Lean UX

Ingeniería de Software 1 (CC3090) - 2026

Lean UX

Semestre 01, 2026

El problema del UX tradicional

El proceso de UX clásico funciona en fases secuenciales.

Un equipo de diseño trabaja durante semanas.

Luego entrega especificaciones al equipo de desarrollo.

El desarrollo implementa lo que se diseñó.

El problema

Los equipos ágiles trabajan en ciclos de 1 a 2 semanas.

Un proceso de diseño que tarda meses no encaja.

- El diseño va siempre adelante o atrás del desarrollo.
- El desarrollador no entiende por qué se diseñó algo así.
- El cliente no ve resultados hasta que ya es tarde para cambiar.

UX en cascada dentro de Agile

Muchas organizaciones adoptan Agile pero mantienen UX en cascada.

Esto genera tres problemas crónicos:

- Problema de sincronización — diseño y desarrollo nunca están al mismo ritmo.
- Vacío de retroalimentación — los diseños se construyen antes de validar con usuarios.
- Efecto silo — diseñadores y desarrolladores trabajan aislados.

Definición

Lean UX es un enfoque de diseño que desplaza el foco:

De entregar documentos → a lograr resultados de usuario.

Origen

Creado por **Jeff Gothelf** y **Josh Seiden**.

Formalizado en el libro *Lean UX* (O'Reilly, 2013).

Gothelf lo desarrolló al observar que el UX tradicional era incompatible con equipos ágiles.

La idea central

Todas las decisiones de diseño son hipótesis.

No son requisitos.

Las hipótesis deben validarse antes de invertir en construirlas completamente.

Los tres fundamentos

Lean UX surge de la síntesis de tres disciplinas.

Lean Startup

Eric Ries, 2011.

- Ciclo Build-Measure-Learn.
- El Producto Mínimo Viable (MVP) como herramienta de aprendizaje.
- Toda decisión de producto es una hipótesis que debe validarse empíricamente.

Agile

Manifiesto Ágil, 2001.

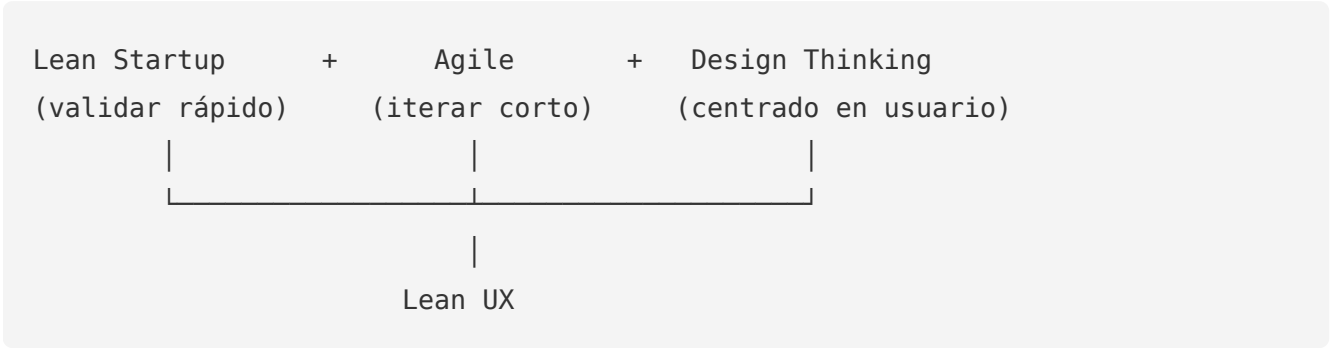
- Ciclos cortos de entrega.
- Equipos multifuncionales y auto-organizados.
- Software funcionando sobre documentación exhaustiva.
- Responder al cambio sobre seguir un plan.

Design Thinking

Stanford d.school / IDEO.

- Diseño centrado en el ser humano.
- Empatía con el usuario como punto de partida.
- Aceptación de la ambigüedad y la iteración.
- Prototipado rápido antes de comprometerse con una solución.

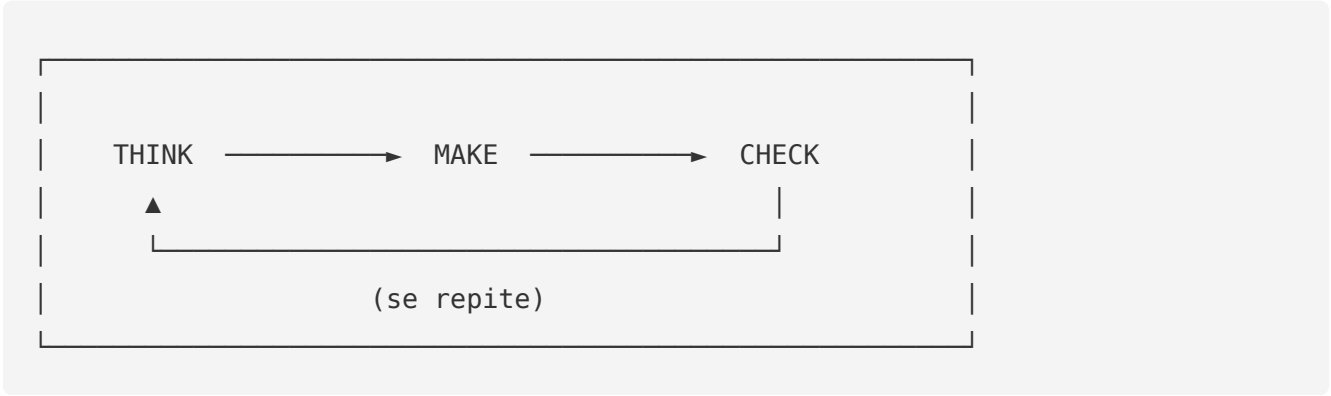
La síntesis



El ciclo Lean UX

El ciclo fundamental se resume en tres fases.

Think → Make → Check



Este ciclo se ejecuta dentro de cada Sprint.

Cada vuelta genera aprendizaje validado.

THINK — Pensar

- Identificar el problema a resolver.
- Declarar los supuestos del equipo.
- Convertir los supuestos más riesgosos en hipótesis.
- Definir qué evidencia confirmaría o refutaría cada hipótesis.

MAKE — Construir

- Crear el artefacto mínimo necesario para probar la hipótesis.
- No un producto terminado — un experimento.
- Puede ser: papel, prototipo clickeable, página web, servicio manual.
- Velocidad y baja fidelidad son una ventaja, no una debilidad.

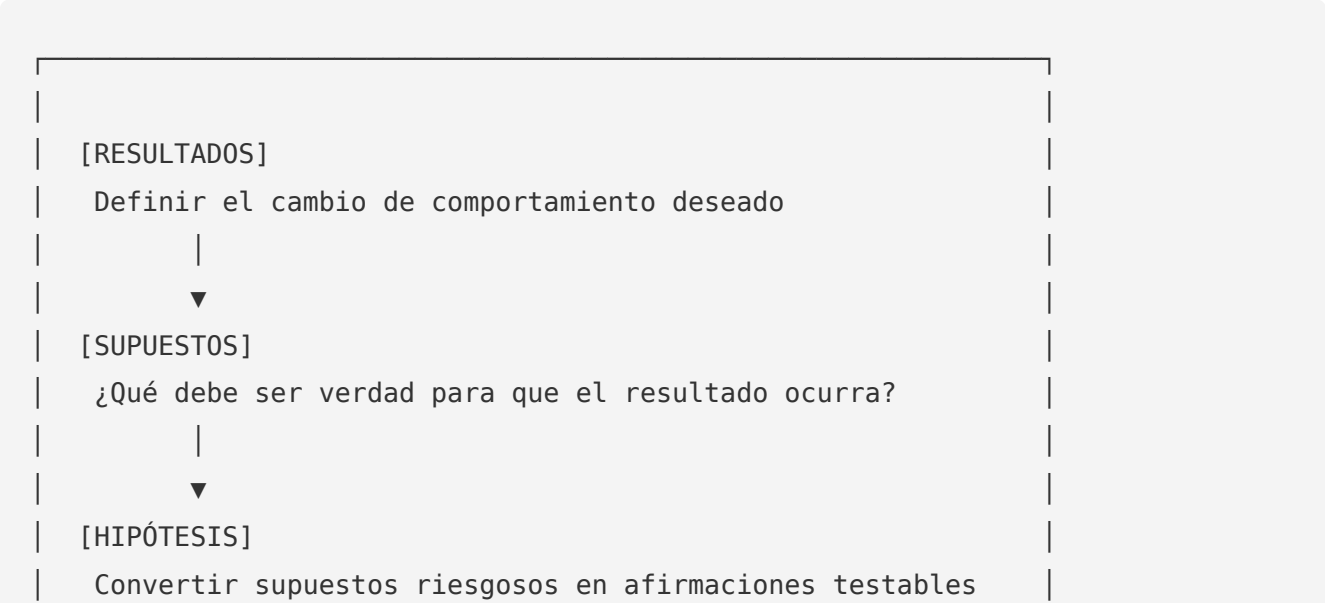
CHECK — Verificar

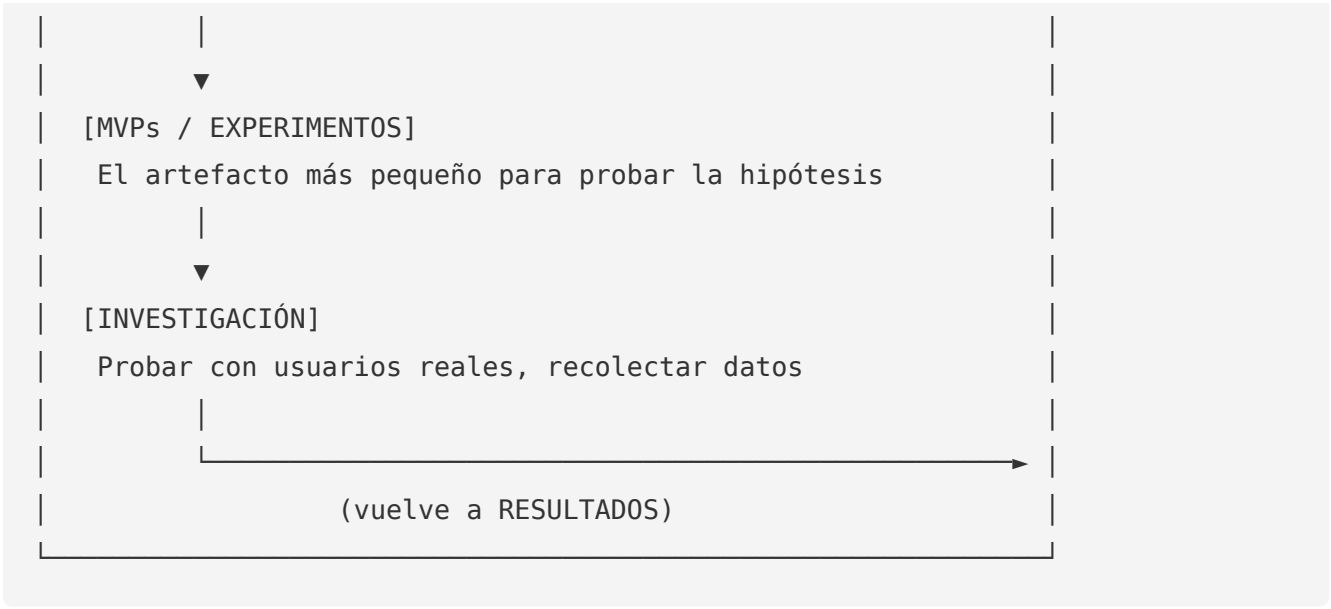
- Poner el artefacto frente a usuarios reales.
- Medir contra los criterios de éxito definidos en THINK.
- Decidir: continuar, pivotar o abandonar.
- Los aprendizajes alimentan el siguiente THINK.

El ciclo completo

A nivel de producto, el ciclo se expande en seis fases.

Diagrama





Relación entre los dos ciclos

Think → Make → Check corre dentro de cada Sprint.

El ciclo completo corre a lo largo de múltiples Sprints.

Son el mismo proceso visto a diferente escala.

Resultados

Lean UX comienza por definir resultados, no características.

Un resultado es un cambio en el comportamiento humano que genera valor de negocio.

Resultado vs. característica

| Característica | Resultado | | --- | --- | | Agregar botón de registro con Facebook | Incrementar la tasa de registro completo de 30% a 60% | | Rediseñar el checkout | Reducir el abandono del carrito en 20% | | Construir un chatbot | Reducir tickets de soporte en 35% |

La característica describe qué se construye.

El resultado describe qué comportamiento se quiere cambiar.

Importancia

Un equipo que define resultados puede decidir cómo lograrlos.

Un equipo que define características solo puede decidir cuándo terminarlos.

Supuestos

Un supuesto es una creencia del equipo que todavía no ha sido validada.

Todo proyecto se construye sobre supuestos.

Lean UX los hace explícitos para poder priorizarlos y probarlos.

Categorías de supuestos

- Supuestos de negocio — ¿Quién es el mercado? ¿Cómo se genera valor?
- Supuestos de usuario — ¿Quién es el usuario? ¿Qué problema tiene?
- Supuestos de diseño — ¿Qué solución adoptarán? ¿Qué flujo funcionará?

Priorización

No todos los supuestos son igual de peligrosos.

Se priorizan por dos dimensiones:

- Riesgo — si este supuesto es falso, ¿cuánto daño hace al producto?
- Desconocimiento — ¿qué tan seguro está el equipo de que es verdad?

Los supuestos de alto riesgo y alta incertidumbre se prueban primero.

Hipótesis

Cada supuesto riesgoso se convierte en una hipótesis.

Una hipótesis es específica, falsable y medible.

Formato

```
"Creemos que [hacer esto / construir esta funcionalidad]
 para [este usuario]
 logrará [este resultado].
 Sabremos que es verdad cuando veamos [esta señal / métrica]."
```

Ejemplo

```
"Creemos que agregar un indicador de progreso
 al proceso de pago
 para compradores por primera vez
 reducirá el abandono del carrito.
 Sabremos que es verdad cuando veamos
 que el abandono cae al menos un 15%
 en una prueba A/B de dos semanas."
```

Formato

Obliga al equipo a ser específico sobre:

- Para quién se diseña.
- Qué cambio de comportamiento se espera.
- Qué evidencia contaría como prueba.

Sin este formato, el equipo puede lanzar una función y declarar éxito sin medir nada.

MVPs y Experimentos

Un MVP en Lean UX es una herramienta de aprendizaje, no un lanzamiento de producto.

Es el artefacto más pequeño que puede generar la evidencia necesaria.

Tipos de experimentos

Prototipo en papel

Pantallas dibujadas a mano; alguien simula la respuesta del sistema.

Prueba: flujo, arquitectura de información, comprensión del concepto.

Prototipo clickeable

Mockup digital (Figma) sin backend real.

Prueba: interacciones, comprensión del usuario, completitud de tareas.

Landing page falsa

Página que describe una funcionalidad que no existe aún.

Prueba: demanda — ¿la gente quiere esto?

MVP Concierge

El servicio es real pero entregado manualmente por personas.

Prueba: si el servicio genera valor y cómo debe ser el flujo.

MVP Mago de Oz

Parece automatizado pero funciona manualmente sin que el usuario lo sepa.

Prueba: si los usuarios confían y adoptan la experiencia automatizada.

El principio

Un prototipo en papel tarda 2 horas.

Una función completa tarda 2 semanas.

Si el prototipo en papel muestra que los usuarios no entienden el concepto, se ahorran 2 semanas de desarrollo.

Lean UX vs. UX Tradicional

Punto de partida

UX Tradicional: requerimientos de stakeholders.

Lean UX: supuestos + resultados deseados.

Orientación

UX Tradicional: entregables primero.

Lean UX: resultados primero.

Forma del proceso

UX Tradicional: secuencial.

Lean UX: iterativo, ciclos cortos.

Documentación

UX Tradicional: pesada — personas, journeys, specs.

Lean UX: ligera — solo lo necesario para probar.

Métrica de éxito

UX Tradicional: calidad y completitud del entregable.

Lean UX: cambio de comportamiento logrado.

Rol del diseñador

UX Tradicional: dueño único de las decisiones de diseño.

Lean UX: colaborador y facilitador.

Investigación

UX Tradicional: gran fase inicial.

Lean UX: continua, en pequeños lotes cada sprint.

Relación con desarrollo

UX Tradicional: diseñador entrega; dev implementa.

Lean UX: diseñadores y devs trabajan juntos desde el inicio.

Fidelidad del prototipo

UX Tradicional: alta — mocks pulidos antes de desarrollar.

Lean UX: baja — el mínimo para probar la hipótesis.

Colaboración multifuncional

Lean UX requiere equipos multifuncionales desde el día uno.

No silos que se pasan documentos entre sí.

Composición del equipo

Un equipo Lean UX típico incluye:

- Diseñador UX / Diseñador visual
- Desarrolladores frontend y backend
- Product Manager
- QA
- Analista de datos

Máximo 8 a 10 personas. Dedicados a un mismo problema.

Prácticas colaborativas

Design Studios — sesiones de bocetado donde todo el equipo (incluyendo devs) genera ideas antes de que el diseñador las refine.

Muros compartidos — bocetos, hipótesis y hallazgos de investigación visibles para todo el equipo.

Investigación conjunta — los desarrolladores asisten a las entrevistas de usuario, no solo ven los highlights.

¿Por qué involucrar a los desarrolladores desde el inicio?

- Conocen qué es técnicamente factible de forma rápida y barata.
- Pueden redirigir el diseño antes de que se invierta tiempo en un callejón sin salida.
- La exposición directa a usuarios genera empatía y compromiso, no solo en el diseñador.

Principios clave

Getting Out Of the Building (GOOB)

Concepto de Steve Blank.

Las respuestas sobre el producto están afuera, con los usuarios reales.

No en una sala de conferencias.

Las opiniones internas no sustituyen la evidencia real.

Entendimiento compartido

El equipo completo desarrolla un modelo mental común sobre:

- Quién es el usuario y qué necesita.
- Qué resultado se busca lograr.
- Qué se cree actualmente y por qué.

Este entendimiento reemplaza a la documentación como mecanismo de alineación.

Hacer en vez de analizar

Construir algo — aunque sea tosco — genera retroalimentación real.

Un debate genera opiniones.

Un prototipo genera evidencia.

Permiso para fallar

Los experimentos fallarán.

Eso es valioso.

Fallar en un prototipo de papel cuesta nada.

Fallar después de 6 meses de desarrollo cuesta todo.

Casos reales

Dropbox

- Hipótesis: la gente paga por sincronización de archivos si funciona bien.
- Experimento: un video de 3 minutos mostrando cómo funcionaría el producto.
- Resultado: 75,000 registros en un día antes de que existiera software listo para producción. Validaron demanda con costo casi cero.

Airbnb

- Hipótesis: hay personas dispuestas a rentar espacio en su hogar a extraños.
- Experimento: un sitio web mínimo ofreciendo colchones inflables en el apartamento de los fundadores. Tres huéspedes pagando = demanda validada.
- Aprendizaje: iteraron el modelo de dos lados (anfitriones y huéspedes) con retroalimentación directa antes de construir cualquier automatización.

Spotify

- Hipótesis: la piratería es un problema de UX, no de moralidad. La gente paga si la experiencia legal es mejor que la ilegal.
- Experimento: lanzamiento solo por invitación en Escandinavia con catálogo limitado.
- Resultado: validaron que los usuarios adoptaban el modelo de pago cuando el acceso era conveniente e instantáneo. Expandieron geográficamente e iteraron modelos de precios con datos reales.

Errores comunes

- Saltarse la investigación → tratar las hipótesis como verdades confirmadas antes de probarlas.
- Confundir entregables con resultados → lanzar funciones y declarar éxito sin medir cambio de comportamiento.
- Usar Lean UX como excusa para hacer menos UX → recortar investigación y pruebas de usuario.
- No involucrar a los desarrolladores desde el inicio → sorpresas técnicas costosas al final.
- Tratar la hipótesis como definitiva → una hipótesis confirmada revela el siguiente conjunto de supuestos a probar.
- No externalizar el trabajo → sin visibilidad compartida, cada disciplina opera con su propio modelo mental.