueve una visión global del proyecto para mejorar la colaboración y alineación con los objetivos del cliente. Algunas herramientas útiles son:
    a. Mapas de procesos (Value Stream Mapping - VSM): Identifican ineficiencias en el flujo de trabajo y optimizan procesos (Team Asana, 2025).
    b. Gestión visual con Kanban: Permite visualizar tareas y cuellos de botella en tableros como los de Trello o Jira (Atlassian, 2024).
    c. Arquitecturas de software modulares: Fomentan una visión estructurada del sistema, facilitando el mantenimiento y escalabilidad.
    d. Diagramas UML: Representan gráficamente los componentes y relaciones del software, mejorando la comprensión del sistema.

    (Huet, 2022)

Conclusiones

- Lean Software Development (LSD) optimiza el desarrollo al reducir desperdicios y mejorar la eficiencia.
- Sus principios fomentan entregas rápidas, decisiones informadas y calidad desde el inicio.
- Herramientas como Kanban, VSM e integración continua mejoran la visualización y optimización del flujo de trabajo.
- Su enfoque flexible y adaptable es ideal para proyectos con cambios constantes y equipos que buscan eficiencia.

Referencias

Asana. (2025). ¿Qué es VSM y cómo se hace un Value Stream Mapping? Asana. https://asana.com/es/resources/value-stream-mapping

Atlassian. (2024). ¿En qué consiste la integración continua?. Atlassian. ¿En qué consiste la integración continua? | Atlassian

Do, D. (2022). Introducción al diseño basado en conjuntos. Lean Construction Blog. https://leanconstructionblog.com/espanol/Introduccion-al-diseno-basado-en-conjuntos.html

Hamilton, T. (s.f.). ¿Qué es el desarrollo basado en pruebas (TDD)? Ejemplo. Guru99. ¿Qué es el desarrollo basado en pruebas (TDD)? Ejemplo

Herranz, R. (2025). What is Lean Software Development? Scrum Alliance. https://resources.scrumalliance.org/Article/lean-software-development

Huet, P. (2022, August 24). Arquitectura de software: Qué es y qué tipos existen. OpenWebinars.net. https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/

Martins, J. (2025). ¿Qué es la metodología Kanban y cómo funciona? Asana. https://asana.com/es/resources/what-is-kanban

Narvaez, M. (s.f.). Diagrama de Ishikawa: Qué es y cómo realizarlo. QuestionPro. Diagrama de Ishikawa: Qué es y cómo realizarlo

Proyectosagilesorg. (s.f.). Qué es Scrum. https://proyectosagiles.org/que-es-scrum/

Rayo, A. (2016). Lean Software Development (LSD): los siete principios. Net Mind. https://netmind.net/actualidad/lean-software-development-lsd-los-siete-principios/

Team Asana. (2025, February 8). ¿Qué es VSM y cómo se hace un Value Stream Mapping? [2025] • Asana. Asana. https://asana.com/es/resources/value-stream-mapping

¿Qué es la integración continua? La clave para el desarrollo de software. (2024, June 4). ACKstorm. https://www.ackstorm.com/blog/que-es-la-integracion-continua/

Anexos

Repositorio de GitHub:

https://github.com/Isabella334/Proyecto_DAPA.git

Enlace al Google Docs:

https://docs.google.com/document/d/1LRyIRwn3sqLgIEwkDPBiBjNKzYc-ntbOSLxoe1FhnU0/edit?usp=sharing

Gestión del tiempo:

Nro.	Tarea	Encargado	Fecha inicio	Fecha entrega
1	Redactar los conceptos principales	Víctor Pérez	26/03/2025	26/03/2025
2	Redactar los principios de Lean SW Development	Juan Solís	26/03/2025	27/03/2025
3	Redactar las herramientas de los principios	Diego Flores y Nils Muralles	27/03/2025	30/03/2025
4	Redactar el resumen	Isabella Recinos	30/03/2025	30/03/2025
5	Redactar la introducción	Isabella Recinos	30/03/2025	30/03/2025
6	Redactar las conclusiones	Todos	30/03/2025	30/03/2025
7	Elaborar la presentación de la investigación	Isabella Recinos	30/03/2025	30/03/2025