Lean Software Development

Ingeniería de Software 2 (CC3091) - 2026

Lean Software Development

Semestre 02, 2026

Introducción

Lean Software Development es una adaptación de los principios de manufactura esbelta (Lean Manufacturing) al desarrollo de software.

Su objetivo es entregar el máximo valor al cliente utilizando la menor cantidad de recursos, eliminando todo aquello que no aporta valor.

Fue formalizado por Mary y Tom Poppendieck en su libro *Lean Software Development: An Agile Toolkit* (2003), donde tradujeron las ideas de Lean al mundo del software.

Origen: el Sistema de Producción Toyota

Lean nace del Sistema de Producción Toyota (TPS), desarrollado en Japón después de la Segunda Guerra Mundial.

Su principal arquitecto, Taiichi Ohno, buscaba producir con menos desperdicio, mejor calidad y mayor rapidez que la producción en masa tradicional.

Dos ideas centrales del TPS son:

- **Muda**: toda actividad que consume recursos pero no agrega valor al cliente.
- **Kaizen**: mejora continua, en pequeños pasos, con la participación de todo el equipo.

Valor y desperdicio

El punto de partida de Lean es distinguir entre valor y desperdicio.

Valor es aquello por lo que el cliente está dispuesto a pagar: una funcionalidad que resuelve su problema.

Desperdicio (muda) es todo lo demás: pasos, actividades o artefactos que consumen tiempo y esfuerzo sin acercar el producto al cliente.

La meta de Lean es maximizar el flujo de valor y reducir el desperdicio de forma sistemática.

Los 7 desperdicios del desarrollo de software

Los Poppendieck adaptaron los 7 desperdicios de la manufactura al software:

- **Trabajo parcialmente hecho**: código escrito pero no integrado, probado ni desplegado. Es inventario que no genera valor.
- **Funcionalidades extra**: características que se construyen pero nadie usa.
- **Reaprendizaje**: volver a aprender algo que ya se sabía, por pérdida de información o conocimiento.
- **Entregas y traspasos (handoffs)**: pasar el trabajo de una persona a otra pierde conocimiento tácito en cada traspaso.

- **Cambios de tarea (task switching)**: alternar entre tareas fragmenta la atención y reduce la productividad.
- **Esperas y retrasos**: tiempo perdido esperando decisiones, aprobaciones o dependencias.
- **Defectos**: errores que llegan al cliente o que obligan a rehacer trabajo.

Los 7 principios de Lean Software Development

Mary y Tom Poppendieck definieron siete principios que guían la aplicación de Lean al software.

1. Eliminar el desperdicio

Identificar y remover toda actividad que no agrega valor al cliente.

Para lograrlo se usa el **mapeo del flujo de valor (value stream mapping)**: dibujar todos los pasos desde la idea hasta la entrega y detectar dónde se genera desperdicio.

2. Amplificar el aprendizaje

El desarrollo de software es un proceso de aprendizaje continuo, no una línea de ensamblaje.

Se amplifica el aprendizaje con iteraciones cortas, retroalimentación frecuente del cliente, revisiones de código, programación en parejas y refactorización.

3. Decidir lo más tarde posible

Retrasar las decisiones importantes hasta el **último momento responsable** (last responsible moment), cuando se tiene la mayor cantidad de información.

Esto reduce la incertidumbre y evita comprometerse temprano con decisiones que luego resulten costosas de cambiar. Se apoya en el **desarrollo basado en conjuntos (set-based development)** y en el **pensamiento de opciones (options thinking)**.

4. Entregar lo más rápido posible

Entregar valor al cliente en ciclos cortos permite obtener retroalimentación temprana y reaccionar al cambio.

Se apoya en integración continua, desarrollo *just-in-time*, sistemas de trabajo tipo **pull** y tableros **Kanban** para visualizar y limitar el trabajo en progreso.

5. Empoderar al equipo

Las personas que hacen el trabajo son las que mejor conocen cómo mejorarlo.

Se respeta al equipo dándole autonomía para tomar decisiones, comunicación abierta y liderazgo que va al *gemba* (el lugar donde ocurre el trabajo) en lugar de imponer desde arriba.

6. Construir la calidad desde el inicio

La calidad no se inspecciona al final: se integra en cada paso.

Se busca **integridad conceptual** (la arquitectura interna es coherente) e **integridad percibida** (el usuario siente que el producto funciona bien). Prácticas clave: TDD, refactorización, programación en parejas y automatización de pruebas.

7. Optimizar el todo

Optimizar una parte aislada del sistema suele perjudicar al conjunto.

Se debe ver el sistema completo: el flujo de valor de principio a fin, alineando incentivos hacia el desempeño global y no hacia métricas individuales.

Herramientas de pensamiento (thinking tools)

Los Poppendieck proponen herramientas concretas para aplicar los principios:

- **Value stream mapping**: visualizar el flujo de valor y localizar el desperdicio.
- **Sistemas pull y Kanban**: dejar que la demanda del cliente jale el trabajo, limitando el trabajo en progreso.
- **Last responsible moment**: decidir cuando se tiene más información, no antes.
- **Set-based development**: explorar varias alternativas en paralelo antes de comprometerse con una.
- **Options thinking**: tratar las decisiones como opciones que se mantienen abiertas mientras aporten valor.
- **Pull authority to the point of work**: dar poder de decisión a quien hace el trabajo.

Lean y Agile

Lean y Agile comparten valores: iteraciones cortas, retroalimentación temprana, respeto por las personas y adaptación al cambio.

Agile se enfoca en cómo trabaja el equipo de desarrollo; Lean aporta una mirada más amplia sobre todo el flujo de valor, desde la idea hasta el cliente.

Metodologías como Kanban y conceptos como el trabajo en progreso limitado provienen directamente del pensamiento Lean.

Conclusiones

Lean Software Development traslada la filosofía de manufactura esbelta al software.

Su esencia es maximizar el valor para el cliente y eliminar el desperdicio de forma continua.

Los siete principios y las herramientas de pensamiento ofrecen una guía práctica para lograrlo.

Lean complementa a Agile al mirar el sistema completo, no solo el trabajo del equipo.