**Lean**

**Definiciones:**

* **El software y su desarrollo son únicos:** El proceso de crear software es diferente de la fabricación de productos físicos como bicicletas, debido a sus características propias.
* **Proceso de desarrollo distinto:** El desarrollo de software es transformar ideas en productos, un proceso empírico (enfocado en la experiencia) basado en la adaptación y la iteración. La variabilidad del software hace difícil definir un proceso común para todos los tipos.
* **Software como producto desarrollado:** El software en funcionamiento es un sistema complejo que organiza la actividad de muchos elementos (personas, dispositivos, otros software). La detección y corrección de errores se realiza en el sistema en funcionamiento.
* **Fabricación de un producto:**
* **Entradas**: materias primas y componentes
* **Salida**: producto terminado (automóvil, teléfono móvil, televisor, etc.)
* **Proceso**: diferentes máquinas y pasos en la cadena de montaje
* Si vemos el **desarrollo software como un proceso iterativo** podemos definir un proceso de desarrollo general:
* **Entradas**: software funcionando e ideas de nuevas características (features) en forma de casos de uso, historias de usuario, etc.
* **Salida**: software funcionando al que se le ha añadido las nuevas características.
* **Proceso**: cada característica debe ser analizada, desarrollada, probada, añadida y entregada.
* El **desarrollo de software iterativo** tiene 2 ejes de calidad: elegir qué característica (right product) y desarrollar correctamente la característica (product right).

**De lo mencionado anteriormente se observan 2 cosas:**

* El desarrollo de software es distinto debido a su proceso iterativo
* Es posible aprender de los procesos de fabricación tradicionales. 🡪 Fabricación **Lean**

Centrada en:

* Filosofía de “Hacer más y mejor con menos”
* **Eliminar desperdicios** (waste), quitar aquellos elementos que no añaden valor al producto, ejemplos de desperdicios en procesos:
  1. **Espera**: personas o hitos del proceso esperando que termine otro proceso o que llegue cierta información.
  2. **Movimiento**: movimiento físico o mental que no añade valor.
  3. **Inventario**: almacenar servicios y componentes extra que el cliente no ha pedido.
  4. **Defectos**: errores que hay que corregir.
  5. **Sobre-procesamiento**: excesiva documentación, informes excesivos, partes no necesarias.
* Se reducen los gastos en inventario al almacenar la cantidad de materiales para los productos justos que se van a producir. (**Just-in-Time**)
* Para eliminar desperdicios se deben identificar los pasos de la cadena de valor (**value stream).** Al mínimo error se “**para la cadena**”.
* Cultura de **mejora continua** en todos los niveles de la organización. Es una **mentalidad** (**mindset**) y no un conjunto de reglas, mejorando la calidad de producción volviéndose fluida (**flow**), haciendo que se reduzcan los costos y el tiempo.
* Pensar en el **conjunto**, procesos y equipos, **visualizándolo** con diversas herramientas.
* El proceso siempre puede ser mejorado y los trabajadores son los que pueden proponer estas mejoras mediante el **método científico** (crear hipótesis, probarla, analizar resultados, y si la hipótesis se confirma se hacen los cambios).
* Consecuencias de aplicar Lean: Alta disciplina y respuesta al cambio.
* Uno de los pilares de la **fabricación Lean** es el sistema de fabricación **Pull:**
  1. En un proceso de fabricación tenés: un conjunto de pasos que necesitan recursos (**celdas),** y celdas que van a necesitar los resultados que produjo el proceso anterior(**upstream).** Un proceso está compuesto por múltiples celdas.
  2. En un sistema **Push**: primero se planifica la cantidad de trabajo, los procesos upstream producen componentes que serán empujados a la siguiente celda para que se siga procesando. Esto puede producir sobrecargas en las celdas.
  3. En un sistema **Pull**: El flujo de trabajo se origina al final de la cadena, donde está el cliente con el producto solicitado, cuando se consume el producto de la última celda, se envía una señal a la celda anterior (**pull signal**) indicando que se necesita recibir un nuevo componente o materiales a transformar. Cada celda tiene un número máximo de productos (**WIP limit**) y se busca reemplazar solo aquello que se consumió y lo que fue inmediatamente entregado.
     + Los **Tableros Kanban** serán los que nos permitirán llevar a cabo estas **pull signal** de formavisual, a través de tarjetas en un tablero que indican que un centro de trabajo ha finalizado un proceso, necesita trabajo o necesita más materiales.

**Llevando la fabricación Lean al campo del desarrollo de software:**

* Implementación de metodología Kanban por medio de **Tableros Kanban**. Se trata de un método visual de gestión de proyectos que permite a los equipos visualizar sus flujos de trabajo y la carga de trabajo. En un tablero Kanban, el trabajo se muestra en un proyecto en forma de tablero organizado por columnas. Tradicionalmente, cada columna representa una etapa del trabajo. El tablero Kanban más básico puede presentar *columnas* como Trabajo pendiente, En progreso y Terminado. Las tareas individuales, serán representadas por tarjetas, que serán usadas como *señales visuales* del tablero, que avanzan a través de las diferentes *columnas* hasta que estén finalizadas. Además, en cada columna se encontrará el *límite del* ***trabajo en curso*** *(****WIP****)* para establecer la cantidad de tarjetas que se puede asignar ese estado.
* **Los 7 principios del desarrollo Lean de software**:
  1. **Eliminar desperdicios** (waste): Implica reducir cualquier actividad que no añada valor al producto final. Esto incluye producir solo lo necesario, evitar sobrecarga de requisitos, exceso de arquitectura y código en desarrollo de software, evitar funciones innecesarias en productos finales y desperdicio de tiempo y energía en tener que re-aprender debido a la falta de comunicación.
  2. **Fomentar la calidad:** En el desarrollo de software, se debe crear un producto intuitivo que tenga sentido para los usuarios y que satisfaga al cliente. Minimizar la deuda técnica mediante (software con bugs y parches) refactorización y revisiones de código es crucial, al igual que enfocarse en pruebas rigurosas para evitar defectos. Un software que funciona debe cumplir con las features que demanda el cliente y debe ser consistente en su integridad, es decir esas features deben funcionar correctamente en conjunto.
  3. **Crear conocimiento**: Entender tanto el producto como el proceso de desarrollo. Utilizar feedback y representaciones gráficas (como diagramas de flujo acumulativo) ayuda a entender mejor el software que se está desarrollando, como la metodología, la arquitectura y tecnología.
  4. **Decidir los más tarde posible**: Toma decisiones importantes lo más tarde posible, con máxima información.
  5. **Entregar rápido**: Las entregas y desarrollos rápidos pueden permitir proporcionar más características, ya que el prototipo que se le muestre al cliente le va a permitir decidir con más información. Además, hay que entender que los retrasos significan un aumento en costos. Por lo tanto se deben usar técnicas de entregas continuas (continuous delivery).
  6. **Respetar a la gente y potenciar el equipo**: Se busca un conjunto de personas comprometidas con conseguir un objeto común, con respeto, mediante trabajo centrado, efectivo, y de manera sostenida con toda la energía posible. Tener “héroes” que hagan la mayoría del trabajo no es potenciar al equipo.
  7. **Optimizar al conjunto**: Para esto es esencial que todo el equipo comprenda el objetivo final y cómo el software añade valor al usuario, si no lo hace debemos como preguntarnos *por qué* se producen tales errores, *por qué* determinada funcionalidad no hace lo que le pido, *por qué* el usuario hace o *no* *hace* tal cosa, o hasta preguntar *por qué* no se ha añadido determinada funcionalidad. A partir de las preguntas se deberá reflexionar para encontrar las respuestas.