**LEAN**

Es una metodología de gestión originada en Japón, por Toyota. Su objetivo principal es maximizar el valor o producto al paciente mientras que se reduce o elimina el desperdicio.

Para ello hay que identificar y eliminar procesos que no agregen valor al producto.

Los principios clave de la metodología Lean serian:

1. **Identificar el valor**: Consiste en identificar el valor del producto, que puede ser para una parte interesada interna (persona que participa en el proyecto) o para una parte interesada externa (Usuario o cliente al que va dirigido el producto).
2. **Mapea el flujo de valor (VSM):** Es una herramienta visual que se utiliza para representar, analizar y mejorar los flujos de material y/o información de un proceso de producción. Tiene inicio desde la Idea hasta que se entrega el producto al cliente. El objetivo será identificar y eliminar procesos innecesarios o repetitivos. “Desperdicios”
3. **Crea un flujo continuo**: Se eliminarán los “Desperdicios” identificados en el anterior apartado, con ello se creará un flujo de trabajo optimo y sin interrupciones.
4. **Establecer un Sist.Pull:** Se establece un sistema con el que se incorporaran trabajos al proceso, cuando otros trabajos se van terminado. De este modo solo se producirá cuando sea necesario.

Ejemplo de sistema “pull” en desarrollo de software:

* 1. El diseñador técnico termina su tarea y marca el producto para revisión.
  2. La marca de revisión indica que debe comenzar la etapa de codificación.
  3. El codificador termina su tarea y marca el producto para revisión.
  4. La marca de revisión indica que debe comenzar la etapa de pruebas.
  5. El responsable de las pruebas de productos termina su tarea y lo marca como listo para revisión.
  6. Llevas a cabo la revisión final del producto.

1. **Perfección o mejora continua**: No se realizará una sola vez, sino que será un proceso iterativo, buscando mejoras continuas en al fujo de trabajo.
2. **Kaizen**: Fomentar una cultura de mejora continua, donde se realizan pequeñas mejoras de manera constante.



**SCRUM**

Scrum se compone de:

**Product owner**: Es el encargado de priorizar la lista de requisitos (backlog)

**Scrum master**: Es quien coordina al equipo scrum además de ayudarlos a superar impedimentos y se asegura que se sigan las prácticas de scrum.

**Development team/Scrum team**: Grupo multifuncional encargado de desarrollar y entregar las entregas realizadas.

**Product backlog**: Es una lista de requisitos.

**Backlog del Sprint**: Una parte de la lista de requisitos que se realizara durante un sprint.

**Sprint**: Es un periodo de tiempo, entre 2 y 4 semanas, durante el cual se realizan el resto de eventos o ceremonias. El sprint debe de ser continuo, y vendrá determinado por el propio *equipo scrum* y el *product owner*. En scrum no habrá fases, sino sprints y cada sprint tendrá 5 eventos o ceremonias:

1. **Sprint planning**: Es una reunión que se realiza al comienzo de cada sprint, y en la cual participa todo el equipo scrum. En esta reunión se inspecciona el *Product backlog* y se seleccionaran los ítems (*Sprint Backlog*) a trabajar durante el sprint. También se tratará el como se va a realizar, por lo que la primera parte de la reunión estará encabezada por el *Product owner* mientras que en la siguiente destacará más el *development team.* La reunión puede durar entre 6-8h para un sprint de 4 semanas.
2. **Daily scrum**: Es una reunión diaria de unos 15 minutos, exclusivamente para el development team. En esta reunión se realiza una pequeña auto organización de las tareas diarias. Para ello se utilizan 3 frases; ¿Qué hice ayer? ¿Qué voy a hacer hoy? ¿Hay algo que me impida mi tarea?
3. **Sprint review**: Es una reunión que se realiza al final del sprint en la cual el Product owner y el development team presentan la entrega terminada al cliente(*stakeholders)*. Esta entrega o software ha sido validado anteriormente por el Product owner.
4. **Sprint retrospective**: Es una reunión con el objetivo de reflexionar sobre el anterior sprint y buscar posibles mejoras para el siguiente sprint. Normalmente en esta ceremonia será el scrum master quien tomará un rol de facilitador, pero también puede ir rotando entre los demás.
5. **Sprint grooming**: En esta reunión se realiza un refinamiento del *Product backlog.* Servirá por tanto como planificación para futuros sprints.

**KANBAN**

Es una categoría de gestión ágil de proyectos originada en Japón. Se basa en el flujo continuo de trabajo y es implementada mediante tableros. Este tablero permite visualizar los flujos y la carga de trabajo. El tablero es una forma visual de gestionar el proyecto, en el cual se muestra el trabajo organizado por columnas.

**Conceptos**:

1. **Tablero Kanban**: Es una herramienta visual que estará tradicionalmente dividida en 3 columnas, las cuales representan diferentes etapas del proceso.
2. **Tarjetas Kanban**: cada tarea se representa mediante una tarjeta, que contendrá información del trabajo a realizar, una breve descripción, la prioridad, etc.
3. **Columnas**: son las columnas representadas en el tablero, y cada una representa el momento por el cual se encuentra una tarea (tarjeta). Tendremos 3 columnas, que pueden ser “Por hacer”, “En proceso” y “Hecho”

**Prácticas en la metodología Kanban**:

1. **Visualizar**: Se puede visualizar como una tarea avanza por las diferentes etapas, con ello tendremos una idea general de como avanza el proyecto, pero además también tendremos información en tiempo real del trabajo.
2. **Wip** Limites del trabajo en proceso: Se establecerá un limite de trabajo en cada columna para así evitar sobrecarga en alguna etapa. Esto implicara que las tareas se muevan rápidamente de una columna/etapa a otra.
3. **Gestionar el flujo de trabajo**: se utilizará un sistema “pull” (solo se inicia trabajo si hay capacidad en la siguiente columna/etapa.) para evitar la sobrecarga y que el flujo de trabajo sea lo más estable posible.
4. **Implementar ciclos de comentarios**

**EXTREME PROGRAMING**

Es una metodología ágil basada en la mejora de la calidad del software.

**Características fundamentales:**

* Desarrollo iterativo e incremental
* Pruebas unitarias continuas
* Programación en parejas
* Integración del equipo de programación con el cliente o usuario
* Corrección de todos los errores
* Refactorización del código
* Propiedad del código compartida
* Simplicidad en el código

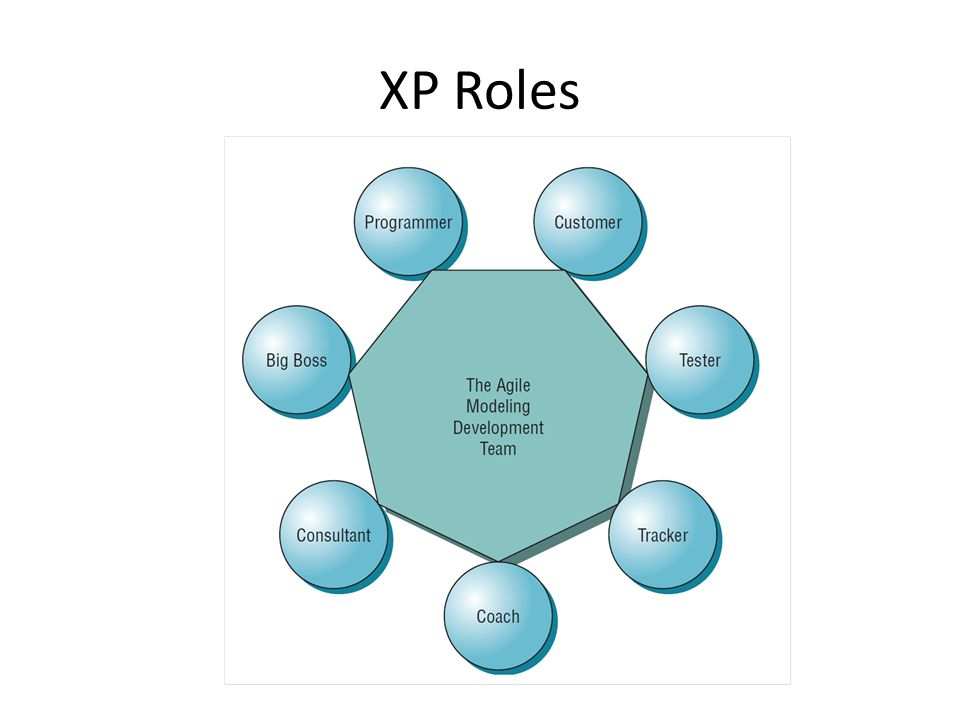
**Proceso Extreme Programing**: El ciclo de vida ideal de XP consiste de seis fases [2]: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. En cada una de las iteraciones se crea un ciclo completo de diseño, pruebas, análisis y desarrollo mediante un conjunto de reglas y técnicas específicas de XP. Habitualmente, el desarrollo de una XP supone entre 10 y 15 iteraciones.

1. **Fase I: Exploración:** En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.
2. **Fase II: Planificación de la Entrega:** En esta fase el cliente establece la prioridad de cada historia de usuario y los programadores estiman el esfuerzo para la entrega de cada una. Se determina un cronograma en conjunto al Cliente. La planificación se puede realizar basándose en el tiempo o el alcance.
3. **Fase III: Iteraciones:** Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. Al final de la última iteración el sistema estará listo para entrar en producción. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.
4. **Fase IV: Producción:** La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente.
5. **Fase V: Mantenimiento:** Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones
6. **Fase VI: Muerte del Proyecto:** Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Se genera la documentación final del sistema

**Valores**:

1. **Simplicidad**: Es la base de la programación extrema. Se simplifica el diseño y el código para así facilitar la compresión y mantenimiento. Para mantener la simplicidad es necesaria la refactorización del código, esta es la manera de mantener el código simple a medida que crece.
2. **Comunicación**: Comunicación constante entre los miembros del equipo y también entre el equipo y cliente de forma fluida.
3. **Feedback**: El cliente se encuentra integrado en el proyecto y así su opinión se conoce en tiempo real. Además, se hacen ciclos muy cortos de presentación de resultados pudiéndose realizar ajustes según las necesidades.
4. **Valentía**: Las practicas del XP requieren valentía ya que se programa para el momento y no en futuro, además de reconocer errores tan pronto como se cometan. Igualmente, se le puede denominar valentía a la persistencia.
5. **Respeto**: Es una virtud vital. Debe producirse entre los miembros del equipo, así como entre el equipo y cliente. Cuando existe respeto mutuo el equipo trabajara de una forma mas eficiente y con un mayor ritmo de producción.

**Roles**:

* **Programador**: Es el encargado de producir el código del sistema.
* **Test developer**: Produce el código de los test unitarios del sistema.
* **Cliente**: Se encargan de escribir las historias de usuario y las pruebas funcionales para validar su implementación. Asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración.
* **Tester**: Interpreta el pedido del cliente y ayuda al equipo de desarrollo a escribir las pruebas funcionales. Ejecuta pruebas y difunde resultados en el equipo.
* **Coach**: Es responsable del proceso global y se encarga de guiar a los miembros del equipo para seguir el proceso correctamente.
* **Manager**: se encarga de agendar las reuniones, se asegura de que el proceso de juntas sea seguido, registra los resultados de las reuniones para futuros reportes para el Tracker
* **Gestor:** Es el vínculo entre clientes y programadores. Su labor principal es la de coordinación.
* **Tracker**: Es el encargado de seguimiento, proporcionando el feedback al equipo.